



1

# LeSS Practitioner

Craig Larman  
[craiglarman.com](http://craiglarman.com) & [less.works](http://less.works)

Please...  
Do not copy or share this material, or re-use for other education.  
Exceptions require prior written consent of the author.  
Copyright © 2017 Craig Larman, All rights reserved.  
May not be reproduced without written consent of the author.

v.55

2

# Opening Topics

3

# Scaling?

4

first, a caution...

5



One of the directors of SAGE was discussing why the programming had gotten out of hand. He was then asked, **“If you had it to do all over again, what would you do differently?” ...**

6

His answer:  
**“Find the ten best people and write the entire thing themselves.”**

[Horowitz74]

7

after years working in  
**large**  
**multisite**  
**offshore**  
development,  
our **key advice?** ...

8

large - don't

multisite - don't

offshore - don't

9

but groups still 'scale',  
for reasons...

compelling  
("create LTE")

questionable  
("low-cost sites")

10

so is LeSS for **scaling**?

11

Descaling &  
Simplifying

12

“How can we apply agile at scale in our big complex organization?”

13

13

is this the right question? ...

14

14

traditional large groups are complicated — though not because they need to be, but because **their organizational designs create an illusion of ‘necessary’ complexity**

15

15

**This** is an Important Question...

“How can we **simplify** the unnecessarily big and complex organizational design, and **be agile** rather than **do agile**?”

16

16



“agile”?

17

because the **word “agile”**  
has become a meaningless  
**jargon** synonym for **anything**,  
will avoid it and use...

**adaptive**

18



**BIG Idea**

*be adaptive*  
not do adaptive

19

19



**BIG Idea**

*be agile*  
not do agile

20

20



## BIG Idea

LeSS **descales**  
organizational complexity,  
dissolving unnecessary  
complex organizational  
solutions, and solving in  
simpler ways.

21

21

descaling

simplifying  
over  
scaling

22

22

LeSS  
More with LeSS

23

23



Keeping it Simple

24

24

# Some Big Ideas

25

not here just to explore  
“LeSS system”, rather:

**how to think about  
systems in general**

26

not about “agile coaching”

**organizational  
design  
consulting**

27

**own**

vs

rent

28

**why**

29

don't believe  
anything i say

fads & gurus -> **insight**

rent -> **own**

30

30



jargon

31

31

“It is difficult to get a man to  
understand something when  
his job depends on not  
understanding it.”

— Upton Sinclair

32

32

Job Safety, but not Role Safety



33

Adoption: **Top-Down** & **Bottom-Up**



34

Adoption: **Deep & Narrow**  
over Broad & Shallow



35

Your  
Backgrounds

36



(optional) team re-organization

1. group by role
2. coach records data
3. form diverse teams

> *NB: teams will re-form about 1/2 through the course*

37



## Introductions

38

38



team: **standing**: round-robin

- > **briefly** (30 seconds each), introduce each other
- > **name card** on “somewhere”; use **THICK BLACK marker**
- > when team done, please **SIT**

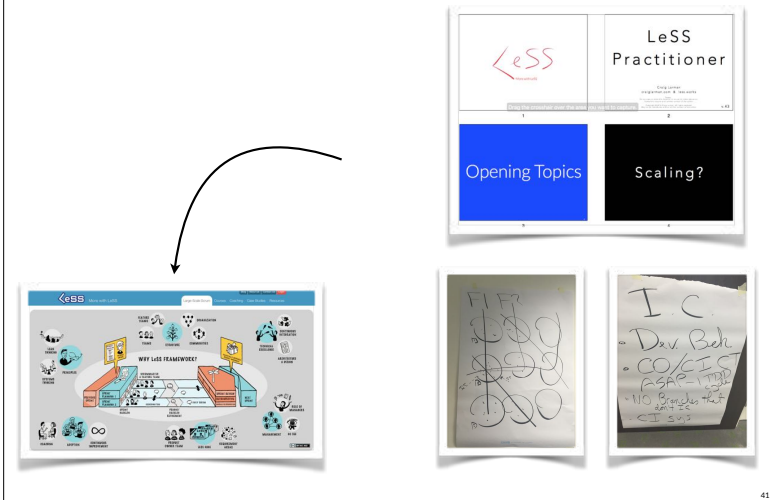
39

39

# Practicalities

40

## Course **PDF** & Room **Photos** at **less.works**



41



42

8:30

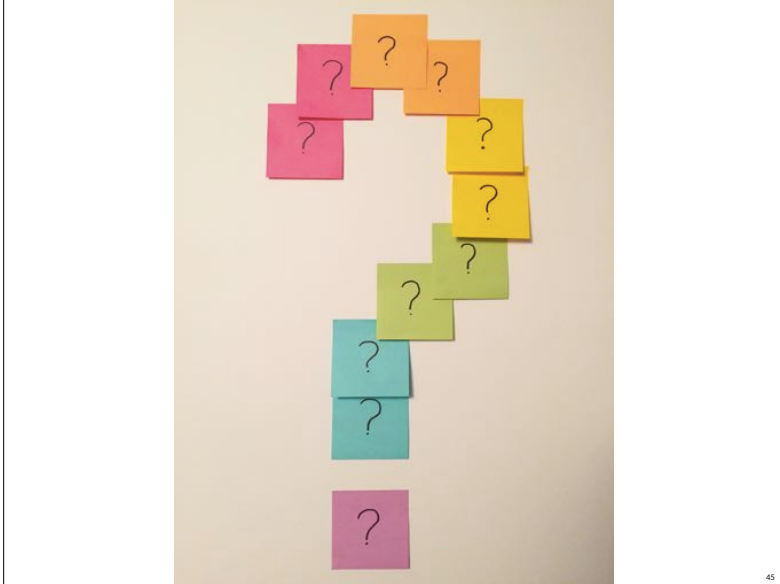


18:00

43

“16:30 pm” drinks?

44



45



46



47



48



**local meetups**  
this week?

49

# Overview & Objectives

50

## Where are We?

1. Opening Topics
2. System Optimization, not  
Local Optimization
3. Organizational Structure
4. LeSS Overview

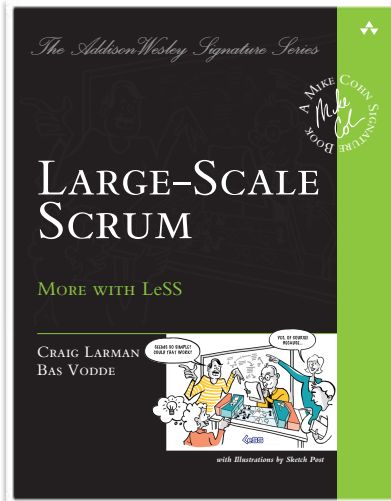
51

## Course Styles

1. **style-1** is **systems-  
modeling oriented**
2. **style-2** is **special topics**  
>latter part of each day

52

## We Will Demote “Easy Topics”



- 1 More with LeSS 1
  - 2 LeSS 5
- LeSS Structure**
- 3 Adoption 41
  - 4 Organize by Customer Value 67
  - 5 Management 111
  - 6 ScrumMasters 139

- LeSS Product**
- 7 Product 155
  - 8 Product Owner 171
  - 9 Product Backlog 197
  - 10 Definition of Done 231

- LeSS Sprint**
- 11 Product Backlog Refinement 249
  - 12 Sprint Planning 275
  - 13 Coordination and Integration 285
  - 14 Review & Retrospective 313

53

53

## Are We Covering All Course Pages?

### Sample Topics in the Course Material

- > Why LeSS?
- > Preparing for Sprint 1
- > Product Backlog Refinement
- > Sprint Planning
- > Technical Excellence
- > Sprint Review
- > Retrospectives
- > Done & Undone
- > DevOps
- > LeSS Huge
- > Feature-Team Adoption Maps (common in *incremental* LeSS Huge adoptions)
- > LeSS Rules
- > LeSS Principles
- > Product Owner
- > Managers
- > Scrum Masters
- > Product Backlog & Tools

54

54

## Likely Objectives: You can...

- > redesign org from **local optimizations** to **global system optimizations**
- > **define a product** broadly
- > motivate & define **LeSS org design** (structure, roles, policies, ...)
- > advise on **LeSS adoption**
- > know & coach **LeSS Sprint** (events, coordination, ...)
- > explain LeSS & LeSS Huge **frameworks**
- > explain **LeSS principles** & make connections
- > answer “**why LeSS?**”
- > explain **roles**

55

55



coach

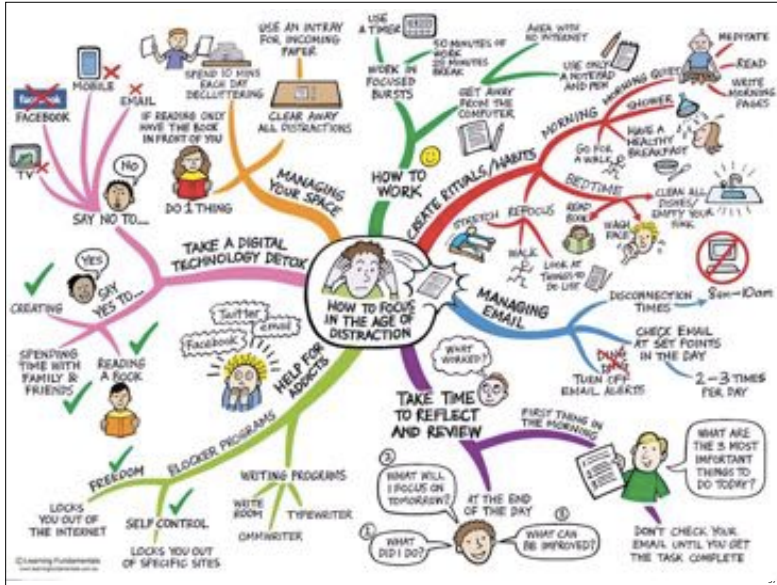
- > discuss mindmap format & why

team

- > mindmap objectives

56

56



57

team

- > update the mindmap with learnings so far

58

System Optimization  
not  
Local Optimization

59

Where are We?

1. Opening Topics
2. **System Optimization, not Local Optimization**
3. Organizational Structure
4. LeSS Overview

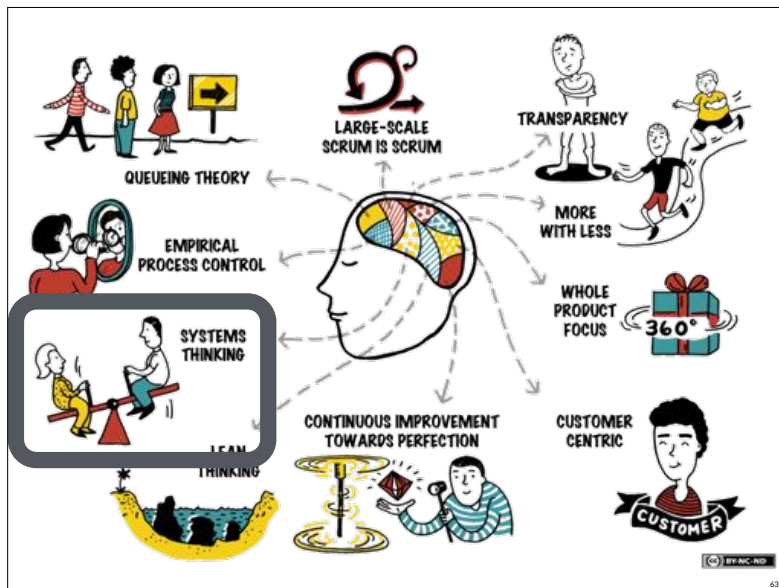
60

what are we  
about to learn?

61

# Systems Modeling

62



63

### Scaling Lean & Agile Development

Thinking and Organizational Tools for Large-Scale Scrum

Craig Larman  
Bas Vodde

### Thinking Tools

- 2. Systems Thinking
- 3. Lean Thinking
- 4. Queueing Theory
- 5. False Dichotomies
- 6. Be Agile

### Organizational Tools

- 7. Feature Teams
- 8. Teams
- 9. Requirement Areas
- 10. Organization
- 11. Large-Scale Scrum

64

64

## Systems Thinking

learn to reason about

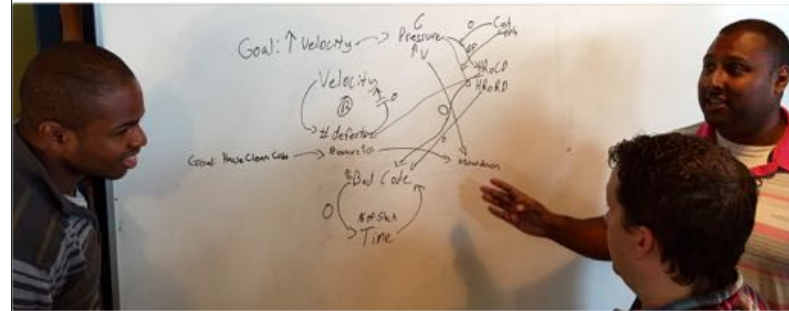
**'any' system**

not just 1 system

*how?...*

65

## Sketch a **System Model**



AKA **causal loop diagram**

66

**we model to have a  
conversation**

the output is shared  
understanding, not a model

67

**own** vs rent  
focus on **why**

68

“all models are wrong,  
but some are useful”

— George Box

69

“mental models”

cognitive bias

false beliefs

self-awareness  
to self-doubt

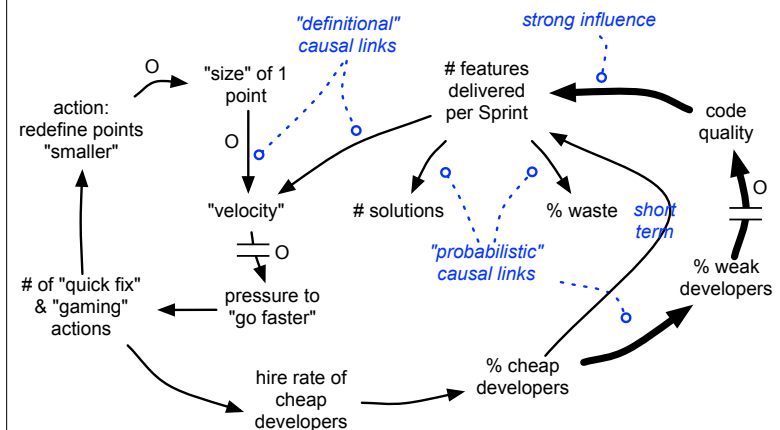
70

coach:

> sketch a system model

71

## Common Elements



72

we'll start with a simple &  
familiar situation

to focus **first** on  
**notation & technique**

rather than “content”...

73



team

> sketch a system model, considering this puzzle:

> **“We don’t have time to create clean code, because we are too busy going slow because of dirty code.”**

> start with these variables; write the **bold words** verbatim

1. **% clean code**
2. **time available to craft clean code**
3. **effort to create a new feature**
4. **velocity** (...of delivering new features)
5. **# defects**
6. **effort handling defects**
7. **pressure to deliver and “go faster”**

74

74



coach: debrief

- > correlations rather than causal relations?
- > “definitional” causal links?
- > “mental models”?

75

75

**we model to have a  
conversation**

the output is shared  
understanding, not a model

76

76

**own** vs rent  
focus on **why**

77



group

- > in **LeSS**, **when** do systems modeling?
- > in what contexts or meetings can it help?

78

Local  
Optimization

79

in traditional large-scale  
organizational design, the  
**overarching & repeating**  
theme?

**local optimization**

80





individual:

> write a definition of what you think is **local optimization**

coach: review

81

81

examples of  
local optimization...

82

82



83



84



individual

- > identify 1 *specific* example you've seen of **"working to job title" rather than "moving the ball"**
- > identify 1 *specific* example you've seen of **"chopping onions" rather than "delivering dishes"**

coach: review

85

85

## Local Optimization *Cognitive Bias*

"Everyone is **busy** and doing **their best**, **working efficiently** on **their task**, yet the **system** is **delivering slow** and **not delighting** the user."



86

86

## Local Optimization *Cognitive Bias*

"It's more **efficient** or **productive** when a person/group does **one specialization.**"



87

87

## Local Optimization **Consequence**

justified as:

**efficient**  
**productive**  
**best**  
**good**

... but consequence is:

**system sub-optimization** of (e.g.)

customer value  
customer cycle time  
customer delight  
company robustness  
company adaptiveness

88

88



## Why Local Optimization?

89

89

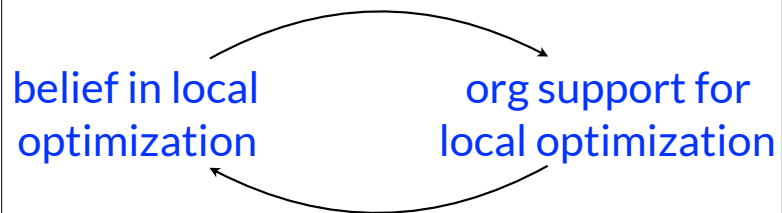
local optimization  
is a  
**cognitive bias**

list of cognitive biases?

90

90

## Reinforcing Loop



91

91

## **Local Optimization** is related to...

- > management by objectives
- > measurement & metrics
- > resource management
- > performance management
- > appraisals

92

92



group

- > examples that led to local optimization due to:
  - > **metrics** or
  - > **objectives** & **appraisals** or
  - > **resource management**

93

# System Optimization

94

Systems Optimization

“watch the **ball**,  
not the players”

“deliver the **dish**,  
not the onions”

95

Systems Optimization

goal:  
optimize  
system

96

the **One True**  
system optimizing goal?

97



## BIG Idea

leadership needs to  
agree on the system  
optimizing goal

98



## BIG Idea

organizational design  
elements should be  
consistent with the  
system optimizing goal  
(i.e. pass the “fitness  
function” test)

99

## System Goals vs Indirect Wishes & Constraints

- > **goals** that the system can be designed to **definitely & directly influence**
  - > highest value, high agility, low cycle time
- > **indirect wishes** (not “goals” in this usage)
  - > increased market share
- > **constraints** (seldom would the CEO describe these as the goal of the system or company)
  - > reduced cost
  - > reduced risk
- > **TIP! beware confusing these**

100

## Systems Optimization

- > there are no 'good' or 'bad' organizational systems/goals
- > but if the **observed** behavior is inconsistent with its **espoused** optimizing goal, it is **inconsistent**

101



coach

- > **counter-intuitive** example of a local optimization inconsistent with system optimization goal?

102

system optimizing  
goal(s) of  
**LeSS**  
organizational design?

103

## the **LeSS** System Goals

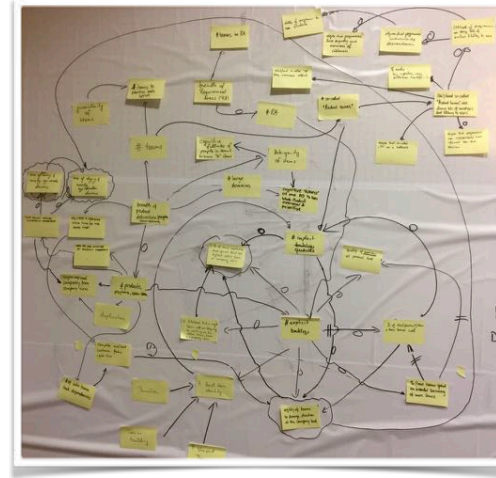
- > highest-possible system optimization for
  - > **deliver highest customer value first**
  - > **cheap & easy adaptiveness, driven by learning** ("turn on a dime, for a dime")

104

# Finding Papers on the Wall ;)

105

## Problem: Finding Variables on the Wall



106

## One Solution: Color Coding!



107



categorization of variables (to find them more easily on the wall):

- > related to or attribute of an **Artifacts & Misc Things**
  - > % clean code, # people in company, % items worked on of highest value from company perspective, # items in the PB to prioritize each Sprint, revenue of product, usability of feature, # roles in groups
- > related to or attribute of an **Action/Activity**.
  - > effort to implement a new feature (e.g. in person hours), effort to refine, pressure to "go faster", effort to decide

108



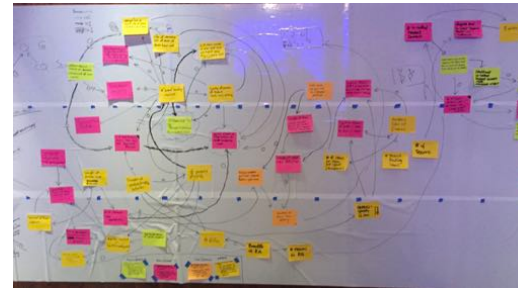
- > related to or attribute of **Person/Team/Group Behavior or Cognition**
  - > **degree of empathy for customers, # skills, “velocity”, breadth of domain knowledge**
- > (excluding effort by Person/Team/Group on a specific activity) **Time/Duration**
  - > **length of Sprint, duration until feedback, time available to craft clean code**

109

109

## Color Coding Important?

**NOT IMPORTANT**; it's just a *search tool*



110

110



coach & group

- > clarify legend colors
- > make legend
- > put on wall

111

111

# Local Optimization in Backlogs

112



let's start to apply  
system modeling to  
“scaling agile”  
organizational design  
choices...

113



team:

- > in reality the prior variables should remain, since it's **all one system**
- > but **wall space management...**
- > **tear down the old model**

114



team: sketch a system model, bearing in mind this puzzle:

- > **1 product, many teams, each team has a Team “Product Backlog” prioritized by a Team “Product Owner”**
- > **start** with these variables **verbatim**
  1. **# backlogs** (e.g. 1 backlog per team, 1 backlog for 2 teams, 1 backlog for all teams) (Artifact/Thing)
  2. **% of total (product) items a team knows well (requirements & design)** (Person/Team/Group Behavior/Cognition)
  3. **agility of teams to change direction at the company level** (i.e. the cost of changing) (Person/Team/Group Behavior/Cognition)
  4. **% of items worked on each Sprint that are highest value from a company view** (Artifact/Thing)
  5. **likelihood that a single team will see they may be working on low-value items, from a company view** (Person/Team/Group Behavior/Cognition)
  6. **local team identity** (Person/Team/Group Behavior/Cognition)

115



coach: relationship of these  
variables?

1. **agility of teams to change direction at the company level**
2. **% of items worked on each Sprint that are highest value from a company view**

116



coach

- > if system optimizing goals are
  - > **highest value & adaptiveness**,  
at company level
- > ... *how many backlogs?*
- > ... *is the answer "good" or "bad"?*
- > ... *did the coach tell you the answer?*

117

117



## BIG Idea

the purpose of  
**adaptiveness/agility?**

...to cheaply & easily support  
changing direction (**re-  
prioritization**) to work on  
**continually newly-discovered**  
highest value

118

118



## BIG Idea

the purpose of  
**adaptiveness/agility?**

*turn on a dime,  
for a dime*

119

119



## BIG Idea

the purpose of  
**adaptiveness/agility?**

**RE-prioritization**  
from continual **learning**,  
not "prioritization"

120

120



## BIG Idea

continual  
**RE-prioritization**

not  
“prioritization”

121

121

support **secondary goals**  
or **constraints**  
(e.g. “team local identity”)

**without sub-optimizing  
the system goal**

122

122

## Local Optimization *Cognitive Bias*

what is the  
misunderstanding  
when someone says  
“**efficient**” or  
“**productive**”?



123

123

“**local  
optimization**  
in backlogs”

124

124

therefore...

125

LeSS Rule(s)

1 Product Backlog  
(and no  
Team “Product” Backlogs)

126

126

1 Product Backlog ...

**and no FAKE “redefining”  
by calling a set of team  
backlogs “*part of 1  
Product Backlog*” or  
“*views on the 1 backlog*”**

127

127

Reflections

128

notice that  
the coach has  
**not** “taught” the  
elements of LeSS

129

129

**own**

vs

rent

130

130

focus on **why**

131

131

biased by  
choice of variables?

132

132

## How to **Find** Useful Variables?

strongly related to...

**optimizing goal(s)**

**secondary goals**

**indirect wishes**

**constraints**

133

133

“driving variables”

134

134

**highlight** variables in  
the model strongly  
related to (or actually)  
the **system optimizing**  
**goal(s)**

135

135

**definitional** links

**probabilistic** links

136

136

grasp the relative  
magnitude of variables  
& influence

e.g.

**effort to coordinate** vs  
**effort to clarify**

137

137

preparing for the  
next exercise...

138

138



tip:

share/rotate the **PEN**

look for ways for  
everyone to be engaged

139

139



team

> now you've worked together  
for a little time, have a  
“**norming**” discussion (a  
retrospective)

140

140



team

- > **update** your system model to be like the “**good enough**” model refined during the debrief
- > **duplicate** the same physical **layout** as the good-enough model; this will help later on

141

141



team

- > **highlight** variables equal to or strongly related to the **system optimizing goal(s)**

142

142

Local Optimization  
in Backlogs  
(again)

143

1 Product Backlog  
but still

**constraints due to  
teams knowing  
disjoint items...**

144

144





coach: visualize these variables with a Venn diagram

team: update your system model with:

- > **size % of intersection set of items all teams know well** (Person/Team/Group Behavior or Cognition)
- > **average size % of disjoint set of items only known by 1 team** (this is the complement of the prior variable and could be ignored, but will help clarify a future point) (Person/Team/Group Behavior or Cognition)

145

145

## Minor Note

> very similar variables:

- > **% of total (product) items a team knows well (requirements & design)**
- > **size % of intersection set of items all teams know well**

146

146



coach

- > if system optimizing goals are
  - > **highest value & adaptiveness**, at company level
- > ... *as a trend, should a team learn about more items (as a % of total) or less items?*

147

147

**“implicit** backlogs”

**“team-level views** on the 1 backlog”

148

148

if the “disjoint set”  
value is high

->

**implicit backlogs  
grow stronger**

149

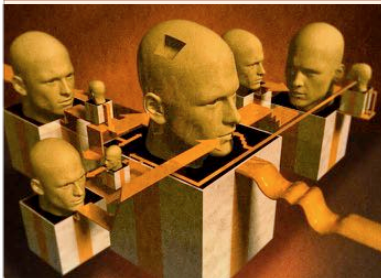
every backlog is a  
**queue**

150

## Scaling Lean & Agile Development

Thinking and Organizational Tools  
for Large-Scale Scrum

Craig Larman  
Bas Vodde



### Thinking Tools

2. Systems Thinking

3. Lean Thinking

4. Queueing Theory

5. False Dichotomies

6. Be Agile

### Organizational Tools

7. Feature Teams

8. Teams

9. Requirement Areas

10. Organization

11. Large-Scale Scrum

151



## COACH & group

> update your system model:

- > **effort teams spend on broader learning of more items** (Action/Activity)
- > **# implicit backlogs** (Artifact)

152



coach

- > if system optimizing goals are
  - > **highest value** & **adaptiveness**, at company level
- > ... *should teams **spend time** learning about **more** items?*

153

153

Q:

“Isn’t it *inefficient* and *wasteful* to have teams learning about many items?”

154

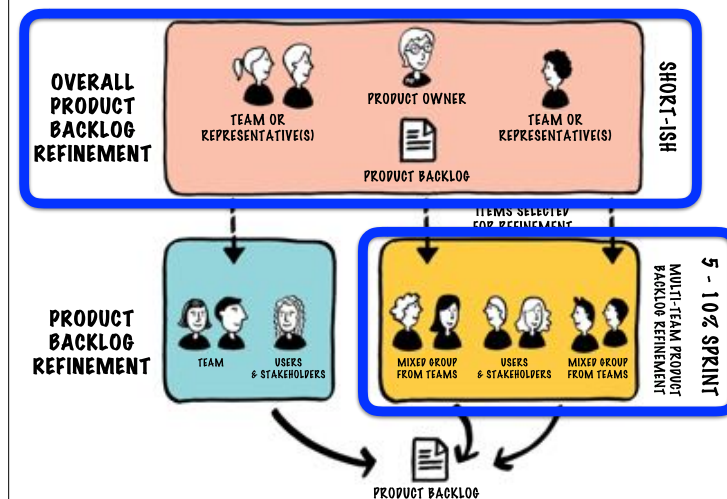
154

therefore...

155

155

### LESS PRODUCT BACKLOG REFINEMENT



156

## LeSS Rule(s)

Do **multi-team PBR**  
and/or **overall PBR** to  
increase shared  
understanding and  
broader learning.

157

157

## Local Optimization Cognitive Bias

Q: “Why does each team  
have a team backlog, and that  
only does narrow learning?”

A: “Because it’s **best, and  
most efficient.**”

158

158

## Descaling with LeSS

remove



local optimization of  
**backlogs...**

**1 Product Backlog**

that comes from:  
team backlogs  
(and all their org  
design elements)

(and no *hidden*  
“team backlogs/  
views”, and avoid  
“implicit backlogs”)

159

159



## team

- > **sync** your system model
- > **duplicate** the same physical **layout** as the good-enough model; this will help later on

160

160

# Reflections

161

team: standing: round robin

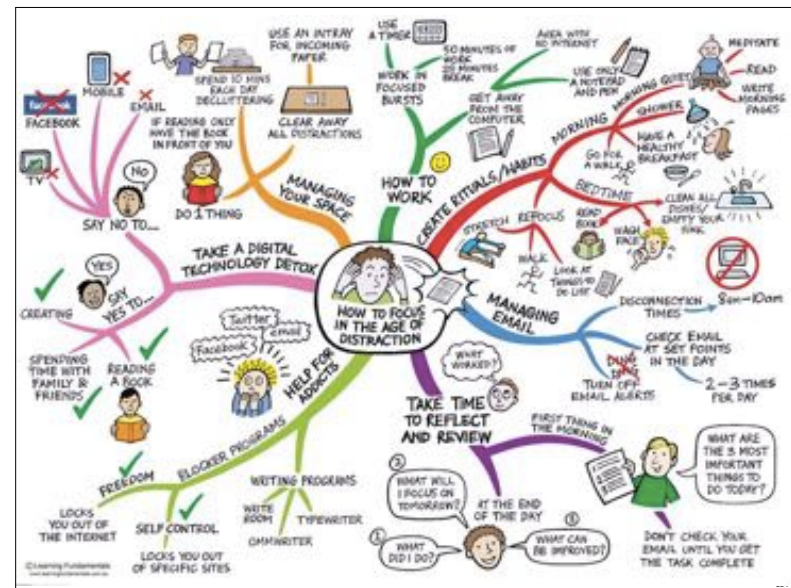
- > most **noteworthy** or **interesting** idea so far?
- > please **sit** when team is done

162

individual

- > most **noteworthy** or **interesting** idea so far?
- > write a summary of it on a separate sticky note
- > put all the notes together on a wall somewhere

163



164

# Opening Topics (again)

165

# Course Misc.

166

related knowledge...

167

## Related LeSS Courses

- > LeSS for Executives (2-4 days)
- > Certified LeSS “Basics” (1-day)
  - > upcoming via Scrum Alliance

168

<eSS

+



169

## Prerequisites

- > understand one-team Scrum
- > completed any pre-readings

170

When I say...

- > **“This question is related to standard 1-team Scrum...”**
- > am not saying this to make people feel bad that they might not know basic Scrum, **but to delineate Large-Scale Scrum rules from Scrum rules**

171

My  
Background

172

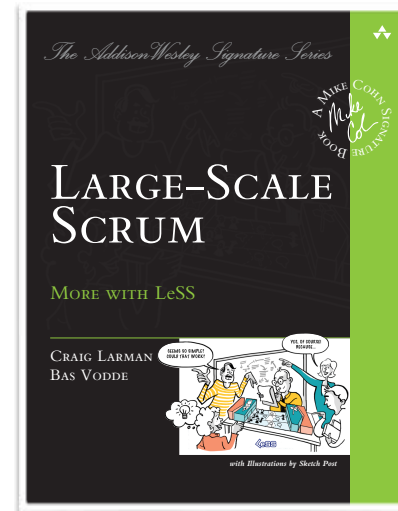
Craig Larman

co-creator of LeSS (with Bas Vodde)

large + multisite + 'offshore'  
large-scale embedded systems  
large-scale financial systems  
large-scale telecom systems

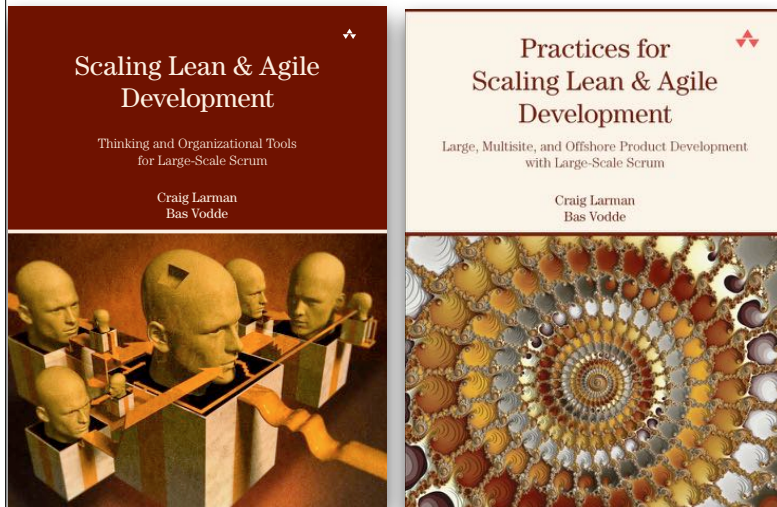
173

3rd LeSS book...



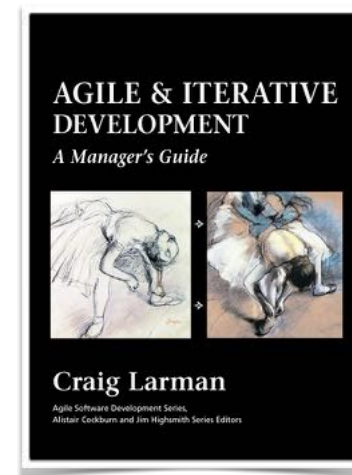
174

First Two LeSS books...



175

Early Agile Book



176

176



## Early Scrum Adopter & Coach...



Chevron  
Research  
Center

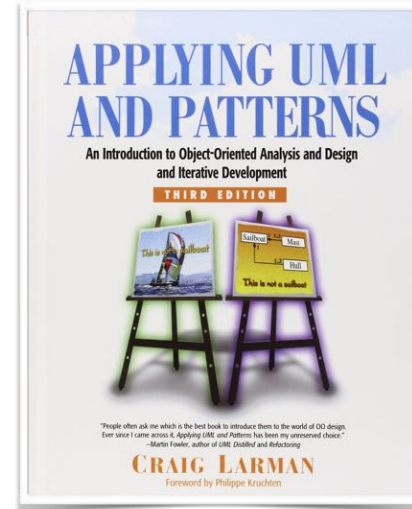
early  
CST

# 1998



177

## Architecture, Patterns, OO Design, ...



178

Valtech  
Global Delivery Center

- > served as chief scientist @ Valtech
- > helped create “agile offshore” in LeSS
- > lived in Bengaluru

179

- > lead coach of lean software development @ Xerox

180

## LeSS consultant @

- > UBS
- > ION
- > BAML
- > Ericsson
- > Nokia Networks
- > CISCO (& Tandberg)
- > JPMorgan
- > Xerox
- > bwin.party, ...

