Better Requirements for Scrum Training

Producore

About This Course

Better Requirements for Scrum is Producore's course that radically improves the outcomes for Scrum teams. Anyone who participates in, works with, or manages Scrum teams, from developers to executive management can attend this course and bring the benefits of what they learned back to their organization. It is delivered online and is available as a public course and can be scheduled privately for your internal team as well.

Why Register?

Regularly failing to meet sprint commitments happens far too often for Scrum teams but it's the effect, not the cause.

Think back to your most recent difficult sprint. Did you experience any of the following?

- Time wasted on rework causing delays
- Old functionality broken by new (regressions)
- Bugs from missing details or misunderstandings
- Time wasted creating unnecessary functionality
- Critical requirement was wrong because it was built from incorrect tribal knowledge
- Having to rewrite requirements from scratch for changes to existing functionality
- Low confidence in safety of changes
- Finding requirements for existing/legacy code next to impossible
- Code, tests, and requirements drifting apart over time
- Team blindsided by external dependency
- Mid-sprint scope creep
- Necessary functionality only discovered after work already started
- Rework because requirements change mid-implementation
- Having to go back to the PO or stakeholders frequently during implementation to get requirements clarified
- Team builds something that turns out to be wrong because of a misunderstanding
- Insufficient details in requirements when implementation begins

Sprints don't just fail on their own. There's almost always one of these factors or something very similar involved, but the issues above, themselves, are merely symptoms. They cannot be fixed, directly.

Instead, the root causes must be resolved:

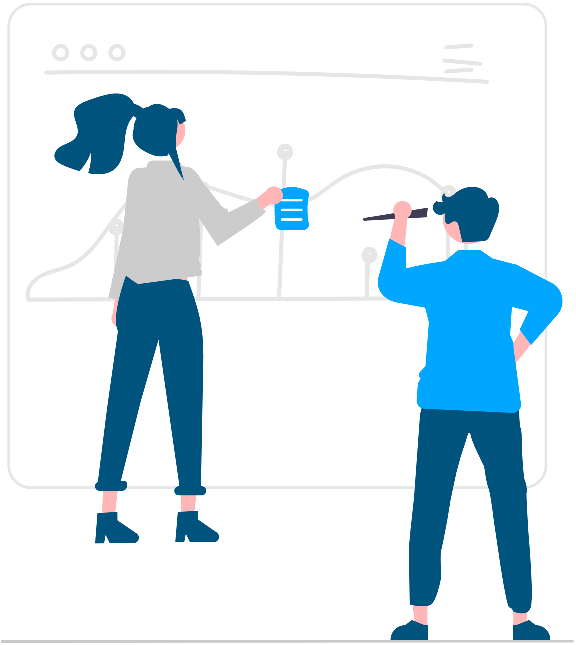
- Knowledge about how a system should work (past and present) is not properly captured
- Implementation often begins on a work item before it is truly ready

The *Better Requirements for Scrum* course directly addresses these two problems by adding eight things to your Scrum process that will completely transform how your team works without violating any portion of Scrum, itself.

Attendees will be taken through the why and the how for each change and might even pick up some details about Scrum they didn't know along the way.

