Adapting Methodologies, Crystal Methodologies, Lean & Kanban + WS4

Robert Feldt

Agile Dev Processes Course, 2012-05-14

Division of Software Engineering

HOSE Lab (Human-fOcused SE)

System Criticality

Judge the "Loss due to impact of defects"

Level	Acronym	Def
4	**L**ife	Loss of life (many, single, degrees of damage to limb)
3	**E**ssential value	Loss of value/money which is hard/impossible to replace
2	**D**iscretionary value	Loss of value/money which can be replaced (but is setback)
1	**C**omfort	Loss of comfort/choice

Precision

"How much you care to say about a topic"

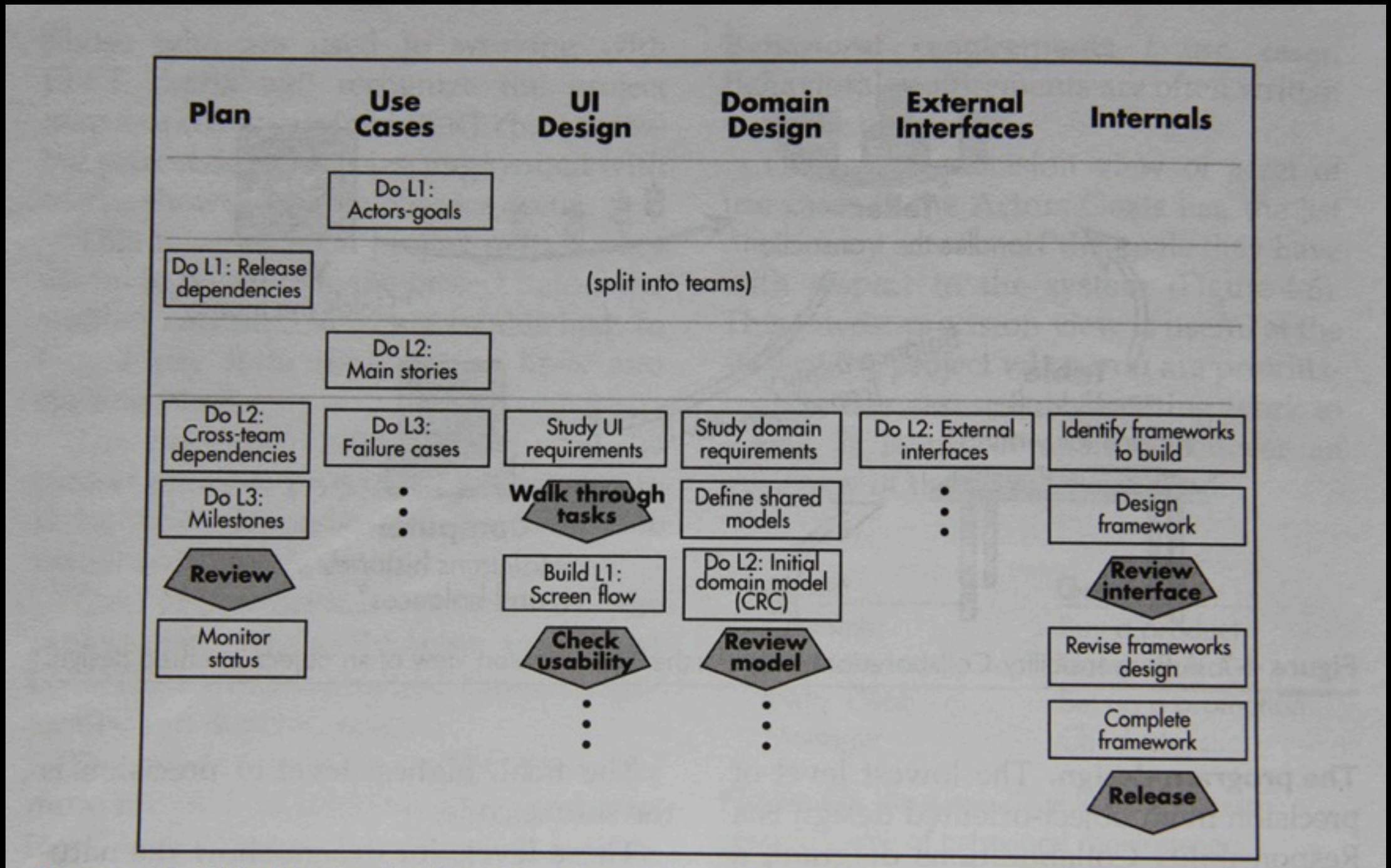


Figure 4-8 Using low levels of precision to trigger other activities.

Tolerance

"How much variation is permitted"

Ceremony

"Amount of precision and tightness of tolerance"

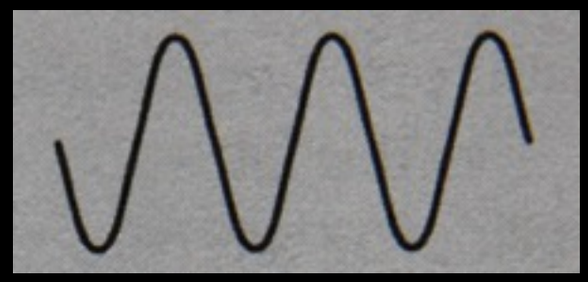
Methodology Size

"Number of elements of control of methodology"

Methodology Weight

*"Methodology Size * Ceremony"*

Stability



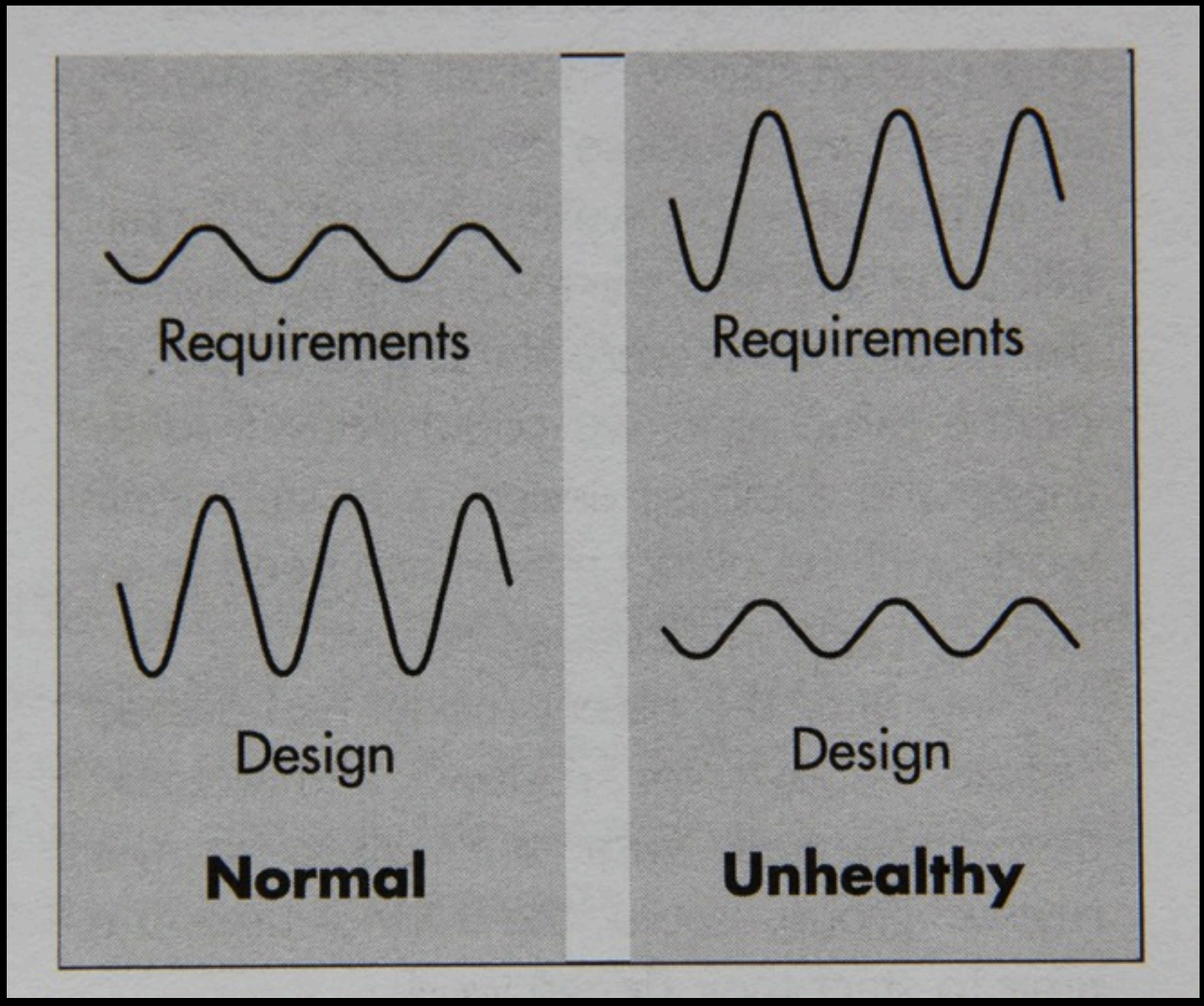
"Likelihood that things will change"

Q: If I ask this question today and in X weeks, how likely I get the same answer?