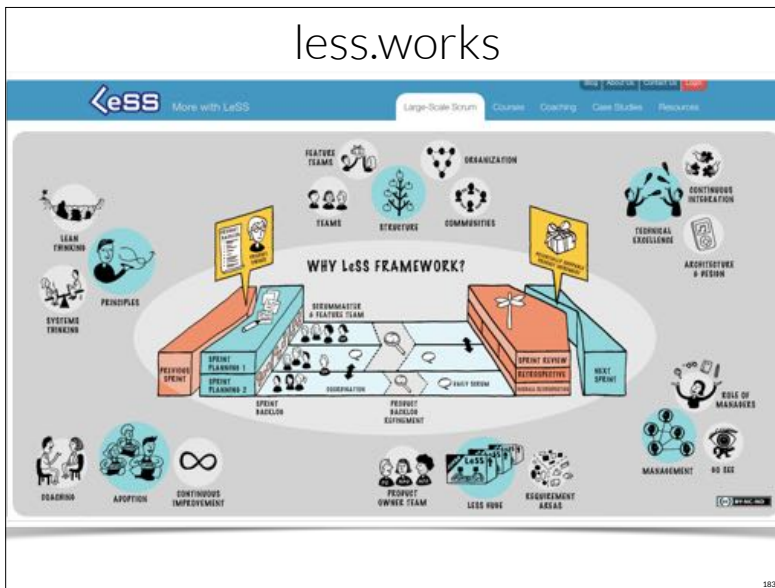
181

181

# Learning Resources

182

## less.works



183

183

- 1 More with LeSS 1
- 2 LeSS 5

**LeSS Structure**

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 133

**LeSS Product**

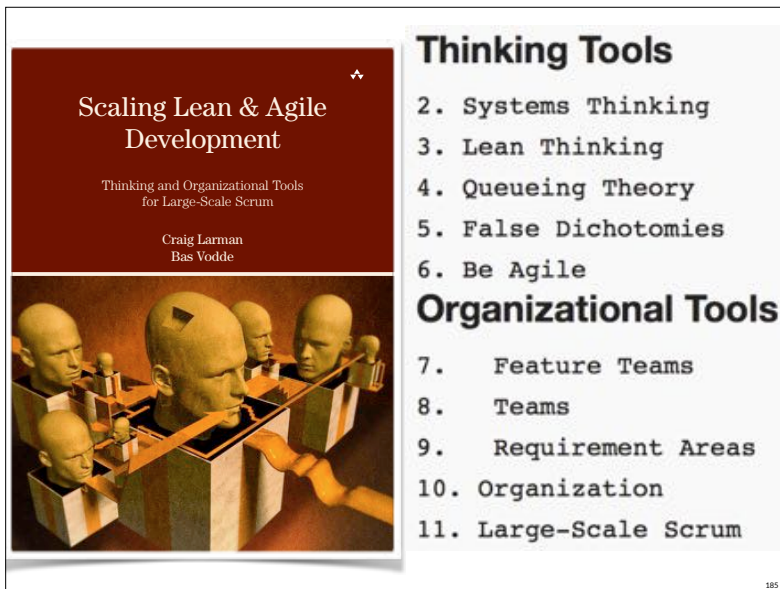
- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

**LeSS Sprint**

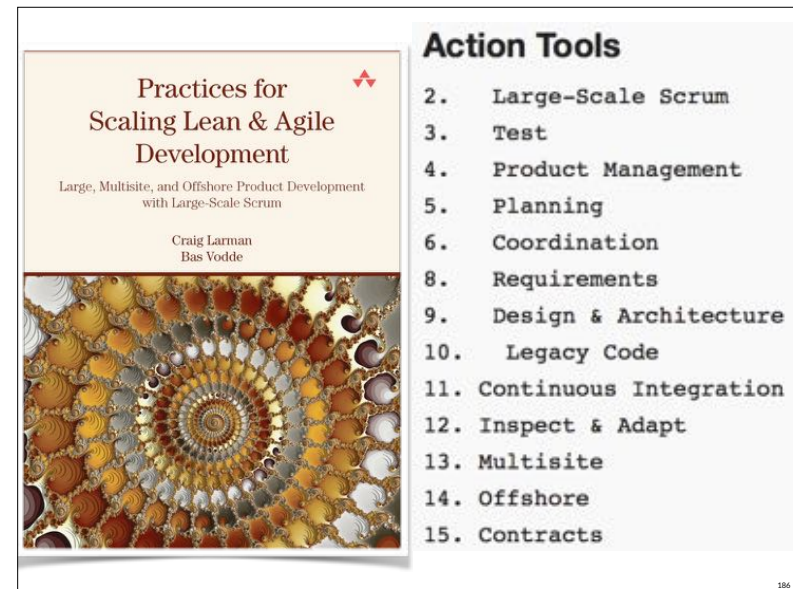
- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

34

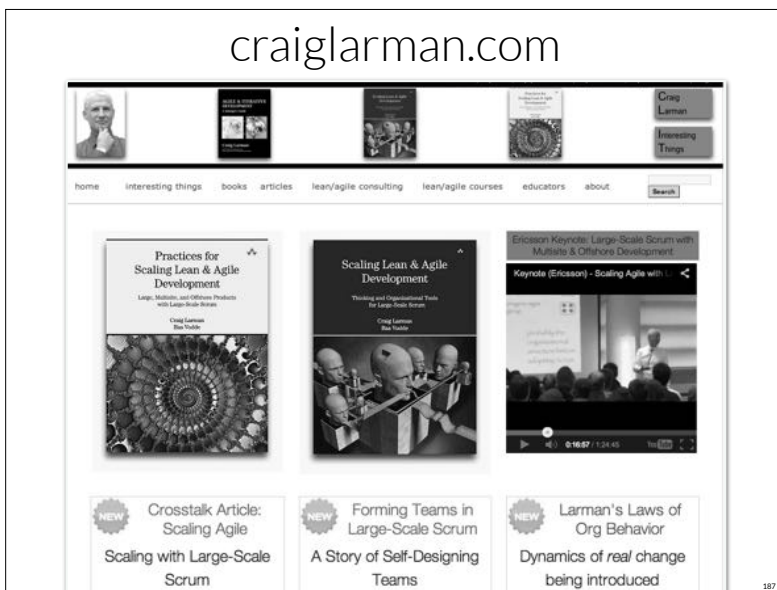
184



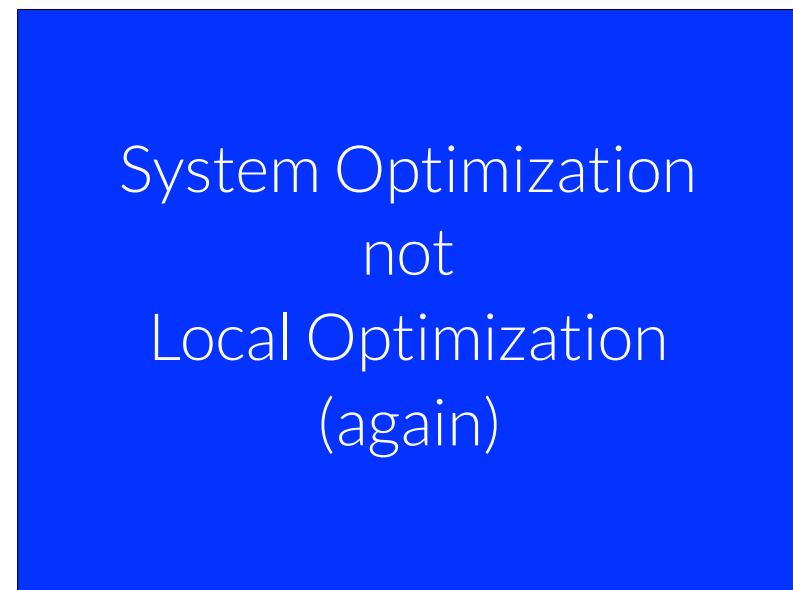
185



186



187



188

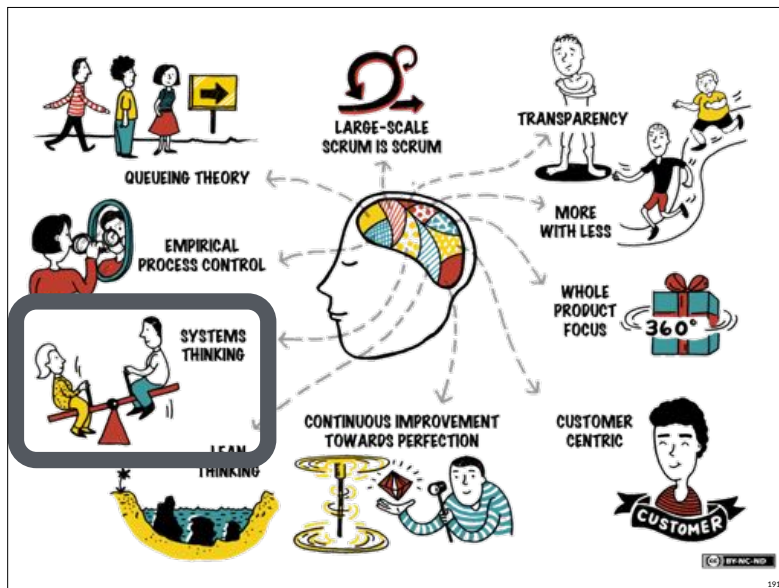
# Systems Thinking

189

after prior exercises,  
**“system”,**  
**“see the whole”,**  
**“optimize the whole”**  
 might be more clear

190

190



191

**Scaling Lean & Agile Development**  
 Thinking and Organizational Tools for Large-Scale Scrum  
 Craig Larman  
 Bas Vodde

## Thinking Tools

2. Systems Thinking
3. Lean Thinking
4. Queueing Theory
5. False Dichotomies
6. Be Agile

## Organizational Tools

7. Feature Teams
8. Teams
9. Requirement Areas
10. Organization
11. Large-Scale Scrum

192

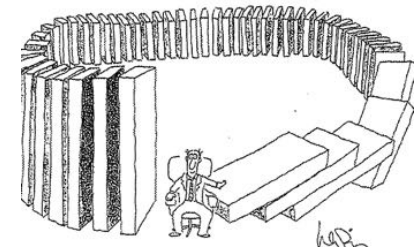
## What is the **SYSTEM**?

- > probably...
- > the entire company +
- > customers/markets +
- > supply chain
- > *it isn't a group within a company*

193

## Systems Thinking

- > see the whole
- > optimize the whole
- > focus on **interaction effects**, not on separate parts



194

or more detailed...

195

## Systems Thinking

- > understand there is a **SYSTEM**
- > learn to reason about 'any' system, not 1 system
- > see the whole, over space and time
- > see how things influence one another and the interaction effects
- > optimize the whole
- > beware local-optimization cognitive bias
- > think & talk about system dynamics by drawing systems model diagrams in groups

196



(optional) individual:

- > draw a **graphic** for each systems thinking idea
- > **when done, please stand**

197

197



(optional) pairs: standing

- > pick one person to play “teacher”
- > with your graphics (*but without looking at course notes*), teacher explain to partner the systems thinking ideas
- > do **NOT** teach both graphics
- > **please sit when done**

198

198

# Local Optimization in Planning: The Contract Game

199

## Where are We?

1. Opening Topics
2. System Optimization, not Local Optimization
3. Organizational Structure
4. LeSS Overview

200

200



## Business-R&D Collaboration Change

[Business] is used to “throwing the project over the wall” and holding engineering/development responsible for meeting needs. Scrum puts this responsibility back on the Product Owner and customers through the inspect and adapt and the Sprint Review.

-Ken Schwaber

201

201

## Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools  
**Working software** over comprehensive documentation  
**Customer collaboration** over contract negotiation  
**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

202

## the Contract/Commitment Game

203

203



team: sketch a system model, bearing in mind this scenario puzzle:

- > **the Contract/Commitment Game exists. e.g., there was “internal contract negotiation”, a project/program, a scope & date deadline “internal contract” with a project/program manager responsible for the project/program, “Business” has “thrown the project over the wall” and holds Development responsible for meeting needs, etc.**
- > **start** with these variables **verbatim...**

204

204



team: sketch a system model, focusing first on the relationship of these variables:

- > **start** with these variables **verbatim** (some may already be in the model)
  1. **gap between true situation and "plan"** (Artifact/Thing) ("plan" is for "internal contract")
  2. **strength of carrots/sticks to "meet plan"** (Artifact/Thing)
  3. **degree of "fear"** (Person/Team/Group Behavior/Cognition)
  4. **pressure to deliver and "go faster"** (Action/Activity)
  5. **transparency** (Person/Team/Group Behavior/Cognition)
  6. **agility to adapt early based on understanding real situation** (Action/Activity)
  7. **agility to change direction based on learning** (Person/Team/Group Behavior/Cognition)
  8. **quality of work** (Person/Team/Group Behavior/Cognition)
  9. **technical & organizational debt** (Artifact/Thing)
  10. **% of effort dealing with consequences of debts** (Action/Activity)
  11. **velocity to sustainably create new features** (Person/Team/Group Behavior/Cognition)
  12. **degree of risk & probability of "failures"** (Artifact/Thing)
  13. **% of items worked on each Sprint that are highest value from a company view** (Artifact/Thing)

205

205



coach & teams:

- > debrief
- > **highlight** variables strongly related to the system-optimizing goal of **adaptiveness** ("agility" — turn on a dime for a dime)

206

206



coach

- > if system optimizing goals are
  - > **highest value & adaptiveness**, at company level
- > ... *should there be the **Project Contract/Commitment Game?***

207

207



coach, (and obviously...)

- > if **eliminating** the **root causes** of **technical & organizational debt** are desired
- > ... *should there be the **Project Contract/Commitment Game?***

208

208



## Scrum ends the Contract Game...

[Business] is used to “throwing the project over the wall” and holding engineering/development responsible for meeting needs. Scrum puts this responsibility back on the Product Owner and customers through the inspect and adapt and the Sprint Review.

-Ken Schwaber

209

209



## BIG Idea

**Scrum** ends the  
Contract/Commitment  
Project Game

(and hence, **Large-Scale Scrum**)

210

210

## Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
**Customer collaboration** over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

211

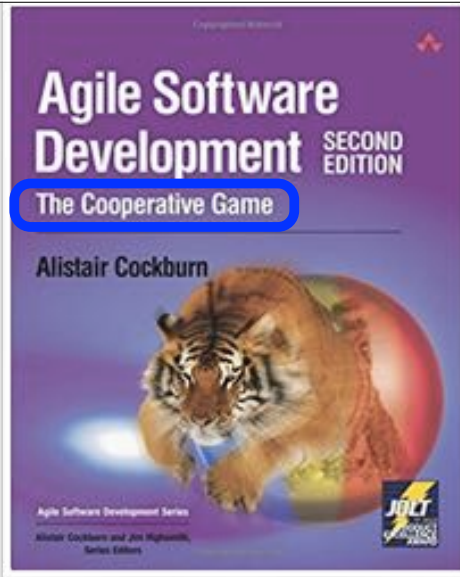


## BIG Idea

“**Agile**” ends the  
Contract/Commitment  
Project Game

212

212



213



## BIG Idea

Scrum & “Agile” is NOT a way to “efficiently deliver the project contract” in the “delivery phase”

214

who is the Product Owner in this case?...

215

## SCRUMGUIDE

By Ken Schwaber

**Tip:** For commercial development, the Product Owner may be the product manager. For in-house development efforts, the Product Owner could be the manager of the business function that is being automated.

216



217



218



219



220

reminder...

1 “50 person” group

not entire company

221

221



coach & group

- > what is the relationship between the “*Contract/Commitment Game*” system dynamics (which assumed a “**6-months duration**”) and the system dynamics of a Team making a **scope commitment** in a **two-week Sprint**?

222

222



coach & class

- > in Scrum, does the Team make a *scope commitment* to delivering “A, B, C, D” items in the Sprint?

223

223

Sprint **Forecast**, not “Commitment”

*Scrum Guide:*

“Sprint Planning Topic One: What can be done this Sprint? The Development Team works to **forecast** the functionality”

224

224

the Contract Game is meant to  
end in **basic Scrum**

why explore this **introductory**  
topic in this course?

225

225

**change implications** are  
especially clear in **large-scale...**

...where Contract Game **elements**  
are “**baked in**”...

226

226

projects

programs

cascading commitments

PMO,

project & program managers  
managing projects/programs

227

227

projects

programs

cascading commitments

PMO,

project & program managers  
managing projects/programs

228

228

there is no blame

229

229

## Local Optimization Cognitive Bias

Q: “Why do you have the Contract/  
Commitment Game in planning?

Why do you have a project/  
program and project/program  
manager to deliver a project?”

A: “Because it’s **best**.”

230

230

focus on **why**

231

231

want to see the explanation again?

The screenshot shows the LeSS website interface. The header includes the LeSS logo and navigation links for Profile, Courses, Event, and Company. A sidebar on the left lists LeSS Resources, Articles, Books & LeSS Chapters, Book images, Videos (highlighted), Discussion Groups, and Graphics. The main content area is titled 'Videos' and lists various video resources related to LeSS, including 'Systems Optimization & Organizational Design: a LeSS Perspective' by Craig Larman, 'Introduction to LeSS (short video)' by Craig Larman, and 'Large-Scale Scrum (LeSS)' by Bas Vodde. The list continues with several other videos from 2011 to 2015, covering topics like Agile Singapore, Valtech France, and Agile India 2013.

232

232



## Descaling with LeSS

remove



local optimization  
of **planning**...

that comes from:  
the Contract Game  
(and all its org  
design elements)

## adaptive planning by a business-side Product Owner, with shipping every Sprint

23

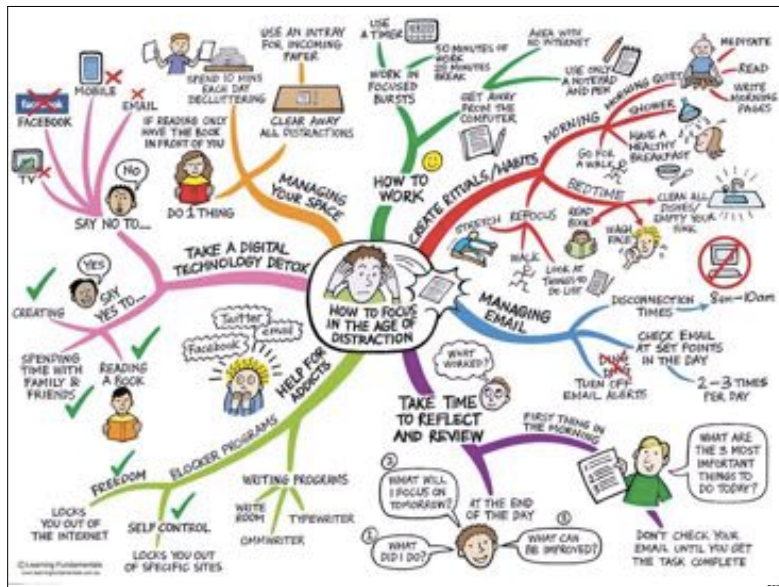
233



## Contract Game & Experts

234

234



235

# Local Optimization in Product Definition

236

## Where are We?

1. Opening Topics
- 2. System Optimization, not Local Optimization**
3. Organizational Structure
4. LeSS Overview

237

237

what are we  
about to learn?

238

238



## Guide: Getting Started

0. Educate Everyone
- 1. Define product**
2. Define 'done'
3. Have appropriately-structured teams
4. Only the Product Owner provides work for teams
5. Keep project managers away from teams

239

239

## Narrow vs Broad Product Definitions



240

240





group

- > suppose we want to do system modeling that includes analysis of **narrow vs wide size of waistline**

> what is the variable?

241

241



group

- > suppose we want to do system modeling that includes analysis of **narrow vs broad size of product definition**

> what is the variable?

242

242



team

- > sketch a systems model, bearing in mind this scenario puzzle:
  - > **“I wonder about the impact of narrow versus broad product definitions on the system optimizing goal(s)?”**
- > start with these variables verbatim:
- > what classification? (e.g. related to *people's cognition*?)
  1. **size/breadth of product definition**
  2. **# products**
  3. **product complexity (re. tech & requirements)**
  4. **# backlogs**
  5. **% of items worked on each Sprint that are highest value from a company view**
  6. **agility of teams to change direction at the company level**

243

243



coach

- > if system optimizing goals are
  - > **highest value & adaptiveness**, at company level
- > ... *broader or narrower product definition?*

244

244

support **secondary goals**  
or **constraints**  
(e.g. “product complexity”)

**without sub-optimizing  
the system goal**

245



### COACH & group

- > start with these variables verbatim:
  1. **# of inter-team task dependencies**  
(i.e. a team probably has to wait for another team to do “their part”)
  2. **strength of “private code” policies**
  3. **average complete end2end customer feature cycle time**
  4. **effort for inter-team coordination**

246

246



coach

- > relationship of?
  - > **low complete end2end customer feature cycle time**
  - > **adaptiveness from learning**

247

247



coach

- > if system optimizing goal is
  - > **low complete end2end customer feature cycle time**
- > ... *broader or narrower product definition?*

248

248



## COACH & group

- > include & discuss; start with these verbatim
  1. **time since reorganized to a broader product** (i.e. merging 2 or more smaller “products”)
  2. **effort spent learning due to broader reorganization**
  3. **effort spent on problems due to broader reorganization**

249

249



coach

- > how **quickly & broadly** should we broaden the product definition?

250

250

therefore...

251

251

## LeSS Rule(s)

The definition of product should be as broad and end-user/customer centric as *practical*. Over time, the definition of product might expand. Broader definitions are preferred.

252

252

focus on **why**

**own** vs rent

253

253

“IO Channels” & Product Definition

Google Maps?

254

254

A “Broad” Product & “**Implicit Backlogs**”

“Oh yes, we have only **one broad product**, and...”

Team-IOS = IOS items

Team-Android = Android items”

255

255

A “Broad” Product & “**Implicit Backlogs**”

“Oh yes, we have only **one broad product**, and...”

Team-IOS = IOS items

Team-Android = Android items”

256

256



teams: **standing**

- > form **new teams**
- > introduce each other?
- > please **sit** when finished

257

257



team

- > claim a wall area ;)

258

258



team

- > synchronize your models

259

259



team

- > start with these at least variables:

1. **cognitive “fullness” of one Product Owner to prioritize and have whole-product overview** (e.g. my head hurts! it's full!)
2. **cognitive “fullness” of people in teams to know ‘N’ items** (i.e. PB items)
3. **heterogeneity of each item**
4. **average size of item a team implements**
5. size/breadth of product definition

260

260



coach

> other variables of relevance to last exercise?

1. **# items to re-prioritize each Sprint**  
(at least enough for Sprint Planning)
2. **# of backlog items**
3. **# teams**

261

261

support **secondary goals**  
or **constraints**  
(e.g. "PO head should not explode")

**without sub-optimizing  
the system goal**

262

262



What happens to the  
Product Owner and  
Developers as the product  
gets broader and broader?

263

263

but no matter what we  
do to help the people's  
brains, at some point,  
we will reach a limit...

264

264

therefore...

265

LeSS Huge

266

**divide**

warning: dividing leads to  
**local optimization**

267

but...  
**divide** by what  
**dimension?**  
(architecture, ...?)

268

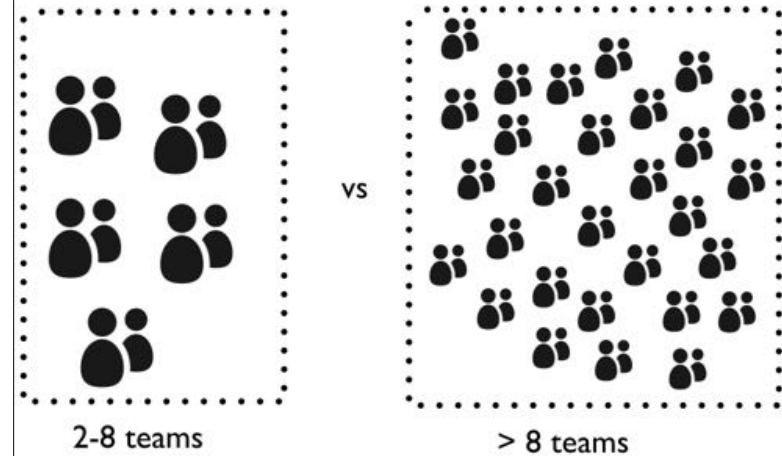
267

268

and this leads us to the  
motivation for  
**LeSS Huge**...

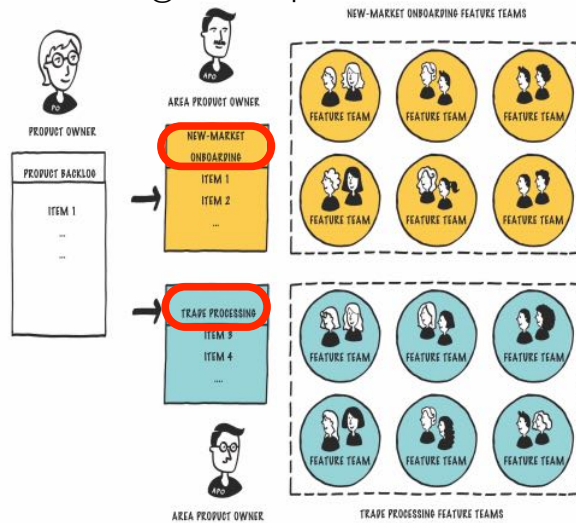
269

## 2 Frameworks: LeSS & LeSS Huge



270

## LeSS Huge: Requirement Areas



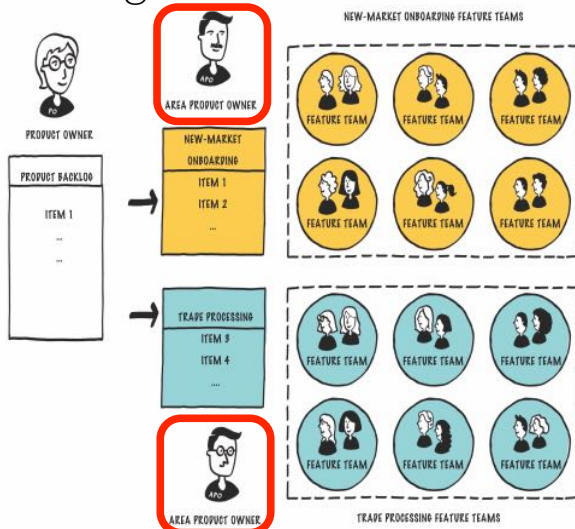
271

divide  
worshipping customers  
not  
worshipping code

272



## LeSS Huge: Area Product Owners



273

## LeSS Huge framework

- > LeSS Huge framework is NOT per se desirable; why? ...
- > **dividing** -> **local optimization**
- > an “uncomfortable art of the possible” so Product Owner & Developer heads don’t explode ;)

274

why not divide into  
separate **products**?

275

smaller products

vs

Requirement Areas

276



## COACH & group

- > add & link following variables:
  1. **breadth of RA** (Requirement Area)
  2. **# RAs**
  3. **# teams in RA**
  4. **# backlogs**

277

277



coach

- > if system optimizing goals are
  - > **highest value & adaptiveness**, at company level
- > ... *bigger or smaller Requirement Areas?*

278

278

therefore...

279

279

LeSS Rule(s)

Each Requirement Area  
has between  
“4-8” teams.

280

280

# Product Definition (again)

281

how to define  
a broader product? ...

282

Would this “Make Sense”?



283



**Guide:** Define Your Product

applying the  
**expanding** &  
**restraining**  
questions...

284

## 1. Expand Product as Broad as *Possible*

- > **customer focus:** What would the end customers answer if we ask them, "What is the product?" What spans the customer journey?
- > **family:** Family of similar products? Do we have components that are shared or functionality that is the same across our current products?
- > **system:** Our product is part of? What problem does the product solve for end customers?

285

285

## Example: Financial Trading



286

286

## 2. Restrain your Product as *Practical*

- > **commonality:** What is the product vision? Who are the customers? What is the product's customer domain?
- > **structural boundaries:** What development is within our company? How much structural change is practical?

287

287

a role of **managers** in a  
LeSS organization? ...

288

288

## Expanding Product Definition

- > sometimes “as broad as **ideal**” isn’t immediately possible
- > a driver for **continuous improvement** by **managers**:  
**“What prevents expanding the product definition?”**

289

289

**internal broad**  
product definition

VS

**narrower external**  
multi-products definitions

290

290

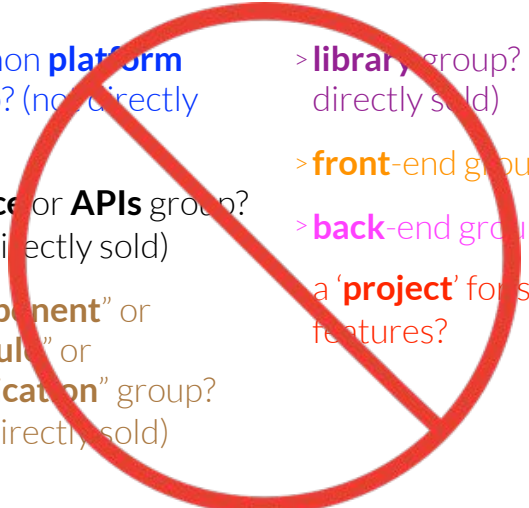
## “Consistent” Product Definitions?

- > common **platform** group? (not directly sold)
- > **library** group? (not directly sold)
- > **front-end** group?
- > **service** or **APIs** group? (not directly sold)
- > **back-end** group?
- > “**component**” or “**module**” or “**application**” group? (not directly sold)
- > a “**project**” for some features?

291

291

## “Consistent” Product Definitions?

- > common **platform** group? (not directly sold)
  - > **library** group? (not directly sold)
  - > **front-end** group?
  - > **service** or **APIs** group? (not directly sold)
  - > **back-end** group?
  - > “**component**” or “**module**” or “**application**” group? (not directly sold)
  - > a “**project**” for some features?
- 

292

292



coach & group: identify cases:

- > max “50” person product group?
  - > smaller LeSS framework
- > huge product group?
  - > LeSS Huge framework

293

293



group

- > split in half

expert on product

- > sketch a “**block architecture**” diagram of
  - > major **software & hardware components**
  - > the **broader context** that “stuff” is within
  - > **don’t** do much **explaining**; focus on sketching

294

294



expert on product

- > **if teams are organized around the components** (which is common), **write for each component**:
  - > the name of the site or sites
  - > # developers
  - > # component testers
  - > # of other people in noteworthy related roles
- > # of other people (by roles), not attached to a component
  - > e.g. BAs? system testers? system engineers?

295

295



coach, for each product

- > apply the expanding & restraining questions to create an **initial** product definition “as broad as **practical**”

296

296



coach, for each product

- > if a complex adoption case,  
**consider predictable work flows through the components**, to identify likely groupings into feature teams

297

297

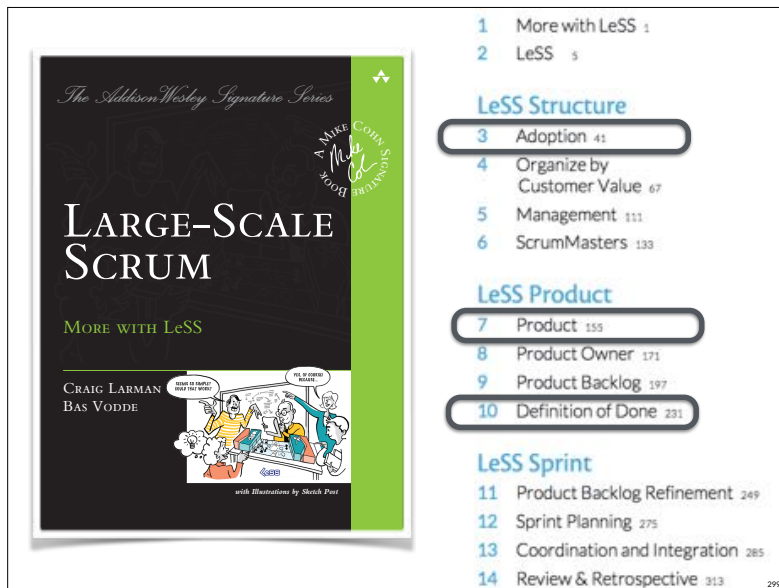


coach

- > as needed, discuss
  - > **incremental LeSS Huge adoption** vs “all-at-once” smaller LeSS framework
  - > **Feature Team Adoption Maps**
  - > **expanding Definition of Done**

298

298



299

299

perhaps the opening  
slides of the course  
make better sense  
now...

300

300

“How can we apply agile at scale in our big complex organization?”

301

301

is this the real problem? ...

302

302

traditional large groups are complicated — though not because they need to be, but because **their organizational designs, based on local optimization, create an illusion of ‘necessary’ complexity**

303

303

**This** is an Important Question...

“How can we **simplify** the unnecessarily big and complex organizational design, and **be agile** rather than **do agile?**”

304

304





## COACH & group

- > **why** have we focused on “**structural**” organizational design elements such as **product definition**? versus:
  - > “**mindset**”, “**team jelling**”, **clean code**, **good Scrum Masters**, **Sprint Planning in LeSS**, **large-scale re-prioritization techniques**, etc.?

305

305



## BIG Idea

in large-scale, the  
**first-order** factors on  
influencing system  
behavior are  
**structural**

306

306

portfolio  
management...

307

307



## COACH & group

- > sketch a systems model, considering this scenario puzzle:
  - > “**We are scaling agile. Therefore we need ‘agile’ portfolio management.**”
  - > start with these variables:
    1. breadth of products/**programs**/“**value-streams**”
    2. # of products/**programs**/“**value-streams**”
    3. **need for and activities of “portfolio management”**
    4. **# people involved in “portfolio management”**
    5. **ease of *first making* & executing large-direction decisions**
    6. **ease of *changing* & executing large-direction decisions**

308

308



coach

- > ... is there a relationship between **narrow** products/programs/ value-streams and the **apparent** need for **portfolio management**?

309

309



coach

- > if LeSS and if there is a **broad** product definition, **who** makes large direction decisions in the product?

310

310

**eliminating programs**

**->**

**elimination of  
program portfolio  
management**

311

311

## **“Artificial” “Portfolio Management”**

the **apparent** need for “program/value-stream portfolio management” is a...

**self-inflicted wound** consequence  
of the **unnecessary complexity...**

the very existence of **programs**

created by the **narrow** product/program/  
value-stream definitions

312

312

So-called “Agile/Lean” Portfolio Management?

- > **narrowly-defined** products/ programs/value-streams must be prioritized and funded
- > it's **big-batch requirements prioritization** driven by the existence of these narrow products, programs, or value streams

313

313

**necessary & real**  
portfolio management  
VS  
“artificial”  
portfolio management

314

314

“portfolio managers”  
=  
**artificial** manager role  
for **artificial** portfolio  
management

315

315

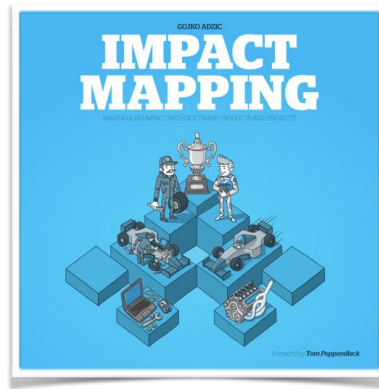
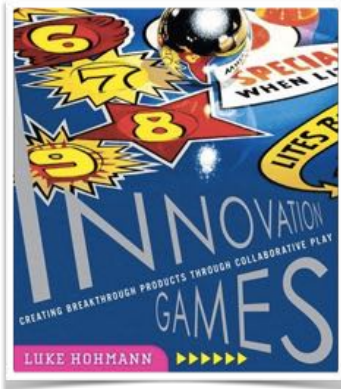
More Likely “Real” Portfolio Management?

the CEO is  
in the room

316

316

## Tips for “Real” Portfolio Management



317

LeSS **simplifies** or  
**eliminates**  
the need for  
“portfolio management” by  
broader product definitions

318

therefore...

LeSS Rule(s)

The definition of product should be as broad and end-user/customer centric as is practical. Over time, the definition of product might expand. Broader definitions are preferred.

