Notes for Improving Scrum Clinic - Part 3 - Ver 40 (Work-in-Progress)

Joe Little — April 27, 2026
© Joseph Little 2026
Webinar — **LeanAgileTraining.com**

Intro



Contents - Part 3

- Intro - slide 2
- Pre-requisites to start? - slide 4
- Basics: Roles - slide 18
- Basics: Events - slide 30
- Basics: Artifacts - slide 38
- Requirements in Agile - slide 46
- Story Points - slide 60
- Business Value Points - slide 69
- Discussion, Questions - slide 75



Are there Pre-Req's to starting Scrum?



Pre-requisites?

- Decision: “Do I stay or do I go?”
- We don’t know with any precision what they are.
- But, common-sense says…[we will add details here later…]
- Your work is important:
  - Everyone should set you all up for success
  - And: not set you up for failure



To Start

- Probably want 3 things:
  - Desire for more transparency and a willingness to face the facts
  - A willingness and some ability to “inspect”
    - The process, the people, the product, everything
  - Some willingness and some ability to adapt. A drive to improve
- We do not need perfection (that never exists)
- But we need some energy in these three areas.



Say “No”!

- If you don’t have those 3 things, do you say NO?
  - Let’s put it this way: You are responsible to the organization to not waste its money. And starting badly will waste money.
    - And manager do not see everything. Never assume they “know” more than you (although it is possible).
  - Also: Sometimes whether those pre-requisites are met …will only be discovered later. Actions speak louder than words.



Maybe

- You (the Team) wants to try Scrum. Or at least willing to try
- You are ok with these ideas:
  - Transparency (more)
  - Inspect (we can some, and willing to try)
  - Adapt (we can some, and willing to try)
- That is: want to get better.



How about these?

- Are these (below) “required”?
- What if you have these problems?
  - They are not cross-functional
  - The Team is not autonomous (they must work with other teams)
  - They don’t know how to build software in small steps
  - They are not working on a new product
  - Note: List from Alan Shalloway. Not the only such list.



Required - to start?

- No.
- Most / All teams are lacking to some degree in these area. Not a reason stop.
- But we should discuss where your Team is, on each of these, and what your plan is to become better.
  - Esp if these are important impediments to success



Cross-functional

- First: learn to help each other.
- Second: See the value of becoming more cross-functional.
- Third: Take actions to become more cross-functional
- Fourth: If really hard - evaluate whether you can succeed with low cross-functionality
- Maybe you can succeed with the level you already have. If done well.
- Fifth: learn to ask for help more.
- Sixth: Teach each other how to help.



Autonomy

- A willingness to treat people as adults.
- The Team must self-organize within its situation.
  - So: if you must work with others (people, teams, groups) start the journey of figuring out how to do that well.
  - And its always different for each team
- Always be transparent.
- Any decisions (actions) that could affect others - tell the right person.
- Not just: Developers decide HOW to do the work



Small Steps

- Some teams can’t figure out fairly quickly how to build the product in small steps
- Small stories and small tasks
- Learn to break down epics into small stories. (Often seems hard at first)
- Learn to break down stories into small tasks and get feedback.
- Benefits are many - ex: focus on faster learning.
- “It ain’t over till its over.” Yogi Berra
- Some people (teams) need to be shown how by a “master.”



Not a “product”

- Some teams are not building what they call a “product”.
- Scrum often fits well with a Team doing “work”
- Work: Might be things delivered in 1 hour or 2 days. Or maybe longer, but we don’t call it a product
- Sometimes a “service”
- Try to give Team a higher-level unified “vision” or “mission”
- Must define something that can be “seen” in the Sprint Review by somebody — to get feedback.



Two More

- Team is somewhat dedicated.
  - Should be 100% (but that is never fully true)
  - Reasonable? 80% for everyone IN the Team (assuming Team = 7)
- One product or project per Team (at a time)
  - Start starting, start finishing
  - Usually: way too many projects are in flight
  - “Fighting fires!”



80% Means what?

