iNNolutioN
agile innovative solutions

Innolution Dependencies Are Killing Your Agility: Learn to Fight Back! Course Agenda

www.innolution.com | hello@innolution.com | 303 827-3333 | @krubinagile | kennethrubin

Module	Description
Course Introduction	• Introductions • Overview of the course agenda • Collection of student discussion topics
What Are Dependencies	• Definition of flow, dependency, blocker • Scope of coordination • Shared dependencies • Introducing the concept of N (the cardinality of the dependency set) • Must-have and nice-to-have dependencies • Upstream and downstream dependencies
Dependency Issues Grow Exponentially	• Dependency permutations • Probability of being blocked by dependencies
Why Are Dependencies Important	• Dependencies affect when work happens • Dependencies setup the problem, but “when” is the killer problem • Dependencies can impact predictability, cycle time, and prioritization • Use lifecycle profits and cost of delay to quantify dependency costs • Discussion of dependency impact on predictability • Discussion of dependency impact on cycle time • Discussion of dependency impact on prioritization
Defining Agile at Scale	• Agile at scale has been evolving • Single-team agility • Multiple collaborating agile development teams • End-to-end business agility (the goal)
Proper Unit for Organizing at Scale	• Creating a dependency solution for a small subset of your organization doesn’t really solve the flow problem • Need to take a more end-to-end business perspective • At the larger perspective, projects prove to be a poor unit of focus • Organizing around the current system architecture can be problematic • We need something more long-lived and business-focused like products, business capabilities, or value streams

Module	Description
Busting Some Common Dependency Myths	• One solution will work for all size dependency problems • Identifying the dependencies mostly solves the problem • Better project management will solve the "when" problem • Centralized demand management (capacity reservation) is the solution • Escalation will solve the dependency prioritization problem
Structural vs. Instantiated Dependencies	• What are structural dependencies (with example) • What are instantiated dependencies (with example) • What are blockers (with example) • Planning and WIP are sources of instantiated dependencies • We need to address both structural and instantiated dependencies
Structural Dependency Modeling and Analysis	• Define target focus • Perform process mapping • Create a collaboration map • Create a dependency table • Aggregate models to create a more holistic view • Model direct and indirect interactions • Techniques for analyzing structural dependency models
Structural Dependency Improvement Strategies Overview	• Organization structure and dependency path modeling process • Modeling frequency, impact (cost of delay), and likelihood of getting blocked • Overview of seven strategies for improving structural dependencies
Structural Dependency Improvement Strategy – Strive for Cross Functional and T-Shaped	• What is cross-functional and T-shaped • Comparing cross-functional and T-shaped • Structural dependency benefits of T-shaping • Structural dependency benefits of completely cross-functional
Structural Dependency Improvement Strategy – Create Feature Teams	• Myth that feature teams are the full solution • Definition of feature team and component team • Feature vs. component team examples • Feature teams can substantially reduce the number of dependencies • Shared services teams are equivalent to component teams • Solution combining feature and component teams • Three feature team impediments

Module	Description
Structural Dependency Improvement Strategy – Establish Communities of Practice	• Some additional feature team concerns • Desirable properties – conceptual integrity and reuse • Organization resistance to restructuring • What is a community of practice • Community of practice example • Spotify Guilds and Chapters • Dependency benefits of communities of practice
Structural Dependency Improvement Strategy – Organize into Coordinated Ecosystems	• Characteristics of coordinated ecosystems • Ecosystems can be of different scales • Coordinated ecosystems help when one feature team is too big • SAFe, LeSS, Spotify, and Nexus examples • Steps for defining a coordinated ecosystem • Dealing with non-aligned resources
Structural Dependency Improvement Strategy – Architect for Build Using / Self Service	• Anti-pattern – “build for me” • Discussion of “build together” (not quite open source) • Discussion of “build in” (I play in your sandbox) • Focus on “build using” (self service) • Four approaches to enable self service
Structural Dependency Improvement Strategy – Establish Team-to-Team Working Agreements	• The concept of pre-defined coordination • Scope and description of services • Intake process • Decisioning process • Interaction process • Service Level Agreements/Expectations • Deliverable-related metrics
Structural Dependency Improvement Strategy – Balance System/Portfolio WIP	• What is system/portfolio-level WIP • Visualizing portfolio-level WIP • Why portfolio-level WIP creates many dependencies • How do you determine the number of items to work on at one time • Goal is to balance system demand and structural capacity

Module	Description
Example of Structural Dependency Improvement	• At scale multiple structural improvement strategies will be necessary • Start by modeling and analyzing structural dependencies • Focus on end-to-end business agility • Decide on proper unit of focus for organizing at scale • Focus on a specific product and define a coordinated ecosystem • Inside the ecosystem organize into feature teams where possible • Use Build Using strategy with product/services outside of the ecosystem • Form working agreements with dependent entities • Have a single product owner to balance system/portfolio WIP

Instantiated Dependency Improvement Strategies	• Instantiated dependency management process
	• 5 strategies for identifying instantiated dependencies
	  ○ Impact analysis (placement of functionality)
	  ○ Groom/refine backlog for independence
	  ○ Definition of ready
	  ○ Story mapping
	  ○ Sprint Planning
	  ○ Discussion of anti-patterns
	• 4 strategies for recording/visualizing instantiated dependencies
	  ○ Dependency board
	  ○ Scrum task board
	  ○ Kanban ticket design (3 methods)
	  ○ Kanban board design (3 methods)
	• 7 strategies for coordinating instantiated dependencies
	  ○ Product backlog prioritization
	  ○ Core Scrum practices
	  ○ Scrum of Scrums
	  ○ Longer-term dependency boards
	  ○ Alerting downstream teams
	  ○ Systemic (inter-team) swarming
	  ○ Prioritize and manage WIP
	  ○ Discussion of anti-patterns
	• Blocker management
	  ○ Record and visualize blocker information
	  ○ Collect blocker-related data
	  ○ Blocker clustering (affinity grouping)
	  ○ Blocker analysis
	  ○ Blocker reduction strategies

Conclusion	• Review of the class • Final Q&A