319

320

## Local Optimization Cognitive Bias

Q: “Why do you have **narrow** products, programs, “component products”, or value streams?”

A: “Because it’s **best, and most efficient.**”

321

321

## Descaling with LeSS

remove



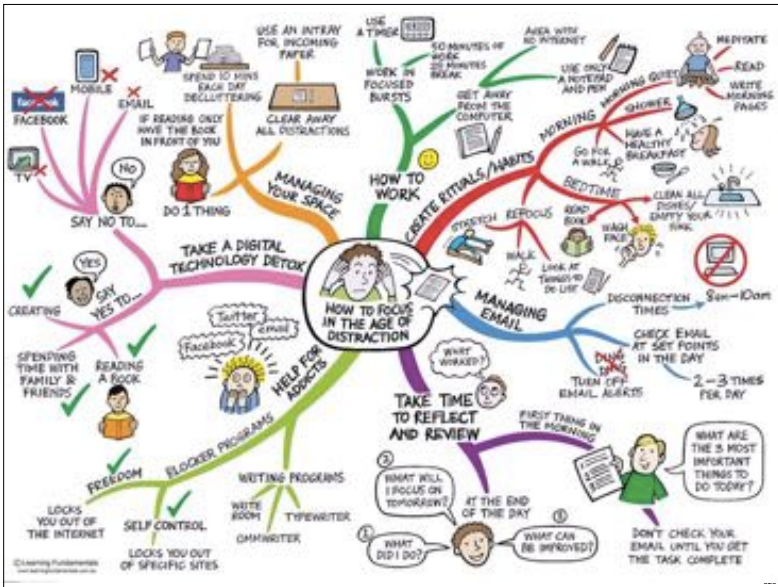
local optimization in  
**product definition**

## broader product definition

that comes from:  
narrow products,  
programs, value  
streams (& their org  
design elements)

322

322



323

# Local Optimization in Programming

324

## Where are We?

1. Opening Topics
- 2. System Optimization, not Local Optimization**
3. Organizational Structure
4. LeSS Overview

325

325



## Guide: Getting Started

### 0. Educate Everyone

1. Define product

2. Define 'done'

### 3. Have appropriately-structured teams

4. Only the Product Owner provides work for teams

5. Keep project managers away from teams

326

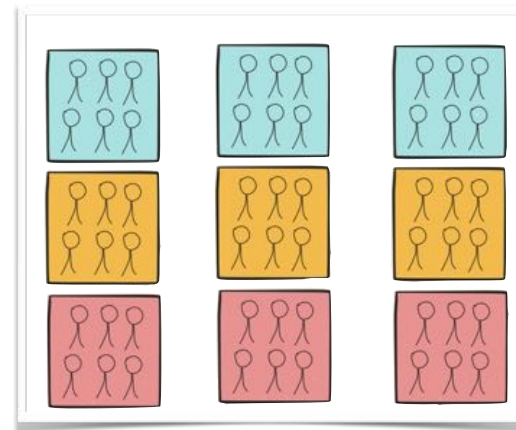
326

**component teams**  
have some advantages,  
but let's start with the  
issues...

327

327

## Component Teams



328

328

is this consistent with  
the system optimizing  
**goals?**

329

“the group goes to a  
**scaling agile** course,  
and learns to scale agile  
and really changes!”

330

therefore...

331

feature teams?

332

therefore...

333

333

LeSS Rule(s)

The majority of the  
teams are  
customer-focused  
**feature teams.**

334

334

“...majority...” ?  
  
contexts for  
component teams?

335

335

focus on **why**

336

336



analysis of component team dynamics could have been done via **system modeling...**

rather than “tell a story”

as a coach, you may want to practice doing it as a model

337

337

## Local Optimization Cognitive Bias

Q: “Why do you have component teams?”

A: “Because it’s **best, and most efficient.**”

338

338

want to see the explanation again?

The screenshot shows the LeSS website interface. The header includes the LeSS logo and navigation links like 'Large-Scale Scrum', 'Courses & Events', 'Coaching', 'Case Studies', and 'Resources'. A sidebar on the left lists 'LeSS Resources' with categories like 'Articles', 'Books & LeSS Chapters', 'Book Images', 'Videos' (highlighted), 'Discussion Groups', and 'Graphics'. The main content area is titled 'Videos' and lists various video titles and dates, such as '2015 Feb - Systems Optimization & Organizational Design: a LeSS Perspective - Craig Larman' and '2015 Feb - In-depth: Component Teams Issues (at 14:15), after short LeSS intro - Craig Larman - Oslo'.

339

339

traditional large groups are complicated — though not because they need to be, but because **their organizational designs, based on local optimization, create an illusion of ‘necessary’ complexity**

340

340

## Descaling with LeSS

remove



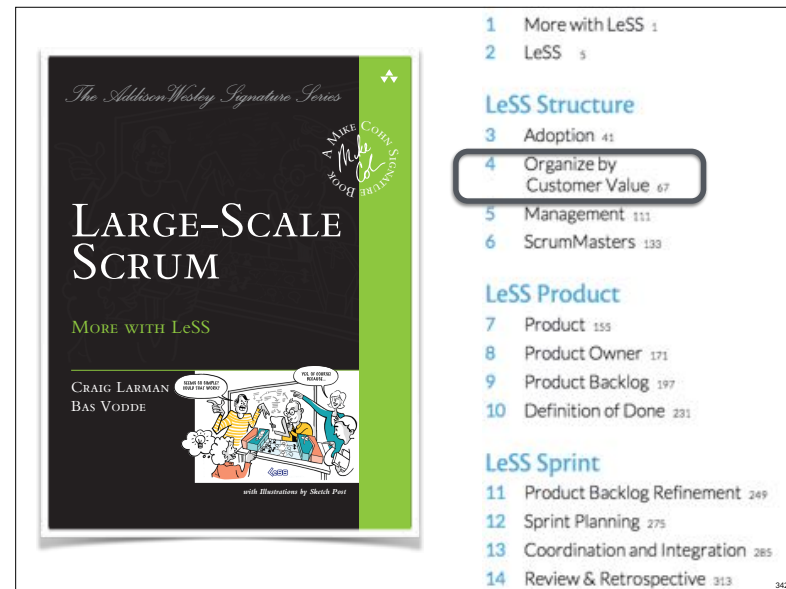
local optimization of  
**programming...**

that comes from:  
component  
teams (a single-  
specialist group)

**feature teams  
coding cross-  
components with  
shared code**

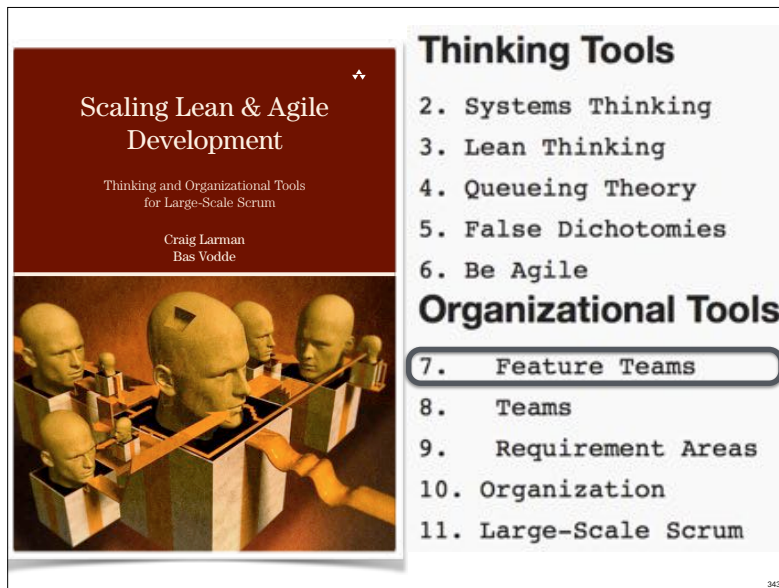
341

341



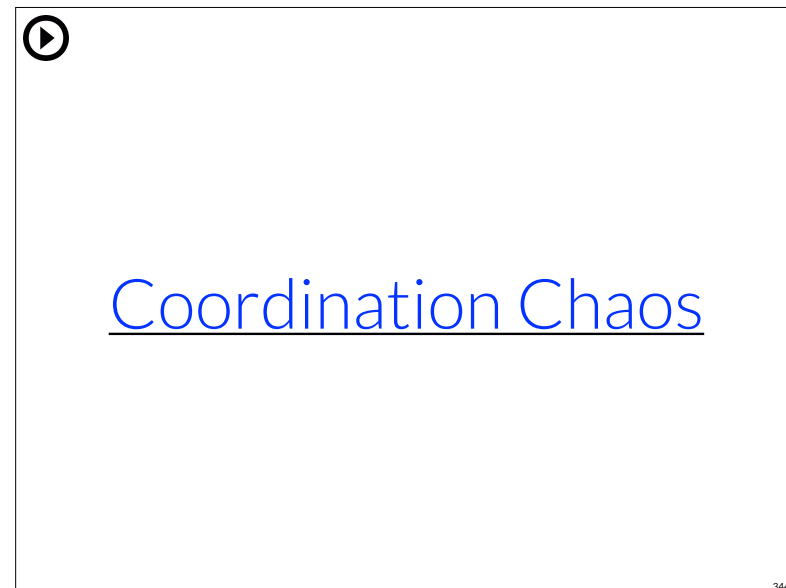
342

342



343

343



344

344

# PBR & Re- Prioritization in LeSS

345



preparation: at end of section,  
you will be **teaching** “all” of its  
ideas with others, **without**  
**referring to notes** 😊

346

346

what are we  
about to learn?

347

347

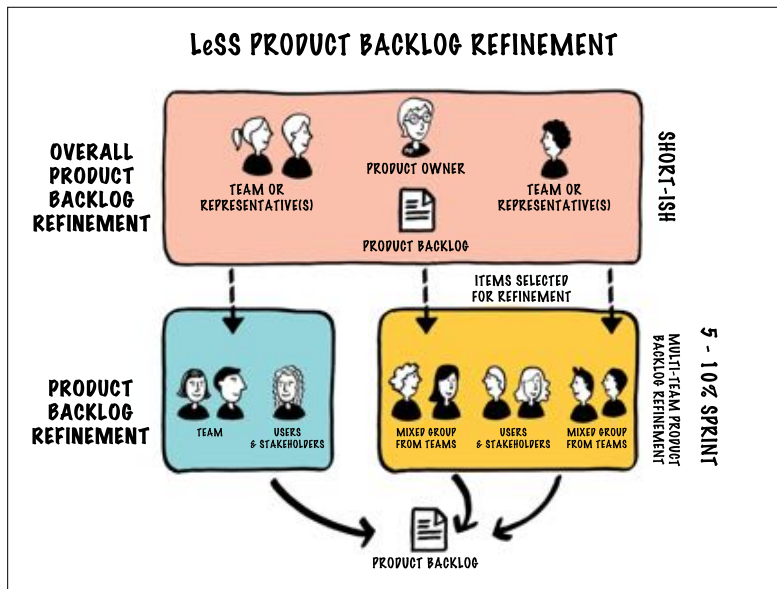


**BIG** Idea

in LeSS (and Scrum)  
cultivate a culture of  
“teams owning the  
product” and caring  
about customers

348

348



349

pairs : standing (without notes)

- > 1 person explain the activities in **Overall PBR**
- > other person explain the activities in **Multi-Team PBR**
- > please **sit** when finished

350

notice that the PO may not be in every PBR discussion...

351

coach & group

- > ways that the one (and only one) Product Owner can learn more about existing items, or get feedback from Teams, if she is not personally clarifying all items

352

topics to motivate a  
discussion of  
re-prioritization...

353

353

re-prioritization  
techniques?  
it's **Scrum**, so **any**  
technique is possible  
for example...

354

354

### Attribute Class Weighted-Sum Re-Prioritization

- > stakeholder preferences
- > technical risk
- > strategic alignment
- > effort & cost
- > driving profit
- > breadth of benefit (local, global)
- > business risk
- > ...

355

355

### Attribute Class Weighted-Sum Prioritization

Item	stakeholder weighted sum	customer-1	customer-2	production support	the 'system'
	<i>weight &gt;&gt;&gt;&gt;&gt;</i>	50	30	10	30
M	80	1	0	0	1
C	30	0	1	0	0

356

356

## Attribute Class Weighted-Sum Prioritization

Item	<i>strategic</i> weighted sum	new regulatory compliance	reduce cus- tomer cost	touch-based interface	transaction-fee based
	<i>weight &gt;&gt;&gt;&gt;&gt;</i>	100	40	30	60
M	100	0	1	0	1
C	130	1	0	1	0

357

357

## Attribute Class Weighted-Sum Prioritization

Item	<i>profit</i> weighted sum	motivates upgrade	hot!	annual OPEX reduced > \$1M	annual extra costs > \$100K
	<i>weight &gt;&gt;&gt;&gt;&gt;</i>	50	20	100	-50
M	100	1	0	1	1
C	20	0	1	0	0

358

358

## Attribute Class Weighted-Sum Prioritization

Item	<i>risk</i> weighted sum	new technology	difficult perfor- mance targets	lack of consen- sus on meaning of item	very uncertain market reac- tion
	<i>weight &gt;&gt;&gt;&gt;&gt;</i>	30	30	80	60
M	170	1	0	1	1
C	30	0	1	0	0

359

359

## Sum of Weighted Sums

Item	sum of weighted sums
M	330
C	135

360

360

does the **PO** need to do  
*detailed clarification*  
to **re-prioritize?** ...

361

361

## Specification by Example

specification with examples (real case)

Order Details											RefData						
currency pair	fx type	Verb	Amount	Dealt	Markup 1	Markup 2	Client Value	Value	Markup Type	Pip Fmt	Success	Total Markup	Wholesale Value	Profit			
EURUSD	Spot	Buy	1,000,000	EUR	2		1.30	0.00	Pts	4	OK	2	1.3002	200.00			
USDJPY	Spot	Sell	2,000,000	USD	10		110.00	0.00	Pts	2	OK	10	109.9	200,000.00			
GBPUSD	Fwd	Buy	1,000,000	GBP	1	3	1.50	0.00	Pts	4	OK	4	1.5004	400.00			
GBPUSD	Fwd	Sell	1,000,000	GBP	1	3	1.50	0.00	Pts	4	OK	4	1.4996	400.00			
GBPUSD	Fwd	Sell	1,000,000	GBP	1	0	1.50	0.00	Pts	4	OK	1	1.4999	100.00			

...  
+ 20 more examples

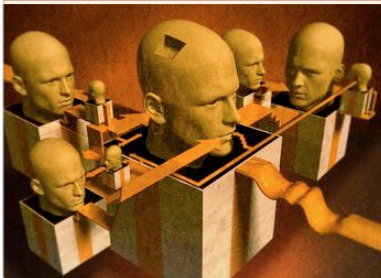
362

362

## Scaling Lean & Agile Development

Thinking and Organizational Tools  
for Large-Scale Scrum

Craig Larman  
Bas Vodde



### Thinking Tools

2. Systems Thinking
3. Lean Thinking
4. Queueing Theory

### 5. False Dichotomies

6. Be Agile

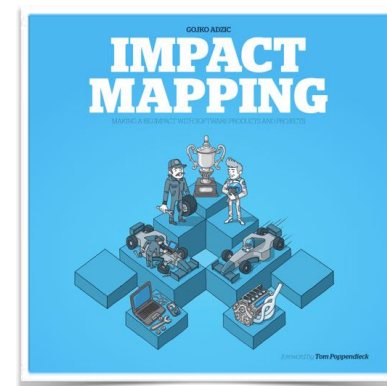
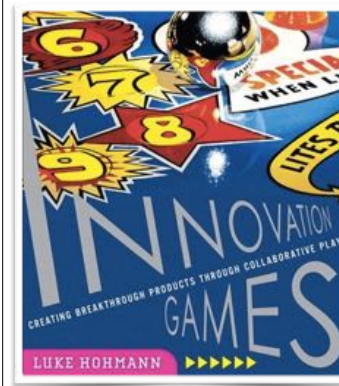
### Organizational Tools

7. Feature Teams
8. Teams
9. Requirement Areas
10. Organization
11. Large-Scale Scrum

363

363

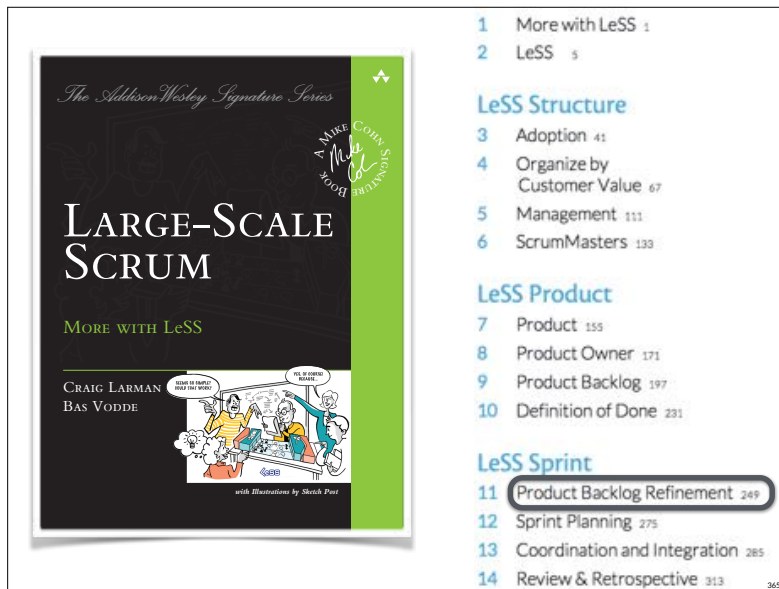
“Buy a Feature”, High Outcome-Low Effort, etc



364

364





365

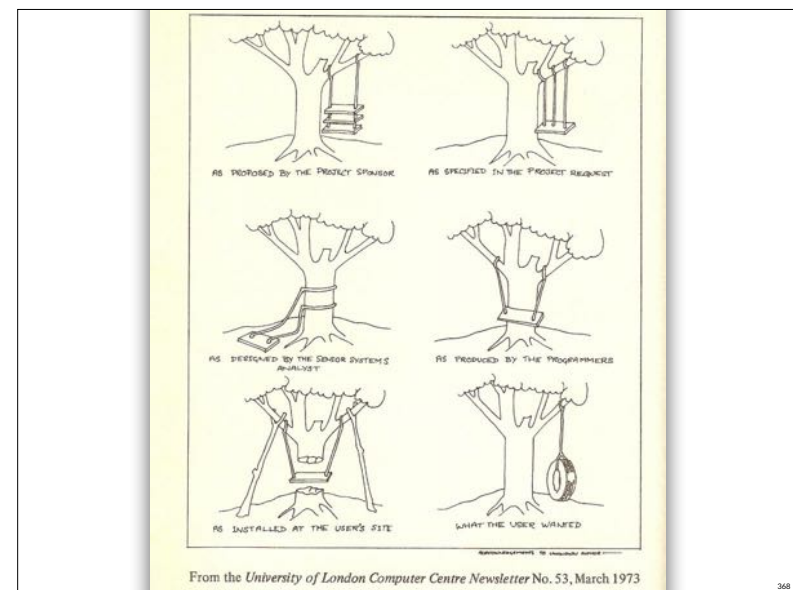


366

## Guide: Getting Started

0. Educate Everyone
1. Define product
2. Define 'done'
- 3. Have appropriately-structured teams**
4. Only the Product Owner provides work for teams
5. Keep project managers away from teams

367



368



## Lean Wastes in Product Development

1. **Over-production**—of intermediate, WIP, or finished things; sooner, faster, greater than demand
2. **Inventory**—intermediate, WIP, or finished things
3. **Over-processing**—& extra processes, rediscovery
4. **Handoff**—& transport
5. **Task switching**—& motion
6. **Waiting**—& delay
7. **Defects & finding/correcting**—tasks to find & correct: test, inspect, review, modify
8. **Not using people's full potential**—working to title, not multi-skilling
9. **Knowledge/information scatter/loss**—& connection to handoff & inventory & rediscovery; communication barriers: indirection, 1-way flows

369

369



## COACH & group

> write: what lean wastes are implied by the cartoon?

370

370



## COACH & group

- > connections between? ...
- > **local-optimization**
  - > **separate analysts or designers**
  - > **lean wastes?**

371

371

the story of stories



372

372

Practices for  
Scaling Lean & Agile  
Development

p. 223

A Stories Story

First sponsored in 1993 by Kent Beck and Grady Booch, the Hillside Group (with Ken Auer, Jim Coplien, Ward Cunningham, Hal Hildebrand, and Ralph Johnson) met to explore patterns and their generativity. Ward invented the wiki—in part—to support ongoing discussion. At a subsequent Hillside Group workshop, Bruce Anderson raised the topic of stories (as in tales) and their power to connect with people.

The implications for development work evolved in Ward's Episodes patterns (notice that 'episodes' relates to 'stories'), especially in the *Implied Requirement* pattern; Ward wrote, "I chose that name because the story only suggested the need to the degree that the developers and customers could talk about it." The implications also evolved in Kent's stories, articulated as part of his—inspired by Ward—agile development method, Extreme Programming (XP), whose first XP book cites Episodes. Kent wrote, "I imagined a user grabbing another user in the hallway and saying, 'I gotta tell you about this wonderful new thing the application does...'. Stories are the stories customers wish they could tell about the system but can't (yet)." (continued...)

Ward wrote, "I chose that name [stories] because the story only suggested the need to the degree that the **developers** and **customers** could **talk** about it."

373

telling stories  
card  
conversation  
confirmation  
between  
**developers & customer**

Developers

Bond  
Trader

As a Bond Trader I want...

not stories

374

“stories” is a

**BEHAVIOR**

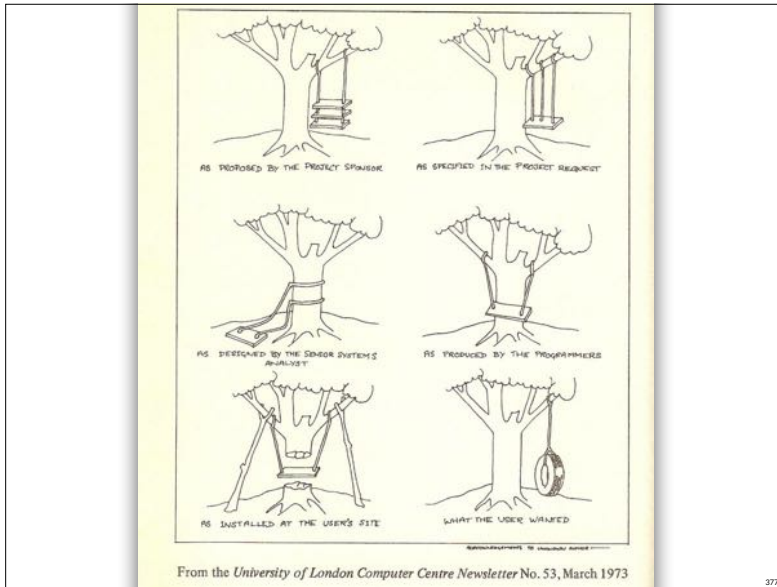
375

why did Ward define stories as

“talking between developer & customer”?

...

376



377

we're not lean & agile 😓

**intermediates** talk to users, create the **artifacts**, and hand them off to developers

378

then the group goes to "Scrum" and "agile" training, and learn...

379

now we're "lean & agile"!

~~intermediates~~ **Product Owners** ~~talk to users,~~  
create the ~~artifacts~~ **stories**, and hand them off to developers

380

*now we're "lean & agile"!*

~~intermediates~~ **Product Owners** ~~talk to users,~~  
~~create the artifacts~~  
**stories**, and hand them  
off to developers

381

381

**1990:** What was this Role Called?



Users &  
Customers



????

talks to users  
clarifies needs  
analyzes  
specifies

...



Dev Team

382

382

**1990:** What was this Role Called?



Users &  
Customers



**Business Analyst/  
Engineer/etc**

talks to users  
clarifies needs  
analyzes  
specifies

...



Dev Team

383

383

**Present Time:** What's Changed?



Users &  
Customers



**"Product Owner"  
(Business Analyst)**

talks to users  
clarifies needs  
analyzes  
specifies

...



Dev Team

384

384

widespread  
misunderstanding of  
the role of  
Product Owner?...


385



The screenshot shows the Scrum.org Blog header with navigation links for 'MAIN SITE', 'ABOUT US', and 'SUBSCRIBE'. The article title is 'Who is the Professional Scrum Product Owner'. The main text discusses the role of Product Owners, stating that many do not have true ownership and are often just proxies for the real Product Owner. A highlighted box contains the text: 'Many people thought that Product Owner is just another name for a Business Analyst.' Below this, a list of four points defines the Professional Scrum Product Owner:

1. Product Owner is an entrepreneur
2. Product Owner is an innovator
3. Product Owner is a systems thinker
4. Product Owner is a Mini CEO

386



The image shows the cover of the 'SCRUMGUIDE' by Ken Schwaber. Below the title, there is a tip:

**Tip:** For commercial development, the Product Owner may be the product manager. For in-house development efforts, the Product Owner could be the manager of the business function that is being automated.

387

“Must Cover in CSM” - Ken’s Direction to CSTs

“Self-managing teams are extremely productive. **When they work closely with the customer** to derive the best solution to a need, they and the customer are even more productive.”

388

so-called  
“Product Owner”  
=  
Business Analyst  
for the Team



389

do  
**real** Product Managers  
do specifications, UI  
design, & analysis? ...

390

### Classic Product Manager

- > “CEO of the product”
- > vision
- > road mapping
- > competitor analysis
- > market & customer analysis
- > pricing
- > channel development

391

so-called  
“Product Manager”  
=  
Analyst/Designer



392



## coach & group

- > where do so-called “Product Managers” (who do analysis, specifications, UX, etc) go in a LeSS adoption?
- > what role does the *real* Product Manager (vision, ...) play?
- > what may happen to the size of an existing “Product Management” group?

393

393



## team

- > sketch a systems model, considering this scenario puzzle:
  - > **1 product, 1 Product Backlog, many teams**
  - > **1 real Product Owner prioritizes the 1 Product Backlog (no team-level “product backlogs”)**
  - > **each Team has a so-called “Product Owner”, who is not doing hands-on development**
- > start with these variables
  1. % of total (product) items a team knows well (requirements & design)
  2. # so-called “Product Owners”
  3. likelihood so-called “Product Owners” are doing lots of analysis & talking to users
  4. likelihood so-called “Product Owners” create intermediate artifacts
  5. % wastes (e.g. inventory, over-production, handoff, info scatter, waiting ...)
  6. likelihood developers are doing lots of analysis & talking to users
  7. ability of developers to communicate effectively with customers/users
  8. degree that developers have empathy and awareness of customers
  9. degree that developers understand the business domain
  10. degree that so-called “Product Owners” are a bottleneck
  11. degree that developers can independently make informed fine-grained decisions

394

394

therefore...

395

395

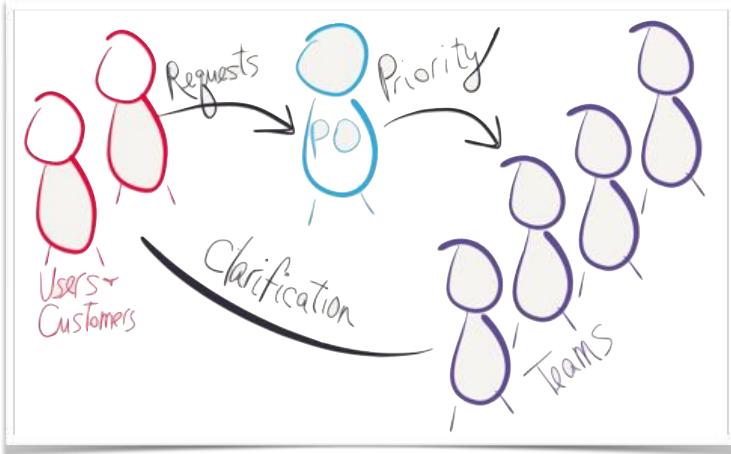
## LeSS Rule(s)

Prioritization goes through Product Owner, but clarification is as much as possible directly between the Teams and customer/users & other stakeholders.

396

396

## Clarification vs Prioritization



397

397



## **BIG** Idea

in LeSS (and Scrum)  
cultivate a culture of  
“teams owning the  
product” and caring  
about customers

398

398



the analyst manager

399

399

therefore...

400

400



## LeSS Rule(s)

1 (and only 1)  
Product Owner

401

focus on **why**

**own** vs rent

402



coach & group

- > in changing to the Scrum organizational structure, where are people in these roles probably meant to go?
  - > analysts & requirements engineers
  - > analyst- or team-“Product Owners”
  - > UX/UI designers
  - > analyst-“Product Managers”

403

scaling Scrum...

404

**naive** Scrum scaled  
**duplicates**  
**“PO”-per-team,**  
unaware of the  
system dynamics...

405

“PO”-per-team leads to...

1. **separate analysts/ designers**
2. **middleman**
3. **handoff**
4. **info scatter**
5. **almost all lean wastes**
6. **silo knowledge/ expertise**
7. **lack of empathy & engagement by developers**
8. **reduction of developers knowing or caring about customers & business**
9. **the “discoverers” vs the “developers”**
10. **(probably) more backlogs**
- 11....

406

naive Scrum ‘scaled’  
**multiple Scrum Teams**

Large-Scale Scrum  
**multiple-Teams Scrum**

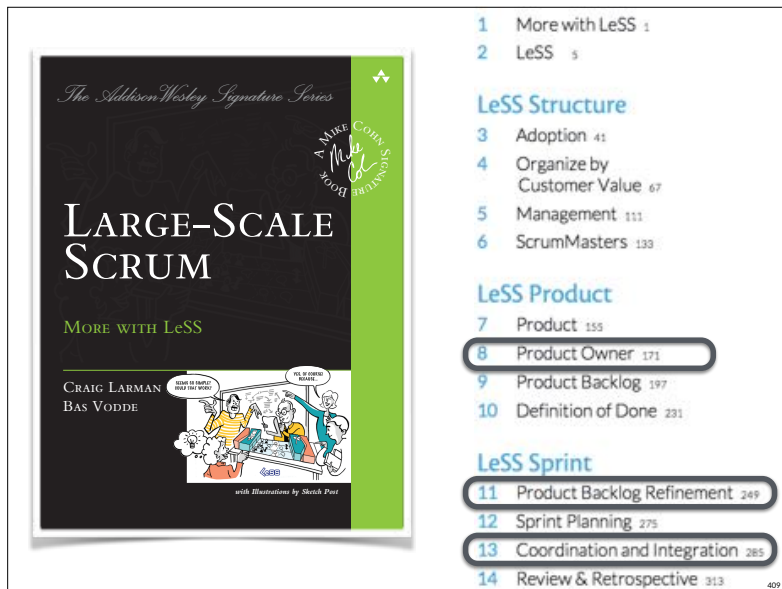
407

**Local Optimization** Cognitive Bias

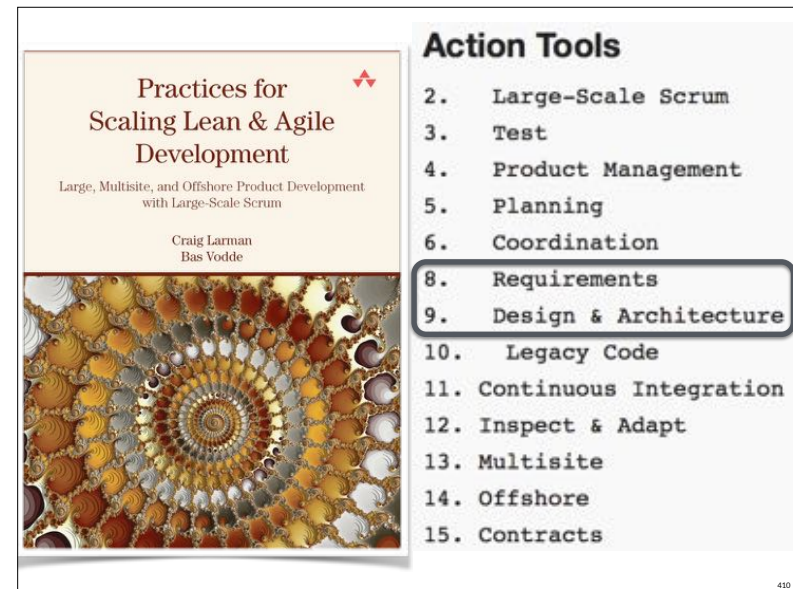
Q: “Why do you have team-level ‘Product Owners’? Why do you have a dedicated person doing analysis & design?”

A: “Because it’s **best, and most efficient.**”

408



409



410

## Descaling with LeSS

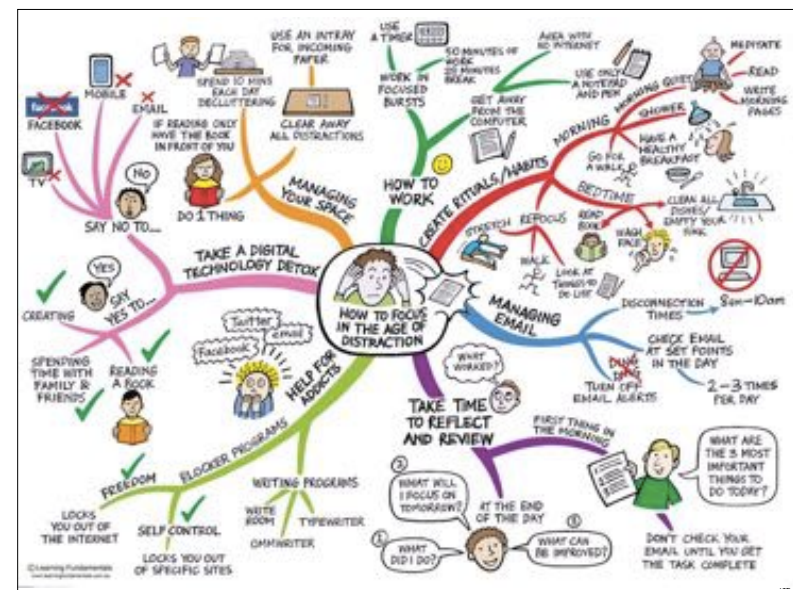
remove →

local optimization of **analysis & design...**

feature teams **clarifying & designing with users**

that comes from:  
separate analysts and designers

411



412

# From Local to Systems Optimization

413

to summarize...

414

## Descaling with LeSS

remove



local optimization of  
**backlogs...**

**1 Product Backlog**

that comes from:  
team backlogs  
(and all their org  
design elements)

(and no hidden  
“team or view  
backlogs”, and avoid  
“implicit backlogs”)

415

415

## Descaling with LeSS

remove



local optimization  
of **planning...**

**adaptive planning  
by a business-side  
Product Owner, with  
shipping every  
Sprint**

that comes from:  
the Contract Game  
(and all its org  
design elements)

416

416

## Descaling with LeSS

remove



local optimization in  
**product definition**

**broader product  
definition**

that comes from:  
narrow products,  
programs, value  
streams (& their org  
design elements)

417

417

## Descaling with LeSS

remove



local optimization of  
**programming...**

**feature teams  
coding cross-  
components with  
shared code**

that comes from:  
component  
teams (a single-  
specialist group)

418

418

## Descaling with LeSS

remove



local optimization of  
**analysis & design...**

**feature teams  
clarifying &  
designing with  
users**

that comes from:  
separate analysts  
and designers

419

419

**descaling &  
simplifying**  
with LeSS

420

420



## BIG Idea

deleting

not

adding

421

421

LeSS  
More with LeSS

422

422

Organizational  
Structure

423

Where are We?