- Each sprint - Team gets 80%
  - Can help other teams, people with other 20%
  - Teaches them to fish, not just gives a fish.
  - Eventually 85-15, 90-10, 95-5



1 Product / Project per Team

- Not addressing scaling.
- Simple: 10 Teams carry 10 products or projects.
- There is “Eu-stress”
- Things will be delivered faster (than otherwise)
- Requires that manager decides which are the top (10) efforts
- Requires saying “later” to some important stuff



Basics: Roles



ScrumMaster - 1

- Guides, cajoles, etc.
- Works on top impediment
- Pulls in the other parties
- The main driver (“the straw that stirs the drink”)
- 100% dedicated given 7-person Team
- Inc Velocity by 100% in 6 months
- Min of Truth and Transparency
- Min of Fun / Happiness



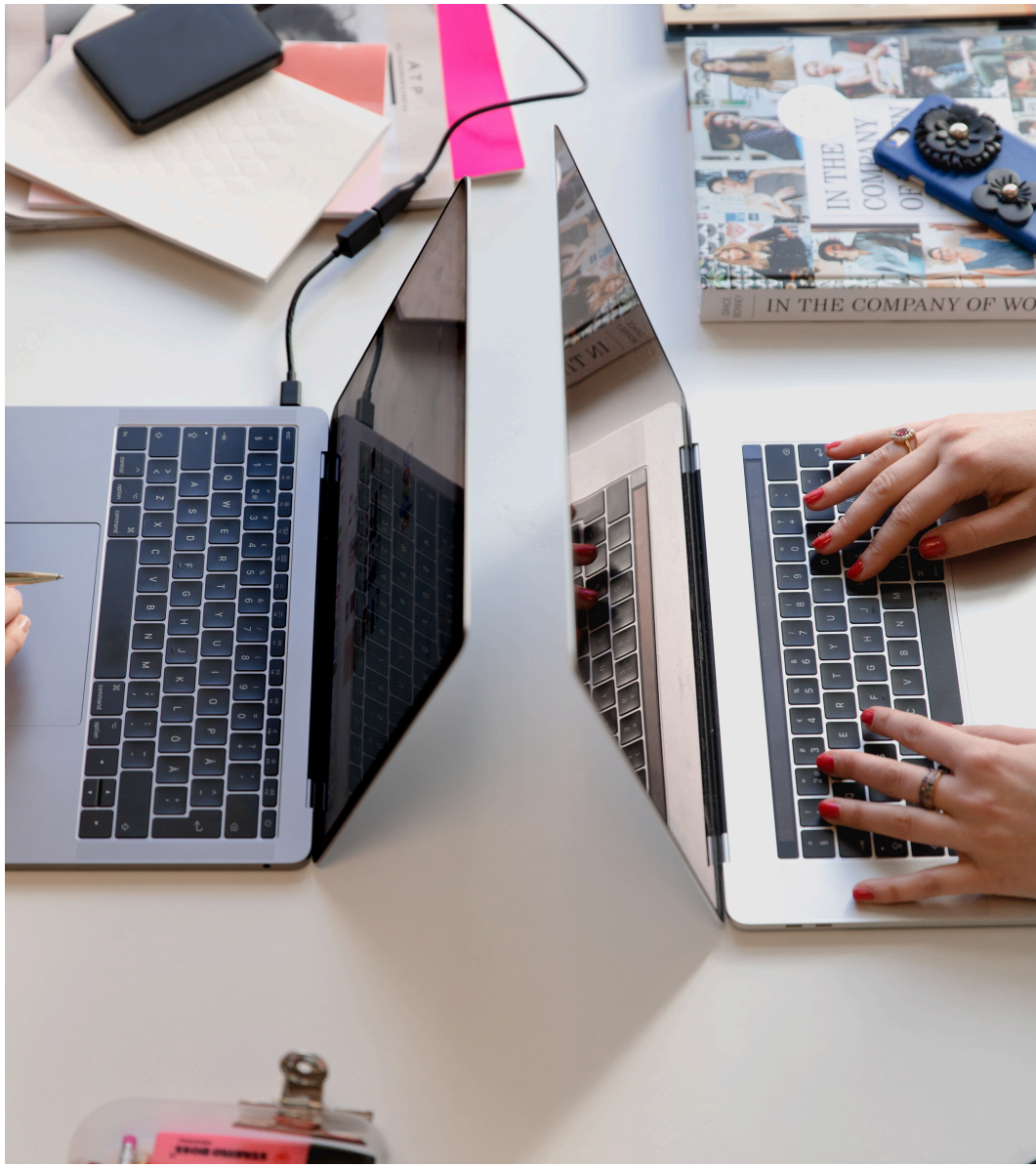
ScrumMaster - 2

- Min of Truth and Transparency
- Min of Fun / Happiness
- Min of Self-Organization



Who fixes impediments?