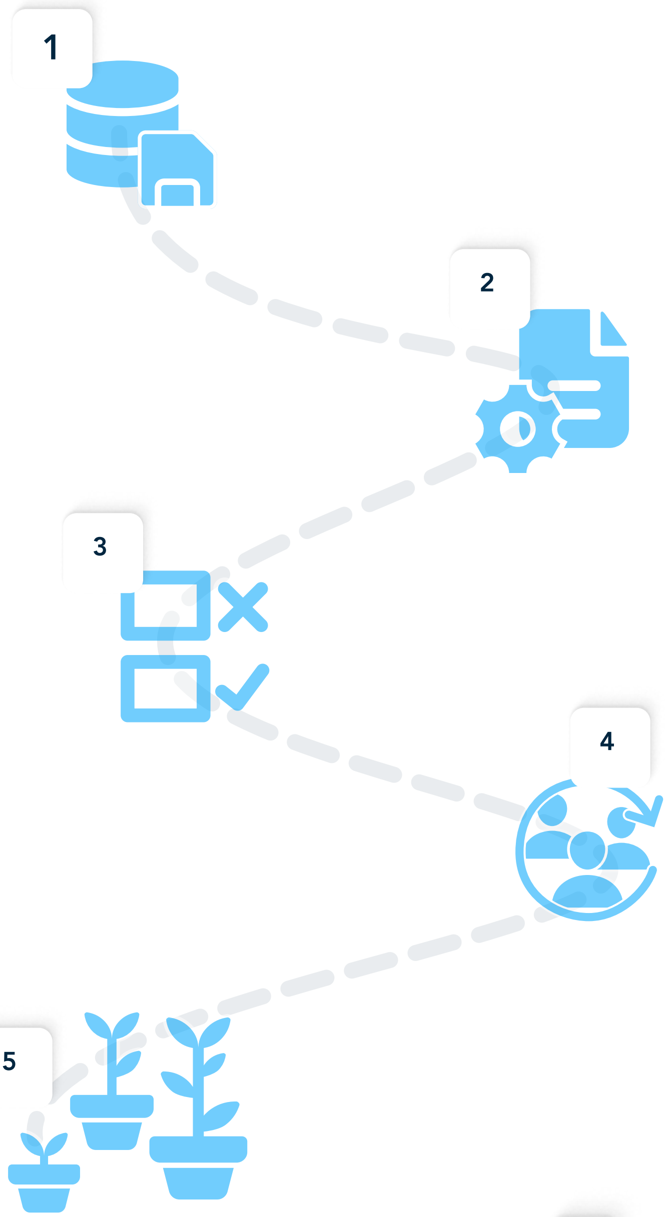
At the end, attendees walk away with a clear picture of how they can enhance their existing application of the Scrum process framework to eliminate the requirements-related problems they may not have even known they had.

What You Will Learn



In order for the course to have its intended effect, we start by making sure everyone is clear on:

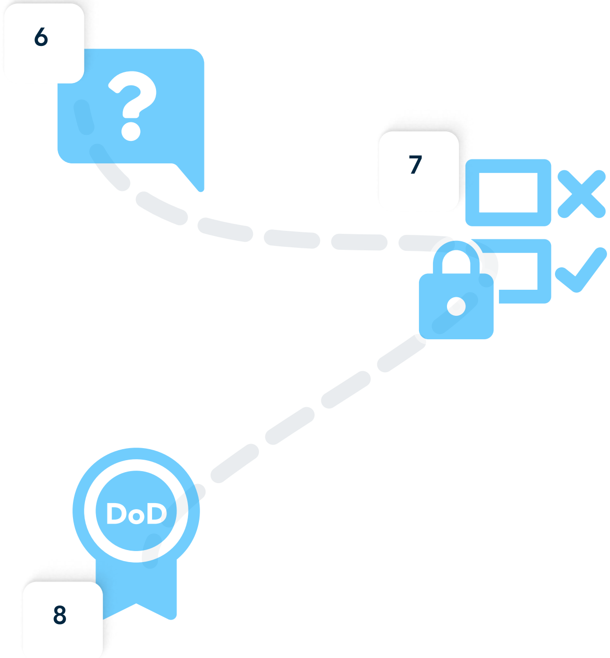
- What the root causes of our problems are
- What Scrum is
- What kinds of additions Scrum permits



We then move on to teaching how to address the problems. This is structured around eight crucial process enhancements every Scrum team should make:

1. Store and maintain requirements in a product knowledgebase
2. Use persisted requirements to drive implementation
3. Give every work item its own doneness and readiness criteria
4. Give Scrum teams the time and processes to be involved in requirements maturation
5. Allow requirements to mature to readiness without deadlines

6. Give Scrum permission to not accept work items they don't understand completely
7. Ensure Scrum teams do not commit to work items until they are ready
8. Give Product Owners permission to not accept items that do not meet their definition of done



We teach how to do each of these things, how they fit into the Scrum process, and how they relate to the larger Producore Framework (our proprietary incremental transformation program).