Level	Desc	Answer
3	**R**elatively stable	Loss of value/money which is hard/ impossible to replace
2	**V**arying	Loss of value/money which can be replaced (but is setback)
1	**W**ildly Fluctuating	Loss of comfort/choice

Stability

"Upstream activities (more stable than) downstream ones"



Serial development

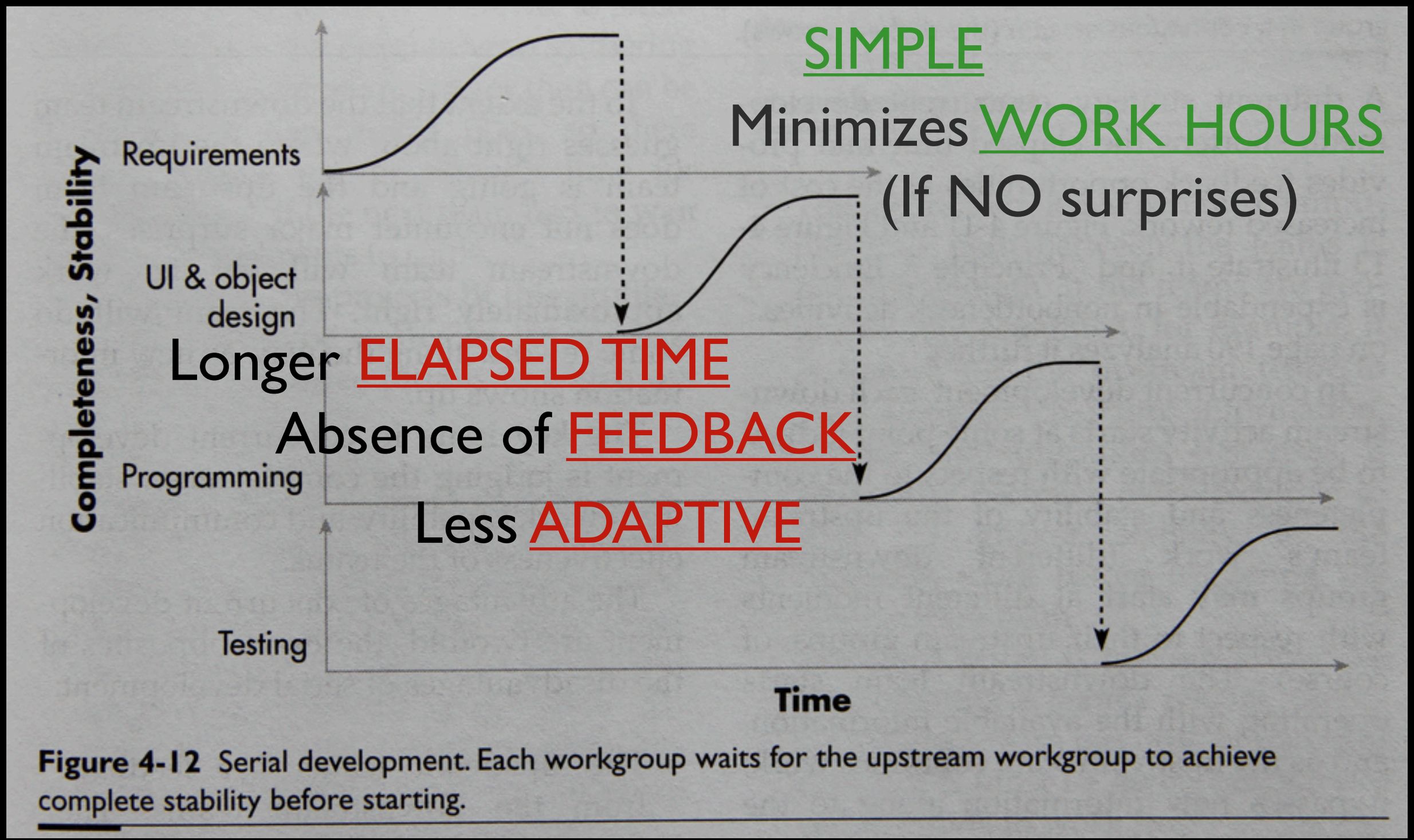
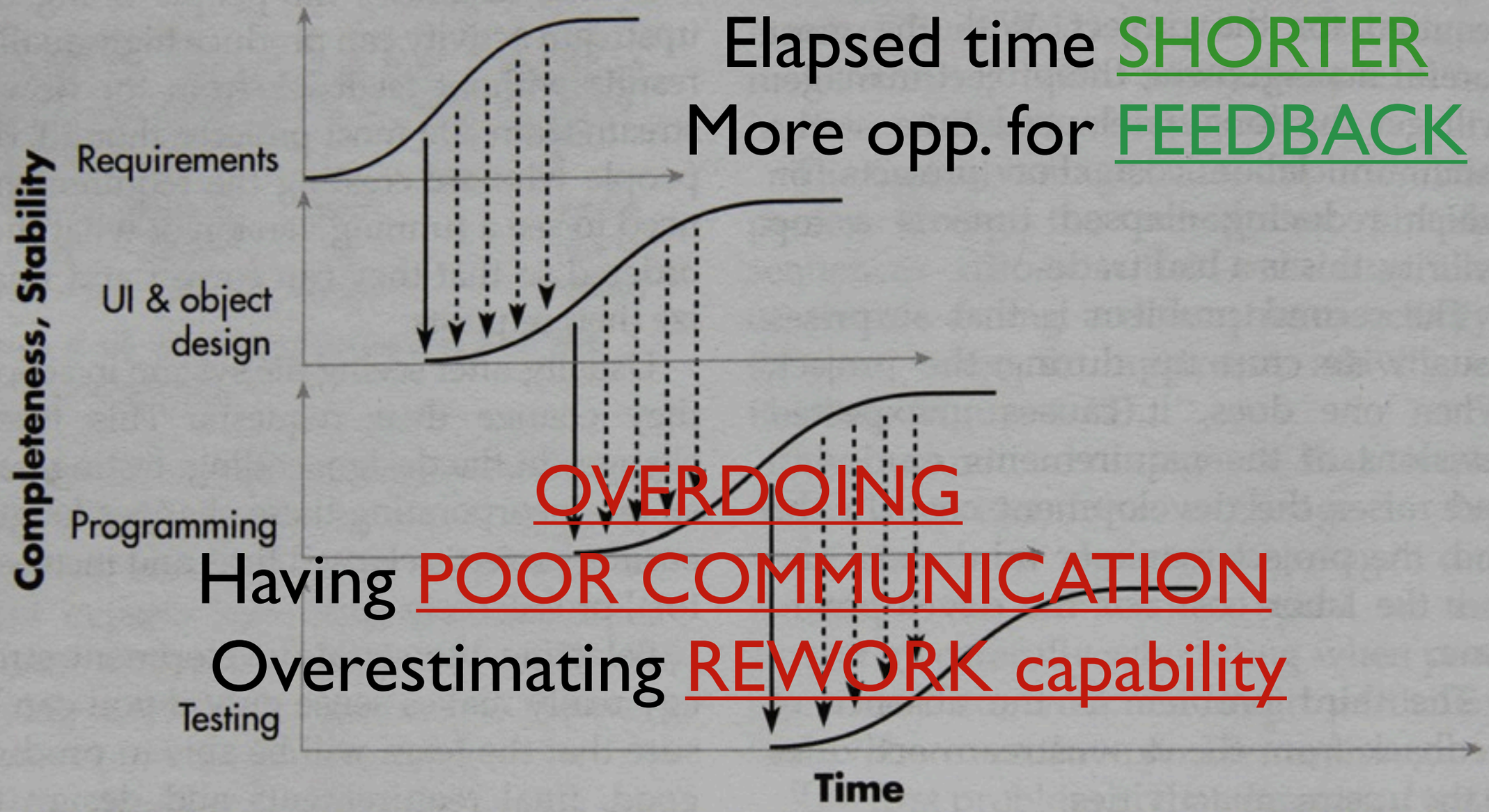


Figure 4-12 Serial development. Each workgroup waits for the upstream workgroup to achieve complete stability before starting.

Concurrent development



Elapsed time SHORTER
More opp. for FEEDBACK

OVERDOING
Having POOR COMMUNICATION
Overestimating REWORK capability

Figure 4-13 Concurrent development. Each group starts as early as its communications and rework capabilities indicate. As it progresses, the upstream group passes update information to the downstream group in a continuous stream (the dashed arrows).

Methodology Design Principles

- 1. Interactive face-2-face comm. is cheapest & fastest comm. channel
- 2. Excess methodology weight is costly
- 3. Larger teams need heavier methodologies
- 4. Greater system criticality => greater ceremony
- 5. Increasing feedback & comm. reduces need to intermediates
- 6. Discipline/Skill/Understanding counter Process/Formality/Docs
- 7. Efficiency is expendable in non-bottleneck activities

Weight-is-costly & Larger-needs-heavier

What size problem can a given number of people attack, using various methodology weights?