- Anyone can…
  - The SM
  - The Team (like a PBI in the sprint). The whole Team… (PO)
  - The Manager
  - Other people outside the Team
  - People outside the company (a vendor)



Impact of SM

- If you did the Top X impediments, could you double the Velocity?
- How long to fix the Top X?



Product Owner - 1

- Decisive under Uncertainty
- Listens to everyone
- Final orderer of PBL
- Decides when we have an MVP
- Decides which changes will be made to a story
- Decides product strategy (within context)
- Inspires the Team



Product Owner - 2

- Answers the Questions quickly
  - 10 secs
  - 10 mins
  - 24 hours
- Gets all Q’s answered before Sprint starts (DOR)



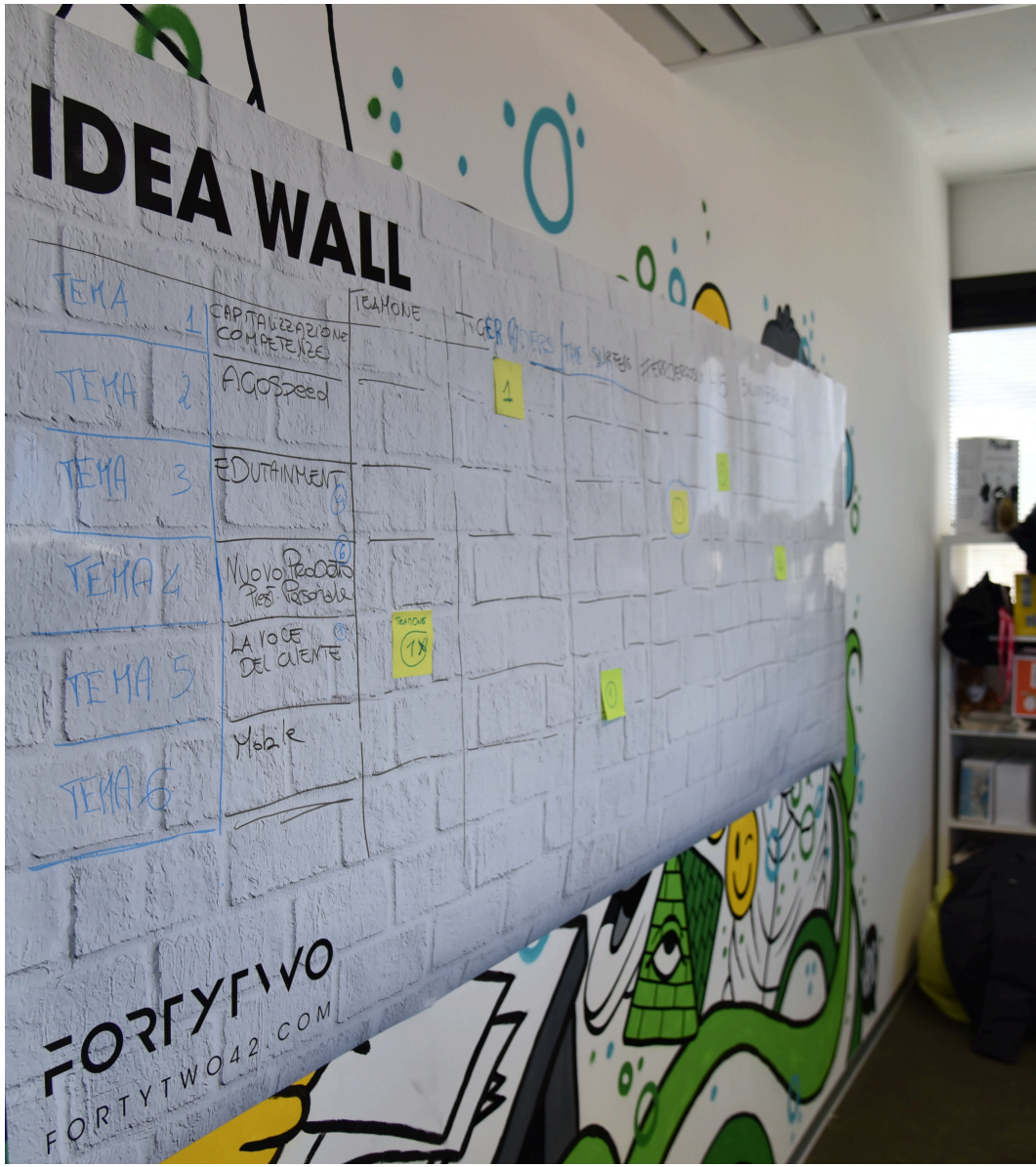
Developers - 1

- In Scrum: The builders and the verification / validation people. Everyone you “need to get it done”
- Team players
- Want to win, together
- Self-organize
- Self-manage
- Whatever it takes
- Figure it out



Developers - 2

- Help each other
- Ask for help (more)



Business Stakeholders

- Recommend: 4
- Pretty good feedback in the Sprint Review re what the customers will want
- They see the Imped List every Sprint (or the Top 3)
- Let the BSHs choose whether to invest in fixing some impediments. (They have some money, normally)
- They should help some with Impediments (maybe only by saying yes)



The "Minions"

- PO has to find these people
- Minions help pull together the details for the stories and answer questions, quickly
- Suspects: BA, SBA, SME, Manager, Boss, Process expert, customer person (of many sorts), senior people, the person who knows more than anyone else, etc.
- 1-20 people. Might change each Sprint, or depending on type of story
- Decides which changes will be made to a story
- Decides product strategy (within context)
- Inspires the Team



The Manager

- A Team should have one manager.
- The manager must help them attack the impediments — in general and specifically.
- The manager must say yes (sometimes) to people, money or approval, or all 3.
- The manager might actually fix some impediments.



Basics: Events



The Sprint

- 2 weeks?
- Is it working well?
- How commonly does Team complete the Sprint successfully?
- All meetings are in the Sprint?
