



INNOLUTION DEPENDENCIES ARE KILLING YOUR AGILITY: LEARN TO FIGHT BACK! COURSE AGENDA



www.innolution.com



hello@innolution.com



303 827-3333



[@krubinagile](https://twitter.com/krubinagile)



[kennethrubin](https://www.linkedin.com/in/kennethrubin)

Module	Description
Course Introduction	<ul style="list-style-type: none"> • Introductions • Overview of the course agenda • Collection of student discussion topics
What Are Dependencies	<ul style="list-style-type: none"> • Definition of flow, dependency, blocker • Scope of coordination • Shared dependencies • Introducing the concept of N (the cardinality of the dependence set) • Upstream and downstream dependencies
Dependency Issues Grow Exponentially	<ul style="list-style-type: none"> • Dependency permutations • Probability of being blocked by dependencies • Definitely-will-happen vs. definitely-will-not-happen dependencies
Why Are Dependencies Important	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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 like products, capabilities, or value streams



Module	Description
Busting Some Common Dependency Myths	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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 Improvement Strategies	<ul style="list-style-type: none"> Organization structure and dependency path modeling process Modeling frequency, impact (cost of delay), and likelihood of getting blocked Five strategies for improving structural dependencies Create feature teams <ul style="list-style-type: none"> Myth that feature teams are the full solution Feature vs. component team examples Feature teams can substantially reduce the number of dependencies 3 feature team impediments Organize into coordinated clusters <ul style="list-style-type: none"> Characteristics of coordinated clusters Must have a single product owner to prioritize work Coordinated clusters help when one feature team is too big Architect for “build using” (self service) <ul style="list-style-type: none"> 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) 4 approaches to enable self service Establish team-to-team working agreements <ul style="list-style-type: none"> Normal class of service working agreement Expedited class of service working agreement Balance (typically reduce) system/portfolio WIP <ul style="list-style-type: none"> What is system/portfolio-level WIP Visualizing portfolio-level WIP Why portfolio-level WIP creates many dependencies Goal is to balance system demand and structural capacity



Module	Description
Example of Structural Dependency Improvement	<ul style="list-style-type: none"> At scale multiple structural improvement strategies will be necessary Consider the people and tech if you spin off your product/value stream/capability as a new product For whole group using coordinated cluster strategy Inside whole group organize into feature teams Use Build Using strategy with product/services outside of whole group Form working agreements with dependent entities Have a single product owner balance system/portfolio WIP
Instantiated Dependency Improvement Strategies	<ul style="list-style-type: none"> Instantiated dependency management process 5 strategies for identifying instantiated dependencies <ul style="list-style-type: none"> Impact analysis (placement of functionality) Groom/refine backlog for independence Definition of ready Story mapping SAFe PI Planning Discussion of anti-patterns 4 strategies for recording/visualizing instantiated dependencies <ul style="list-style-type: none"> SAFe program board Scrum task board Kanban ticket design (3 methods) Kanban board design (3 methods) 6 strategies for coordinating instantiated dependencies <ul style="list-style-type: none"> Product backlog prioritization Scrum of Scrums Alerting downstream teams Systemic (inter-team) swarming SAFe program board Prioritize and manage WIP Discussion of anti-patterns Blocker management <ul style="list-style-type: none"> Record and visualize blocker information Collect blocker-related data Blocker clustering (affinity grouping) Blocker analysis Blocker reduction strategies
Conclusion	<ul style="list-style-type: none"> Review of the class Final Q&A