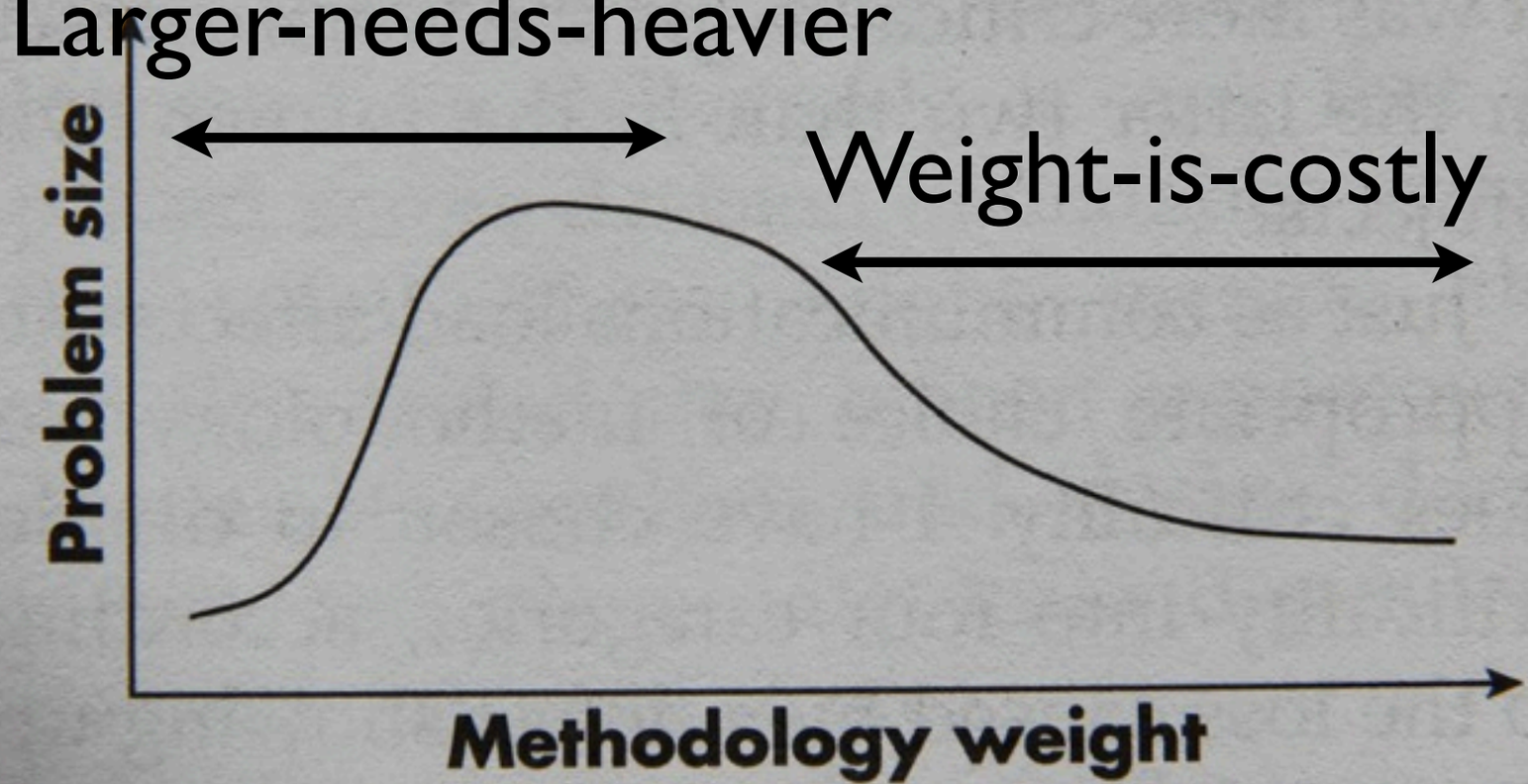


Figure 4-18 Effect of adding methodology weight to a large team.

Different methodologies for different projects

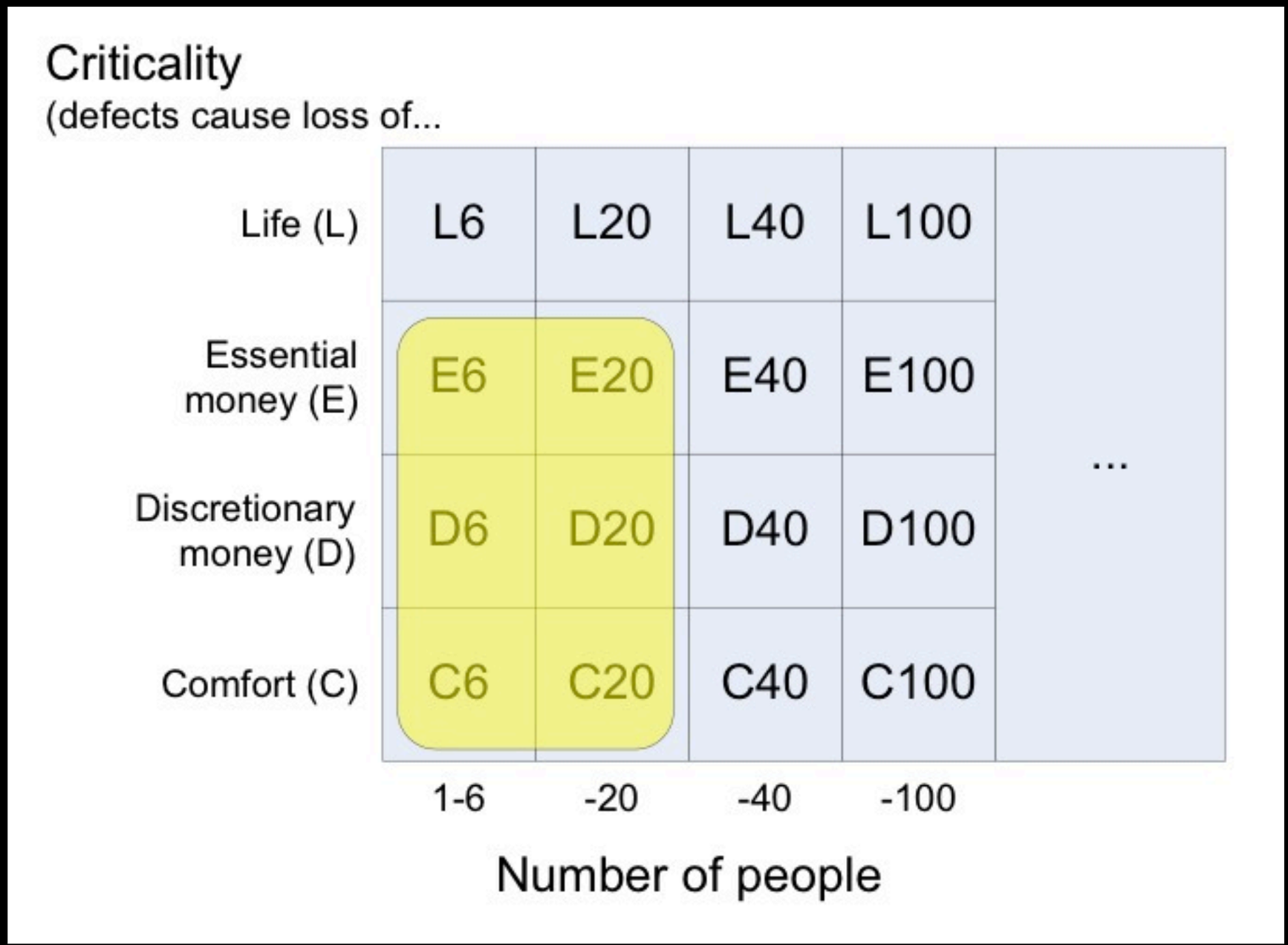
Criticality
(defects cause loss of...

Life (L)	L6	L20	L40	L100	
Essential money (E)	E6	E20	E40	E100	
Discretionary money (D)	D6	D20	D40	D100	...
Comfort (C)	C6	C20	C40	C100	
	1-6	-20	-40	-100	

Number of people

"Cockburn Diagrams"

XP Applicability



Different methodologies for different projects

Crystal = Frequent delivery, Close communication & Reflective improvement

Criticality (defects cause loss of...)					
Life (L)	L6	L20	L40	L100	
Essential money (E)	E6	E20	E40	E100	
Discretionary money (D)	D6	D20	D40	D100	...
Comfort (C)	C6	C20	C40	C100	
	1-6	-20	-40	-100	

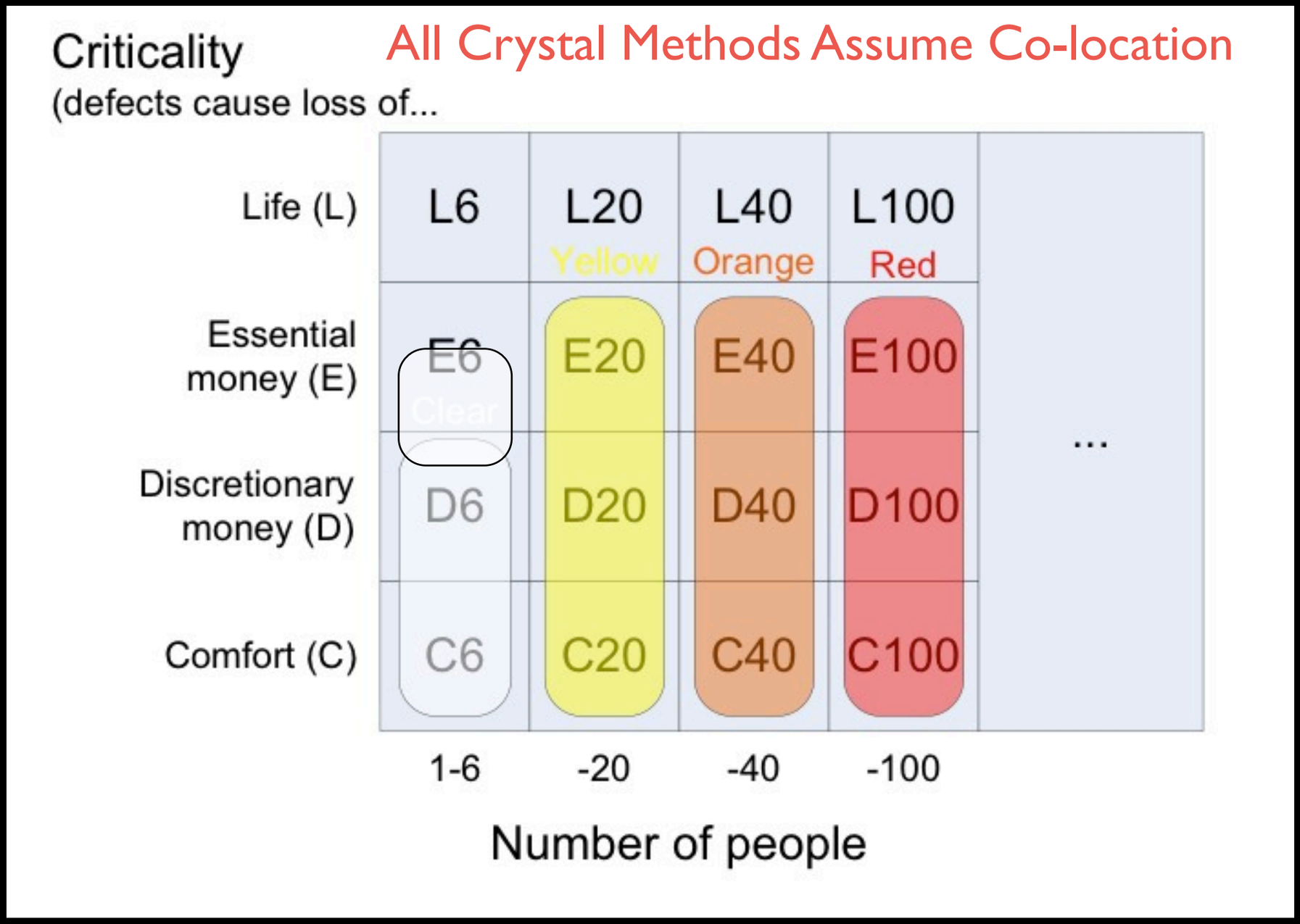
Number of people

"Cockburn Diagrams"