-



Sprint Planning Meeting

- Before:
  - Well-ordered PBL? Small stories?
  - SP are accurate, generally?
  - Good details and other issues discussed / resolved?
- During:
  - BSHs attend?
  - PO and SM co-lead?
  - Devs select stories?
  - "Speak Now" time
  - Do 2 hour tasks?
  - Includes "other" discussion?
  - Everyone suggests tasks, mostly self-assigned
  - Tasks estimated fairly accurately?
  - Discuss having everything, nothing unneeded, accuracy of estimates, reasonableness of assignments, workload balancing?
  - Team commits?



Daily Scrum

- Only 15 mins?
- Use 3 Questions?
- People are honest?
- I suck, we suck, they suck?
- People say useful stuff?
- People listen to each other?
- Info used later?
- Emergent leadership?
- People find meeting useful?



Sprint Review - 1

- Do BSHs attend almost always?
- Quick review?
- Review of average velocity and projected delivery date?
- Discuss top 3 impediments?
- Stories are all sprint-sized?
- Demo done-done work?
-



Sprint Review - 2

- How complete is feedback?
- Discuss BV?
- Discuss positive & negative?
- Any concerns?
- Changes: we get all details now?
- Discuss cost of mis-communication?
- Do summary?
  - Estimate cost of mis-communication?



Retrospective

- How much talking? How much "action"?
- How much working together on the Top Impediment?
- How does or can the Team work together?
- Do we get good results from the Retro that impact the Velocity?
- If so, do we make that clear to the Team?



PB Refinement?

- The Scrum Guide is unclear about this.
- "Product Backlog items that can be Done by the Scrum Team within one Sprint are deemed ready for selection in a Sprint Planning event. They usually acquire this degree of transparency after refining activities. Product Backlog refinement is the act of breaking down and further defining Product Backlog items into smaller more precise items. This is an ongoing activity to add details, such as a description, order, and size. Attributes often vary with the domain of work." SG2020
- What are you doing for that?
- Do you see it as two-parts? (revising the longer term, gathering the details of stories going in the next sprint



Basics: Artifacts



Which artifacts?