1. Opening Topics
2. System Optimization, not Local Optimization
3. **Organizational Structure**
4. LeSS Overview

424

424

# Organize by Customer Value: Feature Teams

425

what are we  
about to learn?

426



## Guide: Getting Started

- 0. Educate Everyone
- 1. Define product
- 2. Define 'done'
- 3. Have appropriately-structured teams**
- 4. Only the Product Owner provides work for teams
- 5. Keep project managers away from teams

427

Decaling with LeSS

replace  
**local optimizations** of  
**single-specialist groups**  
with a *majority* of  
**feature teams**

428

defining  
feature teams...

429

Harvard  
Business  
Review

LEADING TEAMS

## The New New Product Development Game

by Hirotaka Takeuchi and Ikujiro Nonaka

FROM THE JANUARY 1986 ISSUE

### Moving the scrum downfield

From interviews with organization members from the CEO to young engineers, we learned that leading companies show six characteristics in managing their new product development processes:

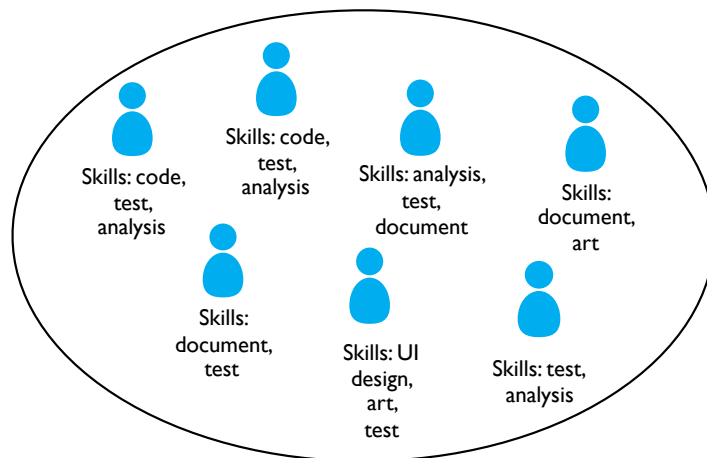
- 1 Built-in instability
- 2 Self-organizing project teams
- 3 Overlapping development phases
- 4 "Multilearning"
- 5 Subtle control
- 6 Organizational transfer of learning

430

429

430

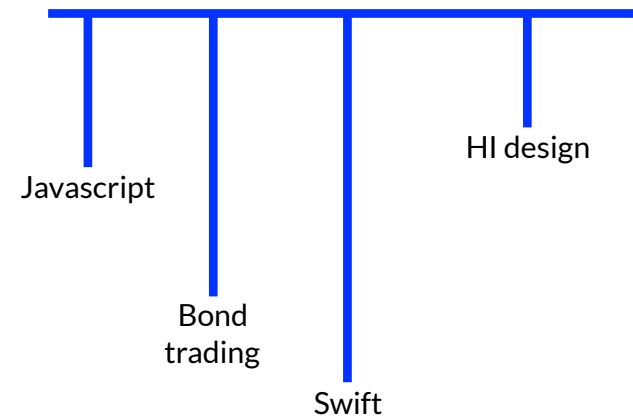
## Learning, Multi-Learning People



431

431

## Generalization & Specialization: Multi-learning Worker

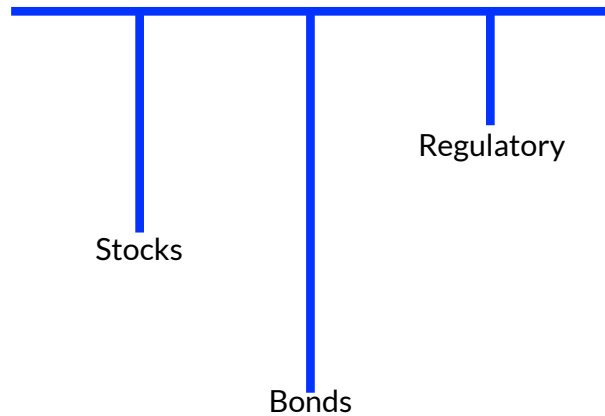


432

432



## Generalization & Specialization: **Multi-learning Team** too!



433

## Multi-Learning Worker Skills

primary

secondary

tertiary

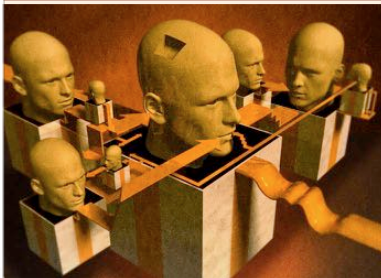


434

## Scaling Lean & Agile Development

Thinking and Organizational Tools  
for Large-Scale Scrum

Craig Larman  
Bas Vodde



### Thinking Tools

2. Systems Thinking
3. Lean Thinking
4. Queueing Theory
5. False Dichotomies

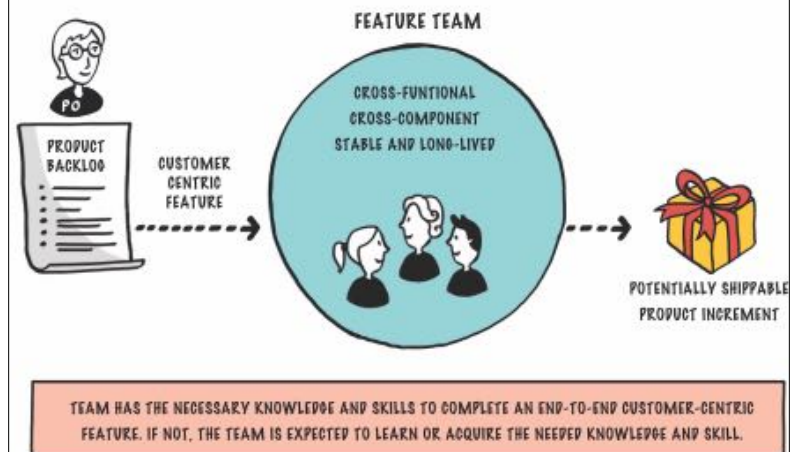
### 6. Be Agile

### Organizational Tools

7. Feature Teams
8. Teams
9. Requirement Areas
10. Organization
11. Large-Scale Scrum

435

## Feature Teams, **Learning**



436

a Feature Team is  
**full stack...**  
works across **all**  
code/components in a  
“shared code” model

437

therefore...

438

LeSS Rule(s)

The majority of the  
teams are customer-  
focused **feature teams**

439

LeSS Rule(s)

Structure the  
organization using  
**real teams**  
as the organizational  
**building block**

440

## LeSS Rule(s)

- Each real team is
- (1) **self-managing**
  - (2) **cross-functional**
  - (3) **co-located**
  - (4) **long-lived**

441

adopting  
feature teams...

442

analysts and/or  
UX/UI designers



DBAs



architects



component-1  
programmers



component-2  
programmers



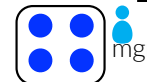
test/QA  
group



a likely traditional  
large-scale  
organizational  
structure before  
adopting Scrum

443

analysts and/or  
UX/UI designers



DBAs



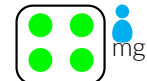
architects



component-1  
programmers



component-2  
programmers

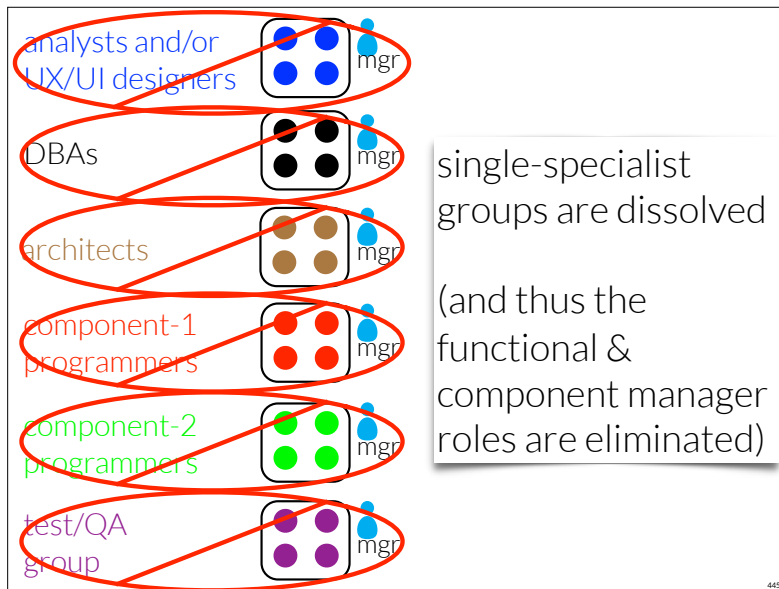


test/QA  
group



a cross-functional  
team in Scrum  
spans **all** functions

444



445

“It is difficult to get a man to understand something when his job depends on not understanding it.”

— Upton Sinclair

446

**Guide:** Job Safety, but not Role Safety

Job safety  
& **salary safety**  
but not  
role safety

447

“let me show you the org chart of my decision for the new feature teams”

448

## Self-Designing Teams Workshop

### How to Form Teams in Large-Scale Scrum? A Story of Self-Designing Teams

5 April 2013



Craig Larman



Ahmad Fahmy

How long does it take an organization to reorganize in order to adopt Scrum?  
Three hours. Really.

449

449

## Self-Designing Teams Workshop



450

450

## Connecting to Scrum Masters?



451

451

(optional)  
connecting to  
line managers

452

452



coach & group

> new roles for

- > ex-functional-team managers
- > ex-component-team managers
- > ex-project & program managers
- > ex-team-leads
- > ex-team-managers
- > ex-team-level "Product Owners"
- > ex-so-called "Product Managers" who are analysts, specifiers, UX/UI designers, etc
- > architects, system engineers?
- > UX/UI designers? BAs?

453

453

Only Title: (Product) Developer

*Scrum Guide:*

"Scrum recognizes no titles for Development Team members other than **Developer**, regardless of the work being performed by the person; **there are no exceptions to this rule.**"

454

454



**BIG** Idea

structural change:  
formally new job titles

e.g. **Product Developer**

455

455

Not a Team of Single-Specialists

*Scrum Guide:*

"Team does not contain sub-teams dedicated to particular domains such as testing or analysis"

456

456

## Managers/Leads Don't Direct Workers

### Scrum Guide:

“...the Team isn't allowed to act on what anyone else says [except the Product Owner] ... Teams are self-organizing...”

hence, no **team/tech leads**

457

457



## Guide: Getting Started

### 0. Educate Everyone

#### 1. Define product

#### 2. Define 'done'

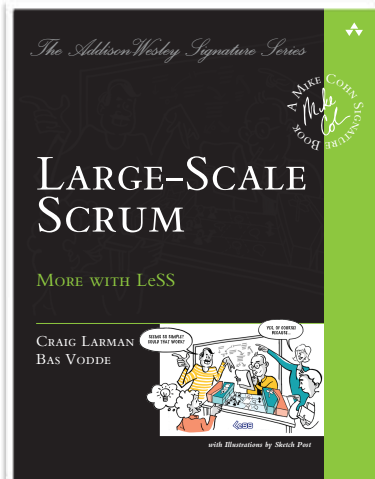
#### 3. Have appropriately-structured teams

#### 4. Only the Product Owner provides work for teams

#### 5. Keep “managers” away from teams

458

458



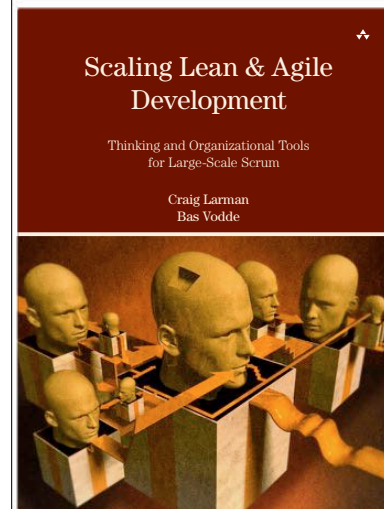
- 1 More with LeSS 1
- 2 LeSS 5
- 3 LeSS Structure
- 4 Adoption 41
- 5 Organize by Customer Value 67
- 6 Management 111
- 7 ScrumMasters 133

- 8 LeSS Product
- 9 Product 155
- 10 Product Owner 171
- 11 Product Backlog 197
- 12 Definition of Done 231

- 13 LeSS Sprint
- 14 Product Backlog Refinement 249
- 15 Sprint Planning 275
- 16 Coordination and Integration 285
- 17 Review & Retrospective 313

459

459



## Thinking Tools

2. Systems Thinking
3. Lean Thinking
4. Queueing Theory
5. False Dichotomies
6. Be Agile

## Organizational Tools

7. Feature Teams
8. Teams
9. Requirement Areas
10. Organization
11. Large-Scale Scrum

460

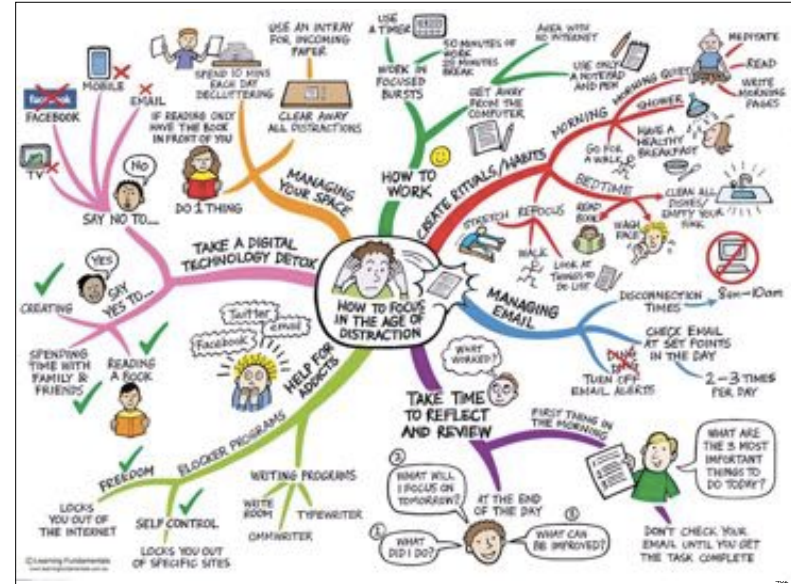
460



team work

461

461



462

why so much?...

**Lean-but**

**Scrum-but**

**Kanban-but**

**DevOps-but**

**AnyChangeIdea-but**

464

464

Larman's Laws of  
Organizational  
Behavior

463



## Larman's Laws of Organizational Behavior

1. Organizations are implicitly optimized to avoid changing the status quo middle- and first-level manager and "specialist" positions & power structures.
2. As a corollary to (1), any change initiative will be reduced to overloading or redefining the new terminology to mean basically the same as status quo.
3. As a corollary to (1), any change initiative will be derided as "purist", "theoretical", "religious", and "needing pragmatic customization for local concerns" — which deflects from addressing weaknesses and manager/specialist status quo.
4. As a corollary to (1), if after *changing the change* some managers and single-specialists are still displaced, they become "coaches/trainers" for the change, frequently reinforcing (2) and (3).
5. Culture follows structure (or behavior/mindset follows system)

465

465

larmanslaws.org  
;)

466

466



team: standing: round robin

- > most **noteworthy** or **interesting** idea so far?
- > write a summary of it on a separate sticky note
- > put all the notes together on a wall somewhere

467

467

LeSS  
Introduction ;)

468

## Where are We?

1. Opening Topics
2. System Optimization, not Local Optimization
3. Organizational Structure
4. **LeSS Overview**

469

469

now that we've  
discovered LeSS for  
ourselves via “why”...

summarize **what**...

470

470

# LeSS Overview

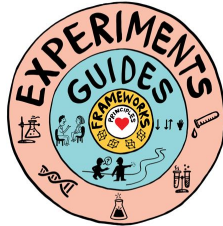


(optional) preparation: at end of  
section, you will be **sketching**  
**and teaching** “all” of its ideas  
with others, **without referring**  
**to notes** 😊

472

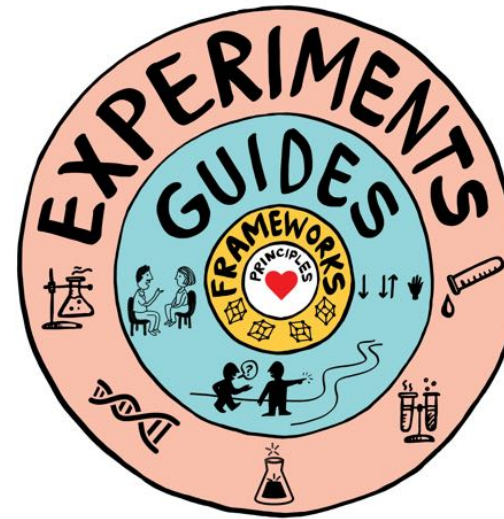
472

471



## LeSS Complete Picture

473

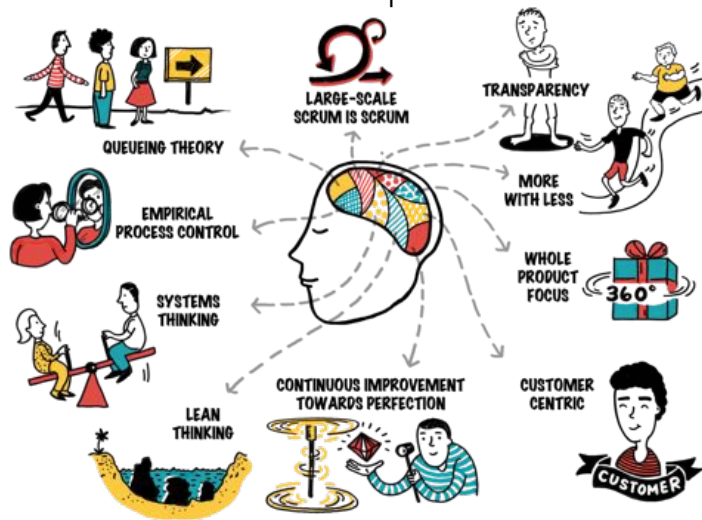


474

473

474

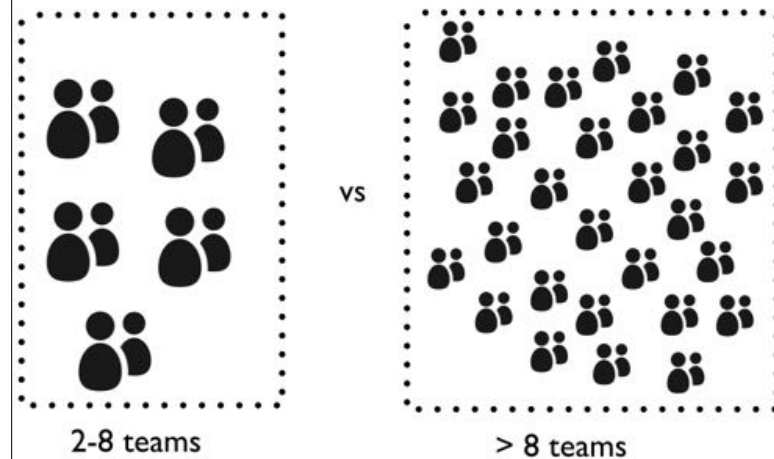
## Principles



475

475

## 2 Frameworks: LeSS & LeSS Huge



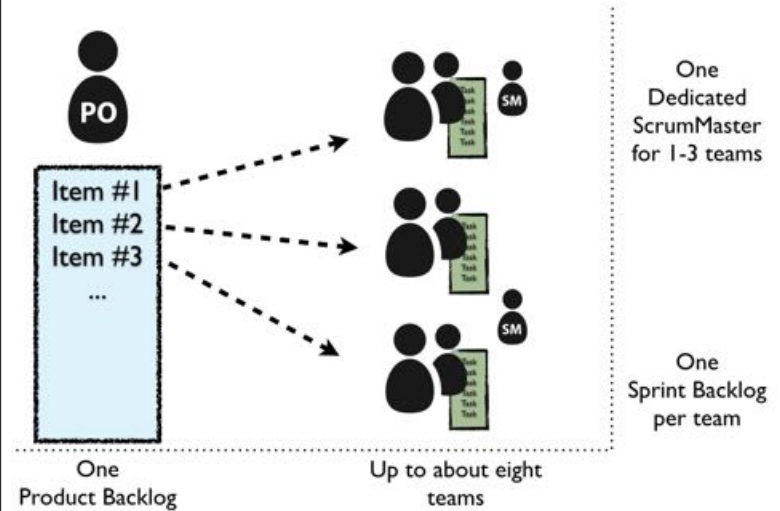
476

476

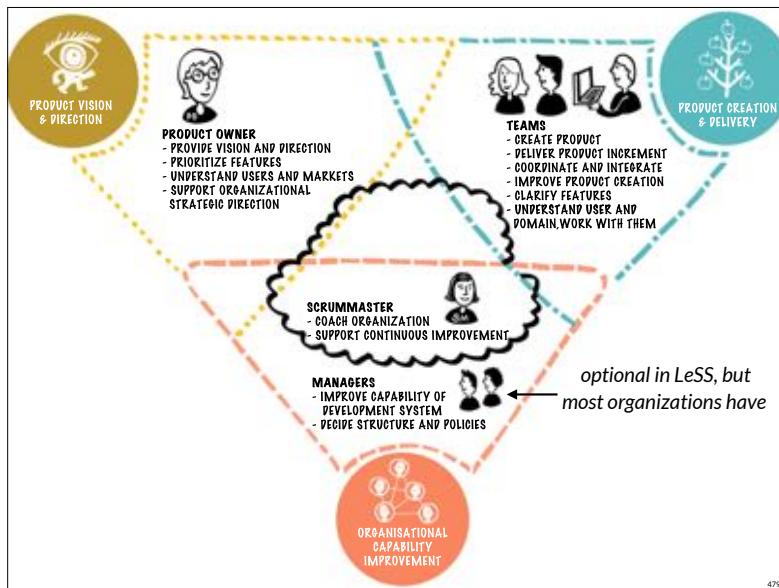
smaller LeSS framework...

477

## (smaller) LeSS Framework

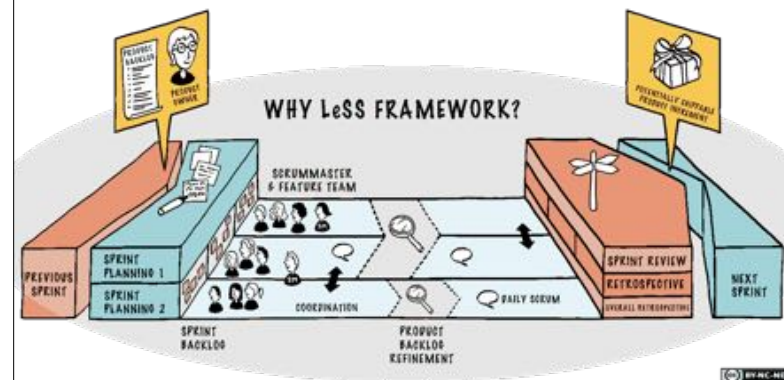


478



479

## 1 Common Sprint, 1 Shippable Product



480

480

## Guide: Sprint Planning One



481

## Guide: Sprint Review Bazaar



482



team: standing

> **why** is there

(1) one **common Sprint** making

(2) a **shippable product**, every Sprint

coach: review

483

## Adoption

> **narrow & deep**

> not broad & shallow

> (smaller LeSS FW): "50" people, 1 product, 1 or 2 sites, "many" months

> **top-down & bottom up**

> **volunteering; don't push**

484

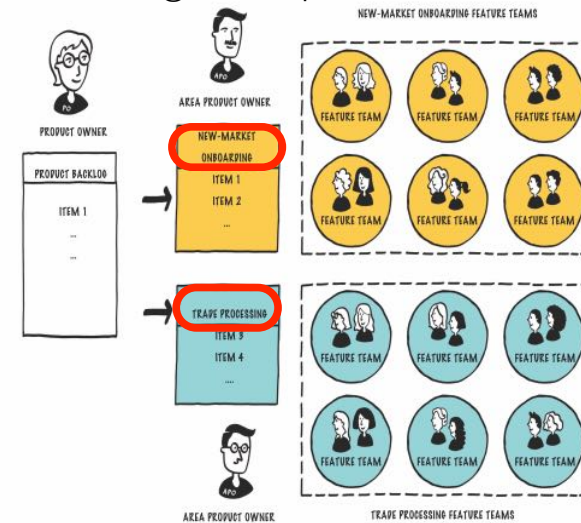


# LeSS Huge...

485

485

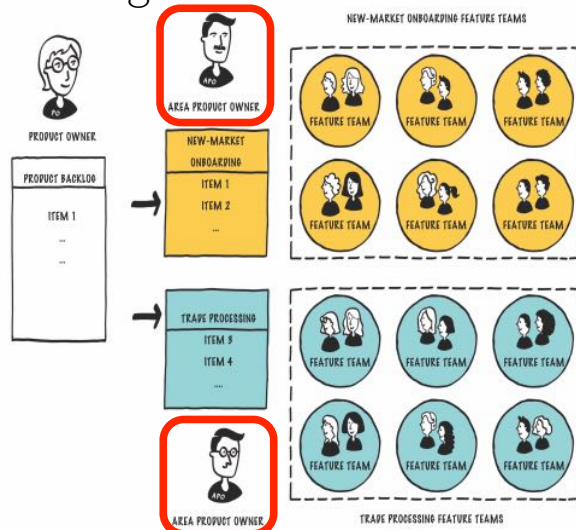
## LeSS Huge: Requirement Areas



486

486

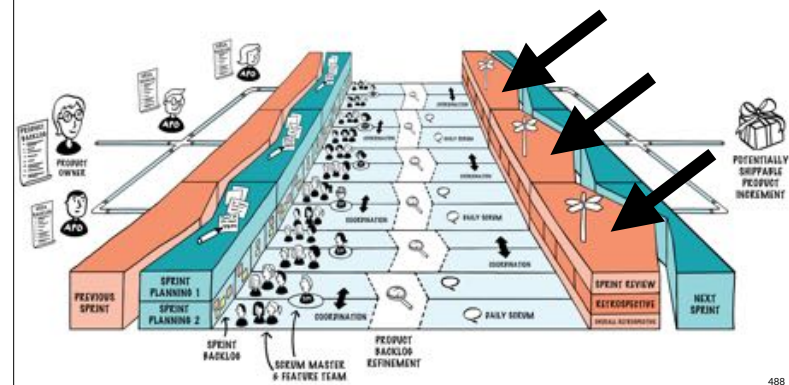
## LeSS Huge: Area Product Owners



487

487

## LeSS Huge Sprint: "Stack of LeSS"



488

488

## Adoption: LeSS Huge

- > not (or rarely) all-at-once
- > two alternatives:
  - > **focused deeper adoption at a part of the product group**
  - > gradual incremental adoption over the whole product group

489



(optional)  
individual  
> briefly review this module

490



(optional)  
pairs: standing: wall/flipchart  
> without referring to notes, one of the two people teach the ideas in this section to your partner, by... talking and sketching the ideas

491

why LeSS...

492

## Why LeSS? (our biases)

- > company-level systems optimization for
  - > **deliver highest customer value**
  - > **agility** (“turn on a dime, for a dime”)
- > transparency
- > whole-product focus
- > empirical process control

493

493

## Why LeSS? Occupational Psychology

my story with the RUP

**owning versus renting**

**“barely sufficient”**

-> more with **less**

494

494

shu - ha - ri

守破離

495

495

## Rules Prescription & Shu - Ha - Ri



“barely sufficient methodology”

496

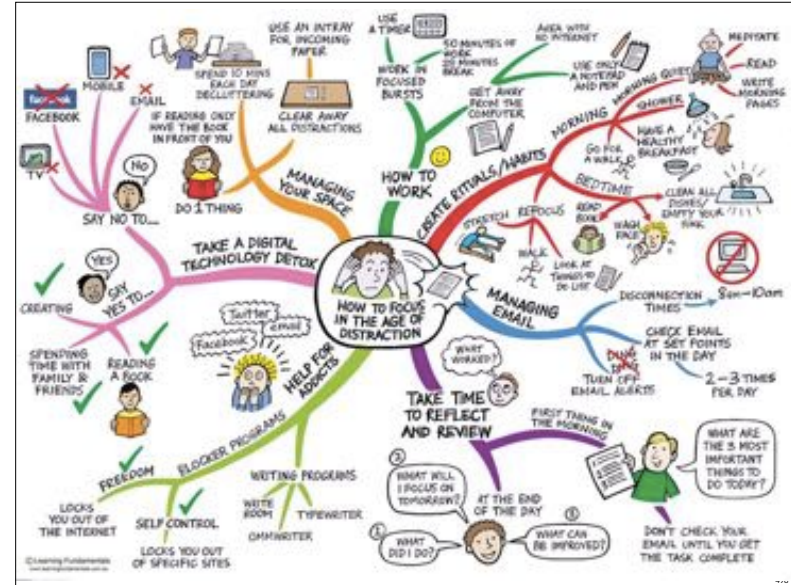
496





one person per team  
> sketch and explain  
“prescriptiveness & LeSS rules”

497



498

Adoption

499

LeSS  
Adoption

500



(optional) team

- > sketch a systems model, considering this scenario puzzle:
  - > **We want to do agile. To focus and accelerate the change, managers have objectives or bonuses or implicit rewards associated with an "agile adoption".**
- > start with these variables verbatim (some may already be in your model)
  1. **strength of carrots/sticks to "meet plan"** (Artifact/Thing) ("plan" is the "agile adoption goals")
  2. **degree of "fear"** (Person/Team/Group Behavior/Cognition)
  3. **gap between true situation and "plan"** (Artifact/Thing) ("plan" in this case is for "agile adoption")
  4. **pressure to deliver and "go faster"** (Action/Activity) ("deliver" in this case is to apparently "deliver the agile change")
  5. **transparency** (Person/Team/Group Behavior/Cognition)
  6. **degree the real change requires elimination of existing mgr & single-specialist positions/ groups, career paths, & financial & HR processes & policies** (Artifact/Thing)
  7. **degree of opaquely re-labeling, re-defining, "gaming", and "changing the change", similar to status quo** (Action/Activity)
  8. **degree of visibly "changing the change" for "pragmatic customization for local concerns", similar to status quo** (Action/Activity) (e.g. "let's create our own agile cookbook")
  9. **agility to adapt early based on understanding real situation** (Action/Activity)
  10. **degree of real improvement from apparently adopting the "change"** (Artifact/Thing)

501

501

"We want to do agile. To focus and accelerate the change, managers have objectives or bonuses or implicit rewards associated with an 'agile adoption'".

???

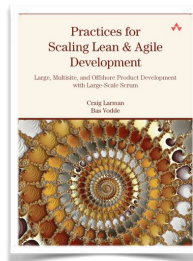
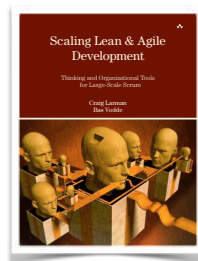
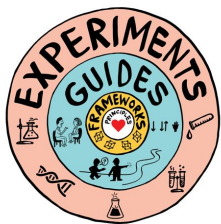
502

502

## LeSS Experiments

> LeSS has **experiments**

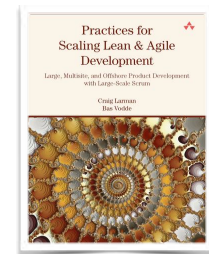
– LeSS books 1 & 2



503

503

"Avoid...  
Agile adoption targets  
or rewards"



### Action Tools

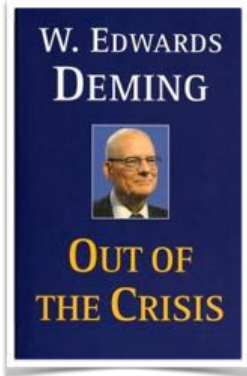
2. Large-Scale Scrum
3. Test
4. Product Management
5. Planning
6. Coordination
8. Requirements
9. Design & Architecture
10. Legacy Code
11. Continuous Integration
12. **Inspect & Adapt**
13. Multisite
14. Offshore
15. Contracts

504

504

## Total Quality Management (TQM)

*(point #11 for managers)*



“Eliminate management by objective.

Eliminate management by numbers, numerical goals.

Substitute leadership.”

505

505

focus on **why**

506

506



preparation: at end of section,  
you will be sharing “all” of its  
ideas with others, without  
referring to notes 😊

507

507

## Pre-Adoption: Building Interest

think & act like a **politician**,  
not like an engineer

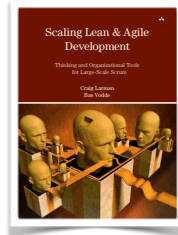


508

508

## Pre-Adoption: Building Interest

- > give “LeSS 1 or 3” **book** to key people
- > book **clubs**
- > send LeSS **video** links to people
- > find internal senior-manager **champion**
- > find & grow **allies**
- > just **talk directly with senior managers**
- > **external expert** talk—“you’re never a prophet in own land”
- > hold & promote **events** to build interest:
  - LeSS Practitioner, LeSS for Executives, Less LeSS



509

## Pre-Adoption: Building Interest?

- > 1 “proof of concept” feature team, working on major high-value end-to-end features, but still surrounded by the traditional organization for the existing product
- > will this **clearly & definitely** build interest? ...

510



team: standing

- > **complications** of introducing 1 feature team while surrounded by a large traditional organization?
  - > e.g. 5 component teams with private code, misc single-function teams (BA, HI, Test, ...)
- > **per definition** the 1 **feature team** is doing
  - > shared/open code across entire product
  - > all functional activities, e.g. analysis, HI design, integration, all testing

511

therefore...

512

all-at-once  
“flip the system”

513

513

How Big Can “All-at-Once” Be Successful?

“50” team members

... and let us know how you  
make it work bigger ;)

514

514

LeSS Rule(s)

For the product group,  
establish the complete  
LeSS structure “at the  
start”; this is vital for a  
LeSS adoption.

515

515

is “**kaizen**” always  
small & incremental in  
Lean Thinking (Toyota)?