Crystal Applicability

"Family of methodologies" =
concrete examples to be tuned

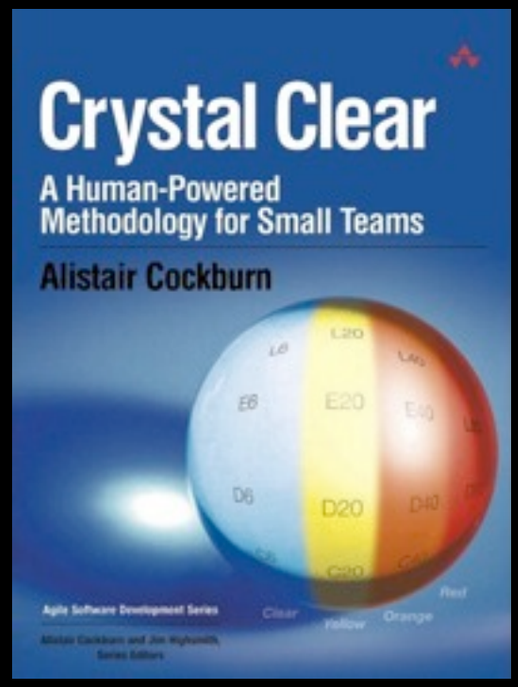
Not Kit Toolbox!



Core of Crystal

- Two values:
  - People- & communication-centric (tools, artefacts, processes supports the humans)
  - Highly tolerant (to varying human cultures and choices)
- Two rules:
  - Incremental dev, 1-3 (max 4) months increments
  - pre- and post-increment reflection workshops, possibly mid-increment also
- Two base techniques:
  - Methodology-tuning via interviews & team workshop
  - Reflection workshops

Crystal Clear



- D6 (E8 with more comm., D10 with more testing)
- 4 Roles: Sponsor, Senior des/prog, Des/Prog, User
- One team, all seated in one office or adjacent offices
- Incremental, regular SW delivery every 2-3 months
- Progress tracked as delivery or major decision, not docs
- Automated regression testing
- Direct user involvement
- Two user "viewings" per release
- Downstream starts when upstream "stable enough to review"
- Product- and methodology-tuning workshops in start and middle

Crystal Clear Artifacts

- Release sequence
- Schedule of user viewings and deliveries
- Annotated use cases or features descriptions
- Design sketches and notes as needed
- Screen drafts
- A common object model
- Running code
- Test cases
- User manual
- Possibly: Templates for artefacts, code & UI & testing standards

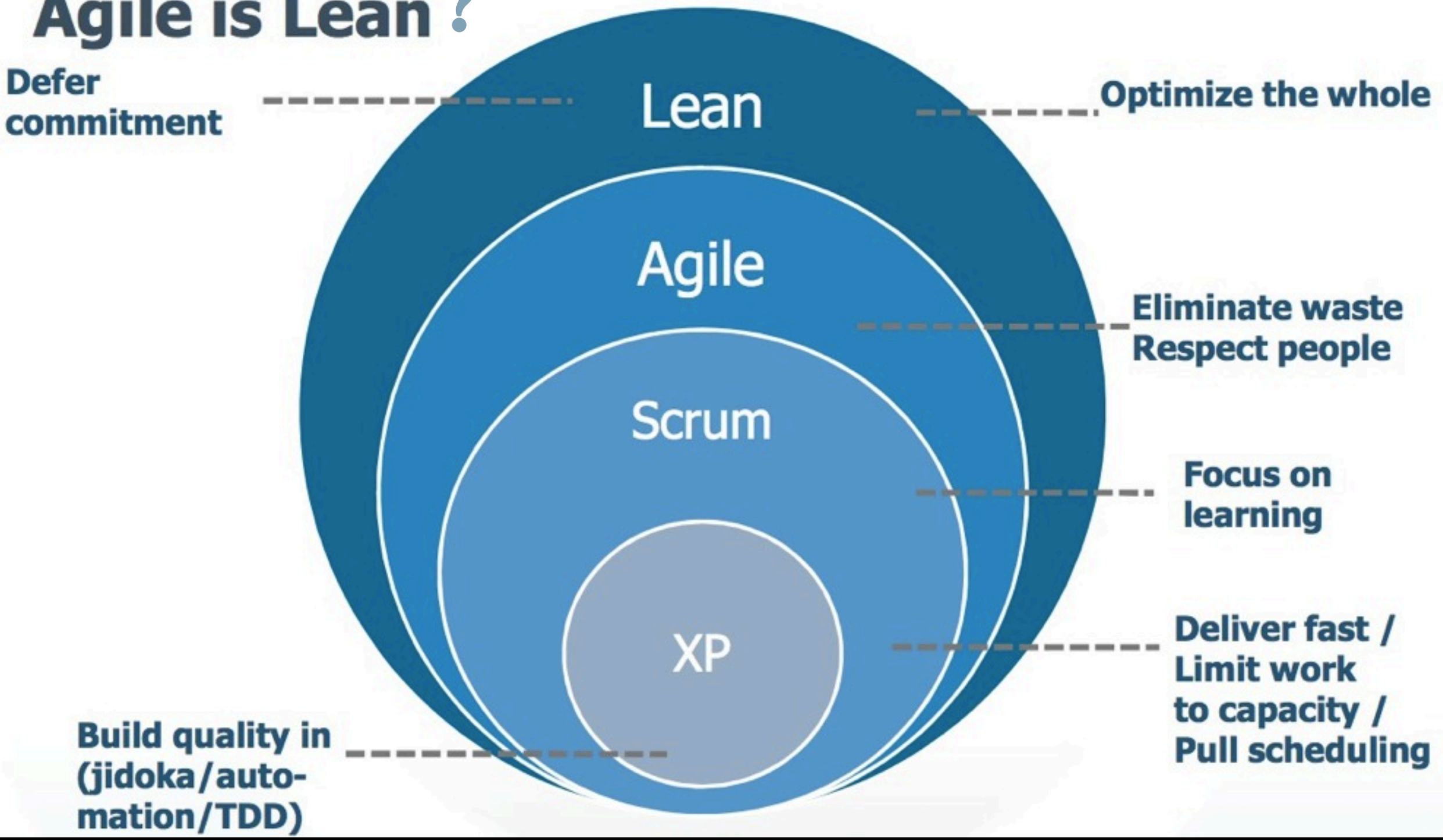
Crystal Orange

- D40 (10 to 40 persons), 1-2 years
- “Medium-sized production project in industrial setting”, not life-critical
- Roles: Sponsor, Business expert, Usage expert, Technical facilitator, Business analyst/des, Project manager, Architect, Design mentor, Lead Des/Prog, Des/Prog, UI designer, Reuse resp, Writer, Tester
- Cross-functional groups:
  - Reduce deliverables, Enhance communication
  - Business analyst/designer, UI designer, 1-2 Des/Prog, possibly tech/db expert and tester depending on group
  - System planning, Project monitoring, Arch, Tech, Functions, Infrastructure, External test

Crystal Orange

- Added artefacts compared to Crystal Clear:
  - Requirements doc
  - Status reports
  - UI design doc
  - Inter-team specs
- Standards/Policies same as Crystal Clear
  - Incremental delivery may be extended to 3-4 months
- Too heavy for 10 people, Light for 40 people

Agile is Lean ?