When we wrap up, we ensure that everyone sees the big picture - how it all fits together - and help people establish a set of action items to implement these changes in their own team and company.

Cost & Delivery

Cost: $ 999.00 USD / person *

Time-Commitment: 12 hours

Venue: Online

Format: Public

Duration: 2-days

* Ask us for group discounts

Frequently Asked Questions (FAQ)

1

How long does the course take?

This is a two-day course. Each day is seven hours long, including a one-hour lunch break.

2

Does this violate Scrum?

No. Everything taught is completely compatible with canonical scrum as defined in the Scrum Guides (www.scrumguides.org).

3

Is this just Waterfall in disguise?

No. Waterfall (in a nutshell) wants you to know everything before you do anything. Knowledge or design work for one area can hold up work in another. We just want you to know the thing you're doing before you do it.

4

Won't all this analysis slow me down?

No. This is focused analysis on imminent work. Not only will it not slow you down, you are definitely going to do it no matter what. In a lot of cases, we're talking about how you organize, track, and execute the analysis work you are already going to do in order to ensure it happens at the right time. Additionally, we want to make sure you capture the output of your analysis efforts, so you don't have to keep repeating it every time you want to make an update.

5

Why is this so important?

Once you see the difference between teams with a clean requirements flow and those working from immature requirements, you can't unsee it.

6

Does anything come with the course?

Yes. The course comes with a companion course book as well as access to the e-book we are writing on the Producore Framework.

7

Are you suggesting that we do comprehensive documentation? 😏

No. Comprehensive documentation, again, refers to the notion of knowing everything before you do anything. By contrast, we are telling you to capture what you are already having to learn in order to implement a product backlog item, so you can use it later.

8

Comments go stale… won't the product knowledgebase (PKB) go stale, too?

Comments go stale because there is nothing connecting them to code. They are typically written after the code they supposedly annotate. Therefore, there's nothing keeping them up to date. Requirements that fall into disuse suffer the same risk but, so long as you drive work through your PKB, it will by definition be kept up to date.

9

How does this work with behavior-driven development (BDD)?

This is perfectly compatible, complementary, and synergistic with BDD. In fact, it will be easier for you to pick this up, if your team already does BDD and vice-versa.

One problem that even most of the successful implementations of BDD experiences is inaccessibility of the requirements. The requirements are stored as tests and the tests might be readable to business stakeholders, but it's very rare for them to be able to actually find and maintain those tests. At best, it becomes a task for engineers to find the scenarios and share them with Product. More often, there are separate conversations for new and existing requirements.

Competency 1, taught in Better Requirements, resolves this problem. The PKB becomes the primary source of record and feature files become derivative artifacts. Requirements always stay in sync with tests and, at the same time, are easy for product workers to find, understand, and update.

10

What about technical practices and skills such as test-driven development (TDD), refactoring, and advanced software design/design patterns... do you not think those work?

We definitely think those things work and our organization contains experts in each. However, we've noticed a lot of teams are held up by something simpler and more fundamental than those relatively advanced disciplines. Those things take years to master and they impact how a developer writes their code. An excellent developer who has a reasonably strong grasp on TDD, design patterns, and refactoring can easily change any code.

That's great but it doesn't do much good if he's never able to figure out what code he's supposed to be writing in the first place. The Better Requirements course addresses the latter problem with simple-to-ratify Organizational Behaviors.

Once the requirements pipeline is aiming the team at the right target, the need for technical practices often becomes apparent. So they're always worth learning. We just don't generally try to get teams to learn them first.

11

What is the Producore Framework and why does it matter to me?

The Producore Framework is our proprietary incremental transformation program. We have divided all the big, daunting skills into small pieces and created a plan of adoption that makes each step easy to digest and also ensures it provides immediate value. Aside from a few things that are unique to our firm, like the product knowledgebase, we're still trying to get you to the same place that a lot of other expert technical or product development consulting firms want you to go. We're just trying to take you along a less painful path with a higher success rate and more immediate dividends.

About the Instructors

Luniel de Beer

Luniel de Beer is a 35-year veteran of the software industry with multiple decades spent as a product manager. He has worked with numerous teams and organizations, leading them to successful implementations of technical product management that improve their engineers' output in both speed and accuracy.