516

516

**system kaizen**

("breakthrough kaizen", "kaikaku")

vs

point kaizen

517

517

focus on **why**

518

518

“informed consent”  
kickoff

519

519

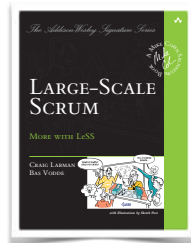
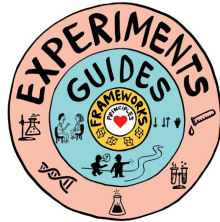
soft

520

520

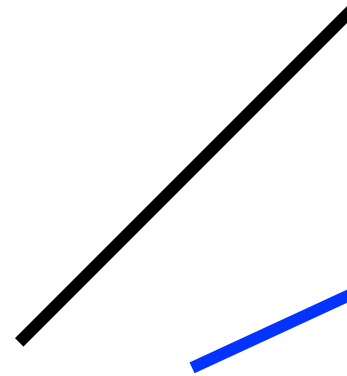
## LeSS Guides

- > LeSS has **guides**
  - LeSS book-3
  - this course



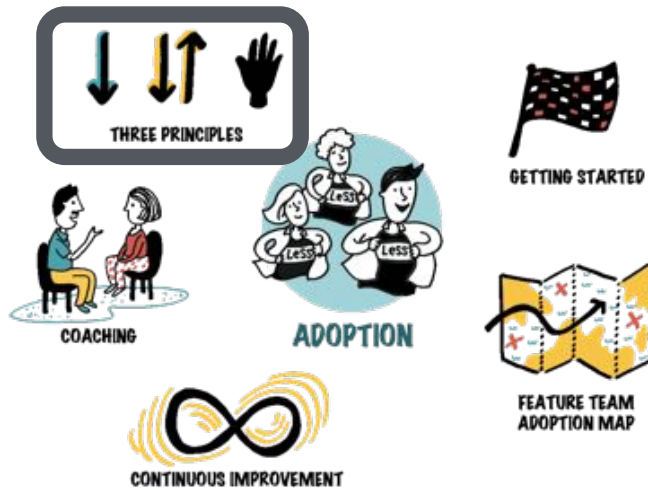
521

## Guide: Parallel Organizations



522

## Guide: Three Principles



523

## Adoption: **Deep & Narrow** over Broad & Shallow



524

524

## Scope of First Adoption

“50” team members

1 product

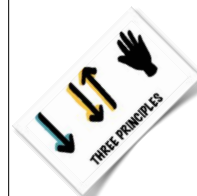
preferably 1 site

“several” months before another

525

525

## Adoption: **Top-Down** & **Bottom-Up**



526

526

## 3. Use Volunteering



versus

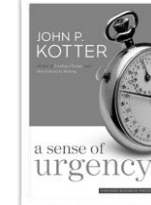
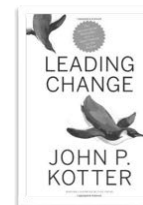


527

527



## Dr. J. Kotter on Resistance to Change

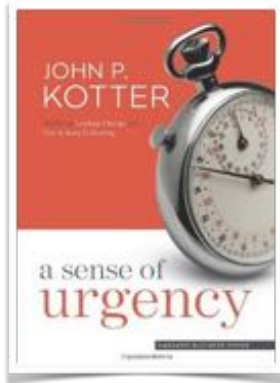


528

528



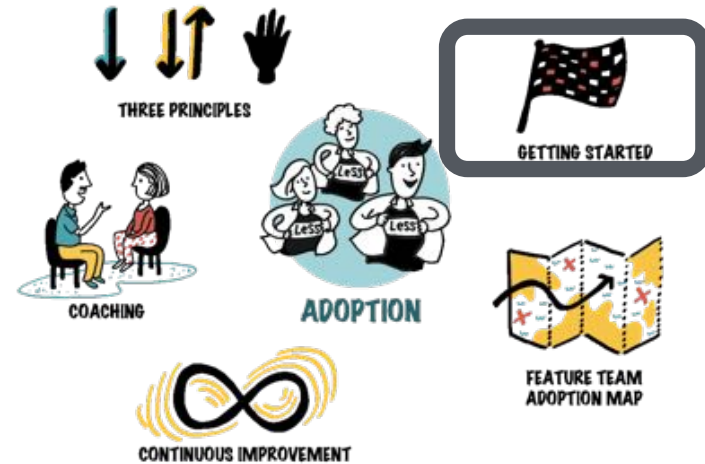
Dr. Kotter...



a sense of **urgency** or **existential crisis** needs to be felt by the senior management, to introduce meaningful change, else it unlikely to succeed

529

## Guide: Getting Started



530

530



## Guide: Getting Started

### 0. Educate Everyone

1. Define product
2. Define 'done'
3. Have appropriately-structured teams
4. Only the Product Owner provides work for teams
5. Keep managers away from teams

531

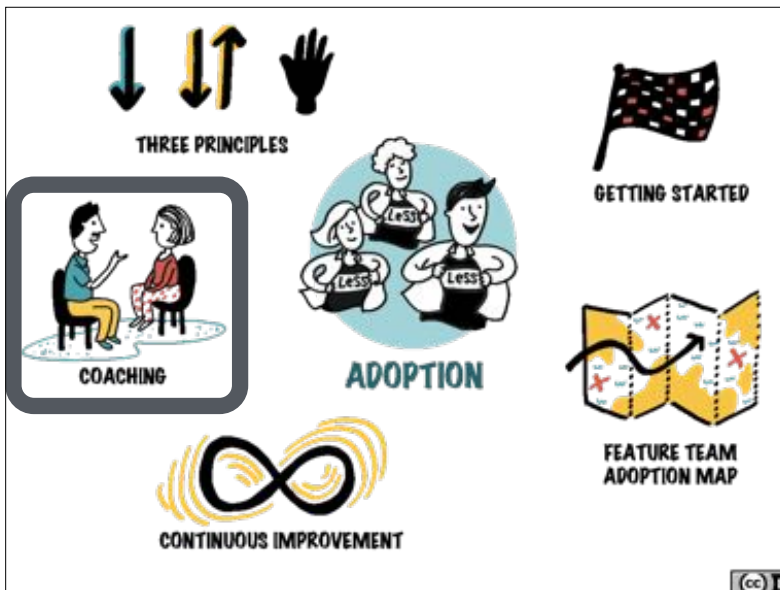
531

## 0. Educate Everyone

- > focus on **why**, not what
- > readings
- > educate **all together** (not role)
- > courses: Scrum, LeSS

532

532



533

prepare for shippable &  
**shipping**  
**awesomeness**  
 by **first** Sprint  
 why?...

534

**shipping**  
**speaks louder**  
**than words**

535

**Guide:** Ship at Least Every Sprint

**Ship**  
**at Least**  
**Every**  
**Sprint**

536

LeSS **Huge**  
adoptions are  
**incremental**, not  
“all-at-once”

537

537

individual  
> briefly review this module

538

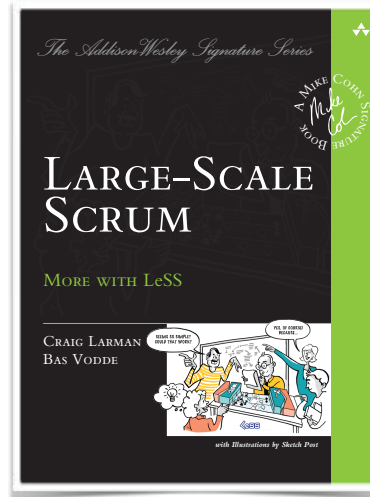
538

pair or team:

- > **teach back** exercise
- > please **sit** when done

539

539



- 1 More with LeSS 1
- 2 LeSS 5

### LeSS Structure

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 139

### LeSS Product

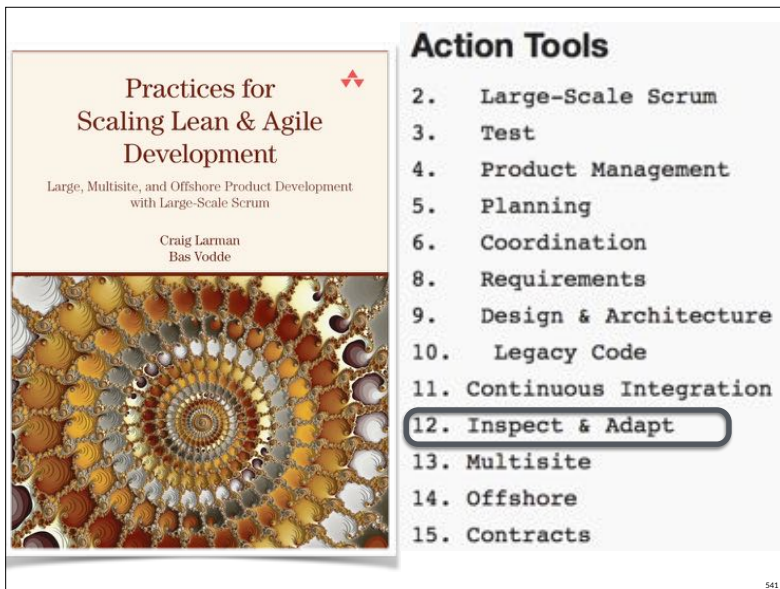
- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

### LeSS Sprint

- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

540

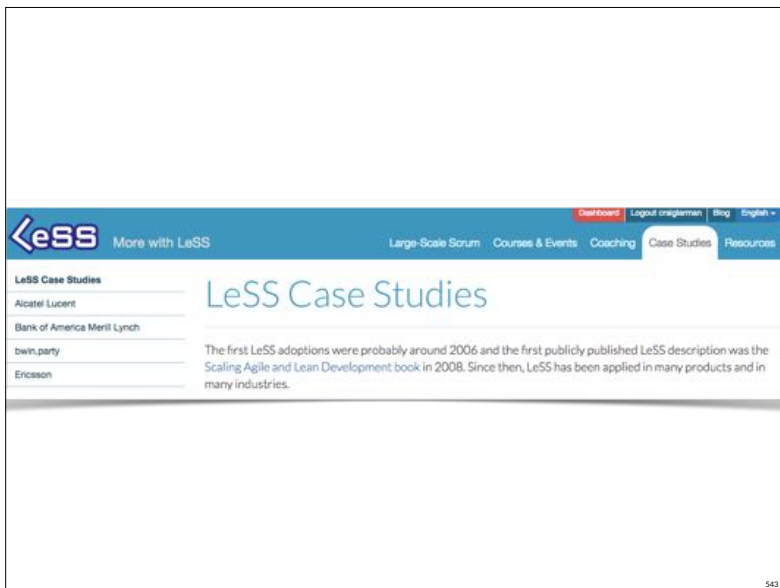
540



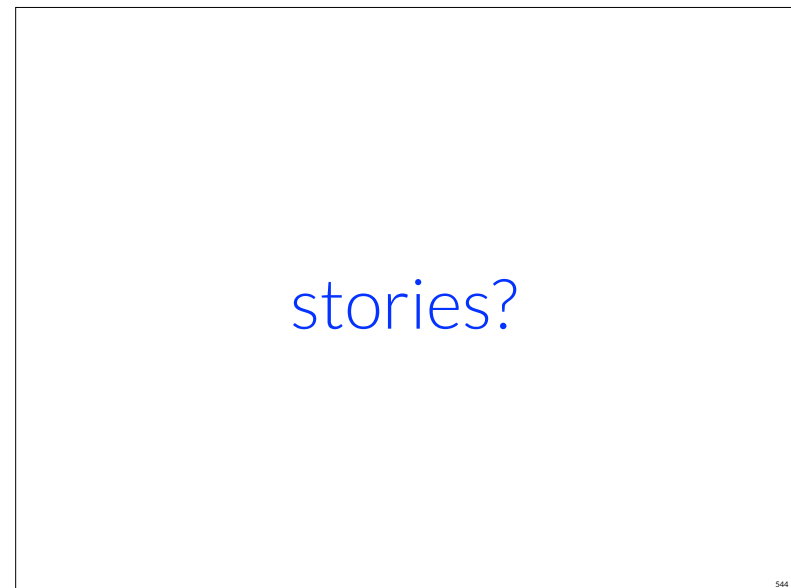
541



542



543



544

reminder...

1 “50 person” group

not entire company

545

545

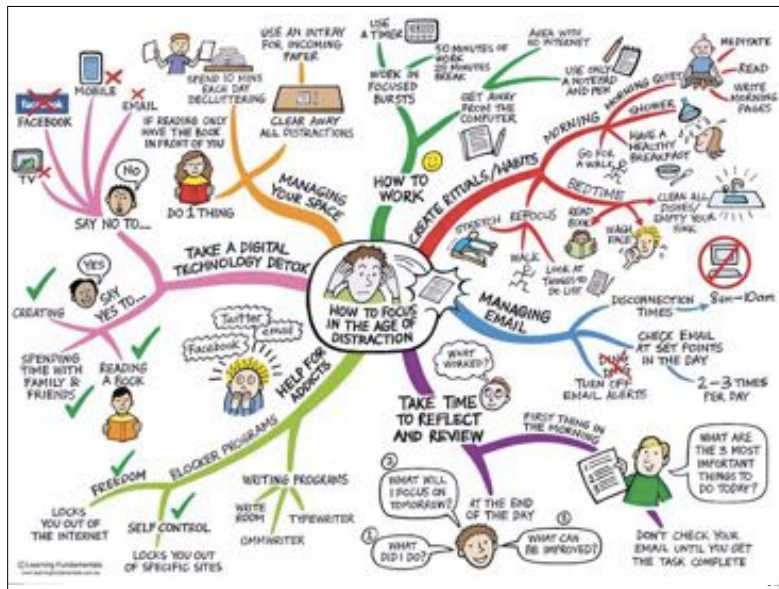


coach

> other adoption questions?

546

546

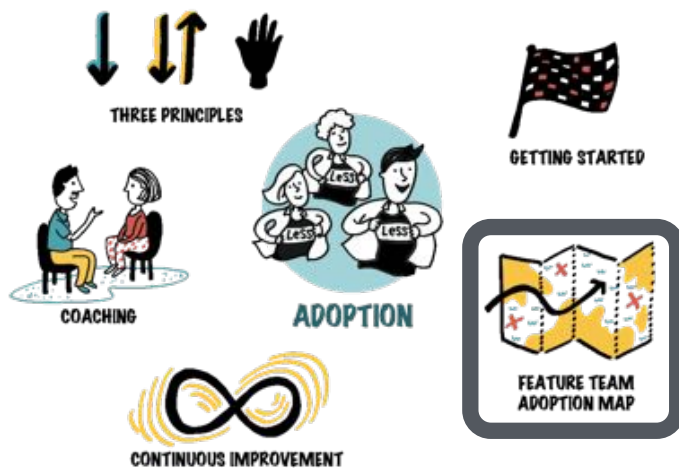


547

# Feature Team Adoption Map

548

## Guide: Feature Team Adoption Map



549

so far, we have  
assumed the initial  
creation of “complete”  
feature teams  
  
but sometimes...

550

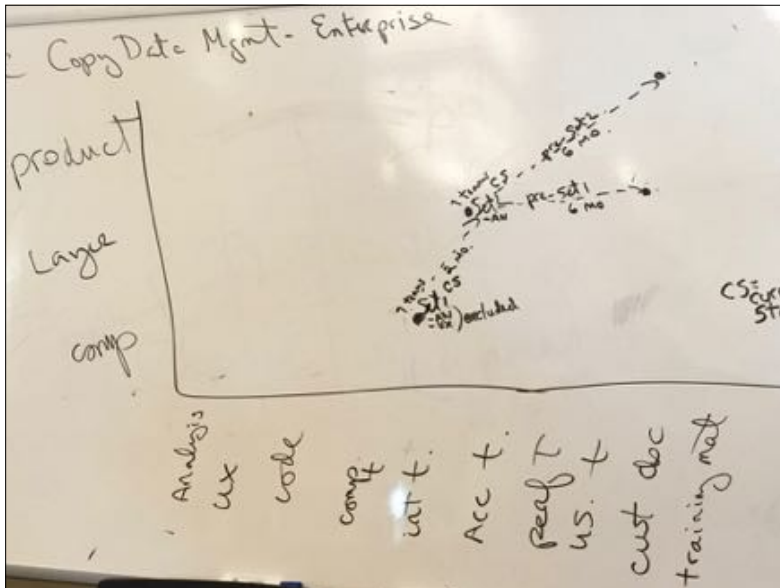
## Incremental Feature Team Adoption

- > extreme multi-site specializations
- > politics related to group structures
- > a full-stack feature involves '20' components and therefore '20' developers
- > extreme or very disparate technologies (“COBOL + JavaScript”)
- > initially-imperfect “done” due to constraints
- > LeSS Huge (many of the above issues)

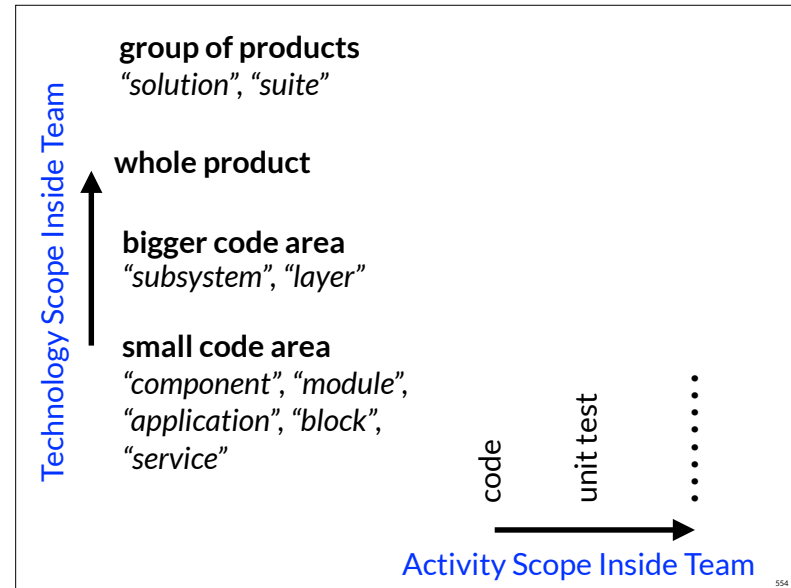
551

when feature-team  
adoption must be  
**incremental**,  
analyze with a  
**Feature-Team  
Adoption Map ...**

552



553



554

coach

- > demonstrate creating a feature-team adoption map, for an incremental adoption case
  - > mark where “potentially shippable”
  - > current state?
  - > next state?
  - > improvement experiments?

555

**LeSS Structure**

- 1 More with LeSS 1
- 2 LeSS 5
- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 139

**LeSS Product**

- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

**LeSS Sprint**

- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

556

# Why LeSS?

557



team

- > sketch a systems model, given this:
  - > **“large, detailed, framework for scaling, with many claimed best practices” is pushed onto a group by senior managers or consultants**
- > start with these variables
  - > what is the ‘driving’ variable in this scenario?
  - > degree of feeling of ownership and engagement by hands-on people in their processes & structures, and in improving them
  - > degree of acceptance of specious arguments (e.g. argument by “best practices”, “gurus”, “sacred texts”, “they do it”, ...)
  - > degree of fear if dissent
  - > degree of public dissent
  - > degree of private dissent
  - > degree of unnecessary or inappropriate processes & practices

558

558

## Agile & Scrum: Original Messages

**“barely sufficient”**

**“empirical process control”**

559

559

## Why LeSS? Occupational Psychology

my story with the RUP

**owning versus renting**

-> more with **less**

560

560





team

- > sketch a systems model, given this:
  - > **“large, detailed, framework for scaling, with many claimed best practices” is pushed onto a group by senior managers or consultants, and then the group is invited to tailor it down.**
  - > in addition to the prior variables, include at least
    - > what is the ‘driving’ variable in this scenario?
    - > degree of explicit or implicit goal to remain similar to status quo
    - > expectation to “get our money’s worth”
    - > expertise by the group to customize the framework
    - > similarity of the full framework to status quo
    - > degree of desire to shift blame “to the framework”, when problems

561

561

More with *less*

Build your  
framework up from a  
few simple core  
elements, based on  
from “why”



don't tailor it  
down



562

562

why not just advise

“think & experiment”?

(i.e., zero prescription)

563

563

shu - ha - ri

守破離

564

564

## Rules Prescription & **Shu** - Ha - Ri



“barely sufficient methodology”

565

## Why “Rules”? **Shu** & Focus on Creating...

- > global systems optimization for:
  - > highest value, agility
- > transparency
- > whole-product focus
- > empirical process control

566



one person per team

- > sketch and explain  
“prescriptiveness & LeSS rules”

567

LeSS  
More with LeSS ...

568

## Why LeSS? (part 1, our biases)

- > global systems optimization for
  - > deliver highest customer value
  - > agility (“turn on a dime, for a dime”)
- > transparency
- > whole-product focus
- > empirical process control

569

569

## Why LeSS? (part 2, our biases)

We don't want <b>more roles</b> , as that	leads to <b>less responsibility</b> to the Teams.
We don't want <b>more artifacts</b> , as that	leads to a <b>greater distance</b> between Teams and customers.
We don't want <b>more dedicated analysts</b> , as that	leads to a <b>greater distance</b> between Teams and customers, <b>more handoff</b> problems, and <b>less engagement &amp; empathy</b> .
We don't want <b>more supplied process &amp; “best practices” &amp; “renting”</b> , as that	leads to <b>less learning &amp; team ownership</b> of process & <b>engaged improving</b> .

570

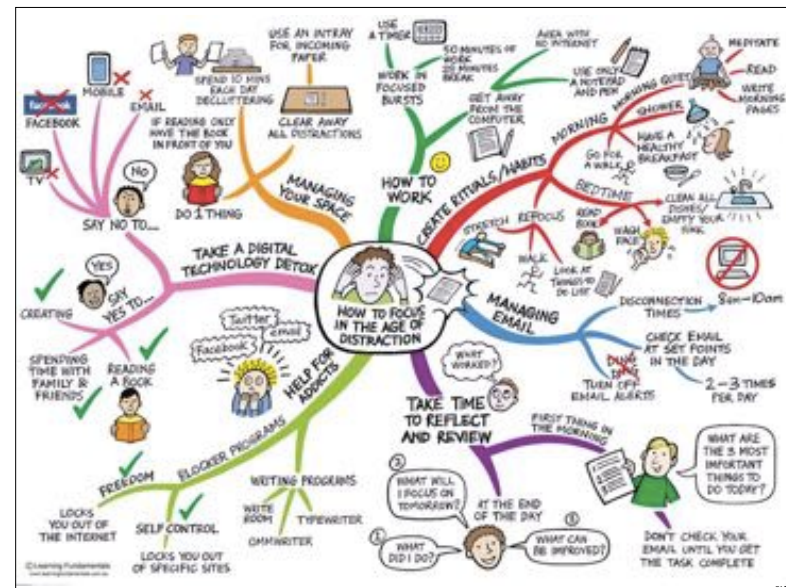
570

## Why LeSS? (part 3, our biases)

We want <b>more responsible</b> Teams	by having <b>less roles</b> .
We want <b>more customer-focused</b> Teams building useful products	by having <b>less artifacts</b> .
We want more <b>customer-focused empathetic</b> Teams that deeply understand requirements	by <b>less dedicated analysts</b> .
We want <b>more Team ownership of process &amp; meaningful work</b>	by having <b>less supplied processes &amp; “best practices”</b> .

571

571

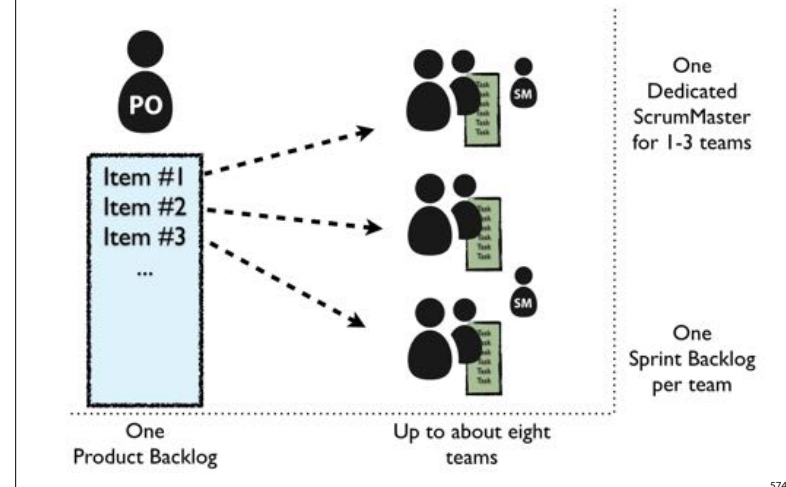


572

# LeSS Sprint

573

## (smaller) LeSS Framework



574

# Preparation Meetings

575

## Preparation Meetings

- > **Educate everyone**
- > **Define Product**
- > **DoD meeting**
- > Feature-Team Adoption Analysis (e.g., with a Map)
- > **Self-Designing Teams meeting**
- > **Community kickoff meetings**
- > **Initial Product Backlog refinement**
- > Current-Architecture Learning Workshop
- > Agile Modeling Design/Architecture Workshop

576

## Self-Designing Teams Workshop

### How to Form Teams in Large-Scale Scrum? A Story of Self-Designing Teams

5 April 2013



Craig Larman



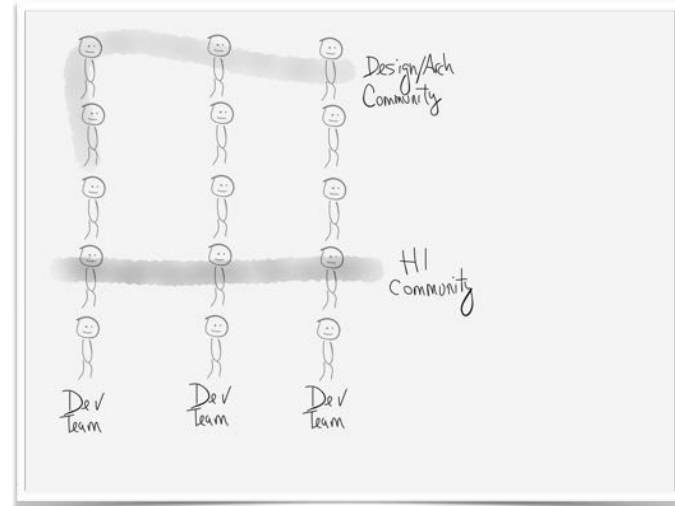
Ahmad Fahmy

How long does it take an organization to reorganize in order to adopt Scrum? Three hours. Really.

577

577

## Community Kickoff meetings



578

578

## Guide: Initial Product Backlog Refinement

create shared understanding

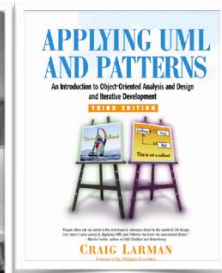
useful activities?



579

579

## Guide: Current-Architecture Workshop



Chapter 39:  
Documenting  
Architecture

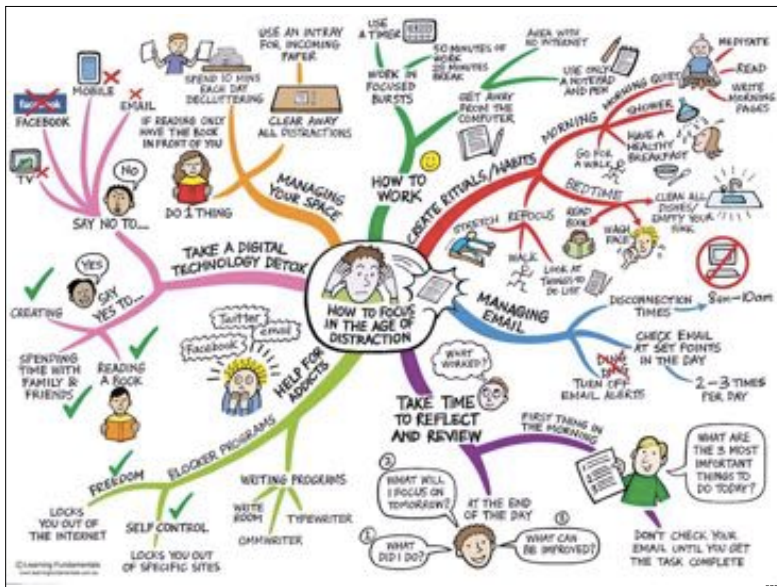
580

580

## Guide: Multi-Team Design Workshop

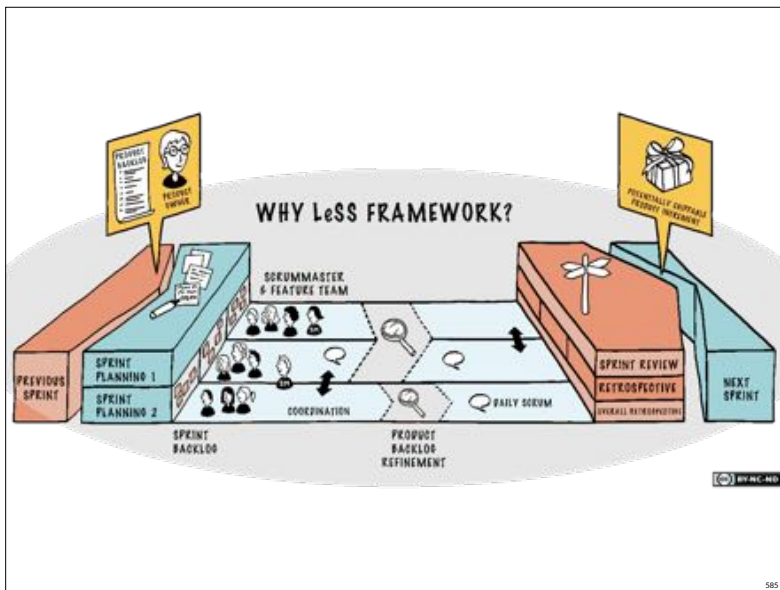


coach: questions?

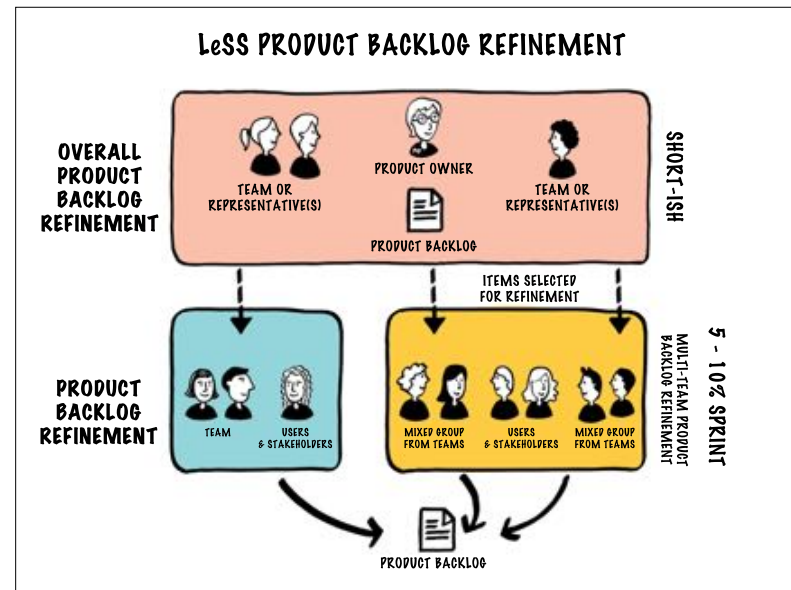


# LeSS Product Backlog Refinement





585



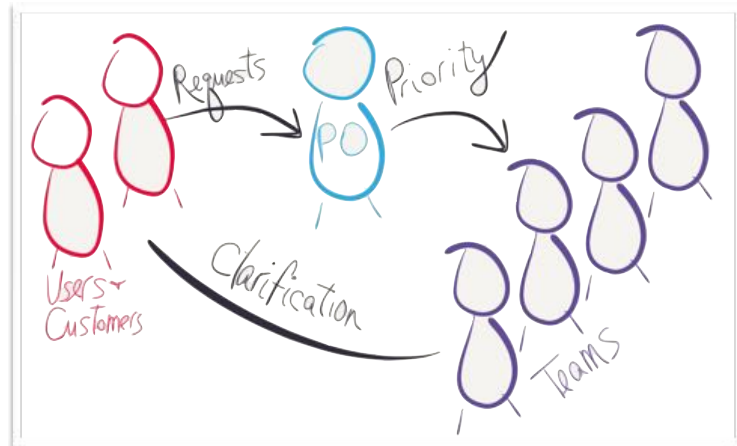
586

## Guide: Multi-Team PBR



587

## Clarification vs Prioritization



588

Teams emphasize  
learning  
**customer domains**  
(not just tech domains)

589

Scrum.org | Blog  
Insights from Scrum.org's community of experts

MAIN SITE ABOUT US SUBSCRIBE

### Who is the Professional Scrum Product Owner

Many Product Owners currently do not really have any ownership with the product they are managing. The label has changed but the behaviour hasn't. They aren't empowered to make decision for the product and they can't say NO to the stakeholders. Basically they're just a proxy for the real Product Owner. Or they're just focusing on requirements engineering, writing perfect requirements. They are still expected to write perfect User Stories. The way requirements are written has changed, but the perspective towards requirements hasn't.

Many people thought that Product Owner is just another name for a Business Analyst.