[Kniberg2008]

Kanban

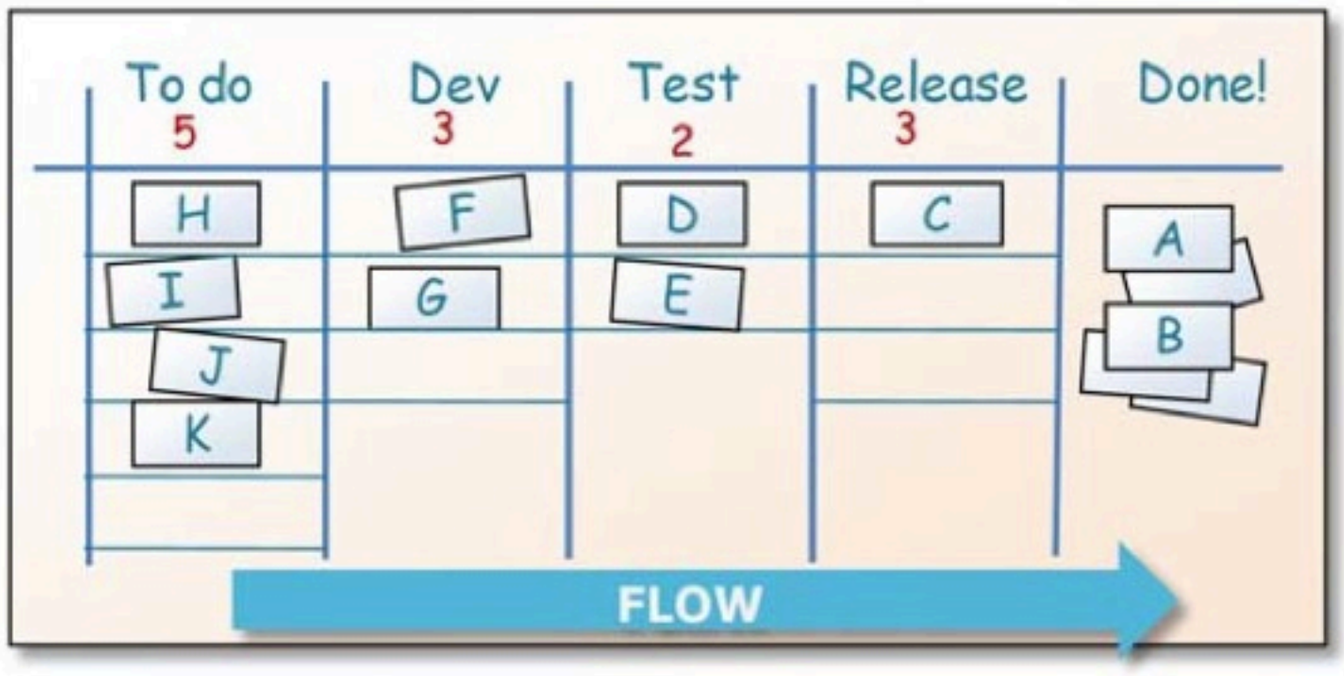
- In Japanese the word Kan means "signal" and "ban" means "card" or “board”.
- A Kanban card is a signal that is supposed to trigger action.
- Therefore Kanban refers to "signal cards".

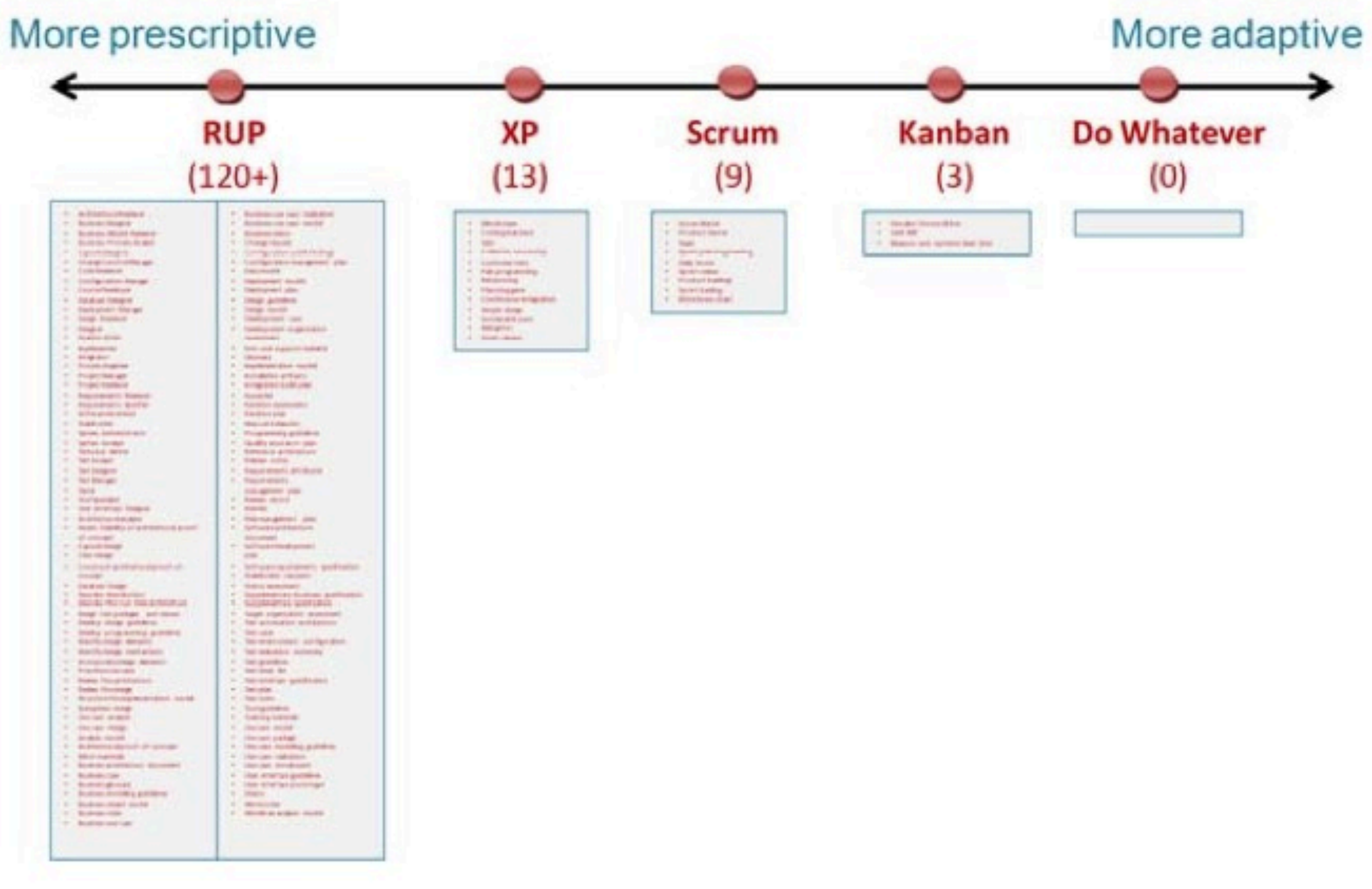
Kanban

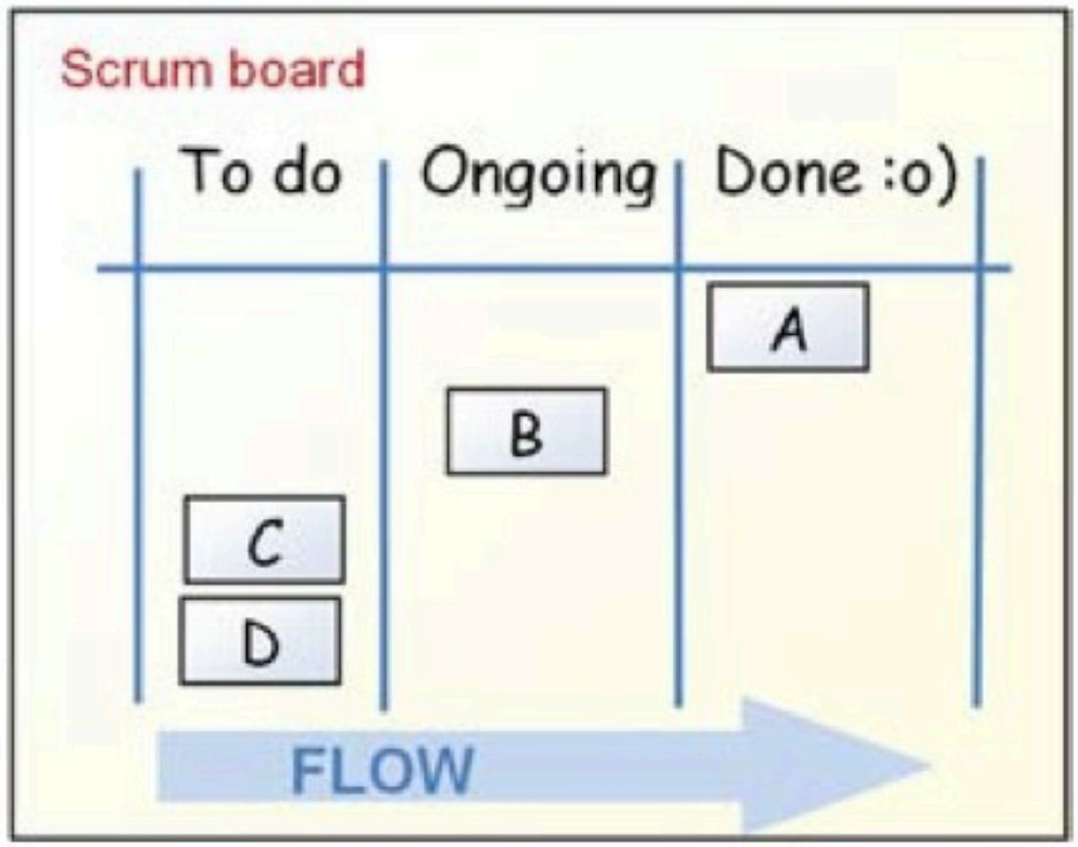
- The basic principles of Kanban for software engineering:
  - Limit Work in Process (WIP)
  - Pull value through (with WIP limit)
  - Make it visible (Visual Control)
  - Increase throughput
  - Fixed Kanban Backlog
  - Quality is embedded in (not inspected in)
- The team continuously monitor the above to improve