- Scrum artifacts vs non-Scrum artifacts
- Let's try to recognize "all"
-



The Scrum Artifacts

- Product Backlog
- Sprint Backlog
- Scrum Board
- Burn Down charts
  - Sprint BD
  - Release BD
- Working Product / Increment
  - DOD
- DOR
  - DOR Process diagram (simple)
- Impediment List.



Other Artifacts?

- Risk List
- List of impediments fixed
- Tracking velocity, average velocity
- Tracking attitude on Agile-Scrum
- Tracking attitude on Product Success
- Happiness-Fun Metric



Impediment List

- Top 20 Impediments
- That stop us or slow us down
-



Transparency

- The Impediment List brings transparency.



How Does It Help?

- Let’s list the ways an **Impediment List** helps.



Benefits

- Increased velocity
- Visibility
- Transparency
- First Things First
- Less complaining
- Higher quality
- Less Stupid
- Higher morale
- They learn to identify the important stuff
- SM work list
- SM less distracted

Requirements in Agile



Ideas

- Not Big Up-Front (BUF) Doc
- Not all written
- Just enough, just in time
- “Everything they need to be successful”
- Clearer.
- Check that the communication was good



Principles (I suggest)

- Emergence
- Continuous improvement
- Continuous learning in 18 domains
- Due to:
  - Change in multiple dimensions
  - Learning
  - Better guessing of the future
- Both sides are responsible for success of communication



A Vision / Prod Goal

- The “first” things we have
- For how much?
  - Duration? 6 mos, 1 year, 5 years?
  - Releases? 1, 2?, 5?
- How clear?
- How much aspirational? Motivational?
- But: To be fair, a start…



Product Backlog

- We quickly identify a PBL of “everything” that we can think of
- In User Story format
- And includes “all the work” for…[the first release?]
- So: we have an overview of the whole thing. Helpful.
- PO is first one responsible for quality - and we all are responsible for quality of PBL



User Stories

- As a [persona x], I can …., so that ……
- Useful format
- Represents: the feature, the bus value, the effort, and the details (to come)
- Good level for planning
- The idea is that most of the details will come later (after the day the story is created)
- AND: Anyone can talk and ask questions about any story. Or identify missing stories



Next level: The Details

- What should they be and when should we “develop” them?
- First answers:
  - What the “developers” need
  - Just enough, just-in-time
  - Also: what the customers, business side, PO, minions, others - can give
- Jira: Lots of little “holes” to stuff the details for a given Story.



DOR - 1

- The Developers (SW: coders and testers) do NOT have to accept crappy requirements into the Sprint
- The Devs can reject stories
- The Devs must give some general advanced indication of what they need. A DOR (kinda like a DOD).
- And the PO must work to get those details. By using other people (1? 20?) I call “the minions”. (The PO can write some details also.)



DOR - 2

- The DOR is a living document. We expect failures in communication.
- When we see failures in the Sprint Review, we consider adding to the DOR, so that the PO+Minions collects that information for “next time”
- So: Comments in the Sprint Review give us feedback on how good the “requirements” were
  - Every sprint
  - And we improve from the feedback - every sprint



Not prescriptive

- The DOR concept does not require you to use any specific thing, document, tool, etc.
- It is “open” to what the (whole) Team wants / needs.
- But with the idea that we MUST improve the communication. The Req’s must be clearer
-



Enabling Specification

- Similar to the DOR concept, with an another metaphor
- The Enabling Spec is like a patent, that enables a person of reasonable ability to produce the product under patent
- Similarly, an Enabling Spec is just enough information to enable our Devs to build the right thing



The Details - 1

- No excess information
- Only what they want
- Varies by Team: eg, their existing knowledge
- Definitely conveyed in part by discussion
- Details often lead to further questions
- Clearer details can change and improve the SPs



The Details - 2

- Lists
- Pictures
- Discussion & Notes
- Questions answered



Related issues

- Lean UX
- User Experience issues
- Against making the process heavy
  - “Minimum Viable Bureaucracy”



Story Points



We need…

- 1 simple measure (guess) of Effort
- A way to make our guesses visible
- A way to generate learning (about everything that goes into effort)
- A way to guess at the delivery date
- A way to enable feedback from different viewpoints
- A way that causes knowledge sharing
- A way that is fun and fairly fast



So…

- We use the wide-band delphi approach; expert estimation
- We make it a game: planning poker
- We use the Fibonacci cards (sequence)
- We do it in a time box. Ex: 50 stories in 60-75 mins (at the beginning)
- We see what we think initially, prioritize our stupidity, learn more, and re-vote.
- Even our “not good” guesses are better than random
-



Who? Why?