- 1. Product Owner is an entrepreneur**
- 2. Product Owner is an innovator**
- 3. Product Owner is a systems thinker**
- 4. Product Owner is a Mini CEO**

590

~~PO or PO Team creating  
any specifications,  
documents, designs,  
mockups, wireframes ...  
and handing them off to  
teams~~

591

## Types of PBR

	Overall PBR	Multi-team PBR	Single-team PBR	Initial PBR
members from	all teams	2+ teams	1 team	all teams
Includes Product Owner?	definitely	depends	rarely	definitely
Includes customers/users?	rarely	probably	probably	definitely
select which teams work on which items?	yes (prefer set of items with group of teams)	no	done already	no
level of clarification	lightweight	in-depth	in-depth	in-depth
length	shortish	0.5-1 day	0.5-1 day	at least 2 days
typical frequency	every Sprint	most Sprints	most Sprints	once

592



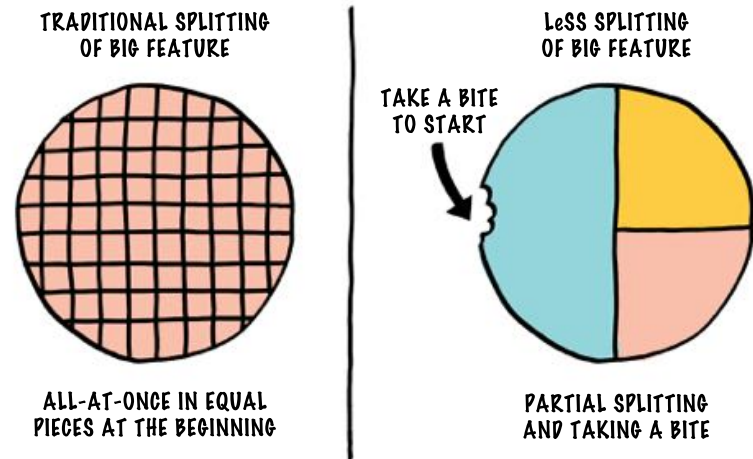
## Estimation synchronization



593

593

## Guide: Take a Bite



594

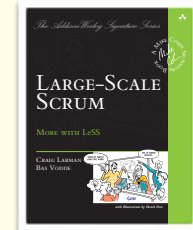
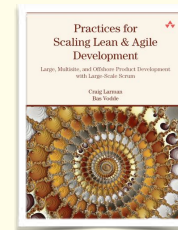
594

## Guide: Splitting

Use cases	the major work flows or use cases	I/O channel	different interfaces, e.g., GUI or command line
Scenario	a specific sequence of steps	Data format	XML, ...
Data part	subset of the data elements	Role or persona	e.g., novice or power user
Type	Varying types of kinds of things	Non-functional	e.g., moderate vs high throughput
Integration	integration between existing (or non-existing) elements	Operation	system operation, e.g., HTTP GET
Configuration	varying configurations, e.g., OS or browser	Stub	working with a fake first

595

595



individual:

> scan the section on splitting big items

596

596

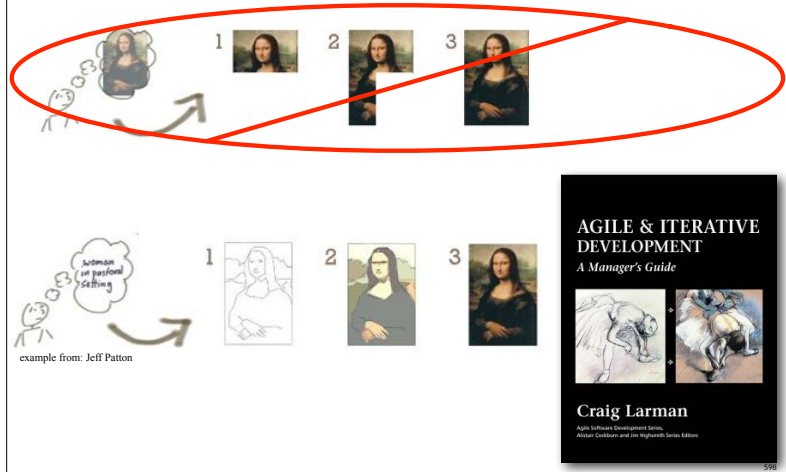


(optional) coach:

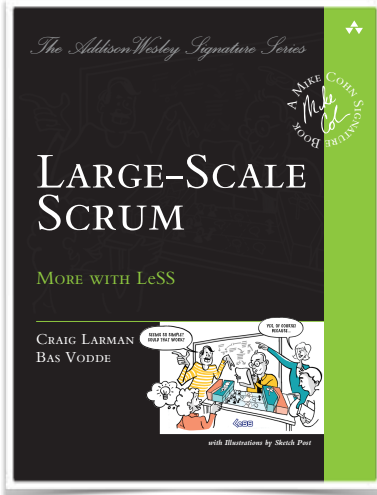
- > find a candidate giant item
- > demonstrate splitting it

597

Vertical Slices, Not “Make Big Parts & Integrate”  
iterative & evolutionary, not “build components”

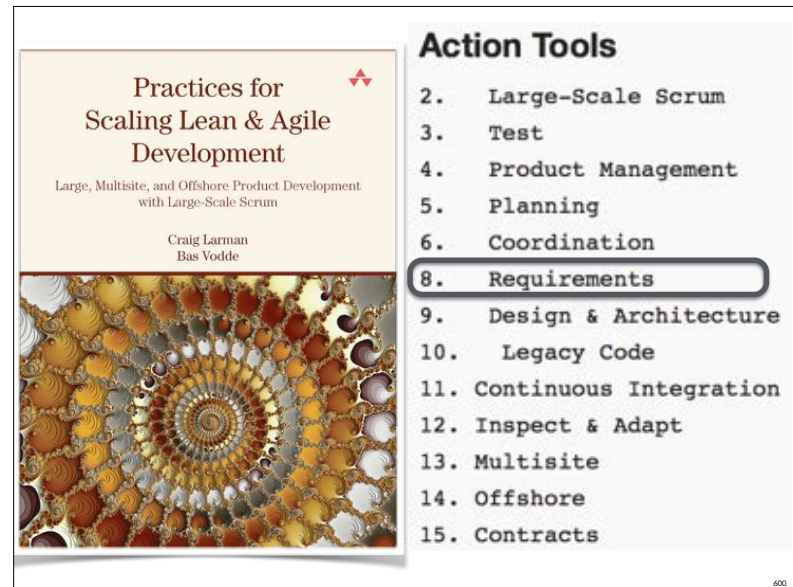


598



- 1 More with LeSS 1
  - 2 LeSS 5
- LeSS Structure**
- 3 Adoption 41
  - 4 Organize by Customer Value 67
  - 5 Management 111
  - 6 ScrumMasters 133
- LeSS Product**
- 7 Product 155
  - 8 Product Owner 171
  - 9 Product Backlog 197
  - 10 Definition of Done 231
- LeSS Sprint**
- 11 Product Backlog Refinement 249
  - 12 Sprint Planning 275
  - 13 Coordination and Integration 285
  - 14 Review & Retrospective 313

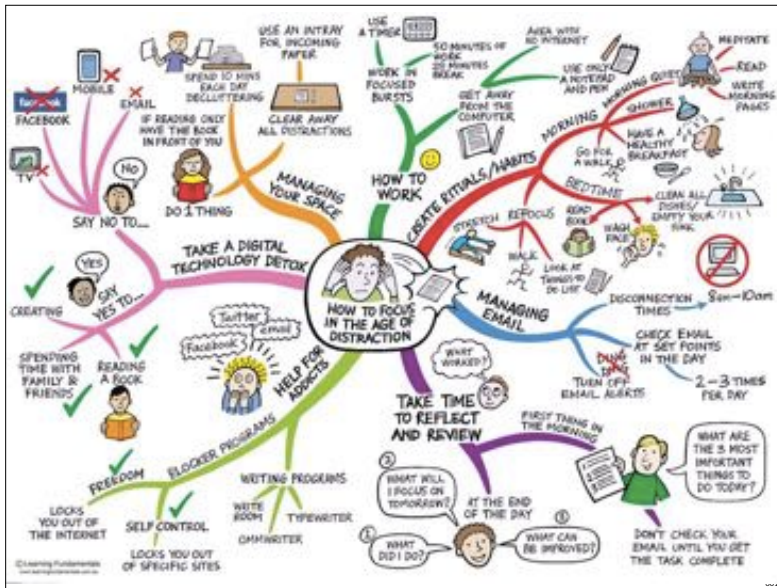
599



### Action Tools

2. Large-Scale Scrum
3. Test
4. Product Management
5. Planning
6. Coordination
8. Requirements
9. Design & Architecture
10. Legacy Code
11. Continuous Integration
12. Inspect & Adapt
13. Multisite
14. Offshore
15. Contracts

600



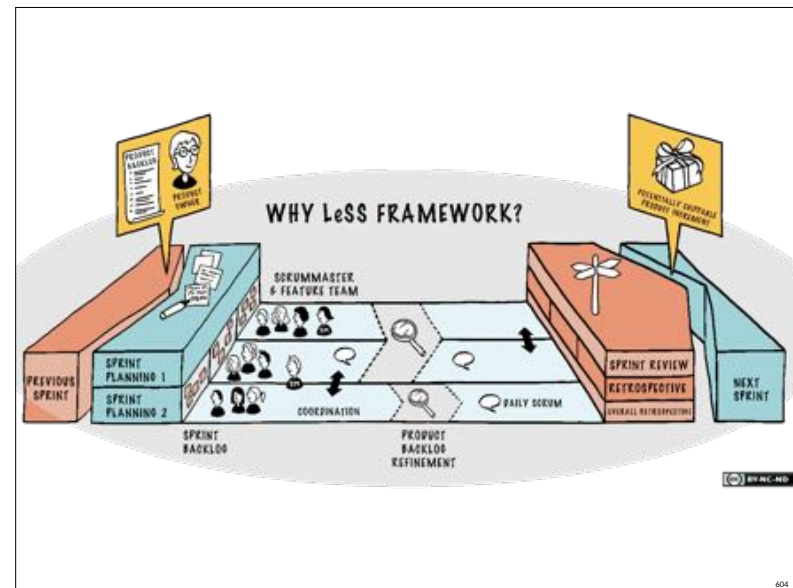
601

# LeSS Sprint Planning

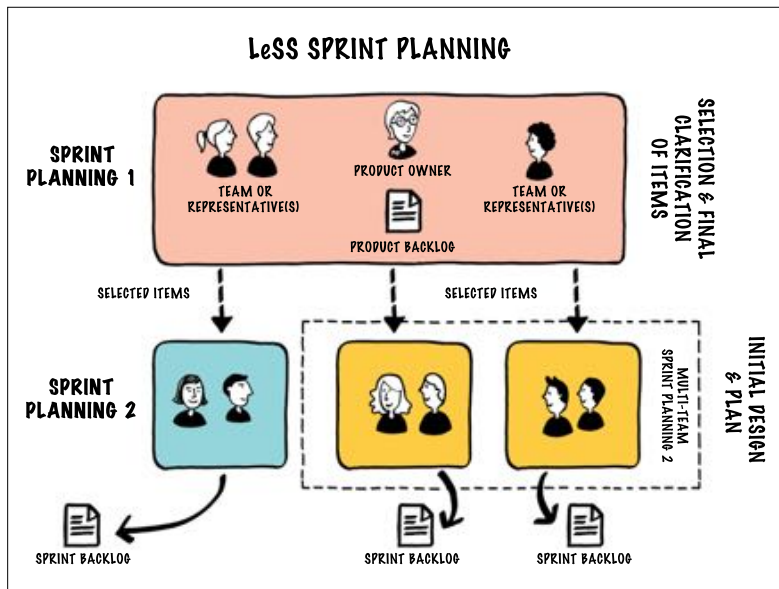
602

preparation: you will be creating a video soon, of Sprint Planning

603



604



605

## Guide: Sprint Planning One



606



class

- > are there **task dependencies between teams**, in 1 product?
- > in Sprint Planning, do people need to analyze and plan for “dependency management”?

607

Task dependencies  
between teams? None  
exist in a LeSS group  
->  
shared work,  
opportunities to work  
together

608

## Sprint Planning Two

609

609

## post-Sprint Planning recap meeting

610

610

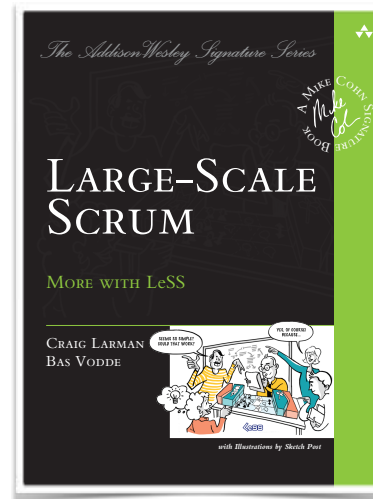


team/class:

- > prepare
- > improv!
- > video it
  - > silent movie. miming! props!
  - > about a “2 minute” complete shot
- > one continuous movie shot

611

611



- 1 More with LeSS 1
- 2 LeSS 5

### LeSS Structure

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 139

### LeSS Product

- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

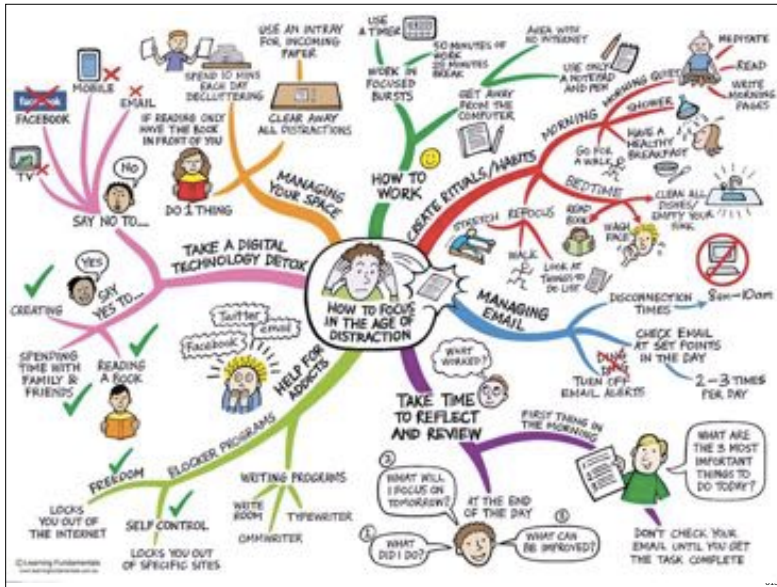
### LeSS Sprint

- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

612

612





613



614

class

> “It’s Tuesday 2pm. I (a Developer) see a coordination problem. The Scrum of Scrums meeting is tomorrow at 11 am.”

615

615

therefore...

616

616

### LeSS Rule(s)

Prefer **decentralized**  
and **informal**  
coordination over  
centralized  
coordination.

617

### We Observe...

> the more formal coordination  
methods in place, **the less**  
**coordination is happening**

618

### LeSS Rule(s)

Cross-team  
coordination is decided  
by the teams.

619



class: why?

620

We Observe...

> coordination techniques need to be especially **situational** and **customizable** in large groups

621

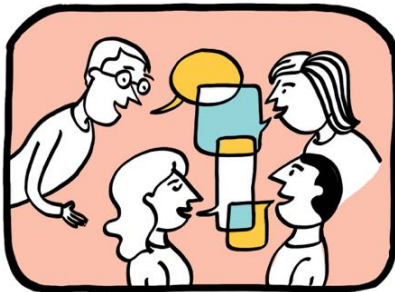
621

therefore, the most advanced coordination technique in LeSS?...

622

622

**Guide:** Just Talk (for Sprint delivery)



JUST TALK

623

623

The problem with large-scale coordination isn't *what* coordination technique to use, but knowing there's a **need to coordinate**, and **who** to talk with.

624

624



how to solve  
“**when**” & “**who**”?

625

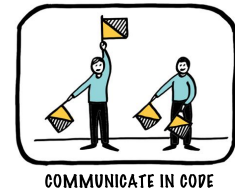
## Guide: Communicate in Code

use **the code** to tell you there's a **need** to coordinate, and **who** to talk with

“social coding” tools such as GitHub or GitLab

plugins that tell you who worked on the code, and initiates chats

integrate continuously...



626



the surprising  
meaning of  
**continuous integration**

627

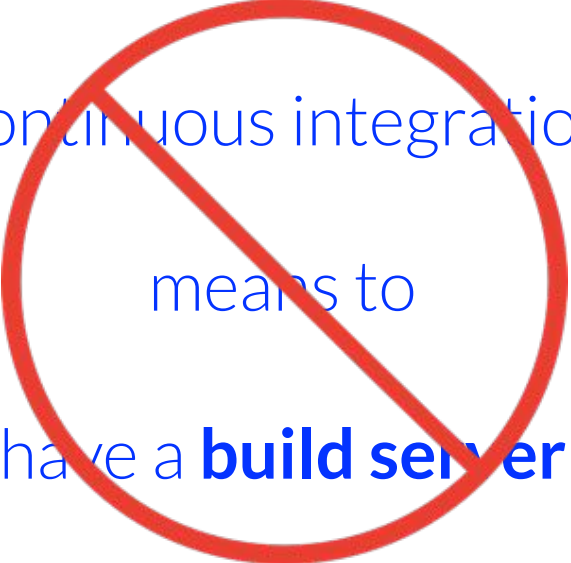
**continuous integration**

means to

**integrate continuously**

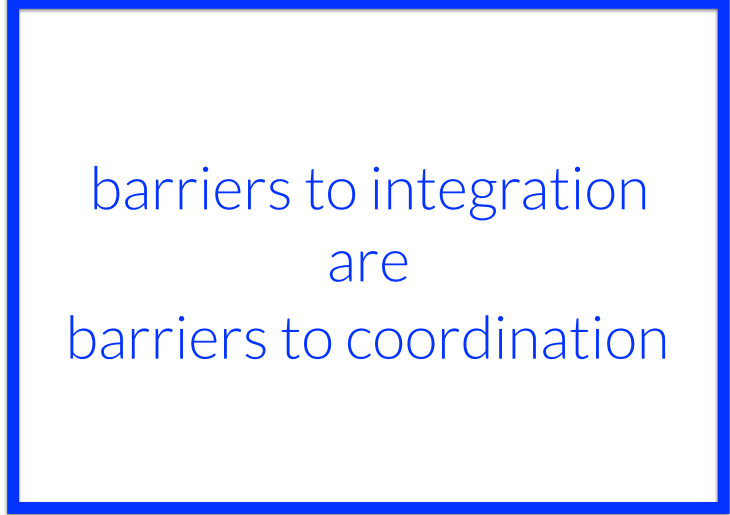
628

continuous integration  
means to  
have a **build server**



629

barriers to integration  
are  
barriers to coordination



630

## **Guide:** Integrate Continuously

> use the code to tell you  
there's a **need** to coordinate,  
and **who** to talk with

631

**“More Jenkins,  
less Jira”**

—Chet Hendrickson

632

traditionally  
**coordination supported  
integration,**  
but we can flip it to  
**integration supports  
coordination**

633

633

## Guide: Communities

- > people from different teams participate for a cross-team concern, e.g. Architecture



COMMUNITIES

- > *(see next section)*

634

634

## Guide: Multi-Team Design Workshop



... with agile modeling

635

635

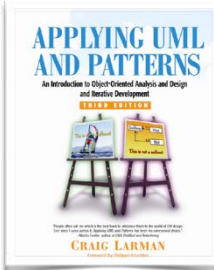
we model to have a  
conversation

the output is shared  
understanding, not a model

636

636

## Guide: Current-Architecture Workshop



Chapter 39:  
Documenting  
Architecture

637

637

## ... Videos, ...



638

638

## Guide: Component Mentors

- > regular feature-team member
- > does NOT approve other's code commits; is not a "committer gate"



COMPONENT MENTOR

639

639

barriers to integration  
are barriers to  
coordination

640

640

## Guide: Traveler



TRAVELER

641

641

## Guide: Maybe Don't do Scrum of Scrums

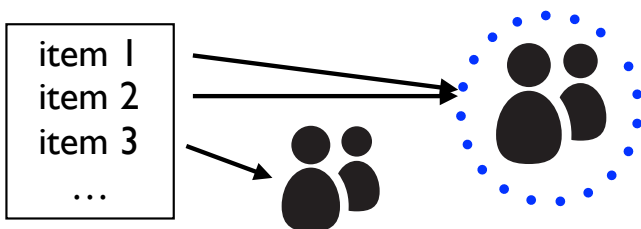


642

642

## Guide: Leading Team

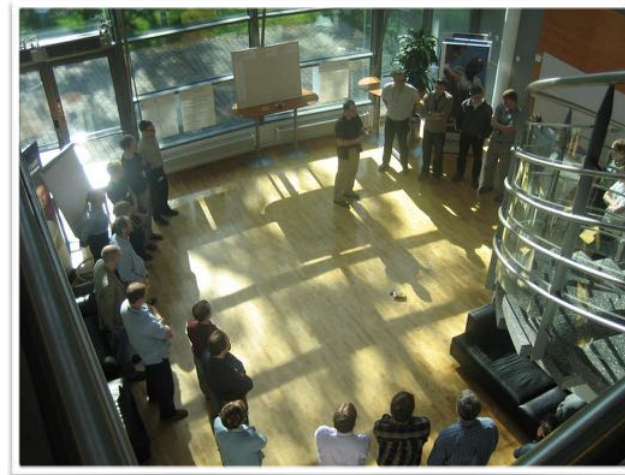
- > start work on a **big** item (or family of items)
- > start solo
- > as more teams join, they educate them



643

643

## Guide: Open Space



644

644

## Rotate Infrastructure Tasks Across Feature Teams

> build system, etc.

> slowly rotate



> NB: no separate specialist infrastructure/tools groups

645

## Guides in LeSS: Coordination

- |                                   |                         |
|-----------------------------------|-------------------------|
| 1. Just Talk                      | 7. Component Mentor     |
| 2. Communicate in Code            | 8. Travelers            |
| 3. Integrate Continuously         | 9. Maybe Don't Do SoS   |
| 4. Communities                    | 10. Leading Team        |
| 5. Multi-Team Design Workshops    | 11. Open Space          |
| 6. Current-Architecture Workshops | 12. Scouts              |
|                                   | 13. Cross-Team Meetings |
|                                   | 14. Mix & Match         |

646



individual

> review section

647

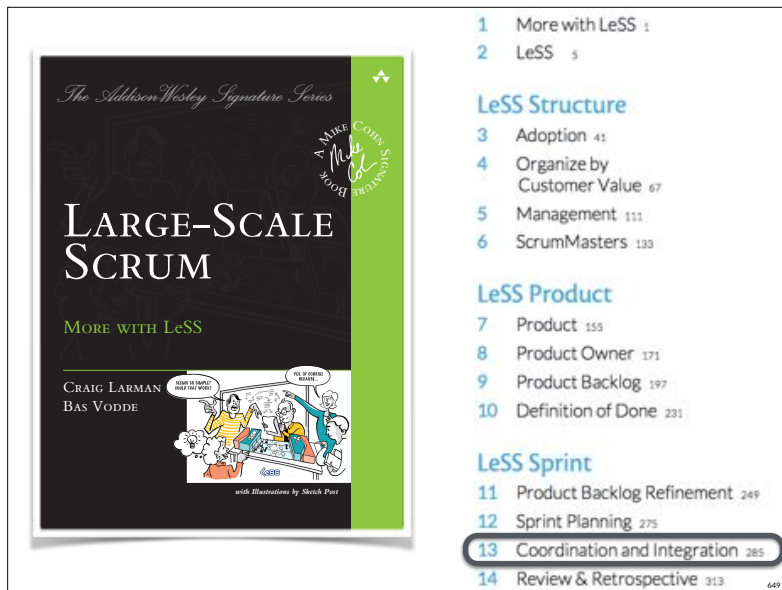


teams or pairs

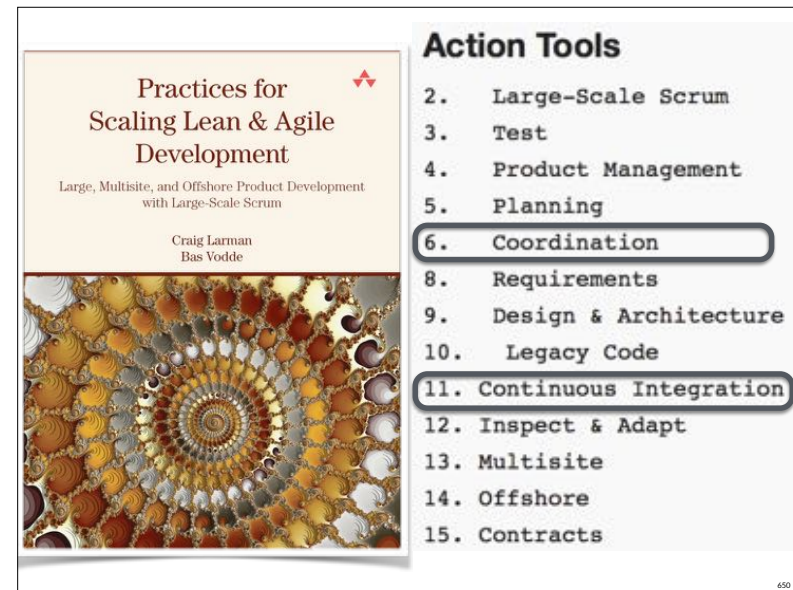
> **teach back** exercise

> please **sit** when done

648



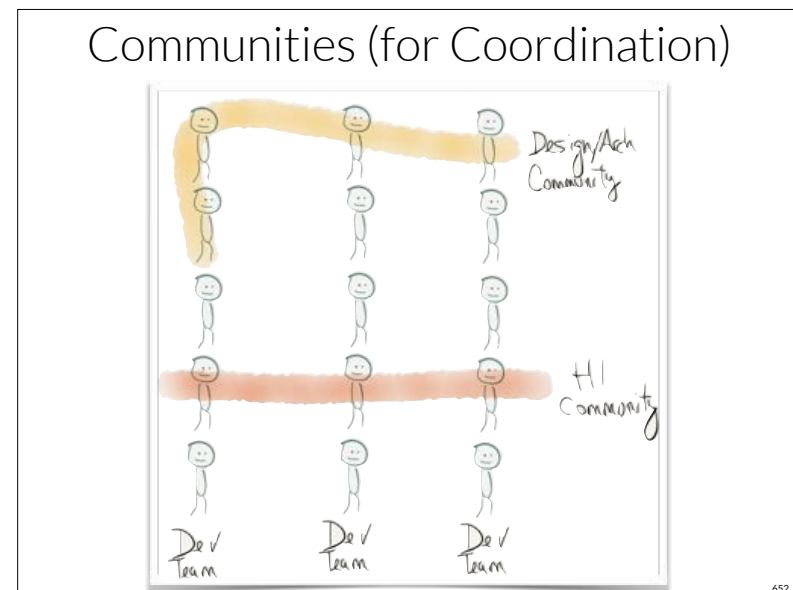
649



650



651



652

## Communities? For...

- > learning
- > speculative solutions
- > cross-team agreements
  - they can't make decisions for teams, but can make proposals that teams decide to adopt

653

653

communities do **not**  
do “the work”

654

654

## Tips for Good Communities

- > have a **community coordinator** with passion for the concern and desire to cultivate a strong community
  - > is an **active hands-on practitioner**
- > actively try to **recruit** participation from most teams
- > focus on **concrete** problem-solving goal
- > has **agreed how** they work and make decisions
- > might have a **Scrum Master** who helps it work
- > are strongly **encouraged** within the organization

655

655

## How to **Kill** Communities

- > there is **no or bad** community coordinator
- > holds frequent meetings just for the sake of it; **blah blah** meetings
- > has members that are **not in** feature teams
- > are considered **secondary** and participation is downgraded because “we’re too busy to participate”

656

656



65

658

69

- 658

65

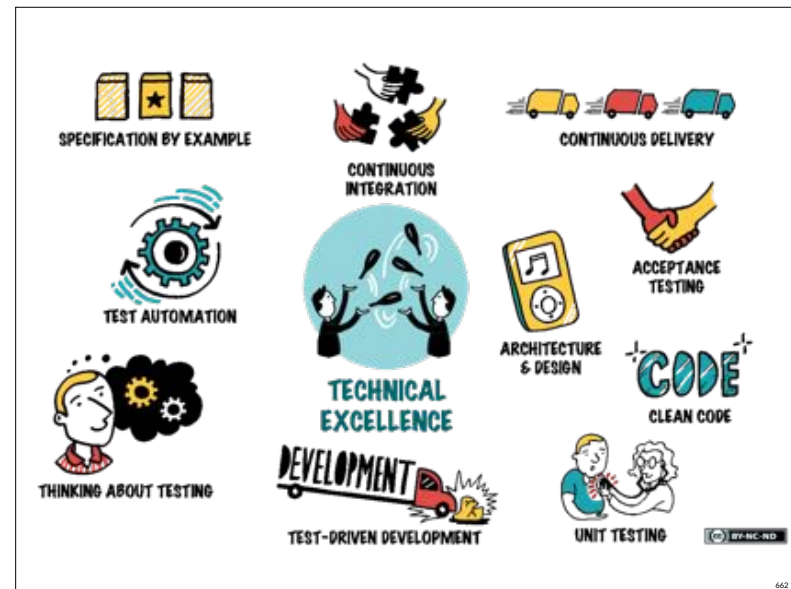
- 658



658

# Technical Excellence

661



662

## Specification by Example

Dev state	Mic Exists	Mic Muted	Input Channel	Input Channel Event	Mic Muted	Mic Muted Symbol	Toggle Mic mute LED	For Fun of mute Postgres	Other OSD feedback etc
Out of call	T	F	RC	"mute"	T	Lit	Lit	NA	NA

663

Dev state	Mic Exists	Mic Muted	Input Channel	Input Channel Event	Mic Muted	Mic Muted Symbol	Toggle Mic mute LED	For Fun of mute Postgres	Other OSD feedback etc
Out of call	T	F	RC	"mute"	T	Lit	Lit	NA	NA
Out of call	T	T	RC	"mute"	F	Unlit	Unlit	NA	NA

664

Dev state	Mic Exists	Mic Muted	Input Channel	Input Event	Mic Muted	Map (Symbol)	Table Mic route LED	For Pin of route Port	Other USD feedback etc
Out of call	T	F	R.C	"route"	T	Lit	Lit	NA	NA
Out of call	T	T	R.C	"route"	F	Unlit	Unlit	NA	NA
"	F	NA	R.C	"	NA	NA	NA	NA	NA
In call	T	F	R.C	"	T	Lit	Lit	Lit	NA
"	T	T	RL	"	F	Unlit	Unlit	Unlit	
"	T	T	RL	END					

665

# Specification with Examples

## specification with examples (real case)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Order Details										RefData					
currency pair	fx type	Verb	Amount	Dealt	Markup 1	Markup 2	Client Value	Value	Markup Type	Pip Fmt	Success	Total Markup	Wholesale Value	Profit	
EURUSD	Spot	Buy	1,000,000	EUR	2		1.30	0.00	Pts	4	OK	2	1.3002	200.00	
USDJPY	Spot	Sell	2,000,000	USD	10		110.00	0.00	Pts	2	OK	10	109.9	200,000.00	
GBPUSD	Fwd	Buy	1,000,000	GBP	1	3	1.50	0.00	Pts	4	OK	4	1.5004	400.00	
GBPUSD	Fwd	Sell	1,000,000	GBP	1	3	1.50	0.00	Pts	4	OK	4	1.4996	400.00	
GBPUSD	Fwd	Sell	1,000,000	GBP	1	0	1.50	0.00	Pts	4	OK	1	1.4999	100.00	

## executable specifications “without change”, with automated validation

Action	currency pair	fx type	Verb	Amount	Dealt	Markup 1	Markup 2	Client Value	Value	Markup Type	Error Code	Total Markup	Wholesale Value	Profit
Verify Profit Details For Order	EURUSD	Spot	Buy	1000000	EUR	2.0		1.30	0.00	Pts	0	2.0	1.3002	200.00
Verify Profit Details For Order	USDJPY	Spot	Sell	2000000	USD	10.0		110.00	0.00	Pts	0	10.0	109.9	200000.00
Verify Profit Details For Order	GBPUSD	Fwd	Buy	1000000	GBP	1.0	3.0	1.5	0.00	Pts	0	4.0	1.5004	400.00
Verify Profit Details For Order	GBPUSD	Fwd	Sell	1000000	GBP	1.0	3.0	1.5	0.00	Pts	0	4.0	1.4996	400.00
Verify Profit Details For Order	GBPUSD	Fwd	Sell	1000000	GBP	1.0	0.0	1.5	0.00	Pts	0	1.0	1.4999	100.00

666

666

More with LeSS

### Practices for Scaling Lean & Agile Development

Large, Multitask, and Offshore Product Development with Large-Scale Scrum

Craig Larman  
Dan Vande

## Action Tools

2. Large-Scale Scrum
3. Test
4. Product Management
5. Planning
6. Coordination
8. Requirements
9. Design & Architecture
10. Legacy Code
11. Continuous Integration
12. Inspect & Adapt
13. Multisite
14. Offshore
15. Contracts

### Architecture & Design

Thinking About Design

#### Behavior-Oriented Tips

- Design workshops with agile modeling
- Just-in-Time (JIT) modeling: vary the abstraction level
- Design workshops each iteration
- Design workshops in the team rooms
- Multi-team design workshops for broader design issues
- Technical leaders teach at workshops
- Architects and system engineers are regular (feature) team members
- Question all early architectural decisions as final
- Don't conform to outdated architectural decisions
- Avoid architecture astronauts (PowerPoint architects)
- Very early, develop a walking skeleton with tracer code
- Incrementally build 'vertical' architectural slices of customer-centric features
- Do customer-centric features with major architectural impact first
- Architects clarify by programming spike solutions
- Don't let architects hand off to 'coders'
- Tiger team conquers then divides
- SAO workshops at end of "tiger phase"

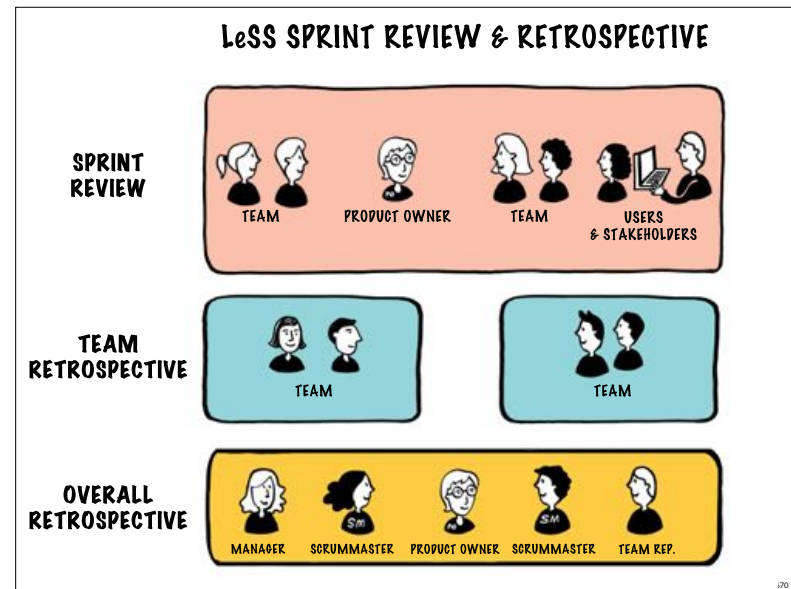
667

# technical excellence

668

# LeSS Sprint Review

669



670

in-Sprint  
early  
item feedback

671

671

## Guide: Sprint Review Bazaar



672

672



## Sprint Review “Bazaar” Q&A



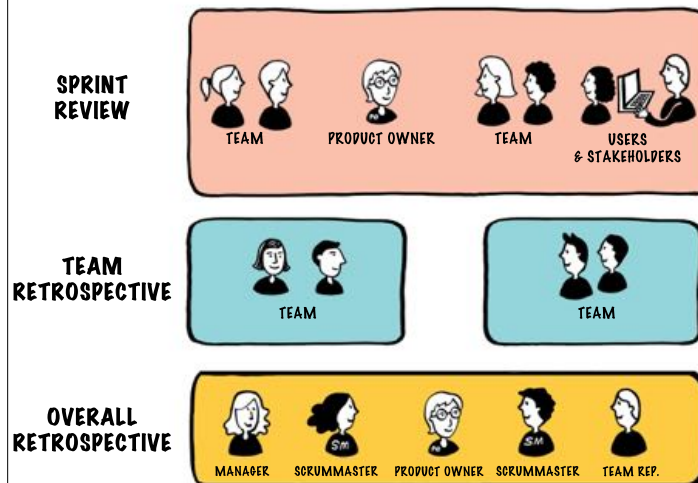
673

673

## LeSS Sprint Retrospective

674

### LeSS SPRINT REVIEW & RETROSPECTIVE



675

675

### Guide: Overall Retrospective (multisite)



676

676

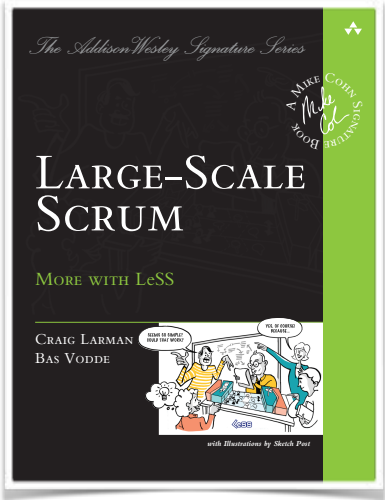
# Systems Modeling

```
graph TD; Velocity[Velocity] --> LOC[LOC]; LOC --> Velocity; Velocity -->|negative| CodeQuality[Code Quality]; HireRate[Hire rate cheap devs] --> PctWeak[% of weak devs]; PctWeak --> CodeQuality; Note((Belief: mngs can hire devs who looking at code)) -.-> HireRate; Note2(only in short term) -.-> Velocity; Note2 -.-> LOC;
```

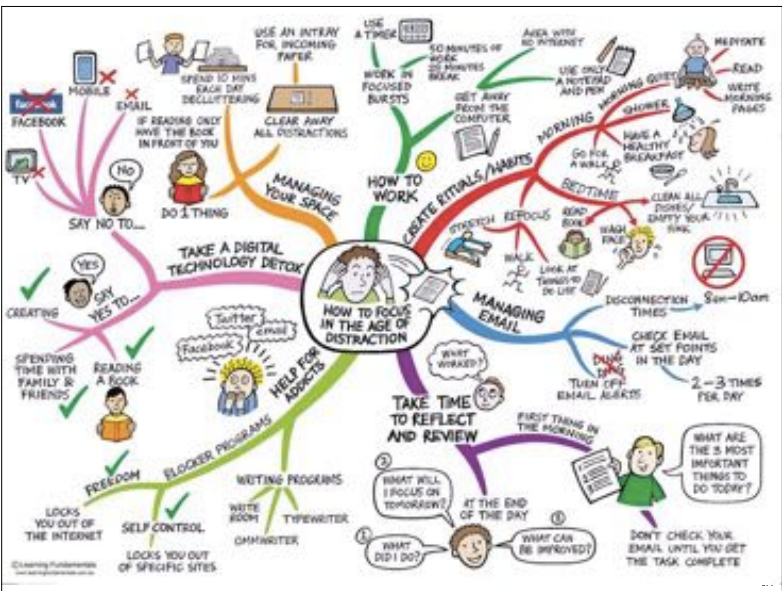
The diagram illustrates a systems modeling concept, likely related to software development metrics and their interdependencies. It features several interconnected components:

- Velocity** and **LOC** (Lines of Code) are connected by a curved arrow, suggesting a relationship or feedback loop.
- Code quality** is shown as a box, with a curved arrow pointing from it towards the Velocity/LOC relationship, indicating its influence.
- % of weak devs** (Percentage of weak developers) is shown as a box, with a curved arrow pointing from it towards the Code quality box, indicating its influence.
- Hire rate cheap devs** (Hire rate cheap developers) is shown as a box, with a curved arrow pointing from it towards the % of weak devs box, indicating its influence.
- A cloud-shaped box contains the text: "Belief: mngs can hire devs who looking at code", with a dotted arrow pointing towards the Hire rate cheap devs box.
- A curved arrow labeled "only in short term" points from the Velocity/LOC relationship towards the Hire rate cheap devs box, suggesting a temporary or short-term effect.