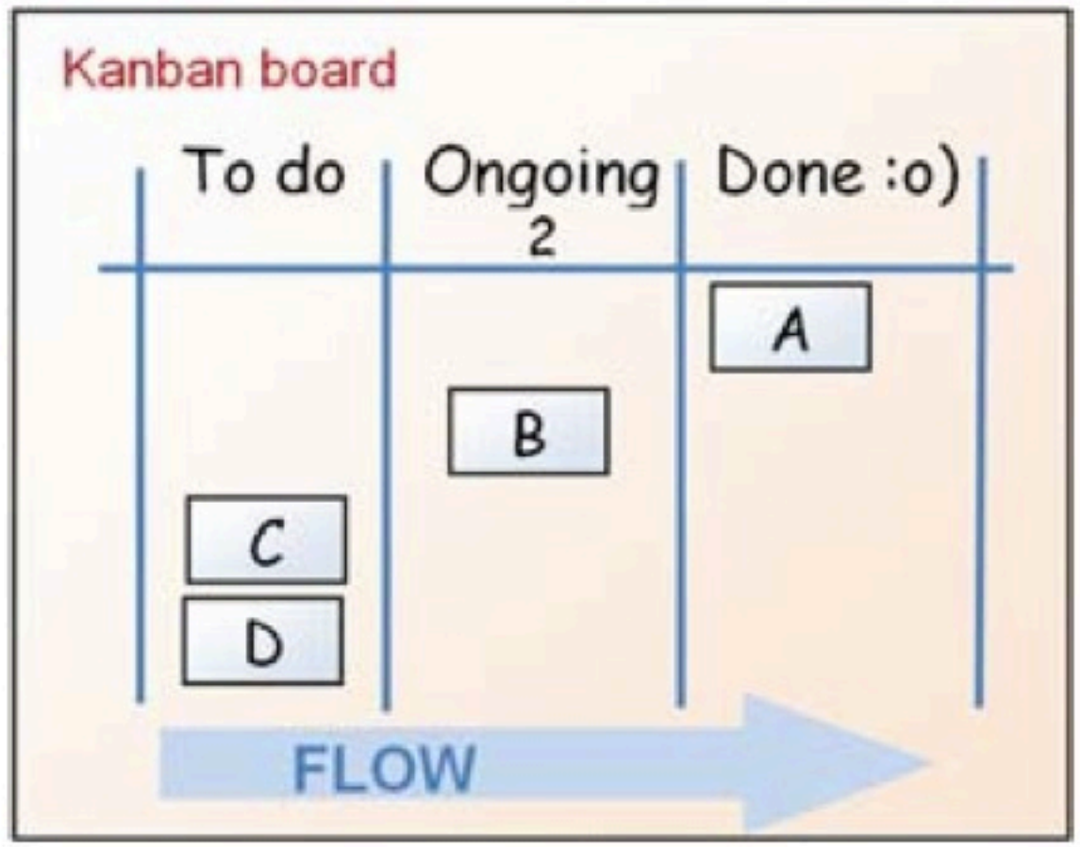
Kanban in a nutshell

- **Visualize the workflow**
    - Split the work into pieces, write each item on a card and put on the wall.
    - Use named columns to illustrate where each item is in the workflow.
- **Limit Work In Progress (WIP)** – assign explicit limits to how many items may be in progress at each workflow state.
- **Measure the lead time** (average time to complete one item, sometimes called "cycle time"), optimize the process to make lead time as small and predictable as possible.




More prescriptive
More adaptive
RUP
(120+)
XP
(13)
Scrum
(9)
Kanban
(3)
Do Whatever
(0)



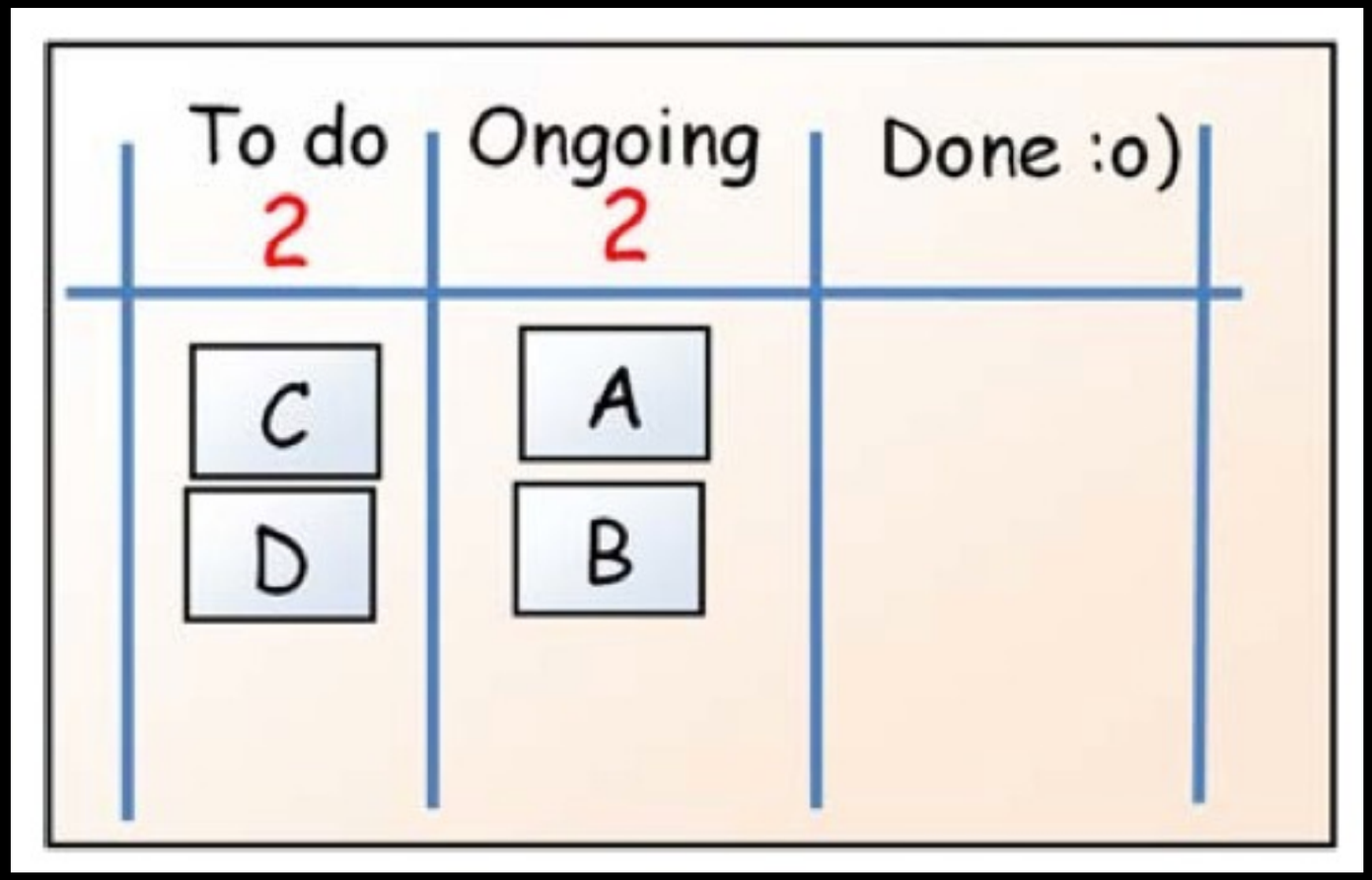


Scrum limits WIP indirectly (via timeboxed iteration, i.e. limit per time unit)

Kanban limits WIP directly (limit per workflow state)

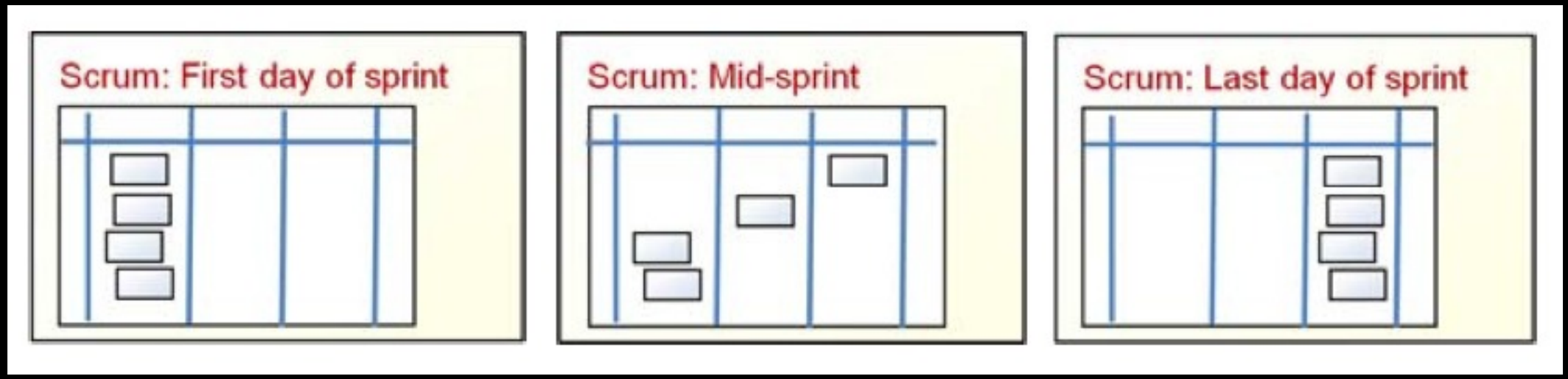
Scrum board related to team, Kanban board related to workflow