- Developers (5) - the PO is there to explain stories. SM there to facilitate.
- Usual: BSHs go get a latte. I'm ok.
- Who comments later? Others
- Morale improves:
  - We estimate our own work
  - We can revise estimates when new information arrives (expected constantly)
  - We are expected to learn. Things are expected the change.
  - It gives us a basis to learn (the first draft)



Let’s not…

- Expect any guess to be precisely accurate (precision NOT needed)
- Over-work the team (SPs are part of protecting the Team from over-work)
- Have too much stress (as we almost always have, I think)
- Disrespect the Customers and the business side, by giving them no clue about the date
- Punish people for their first guess (“estimate”)
- Expect people to know everything; expect no change or learning to happen
- Have “planning” done by one or two “project managers”



Benefits - 1

- Morale improves:
  - We estimate our own work
  - We can revise estimates when new information arrives (expected constantly)
  - We use estimates to “protect” ourselves and stay within sustainable pace.
- We are expected to learn. Things are expected the change.
  - First estimates give us a basis to learn (the first draft)



Benefits - 2

- It's fun! It's a game. (Pressure feels less. Estimates improve, go to middle.)
- **Used to enable better sharing of knowledge**. Which helps multiple ways.
- Once we have a Velocity, then
  - Business and Technology can discuss effort and how it has changed.
  - Better collaboration



How often do we re-vote?

- Whenever we think we are smarter
- Whenever it makes sense
- When we finally have “all” the information

- Never perfect. Fairly fast. Overall, we learn faster.
- Effort is about the future. About discovering exactly how hard it will be to do each story.



Problems

- Some developers have never learned basics of estimating effort
- Often Developers were put under great pressure, and with no “controls”
- So, to some, estimating feels like a special place in Hell. We must overcome that bad attitude

- The basic variation of actual from expected is NOT a problem. Except when it is a lot.
- NO team should ever be expected to estimate every sprint “perfectly”. Not needed.
- So: This is typically a **notable culture shift**.



Business Value Points



We need…

- 1 simple measure of Business Value
- A way to make our guesses visible
- A way to generate learning (about business value)
- A way to enable feedback from different viewpoints
- A way that is fun and fairly fast



So…

- We use the wide-band delphi approach to Business Value
- We make it a game: priority poker
- We use the Fibonacci cards (sequence)
- We do it in a time box. Ex: Initially, 50 stories in 60-75 mins
- We see what we have initially, prioritize our stupidity, learn more, and re-vote.
- Even our “not good” guesses are better than random



Who?

- Business Stakeholders (4) plus PO
- Who listens: The Developers (and they learn)
- Who comments later? Others
- Morale improves:
  - I can see where we are going?
  - Everyone contributes their knowledge
  - It gives us a basis to learn (the first draft)



How often do we re-vote?

- Whenever we think we are smarter
- Whenever it makes sense
- When we finally have “all” the information

- Never perfect. Fairly fast. Overall, we learn faster.
- The Future is a hard subject. About what humans will want in the future.



Business Stakeholders like it

- Yes, they ask questions…
- But once they do it, they like it



Discussion, Questions



Discussion, Questions

- Don’t be shy!
- Turn off the mute
- OR: Type them in the Chat



Other webinars

- Two Types:
  - About courses and workshops (30 mins)
  - About agile questions or issues (60 mins)

- Where:
  - LeanAgileTraining.com
  - MeetUp



More Info about Us

- Courses & Workshops: **leanagiletraining.com/lean-agile-and-scrum-courses**
- Joe Little
- jhlittle@leanagiletraining.com
- (704) 376-8881
- **leanagiletraining.com**