677



678



100

# Done & Undone

# Done

681



class

- > for one sample large-scale product, tasks to have a shippable product?

682

682

**perfect** DoD  
=  
shippable DoD  
=  
potentially shippable

683

683

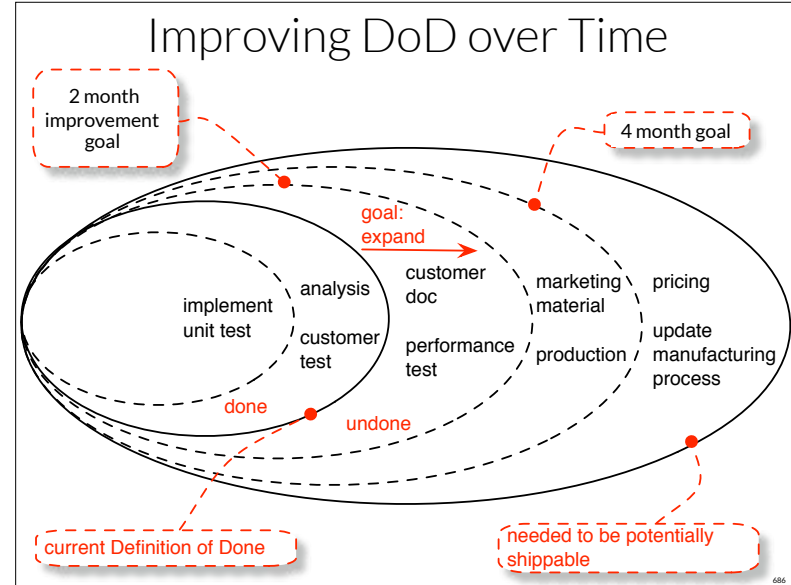
an **imperfect DoD** is  
especially common in  
large-scale groups first  
moving to LeSS

684

684



685



686

Perfect DoD  
- Imperfect DoD  
  
= Undone Work

#### DEFINITION OF DONE:

- CODED (UNIT-TESTING)
- INTEGRATED: INT. TEST, REAL ENV. (WHEN AVAILABLE)
- ACCEPTANCE TESTS PASSED (MANUAL), MOCK, STUB
- DRAFT user doc WRITTEN (CHANGE LOG FILE)
- RELEASED (BUILD <sup>REL NOTES</sup> AVAILABLE ON TRACKER)

#### UNDONE

- PERFORMANCE TEST
- QA TEST (END TO END)
- FULL UAT

688

687



## Handling Undone Work?

*covered in next section*

689

## Perfection Goal

perfect DoD = shippable

no Undone Work

690

## LeSS Rule(s)

1 Definition of Done

not for each team

(teams can extend base version)

691

## LeSS Adoption Tip

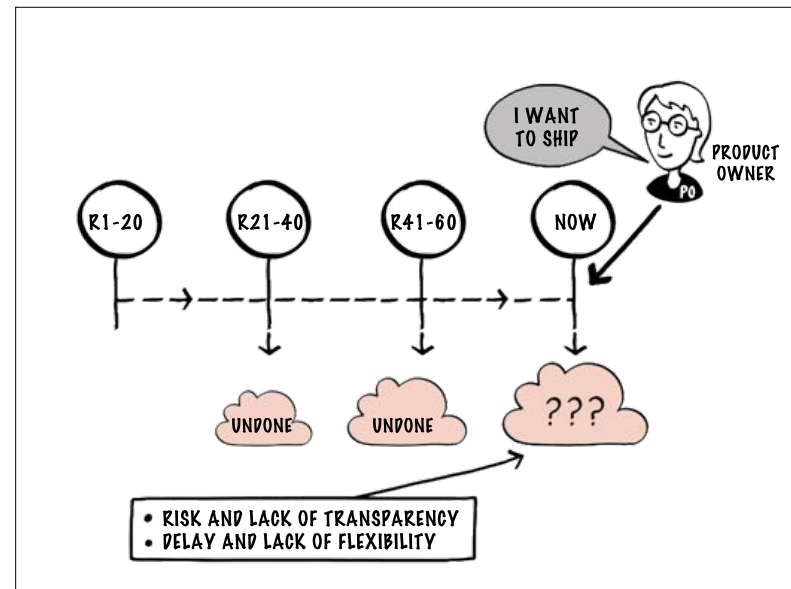
if possible, solve the problems  
so you can have a perfect DoD  
before Sprint 1

why?

692

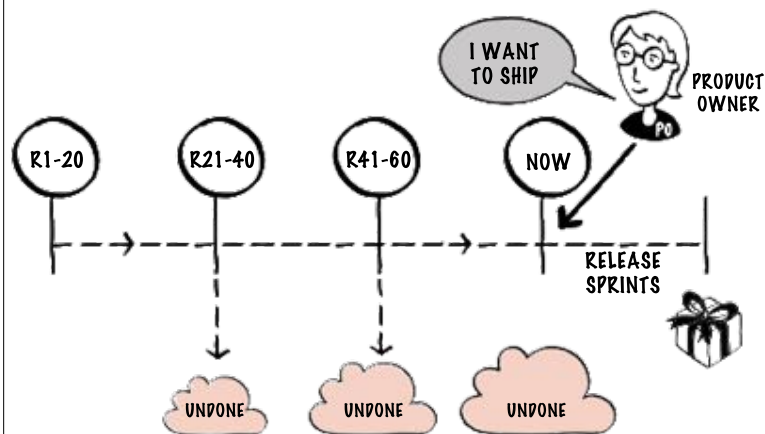
# Undone

693



694

## Release Sprint... by **TEAMS**



695

## Lean Wastes in Product Development

1. **Over-production**—of intermediate, WIP, or finished things; sooner, faster, greater than demand
2. **Inventory**—intermediate, WIP, or finished things
3. **Over-processing**—& extra processes, rediscovery
4. **Handoff**—& transport
5. **Task switching**—& motion
6. **Waiting**—& delay
7. **Defects & finding/correcting**—tasks to find & correct: test, inspect, review, modify
8. **Not using people's full potential**—working to title, not multi-skilling
9. **Knowledge/information scatter/loss**—& connection to handoff & inventory & rediscovery; communication barriers: indirection, 1-way flows

696

and...

to get to perfect DoD

Teams **learn by doing**

697

697

you should NOT need a  
**Release Sprint**; but it  
may be a temporary  
“necessary evil” during  
early transition to LeSS

698

698

what if there were ‘7’  
Sprints before the  
Release Sprint?

(a bad idea; rather, ship every Sprint)

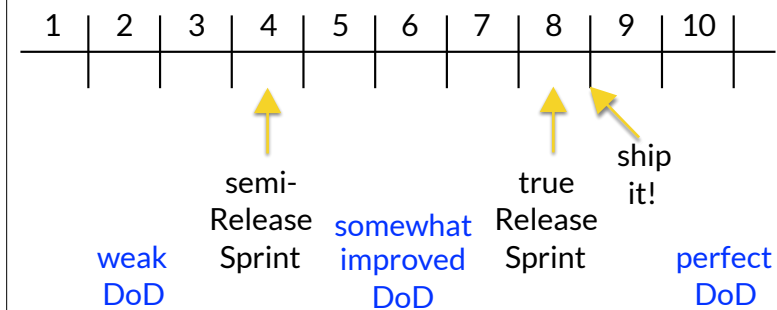
699

699

Frequent “Semi-Release Sprints”

Undone Work has:

**risk**  
**delay**



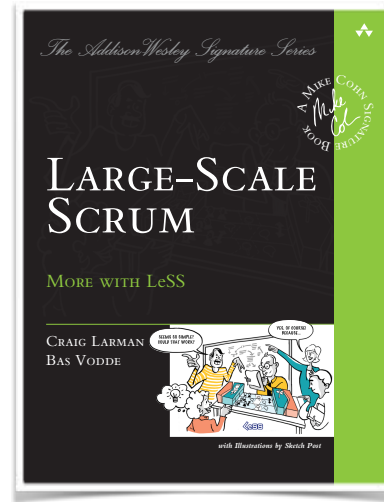
700

700

get to perfect DoD  
**ASAP**

701

701



- 1 More with LeSS 1
- 2 LeSS 5

#### LeSS Structure

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 139

#### LeSS Product

- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

#### LeSS Sprint

- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

702

702



coach:  
> discuss questions

703

703

DevOps

704

since we're exploring  
shipping, what about  
DevOps? ...

705

705

Realizing “DevOps” implies...

- > extending the DoD to include operations tasks
- > increasing the cross-functionality of the feature teams
- > dissolving and merging in the Operations group into feature teams

706

706

“level 1 support”?


707

707

DevOps thought leaders on  
DevOps ...

708

708



**CONTINUOUS DELIVERY**

BLOG PUBLICATIONS TALKS ABOUT

« On Antifragility in Systems and Organizational Architecture Elisabeth Hendrickson Dis

**There's No Such Thing as a "Devops Team"**

Nor should there be "devops specialists"

**Why Segregation of Duties Doesn't Work**

**Why are Functional Silos Problematic?**

709

709

why does  
**"fake DevOps"**  
 arise?

"DevOps team",  
 "DevOps specialist"

710

710

Larman's Laws of Organizational Behavior

1. Organizations are implicitly optimized to avoid changing the status quo middle- and first-level manager and "specialist" positions & power structures.
2. As a corollary to (1), any change initiative will be reduced to overloading or redefining the new terminology to mean basically the same as status quo.
3. As a corollary to (1), any change initiative will be derided as "purist", "theoretical", "religious", and "needing pragmatic customization for local concerns" — which deflects from addressing weaknesses and manager/specialist status quo.
4. As a corollary to (1), if after *changing the change* some managers and single-specialists are still displaced, they become "coaches/trainers" for the change, frequently reinforcing (2) and (3).
5. Culture follows structure (or behavior/mindset follows system)

711

711

LeSS Huge

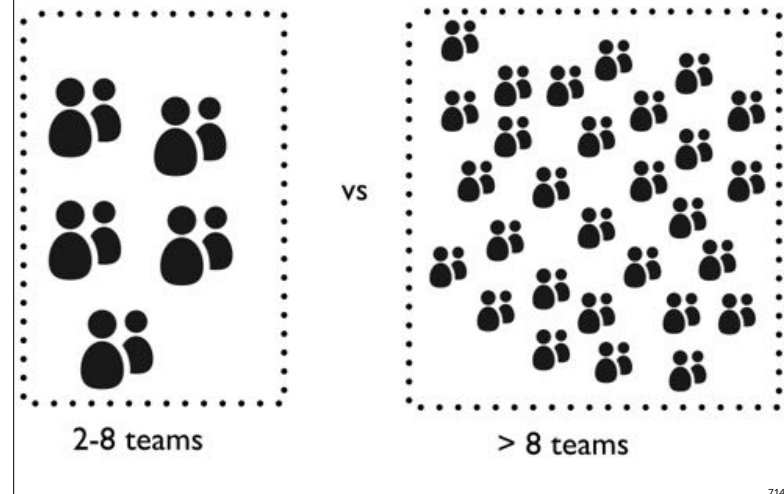
712

712

# LeSS Huge Framework

713

## 2 Frameworks: LeSS & LeSS Huge



714



team

- > sketch a systems model, given:
- > one Product Owner, for a product with 20 teams
- > what are the noteworthy variables?

715

“8” is not a magic  
number

716

## Requirement Areas

Item	Requirement Area	...
B	market on-boarding	
C	trade processing	
D	asset servicing	
F	market on-boarding	
...		

717

717



coach

- > for some participant in a “huge” context, your possible requirement areas?

718

718

## Requirement Areas are SLOWLY DYNAMIC



719

719

## Requirement Areas Are...

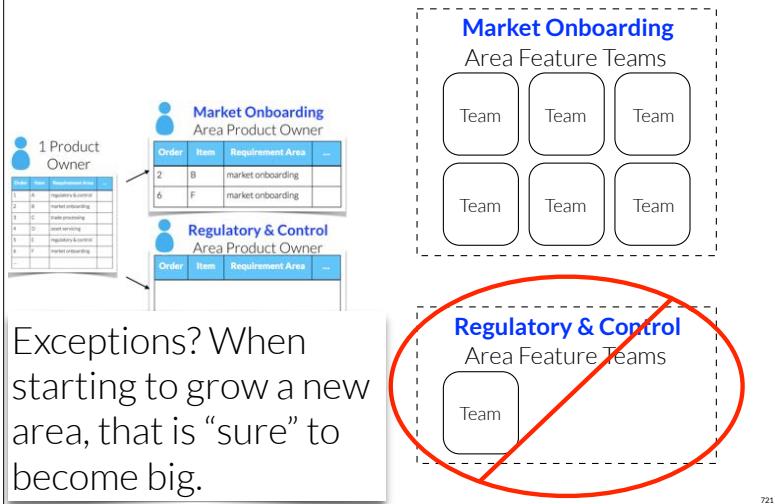
- > ... *not* the same “domains” in Domain-Driven Design
- > though there can be an overlap
- > Requirement Areas can be more dynamic and market driven

720

720



## An Area can't have only 1 Team



721



team

> sketch a systems model, given:

> **many small (e.g. 1 or 2-team) Requirement Areas**

> reminder: each RA has an Area Backlog, and teams are in 1 area

> what are the noteworthy variables?

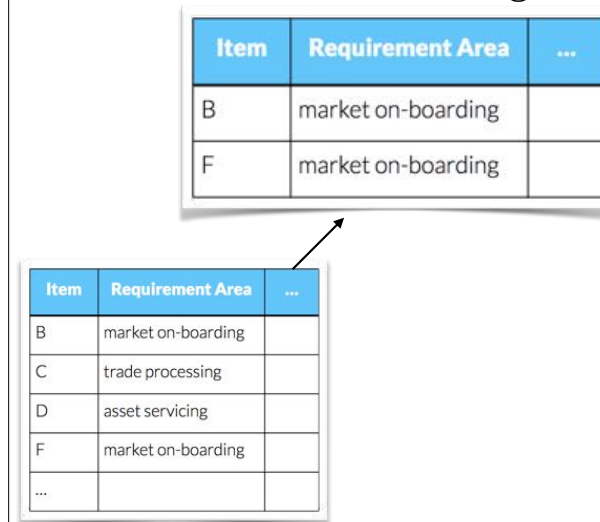
722

## LeSS Rule(s)

Each Requirement Area has between “4-8” teams. Avoid violating this range.

723

## “Area Backlog”



724

723

724

## Area Backlogs via **Views**

Item	Requirement Area	...
B	market on-boarding	
F	market on-boarding	

Item	Requirement Area	...
B	market on-boarding	
C	trade processing	
D	asset servicing	
F	market on-boarding	
...		

a VIEW for one requirement area

it is NOT a separate artifact

725

725

## Area Backlogs via **Separate Artifacts**

Item	Requirement Area	...
B	market on-boarding	
F	market on-boarding	

Item	Requirement Area	...
B	market on-boarding	
C	trade processing	
D	asset servicing	
F	market on-boarding	
...		

an ARTIFACT for one requirement area

it IS a separate artifact

726

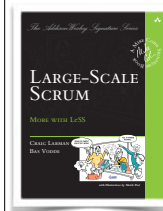
726

Area Backlogs as  
**views**  
vs  
Area Backlogs as  
**separate artifacts**

727

727

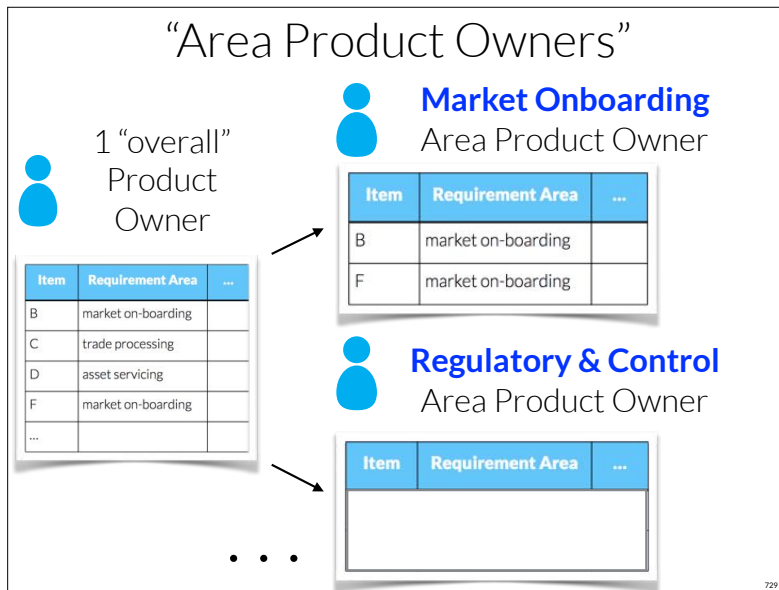
## Area Backlogs via **Separate Artifacts**



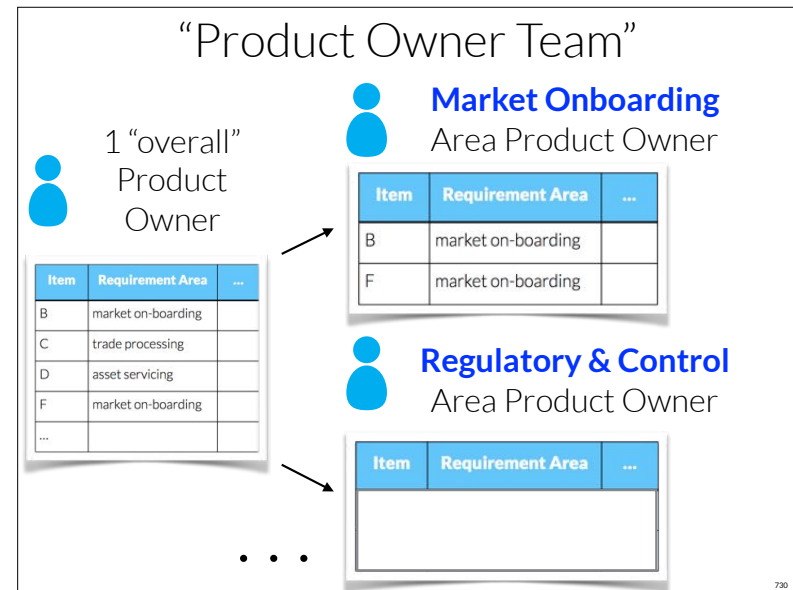
Contents
Product Backlog in LeSS 207
• Guide: Don't "Manage Dependencies" but Minimize Constraints 208
• Guide: Take a Bite 212
• Guide: Dealing with Parents 214
• Guide: Handling Special Items 217
• Guide: Tools for Large Product Backlogs 220
• Guide: More Outcome, less Output 223
LeSS Huge 225
• <b>Guide: Area Backlogs 225</b>
• Guide: Three Levels Max 232
• Guide: New Area for Giant Requirement 233
• Guide: Handling Gigantic Requirements 234

728

728



729



730

~~PO Team with analysts,  
UX/UI designers,  
architects, project  
managers~~

731

responsibilities of the  
Product Owner?

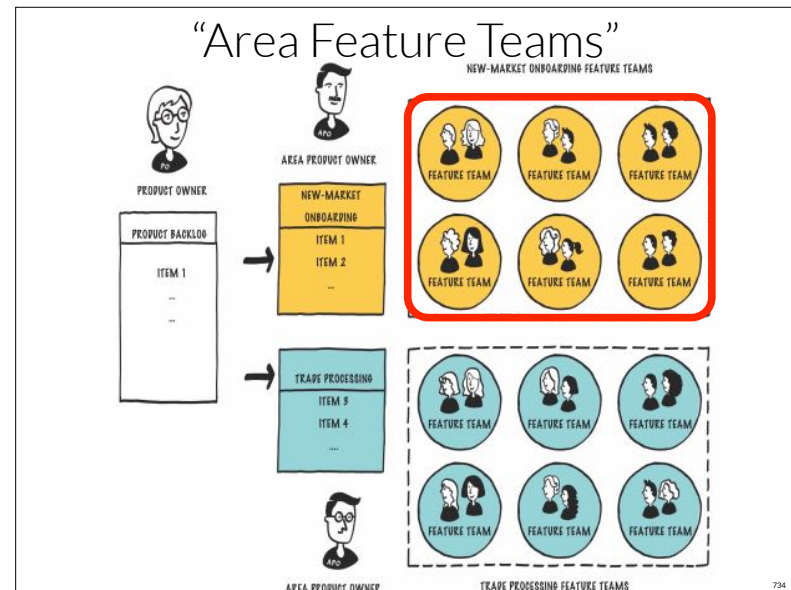
732

## Guide: Product Owner Team Meeting

- > issue: losing whole-product focus or alignment between areas in the choice of themes and items
- > each Area Product Owner shares their situation and upcoming goals, and they discuss opportunities to align
- > Product Owner can provide high-level guidance
- > discuss the results of the previous Sprint Review meetings in each Requirement Area, as input to planning
- > include some team reps for learning and feedback
- > include 1 Scrum Master to support reflection and improvement

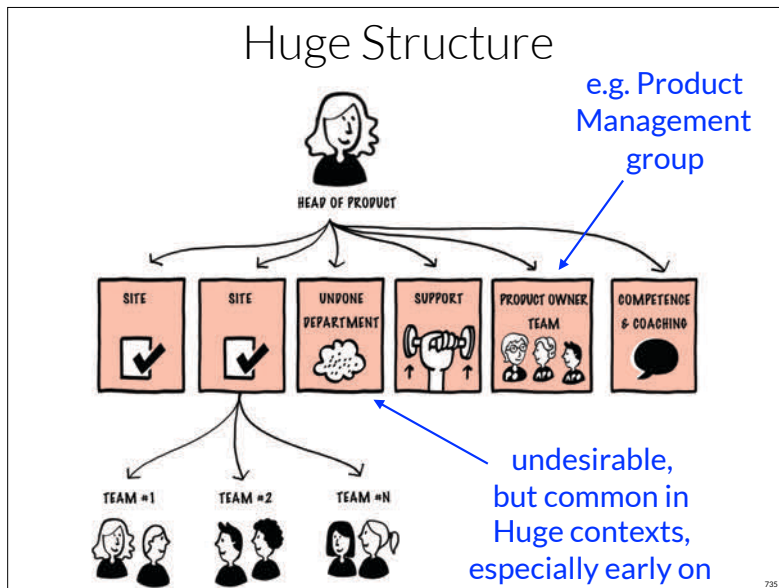
733

733



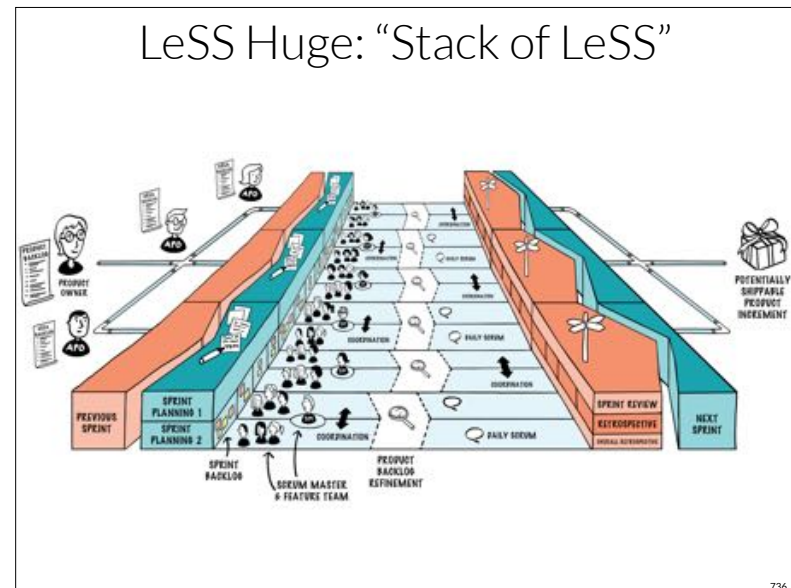
734

734



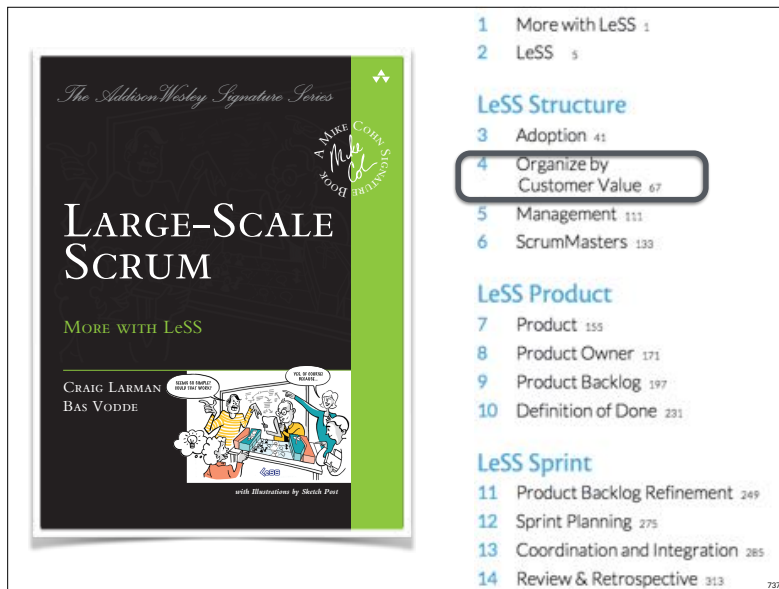
735

735

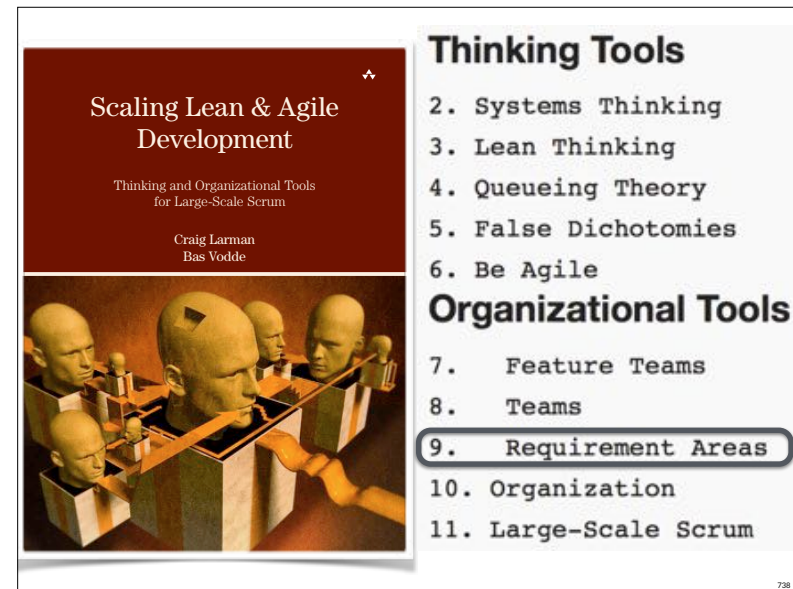


736

736



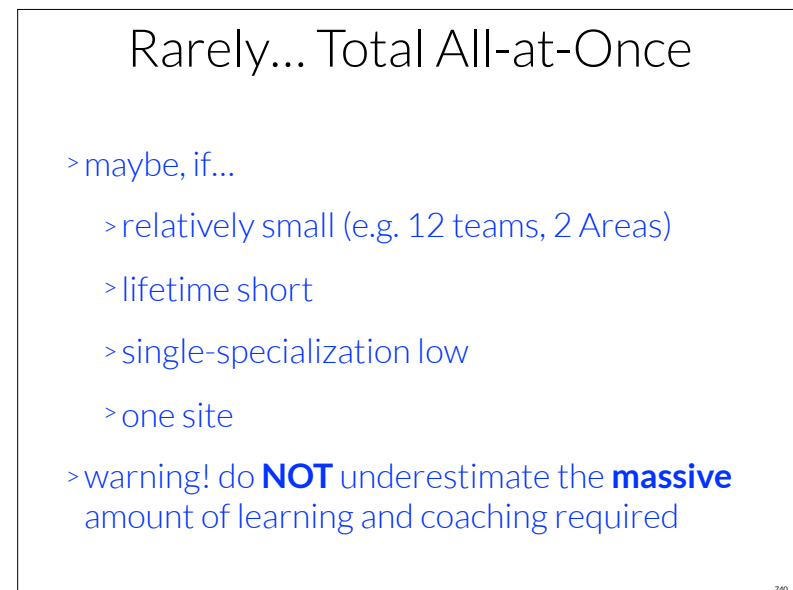
737



738



739



740

but more common...

741

## LeSS Rule(s)

LeSS Huge adoptions, including the structural changes, are done with an **evolutionary incremental** approach.

Remember each day: LeSS Huge adoptions take months or years, infinite patience, and sense of humor.

742

742

## Guide: Evolutionary Incremental Adoption

> two alternatives:

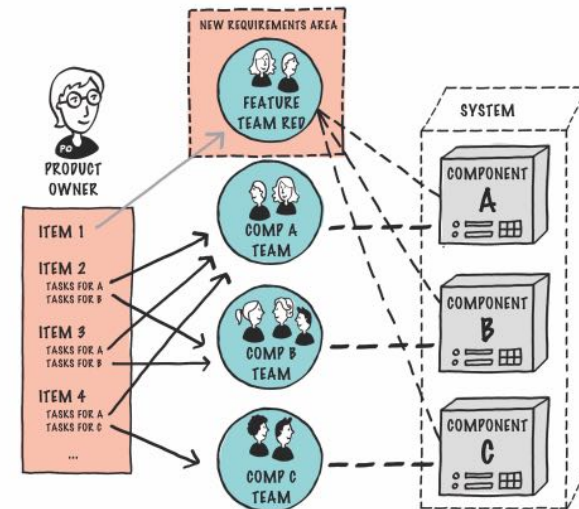
> **focused deeper adoption at a part of the product group**

> gradual incremental adoption over the whole product group

743

743

## Guide: Parallel Organizations



744

744

## Guide: Parallel Organizations

- > *focused deeper adoption at a part of the product group*
- > gradual, low-risk, well-suited for **huge** LeSS Huge product groups
- > key drawback? takes a *long* time
- > must: abandon private code
- > don't: allow branching

745

745

## Guide: One Requirement Area at a Time

- > *focused deeper adoption at a part of the product group*
- > “all at once” in only one Requirement Area
- > is there some high-benefit low-risk area?
- > wicked problem: the new org model exists interacting closely with the old model
- > must: abandon private code

746

746

## Guide: Usual Recommendation...

**Parallel Organizations**

in

**One Requirement Area  
at a Time**

747

747

## Guide: Transitioning to Feature Teams

- > *gradual incremental adoption over the whole product group*
- > **gradually expand component team responsibility**
- > use **feature-team adoption map**
- > context: huge, many sites, high learning across sites required
- > problems:
  - > drawbacks of both feature and component teams while not giving the best benefits
  - > hard to adopt customer-centric Requirement Areas when the teams are still component teams

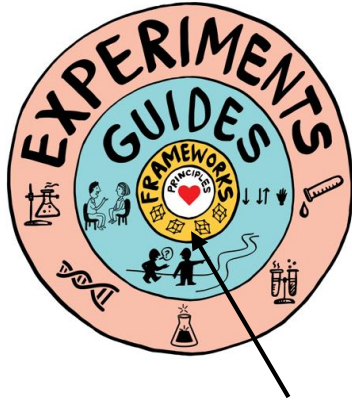
748

748









the LeSS **rules**  
define the 2 frameworks

753

“we don’t want rules”  
so why do they exist?

754

Scaling Sweet Spot & **Shu**-Ha-Ri



“barely sufficient methodology”

755

Why “Rules”? **Shu** & Focus on Creating...

- > transparency
- > whole-product focus
- > global systems optimization
- > empirical process control

756



individual

- > scan the LeSS rules (+ Huge)
- > (optional) record questions

757

757



coach:

- > discuss questions

758

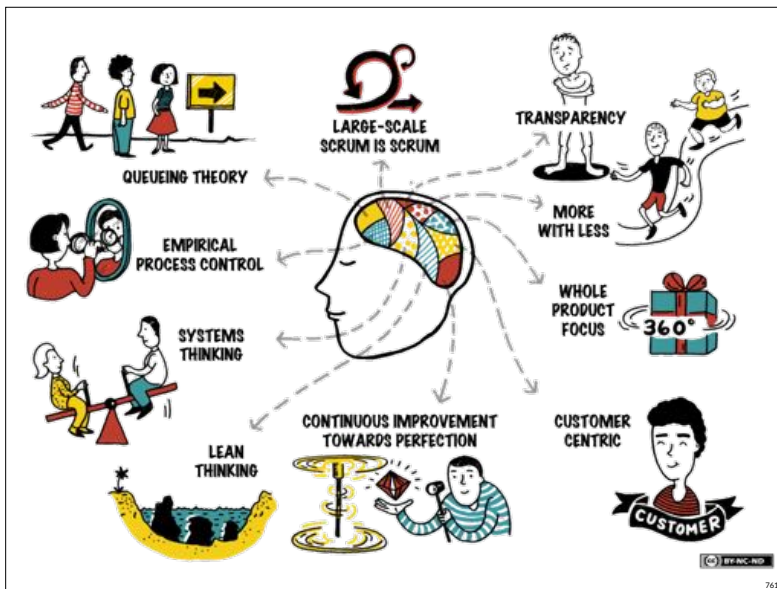
758

# LeSS Principles

759

# LeSS Principles

760



761

## LeSS Principles

**Large-Scale Scrum is Scrum**—It is not “new and improved Scrum.” And it is not “One-team Scrums at the bottom, and something different on top.” Rather, LeSS is about figuring out how to apply the principles, elements, and purpose of Scrum in a large-scale context.

**Empirical process control**—Inspect & adapt processes, organizational design, & practices to craft a contextually-appropriate organization based on Scrum, rather than following a detailed script.

**Transparency**—Based on tangible ‘done’ items, short cycles, working together, common definitions, and driving out fear in the workplace.

**Whole-product focus**—One Product Backlog, one Product Owner, one Shippable Increment, one common Sprint—regardless if there are 3 or 33 teams. Customers want the product, not a part.

**Customer-centric**—Identify value & waste in the eyes of paying customers. Reduce cycle time from their perspective. Do user-centered design. Increase feedback loops with real customers.

762

## LeSS Principles

**Continuous improvement towards perfection**—Create and deliver a product in no time, with no cost and no defects, that utterly delights customers, improves the environment, and makes lives better. Do humble and radical improvement experiments each Sprint towards that.

**Systems thinking**—See, understand, and optimize the whole system (not parts), and do causal-loop modeling to explore system dynamics. Avoid the local and sub-optimizations of focusing on the ‘efficiency’ of individuals and individual teams. Customers care about the overall concept-to-cash cycle time and flow, not individual steps.