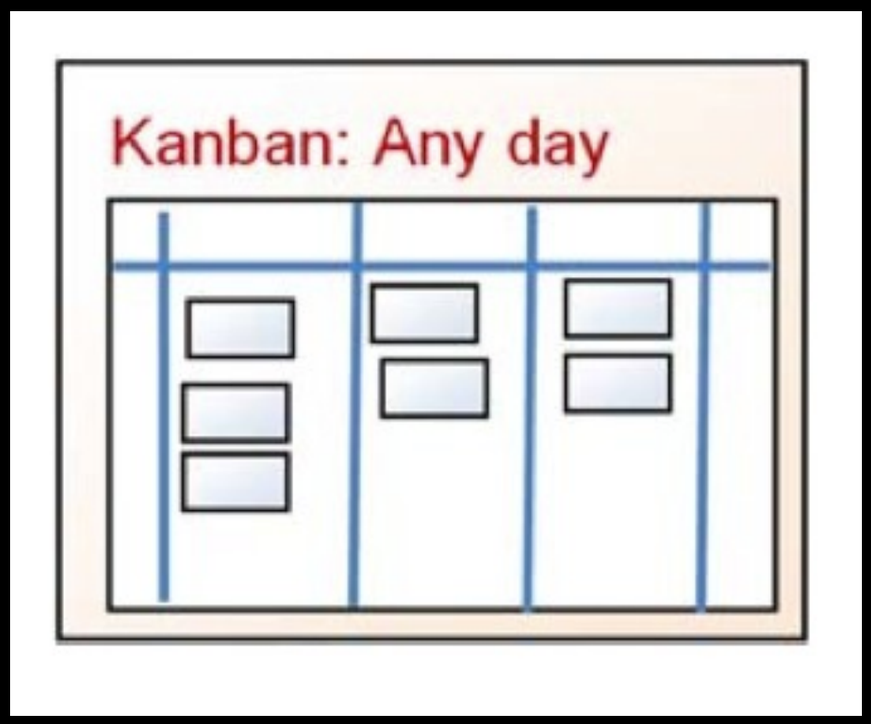
Scrum resists change within iteration



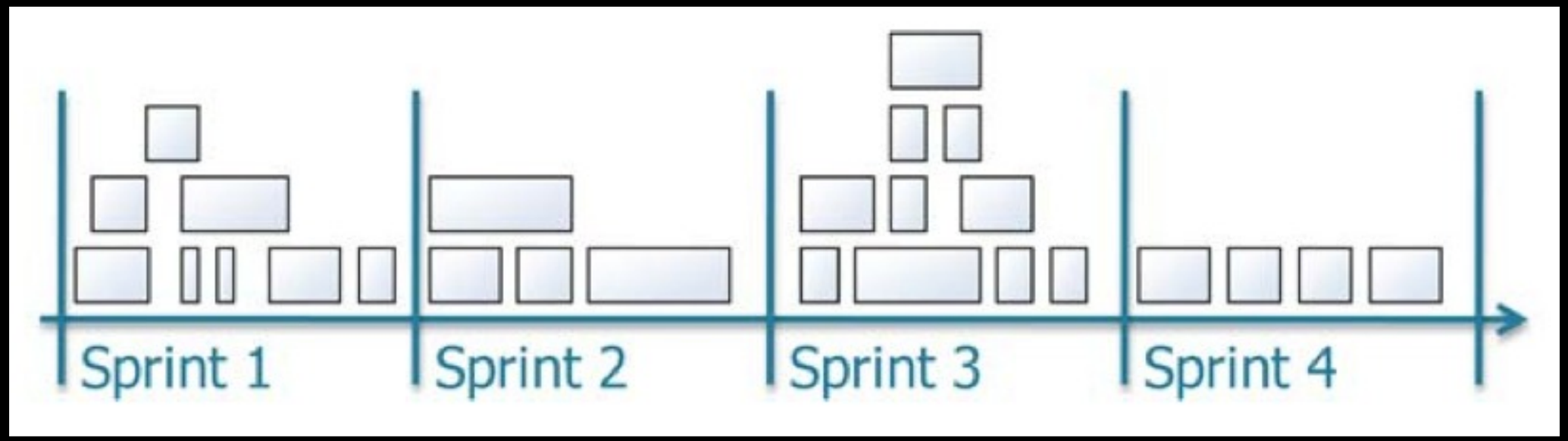
- A-D are in iteration / being processed. User turns up with E.
  - Scrum: "You are welcome with E in next iteration."
  - Kanban: "Feel free to add E to ToDo if you take away C or D"

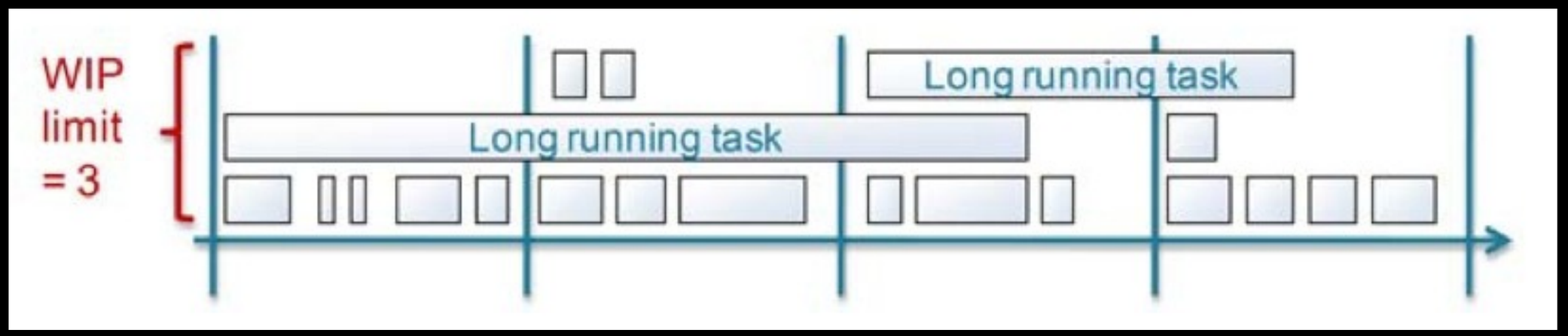
Scrum resets board between iterations, Kanban does not





Scrum items must fit in iteration





Kanban items can be long-running

Kanban Stand-ups

- Focus on WIP not on people. Enumerate items in flow, not people
- => can handle more people/larger teams
- Board shows status, meeting focus on exceptions
- Traverse board from right to left, emphasizing pull
- Standup questions:
  - Do we have a bottleneck? (Congestion or gap in queues)
  - Do we have a blocker not dealt with?
  - Are we keeping our WIP limits?
  - Are priorities clear?
  - Done yesterday, planning today.

Summary of Scrum vs. Kanban

Similarities

- Both are Lean and Agile.
- Both use pull scheduling.
- Both limit WIP.
- Both use transparency to drive process improvement.
- Both focus on delivering releasable software early and often.
- Both are based on self-organizing teams.
- Both require breaking the work into pieces.
- In both, the release plan is continuously optimized based on empirical data (velocity / lead time).

Differences

Scrum	Kanban
Timeboxed iterations prescribed.	**Timeboxed iterations optional.** Can have separate cadences for planning, release, and process improvement. Can be event-driven instead of timeboxed.
Team commits to a specific amount of work for this iteration.	**Commitment optional.**
Uses **Velocity** as default metric for planning and process improvement.	Uses **Lead time** as default metric for planning and process improvement.
Cross-functional teams prescribed.	Cross-functional teams optional. **Specialist teams allowed.**
Items must be broken down so they can be completed within 1 sprint.	No particular item size is prescribed.
Burndown chart prescribed	No particular type of diagram is prescribed

Differences

Scrum	Kanban
WIP limited indirectly (per sprint)	**WIP limited directly** (per workflow state)
Estimation prescribed	**Estimation optional**
Cannot add items to ongoing iteration.	**Can add new items whenever capacity is available**
A **sprint backlog is owned by one specific team**	A **kanban board may be shared by multiple teams** or individuals
Prescribes 3 roles (PO/SM/Team)	**Doesn't prescribe any roles**
A **Scrum board is reset** between each sprint	A **kanban board is persistent**
Prescribes a prioritized product backlog	**Prioritization is optional.**

SWELL SE Census 2012

- Survey over 4 years 2009-2012
- Hot off the press! 2012 survey ended April 30, you are first to see results
- Around 150 responses per year, Industrial developers, Finance and consultancies
- Main division is on dev method used: Agile, Plan, Hybrid/Mix
- Then focus is on Requirements and Testing

Development method?

Method?	Total	2010	2011	2012
Agile	16.9%	13.7%	18.8%	18.7%
Mixed	51.8%	52.3%	49.3%	54.7%
Plan-driven	26.5%	30.7%	26.4%	22.3%
Other	4.1%	3.3%	5.6%	3.6%

Development method?

Agile dev methods are common.

A majority use either a selected/concious mix (52%) or work “purely” agile (17%). Close to a third work mainly plan driven.

No significant differences between years.

Men: definition av “agile”...?!

Plandrivet är fortfarande större än rent agilt, hybrider vanligt

Req/Test Org&Process works?

Response	Requirements			Test		
	2010	2011	2012	2010	2011	2012
Very high degree	7.8%	7.6%	5.8%	14.9%	22.9%	18.0%
High Deg.	35.3%	31.2%	39.6%	55.8%	45.8%	54.0%
Low Deg.	46.4%	42.4%	38.8%	26.6%	26.4%	24.5%
Very Low Deg.	10.5%	18.8%	15.1%	2.6%	4.9%	2.9%
Avg (1-4):	2.4	2.3	2.4	2.8	2.9	2.9

2012

	Practice	AgileAvg	MixedAvg	PlanAvg
1	Use cases/Scenarios	55	71	72
2	Natural Language Reqs	64	61	65
8	Req-specific personnel	36	54	48
4	Non-func requirements	40	48	38
7	Measurable/Testable reqs	43	35	32
5	Prototypes/Mock-ups	30	31	19
6	Modeling (UML)	17	26	22
3	Formal notation	4	6	3

Table 4: Ways of working in Requirements Engineering 2012 (Percentages)

2012

	Practice	AgileAvg	MixedAvg	PlanAvg
3	Manual testing	91	96	91
4	Textual test cases	70	90	93
7	Test-specific personnel	72	84	74
5	Exploratory testing	62	62	42
1	Automated Test Execution	62	55	35
2	Automated Test Generation	21	26	16
8	Testers also work with dev	23	11	14
6	Models in testing	11	15	9

Table 6: Ways of working in Testing 2012 (Percentages)

2012

	Practice	Avg11	Avg12	Diff	Agile12	Mixed12	Plan12
1	Sprint-based development	41	56	15	69	72	6
7	Stand-up meeting	46	52	6	73	62	6
2	Exploratory testing	33	47	14	54	59	13
6	Product/Sprint backlog	38	44	6	65	51	6
8	Daily/Continuous build	26	30	4	58	33	0
4	Test driven development	19	29	10	42	36	6
10	Small/Frequent release	26	29	3	46	33	3
3	Coding standards	12	23	11	35	28	3
5	Refactoring	10	18	8	38	18	3
9	Planning game	12	16	4	27	18	0
13	On-site customer	11	12	1	23	12	3
11	Pair programming	10	12	2	27	11	0
14	Collective code ownership	8	9	0	12	12	0
12	Sustainable pace	2	4	2	15	3	0
15	System metaphor	1	1	0	4	1	0

Table 2: Use of Agile Practices 2012 (Percentages)

Sources

- Kniberg & Skarin, “Kanban and Scrum - Making the most of both”, InfoQ free book, Crips AB
- David Joyce, “Kanban for SE”, BBC, presentation online.

Workshop 4

- Theme: Does Agile work?
- Sub-questions:
  - When does agile methodologies work?
  - Why do agile methodologies work?
  - When do agile methodologies not work?
  - Why do they fail?
  - Are agile methodologies easy to transition to? Why/why not?
- Groups of 5, Each group either positive or negative and discuss effect of of agile on one of:
  - SW Quality, Effectiveness/Time, Team spirit/Happiness, Individuals happiness, Customer interaction/satsifaction, Scale/Large projects

Software team response extensiveness (formative) (1 = 0%; 2 = 20%, 3 = 40%, 4 = 60%, 5 = 80%, 6 = 100%)
To what extent did the software team actually incorporate requirement changes in each of the following categories? (For example, if the project actually incorporated four out of ten different changes in a specific category, your answer would be 40 %.)

1. System scope (EXT1)
2. System input data (EXT2)
3. System output data (EXT3)
4. Business rules/processes (EXT4)

Software team autonomy (reflective) (1 = strongly disagree; 7 = strongly agree)

1. The project team was allowed to freely choose tools and technologies (AUTO1)
2. The project team had control over what they were supposed to accomplish (AUTO2)
3. The project team was granted autonomy on how to handle user requirement changes (AUTO3)
4. The project team was free to assign personnel to the project (AUTO4)

Self-managing

Self-organized

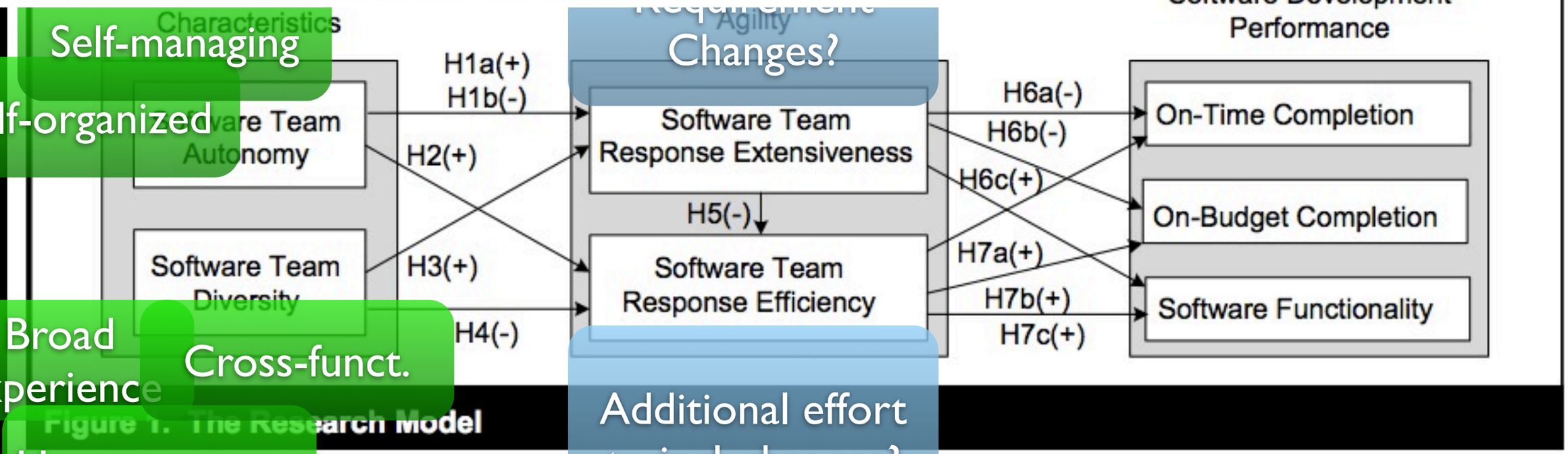


Figure 1. The Research Model

Changes?

Broad experience

Cross-funct.

Heterogeneous

Additional effort to incl. changes?

Software team diversity (reflective) (1 = strongly disagree; 7 = strongly agree)

1. The members of the project team were from different areas of expertise (DIV1)
2. The members of the project team had skills that complemented each other (DIV2)
3. The members of the project team had a variety of different experiences (DIV3)
4. The members of the project team varied in functional backgrounds (DIV4)

llowing changes? (Effort includes time, cost, personnel,

2. System output data (EFF3)
4. Business rules/processes (EFF4)
5. Data structure (EFF5)
6. User interface (EFF6)

Agility alone is not enough!

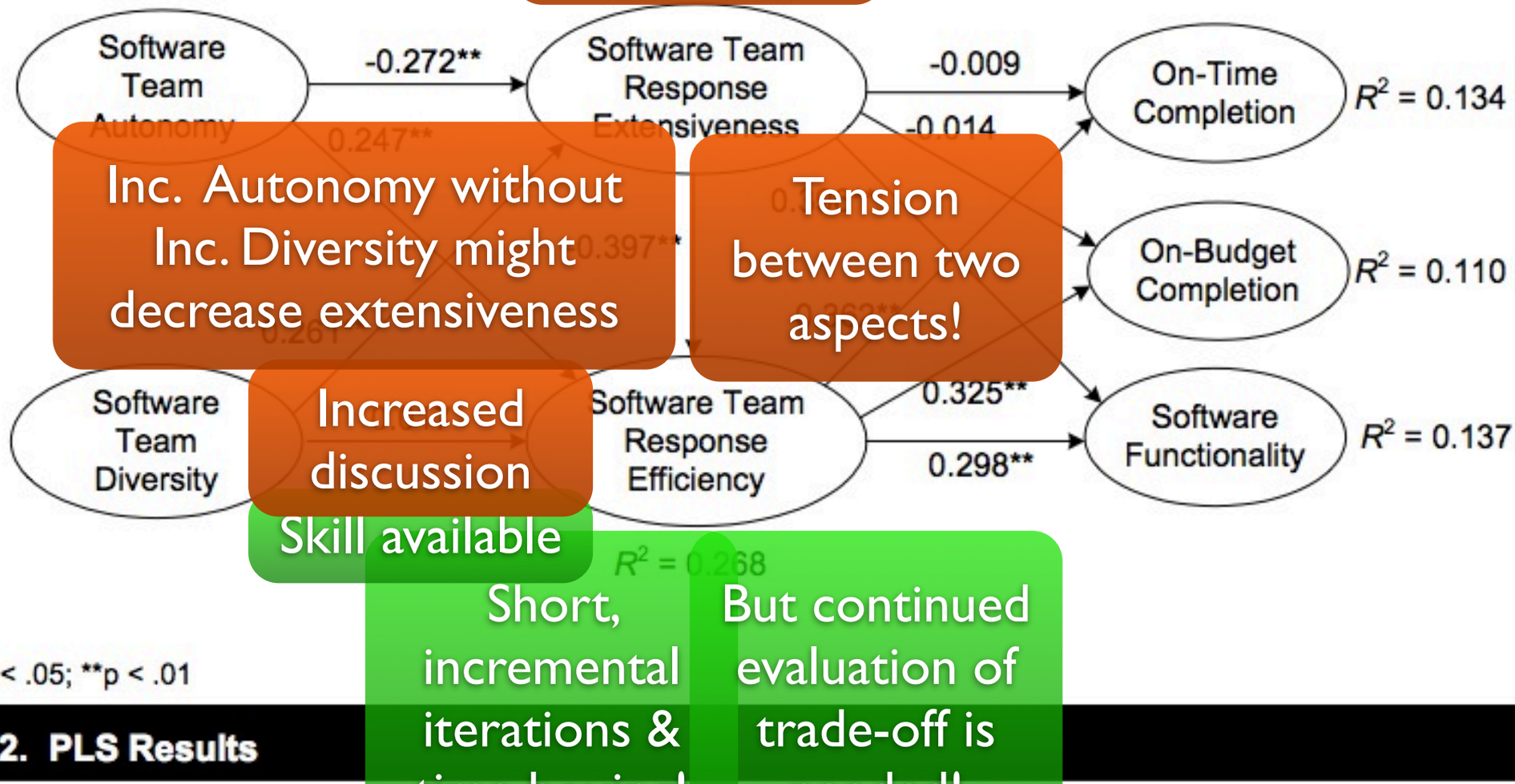


Figure 2. PLS Results

Inc. Autonomy without Inc. Diversity might decrease extensiveness

Tension between two aspects!

Increased discussion

Skill available

Short, incremental iterations & time boxing!

But continued evaluation of trade-off is needed!

Selective response to changes when time & cost top prios!

[Lee2010]

Summary of LEE2010

- Team Diversity:
  - *"The more diverse team will be better able to respond to changes because people will bring different levels of experience, different background, different skill sets. A team that doesn't have that diversity can get tunnel vision on a solution and not be as open to other options."*
  - *"The diversity made it more difficult to communicate and manage change, because the change required interaction amongst a diversity of workgroups, and that made it harder for people to be on the same page and agree to these changes."*

Summary of LEE2010

- Team Autonomy:
  - *"Each team member was able to respond to small system changes individually although the whole team discussed change requests that are important. We were very efficient in responding to change partly due to our authority to make decisions."*