**Lean thinking**—Create an organizational system whose foundation is manager-teachers who apply and teach systems thinking and lean thinking, manage to improve, and who practice Go See and Help at gemba. Add the two pillars of respect for people and continuous improvement. All towards the goal of perfection.

**Queueing theory**—Understand how systems with queues behave in the R&D domain, and apply those insights to managing queue sizes, work-in-progress limits, multitasking, work packages, and variability.

**More with LeSS**—See prior section.

763



team: round robin

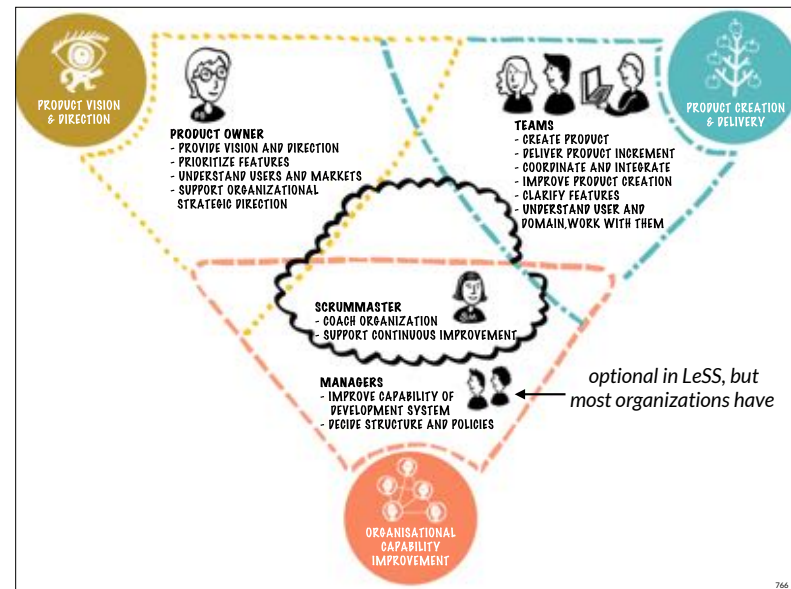
> without notes...

> “charades” to communicate each principle

764

# LeSS Roles

765



766

Where is the  
Product Owner?  
3 Types of  
Development

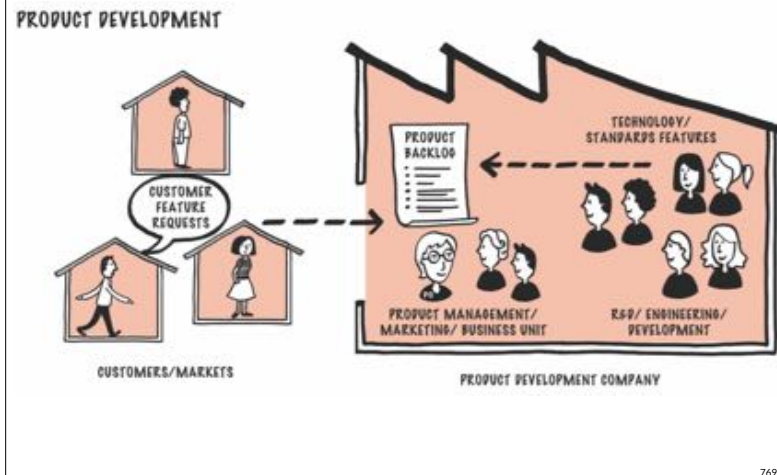
767

Why Learn 3 Types of Development?

- > where is the Product Owner?
- > major “pattern” groupings?

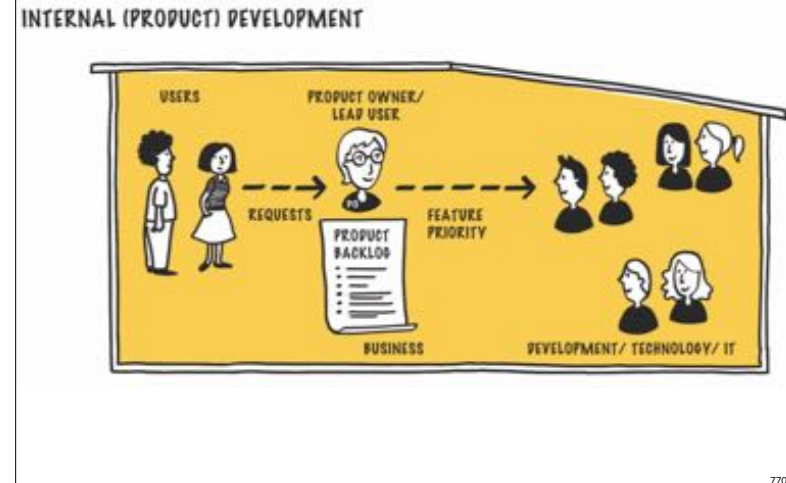
768

## (External) Product Development



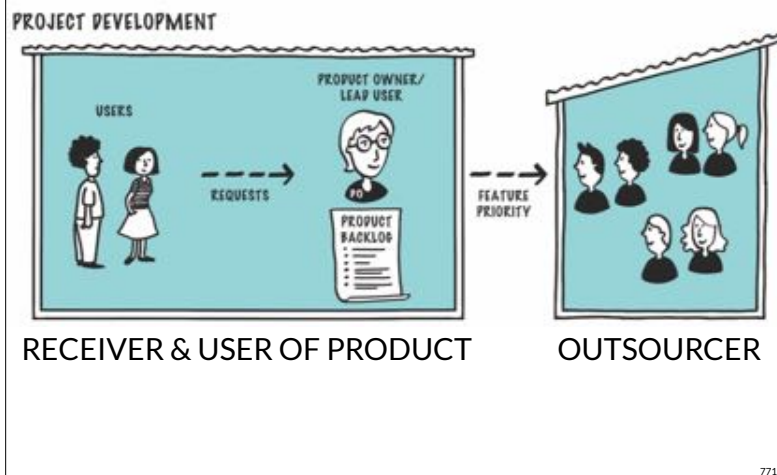
769

## Internal (Product) Development



770

## (Outsourced) Project (Product) Development



771

The book cover for 'Large-Scale Scrum' is shown. It features the title in large white letters on a black background. Below the title, it says 'MORE WITH LeSS'. The authors' names, 'CRAIG LARMAN' and 'BAS VODDE', are listed. There is a small illustration at the bottom showing people working.

- 1 More with LeSS 1
- 2 LeSS 5

**LeSS Structure**

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 130

**LeSS Product**

- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

**LeSS Sprint**

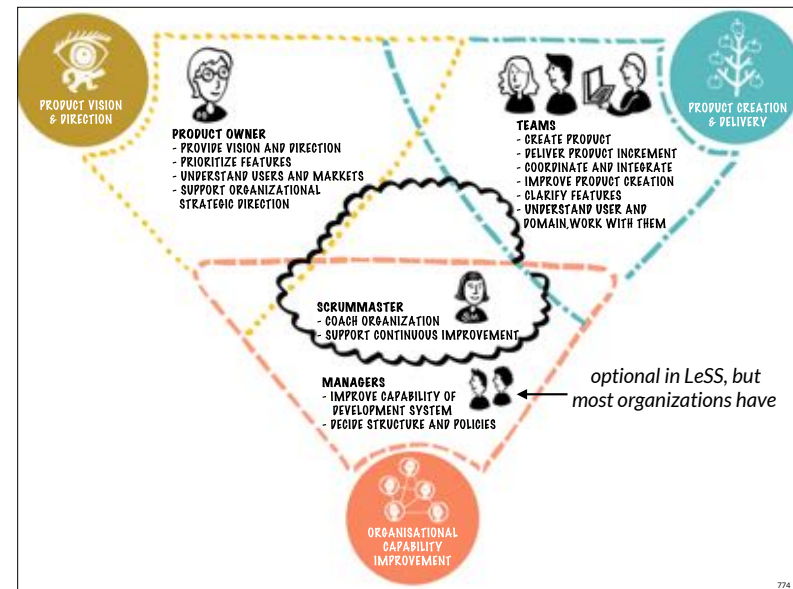
- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

772

772

# Product Owner in LeSS

773



774

## LeSS Rule(s)

There is one Product Owner and one Product Backlog for the complete shippable product.

The Product Owner shouldn't work alone on Product Backlog refinement; it is mostly done by the multiple Teams working directly with customers, users, and other stakeholders.

All prioritization (ordering) goes through the Product Owner, but clarification is as much as possible directly between the Teams and customer, users, and other stakeholders.

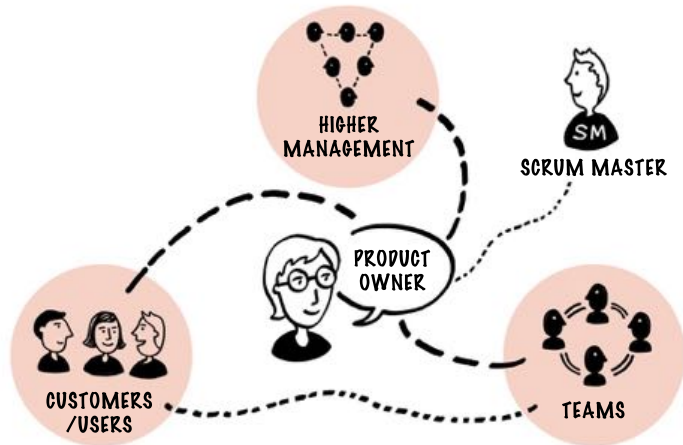
775

## Guides in LeSS: Product Owner

- > Who Should be Product Owner?
  - > Customer Collaborations over...
- > Who are those Users/Customers?
  - > Ship At Least Every Sprint
- > Prioritization over Clarification
  - > Don't Be Nice
- > Don't Do It
  - > Let Go
- > Helpers
  - > Don't Let Undone Work be Your Undoing
- > Five Relationships

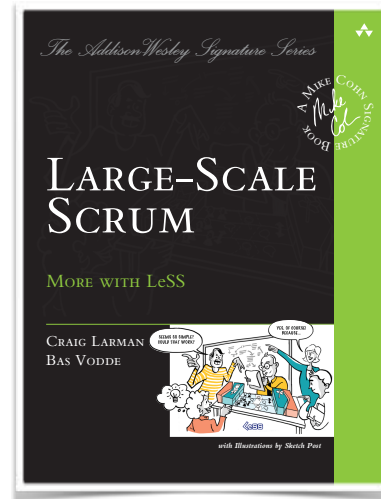
776

## 5 relationships



777

777



- 1 More with LeSS 1
- 2 LeSS 5

### LeSS Structure

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 139

### LeSS Product

- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

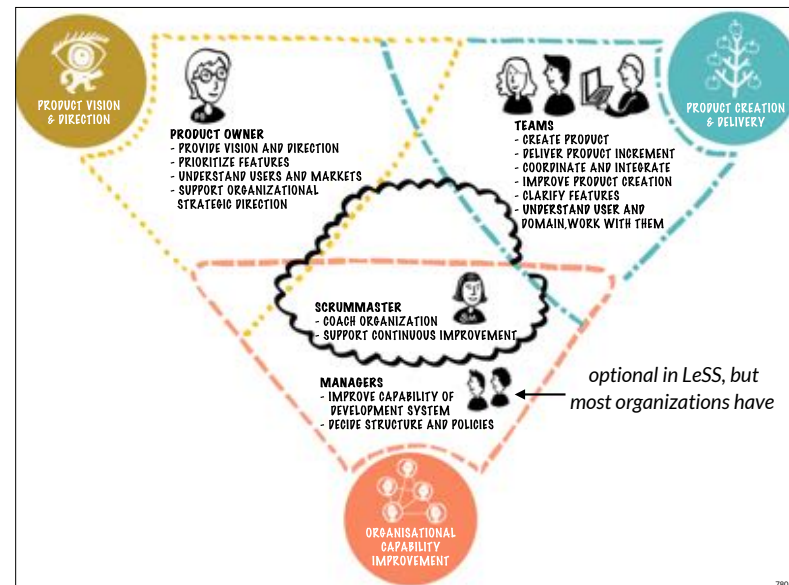
### LeSS Sprint

- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

778

778

# Scrum Master in LeSS



780

780

779



## LeSS Rule(s)

Scrum Masters are responsible for a well-working LeSS adoption. Their focus is towards the Teams, Product Owner, organization, and development practices.

A Scrum Master doesn't only focus on a team but on the overall organizational system.

A Scrum Master is a dedicated full-time role. One ScrumMaster can serve 1-3 teams.

781

781



team: standing

- > What may happen if the big group is moving to LeSS and...
- > “a Scrum Master is only allowed to serve 1 Team”
- > or, “a Scrum Master can serve 6 Teams”

782

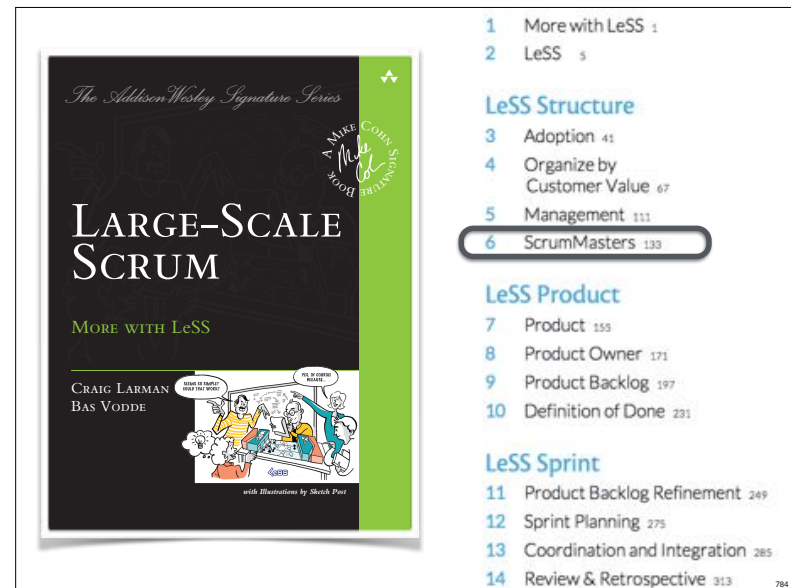
782

## Guides in LeSS: Scrum Masters

- > Scrum Master Focus
- > Five Scrum Master Tools
- > Large-Group Facilitation
- > Promote Learning & Multiple Skills
- > Community Work
- > Scrum Master Surviving Guide
- > Scrum Master Reading List
- > Especially Pay Attention To...
- > Avoid Requirement Area Silos

783

783



784

784



# Managers in LeSS

785



teams

> traditional activities/  
responsibilities of managers  
(program, project, functional,  
component, resource, team, ...)

coach: discuss

786

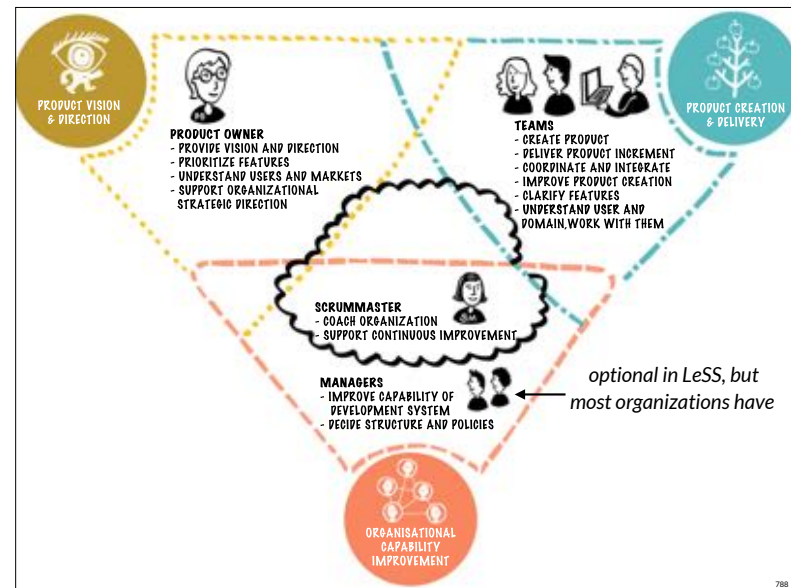
786

## Manager Responsibilities

- > the “other” column tasks
- > corporate admin tasks, etc.

787

787



788

788

## Manager Responsibilities

- > corporate admin tasks, etc.
- > **customer-value-delivery capability of org system**

789

789

do you recall? ...

**imperfect**  
product definition

**imperfect**  
feature teams

790

790

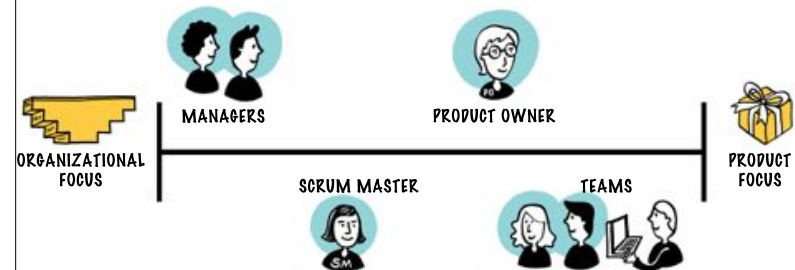
## Manager Responsibilities

- > corporate admin tasks, etc.
- > customer-value-delivery capability of org system
- > **expanding product definition**
- > **expanding feature teams**

791

791

## Focus of Different Roles



792

792

### LeSS Rule(s)

In LeSS, managers are optional, but if managers do exist their role is likely to change. Their focus shifts from managing the day-to-day product work to **improving the value-delivering capability of the product development system.**

793

### LeSS Rule(s)

Managers' role is to improve the product development system by practicing Go See, encouraging Stop & Fix, and "experiments over conformance".

794

**Guide:** Go See at *Gemba*



795

senior managers  
**manage by means**

senior managers  
manage by results

796



coach & group

- > meaning of “senior managers manage by means” (vs ... by results)?

797

797



teams

- > make 2 lists:
  - > Theory-of-mind **X** assumptions & behaviors
  - > Theory-of-mind **Y** assumptions & behaviors

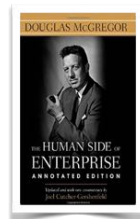
coach: discuss

798

798

**Guide:** Theory (of mind) Y Management

Agile Principle 5: ...Give them the environment and support they need, and trust them to get the job done.



799

799



class

- > scenario:
  - > some manager says, “let’s measure each team’s velocity”
  - > the result is, “TeamRed has a higher velocity than TeamBlue”
- > what dysfunctions arise?

800

800

## Guide: LeSS Metrics with Less Targets

### > who

> metrics created by the **Teams** themselves, or **Product Owner**

### > purpose

> to learn & improve

801

801

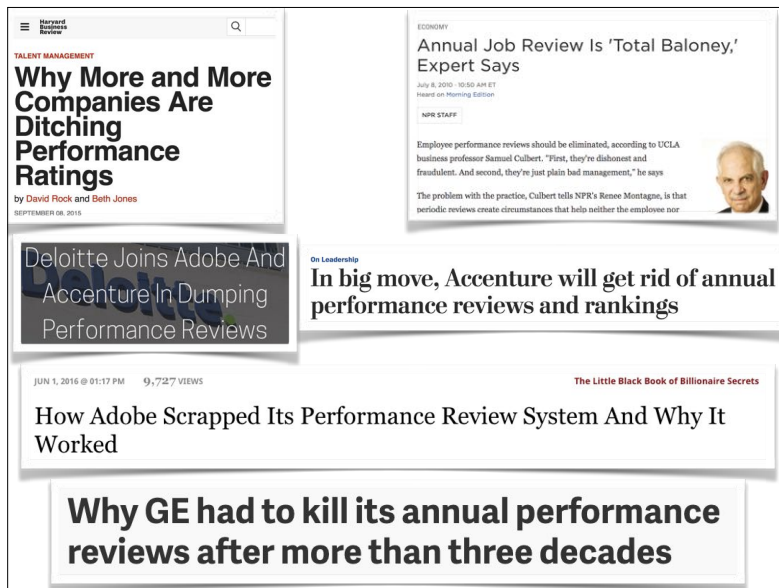
## LeSS Metrics: Don't...

> ...let anyone other than team members or the Product Owner create metrics

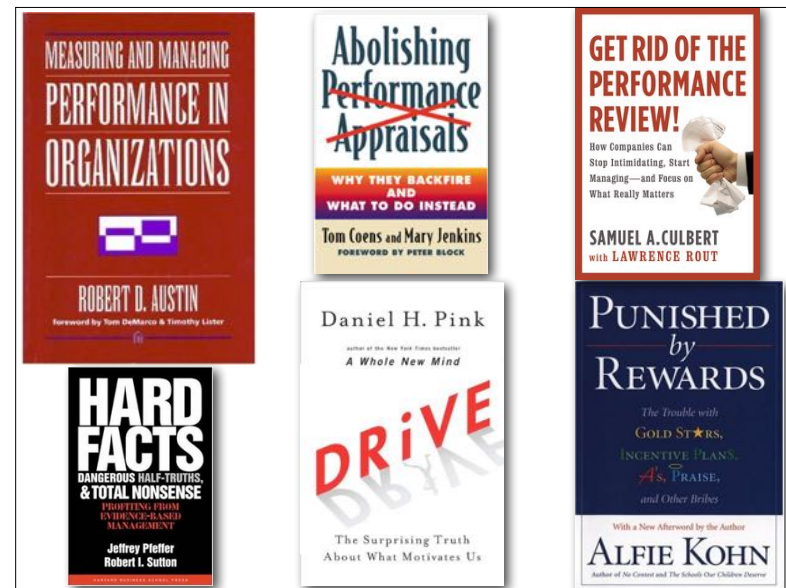
> ...measure for comparing teams or people

802

802

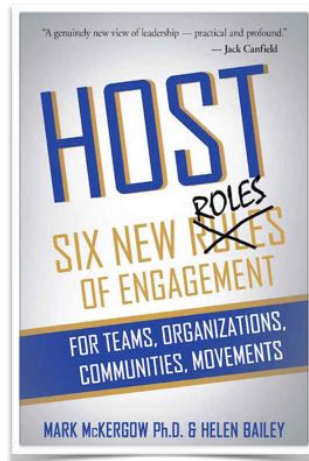


803

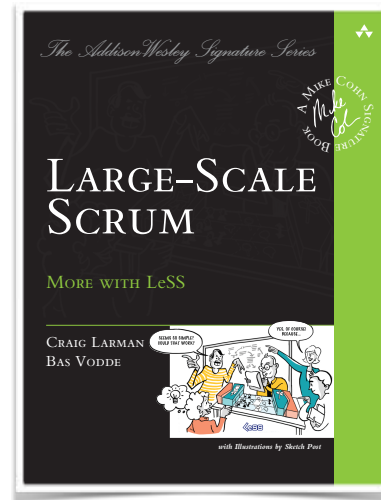


804

## Metaphor: **Host** (... manager)



805



- 1 More with LeSS 1
- 2 LeSS 5

### LeSS Structure

- 3 Adoption 41
- 4 Organize by Customer Value 67
- 5 Management 111
- 6 ScrumMasters 139

### LeSS Product

- 7 Product 155
- 8 Product Owner 171
- 9 Product Backlog 197
- 10 Definition of Done 231

### LeSS Sprint

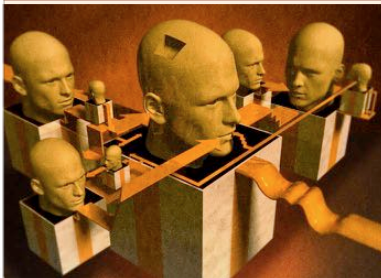
- 11 Product Backlog Refinement 249
- 12 Sprint Planning 275
- 13 Coordination and Integration 285
- 14 Review & Retrospective 313

806

## Scaling Lean & Agile Development

Thinking and Organizational Tools for Large-Scale Scrum

Craig Larman  
Bas Vodde



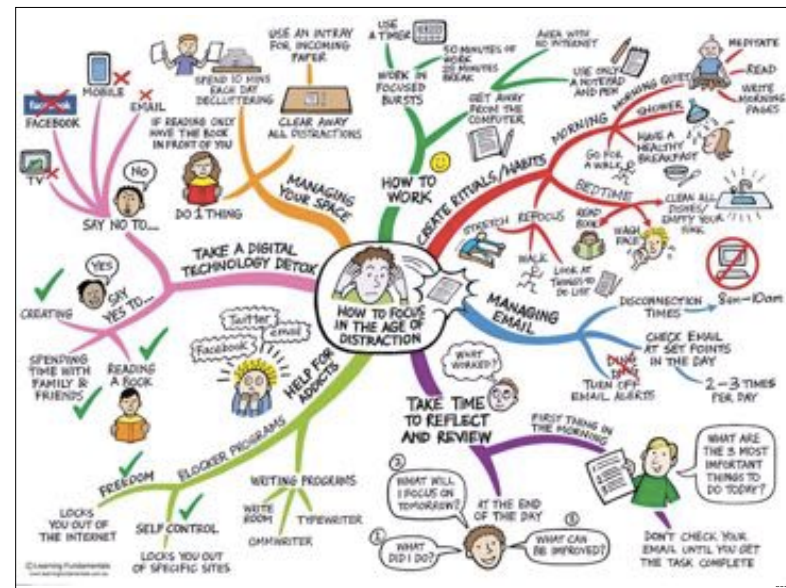
## Thinking Tools

2. Systems Thinking
3. Lean Thinking
4. Queueing Theory
5. False Dichotomies
6. Be Agile

## Organizational Tools

7. Feature Teams
8. Teams
9. Requirement Areas
10. Organization
11. Large-Scale Scrum

807



808



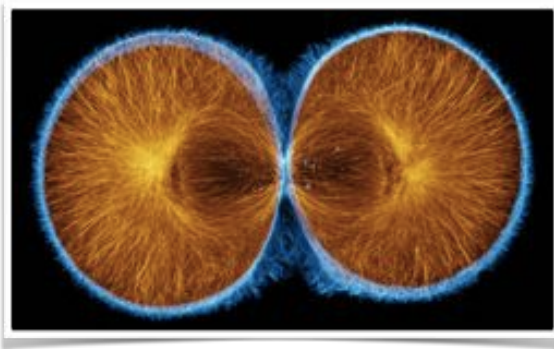
## LeSS Artifacts

809

## Product Backlog & Tools

810

### Guide: Dealing with Parents



811

811

### Dealing with Parents: **Ancestor** Attribute

Order	Item	noteworthy direct/indirect ancestor?
1	settle a buy	settle a trade
2	X	
3	Y	
4	settle a sell	settle a trade

812

812

## Guide: Handling Special Items

- > defects
- > improvements
- > innovation or unusual study

813

813

## Guide: Tools for Large Product Backlogs

- > tools aren't agile; agility is an organizational behavior
- > what Product Backlog tool at scale?
  - > nothing more complicated than a **spreadsheet** and **wiki**
  - > why? ...

814

814

## Why not “Agile Tools”? (1)

- > Focus is on tools rather than the deep systemic problems...
  - > ... and that diverts or avoids focusing on what's important: changing behavior and the system. These tools don't solve the real problems.
- > These tools contain and promote reporting features, reinforcing traditional management-reporting and control behaviors.
- > They convey a facade of improvement or agile adoption, when nothing meaningful has changed; “agile” tools have nothing to do with being agile.

815

815

## Why not “Agile Tools”? (2)

- > They often impose inflexible terminology and workflows to the teams, taking away process ownership and restricting improvement.
- > The Product Backlog is often hidden for most people as access requires an expensive account.
- > These tools enable complexifying rather than simplifying.

816

816



don't use same tool for  
Product Backlog and  
Sprint Backlogs

817

### Guide: Area Backlog

Item	Requirement Area	...
B	market on-boarding	
F	market on-boarding	

Item	Requirement Area	...
B	market on-boarding	
C	trade processing	
D	asset servicing	
F	market on-boarding	
...		

818

817

818

### Area Backlogs via **Views**

Item	Requirement Area	...
B	market on-boarding	
F	market on-boarding	

Item	Requirement Area	...
B	market on-boarding	
C	trade processing	
D	asset servicing	
F	market on-boarding	
...		

a VIEW for one  
requirement area

it is NOT a  
separate artifact

819

819

### Area Backlogs via **Separate Artifacts**

Item	Requirement Area	...
B	market on-boarding	
F	market on-boarding	

Item	Requirement Area	...
B	market on-boarding	
C	trade processing	
D	asset servicing	
F	market on-boarding	
...		

an ARTIFACT for one  
requirement area

it IS a separate  
artifact

820

820

# Area Backlogs as **views** vs Area Backlogs as **separate artifacts**

821

821

## splitting items within a Requirement Area...

822

822

### Splitting Case 1: Minor Discrepancy

#### Overall PB

Item	Area
B	market onboarding
C	trade processing
D	market onboarding

#### Market Onboard- ing Area Backlog

Item	Ancestor
B-1	B
D	
B-2	B

823

823

### Splitting Case 2: Major Discrepancy

Item	Area
B	market onboarding
C	trade processing
D	market onboarding
E	market onboarding
F	market onboarding
...	

Item	Ancestor
B-1	B
B-2	B
D	
E	
F	
B-3	B
B-4	B

824

824

## Splitting Case 2: **Un-splitting** (merging)

Overall PB		Market Onboard-ing Area Backlog	
Item	Area	Item	Ancestor
BX (general-ization of B1, B2)	market onboarding	BX-1 (old B-1)	BX
C	trade processing	BX-2 (old B-2)	BX
D	market onboarding	D	
E	market onboarding	E	
F	market onboarding	F	
BY (general-ization of B3, B4)	market onboarding	BY-1 (old B-3)	BY
		BY 2 (old B4)	BY

825

825

## **Guide:** Three Levels Max

Overall PB

Item	Ancestor	Area
XA	X	trade processing
XB	X	trade processing
...		

2 “levels”

now, 3 levels

the Ancestor column  
also links the the  
Area & Overall PB

Trade Processing Area Backlog

Item	Ancestor
XA-1	XA
XA-2	XA
...	

826

826

prioritization of  
large backlog?  
(see PBR module)

827

827

**Guide:** Don't “Manage Dependencies between Products”  
but Minimize Constraints

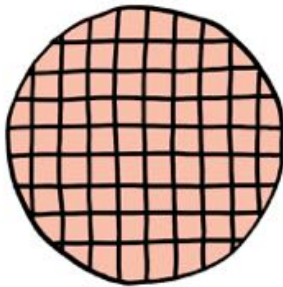
- > Do “their part”
- > Pair-work “their part”
- > Simplify or split item-A so that the other group's change is small
- > Split item-A into (1) item with a stub, and (2) fully integrate item
- > Split item-A into (1) item using an alternative interface, and (2) item using the final interface
- > Explain the constraint
- > Bypass the constraint
- > Achieve the outcome a different way

828

828

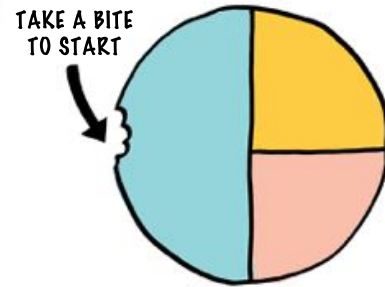
## Guide: Take a Bite

TRADITIONAL SPLITTING  
OF BIG FEATURE



ALL-AT-ONCE IN EQUAL  
PIECES AT THE BEGINNING

LESS SPLITTING  
OF BIG FEATURE



PARTIAL SPLITTING  
AND TAKING A BITE

829

Style 1

->

Style 2

830

what is “Style 2”?

831

## Topics Coach **Will** Start With in Part 2

### > Adoption Story & Adoption

### > Coordination & Integration (architecture, sharing tasks, communities, learning, ...)

### > Why LeSS?

### > Preparing for Sprint 1

### > Product Backlog Refinement

### > Sprint Planning

### > Technical Excellence

### > Sprint Review

### > Retrospectives

### > Done & Undone

### > DevOps

### > LeSS Huge

### > Feature-Team Adoption Maps (common in *incremental* LeSS Huge adoptions)

### > LeSS Rules

### > LeSS Principles

### > Product Owner

### > Managers

### > Scrum Masters

### > Product Backlog & Tools

832



pair/triplet

- > read following slide for ideas, & write new topics/questions, 1 per paper

833

833

## Sample Topics in the Course Material

- > Why LeSS?
- > Preparing for Sprint 1
- > PBR (*Splitting*, ...)
- > Sprint Planning
- > Technical Excellence
- > Sprint Review
- > Retrospectives
- > Done & Undone
- > DevOps
- > LeSS Huge
- > Feature-Team Adoption Maps (common in *incremental* LeSS Huge adoptions)
- > LeSS Rules
- > LeSS Principles
- > Product Owner
- > Managers
- > Scrum Masters
- > Product Backlog & Tools

834

834



teams

- > retrieve any questions already on the wall

coach

- > organize the topic/questions priorities with the group

835

835

# Closing

836

## Likely Objectives: You can...

- > redesign org from **local optimizations** to **global system optimizations**
- > **define a product** broadly
- > motivate & define **LeSS org design** (structure, roles, policies, ...)
- > advise on **LeSS adoption**
- > know & coach **LeSS Sprint** (events, coordination, ...)
- > explain LeSS & LeSS Huge **frameworks**
- > explain **LeSS principles** & make connections
- > answer "**why LeSS?**"
- > explain **roles**

837

837

## Certified LeSS Practitioner

- > i will register you at **less.works**
- > you can change your email address at any time

838

838

## Your Account @ less.works

- flipchart/whiteboard/wall photos
- course notes pdf
- contacts
- certificate
- class photo

839

839

## Connections

LeSS Site: <http://less.works>

LeSS Twitter: [#LeSSWorks](#)

LinkedIn Group: [LeSS - Large-Scale Scrum](#)

LinkedIn Group: [Certified LeSS Practitioner](#)

Slack: <http://less-works.slack.com/>

LeSS Discussion Group:  
<http://groups.google.com/forum/#!forum/largescalescrum>

LeSS on Facebook  
<https://www.facebook.com/less.works>

840

840



#LeSSWorks  
@less\_works  
@lesscraiglarman

841

841

share!

blog!

tweet!

spread the word!

842

842



amazon

**Amazon** reviews of  
new book are  
appreciated ;)

843

843



team: round-robin: standing

> “**how do i feel?**...”

> please **sit** when team is done

844

844

last words

845

845

individual  
> feedback

846

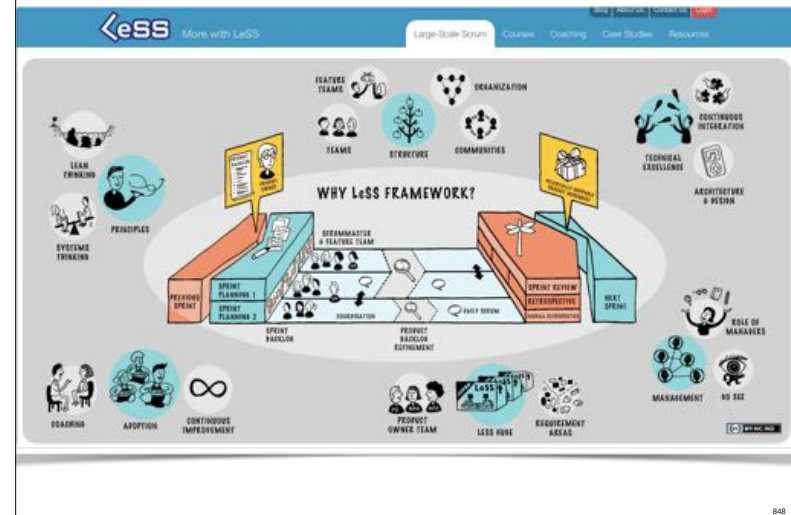
846

class: photo

847

847

less.works



848

848