Max Guernsey

Max Guernsey, III has been a professional software developer and architect for over 25 years. He is an expert in all of the core software engineering skills: test-driven development, refactoring, and advanced software design. He has led countless teams to improve their technical skills, increasing the quality and maintainability of their software. He has also worked with many teams to improve the quality of their conversations with Product, thereby getting better results.

Certification

Upon completion of an optional quiz at the end, attendees will receive certifications in the Producore Framework Competencies 1 and 2.

Course Syllabus

1. **Establish the Problem**
    a. Example problem case
    b. Analysis of why problems happened

2. **Establish the Context**
    a. What is Scrum?
    b. What needs to be added? (high level)

3. **Product Habit 1 - The product knowledgebase**
    a. Addition to Scrum #1: Store requirements somewhere other than where you track work items
    b. Why work-tracking isn't enough
    c. Benefits of persistent requirements
    d. How to store requirements
    e. Ownership of requirements
    f. Version & process control for requirements
    g. The new way of writing work items
    h. How it fits in Scrum

4. **Relationship with the Producore Framework**
    a. Part of a larger framework

   b. Overall structure (Organizational Habits, Competencies, and Masteries)
   c. Better Requirements is one small slice
   d. Tie-in with Addition #1

5. **Product Habit 2 - Use organizational capabilities to drive structure of product knowledgebase**
   a. Why structure is a hard problem
   b. Concept: Business architecture
   c. The business big picture
   d. Problems with other methods knowledge-organization
   e. Business-oriented naming best-practices
   f. Tie-in with Addition #1

6. **Process Habit 1 - Use product knowledgebase to guide implementation**
   a. Addition to Scrum #2: Use persistent requirements to drive testing and coding activities
   b. Problems with writing code from change-orders
   c. Problems with writing tests from change-orders
   d. Driving implementation from PKB
   e. Tracking changes
   f. Tie-in with Addition #2

7. **Competency 1 - All work driven through the product knowledgebase**
   a. Synthesis of 3 Habits
   b. The life of a requirement
   c. Flow of information and work in Scrum
   d. Benefits of Competency 1 (before and after)

    e. Enabling comprehensive alignment between implementation and requirements
    f. Tie-in with Additions #1-2

8. **Process Habit 2 - Collaborate on requirements until shared understanding achieved**
    a. Addition to Scrum #3: Scrum teams need the space, time, and processes to be involved in requirements maturation
    b. Thoughts on wise use of time
    c. The natural flow of information and work in software development
    d. Why so much rework with such long delays?
    e. Collaboration is already happening throughout sprint
    f. No need to wait
    g. How this saves you time
    h. This is not Waterfall (and why)
    i. The value of using Engineering time in requirements analysis
    j. How to budget for Engineering participation in requirements
    k. Tie-in with Addition #3

9. **Process Habit 3 - Work gated by a unique definition of done**
    a. Problems with delayed, poorly-defined, or “standard” definitions of done
    b. Addition #4: Every work item has its own doneness criteria, and they are treated as a contract
    c. Addition #5: Give product owners permission to not accept items that aren’t done
    d. Definition of done template - a starting point for every work item

e. Sample DoD items
f. Work items are unique, require unique DoD
g. Track DoD completion status in work item
h. Accepting a story with a purpose-built DoD
i. DoD cuts both ways (a.k.a.: Yes, you have to accept it when the DoD is met)
j. The two DoD gates (implementation and acceptance)
k. Tie-in with Additions #4-5

10. **Process Habit 4 - Work gated by a unique definition of ready**
    a. Problems with ignored, poorly-defined, or "standard" definitions of ready
    b. Addition #6: Teams cannot commit to work items until they are ready
    c. Addition #7: Give Scrum teams permission to not accept items that aren't ready
    d. Why not start before you're ready?
    e. Definition of Ready (DoR)
    f. Definition of Ready template
    g. Sample DoR checklist items
    h. DoR template: a starting point
    i. Why each work item needs its own DoR item
    j. Hand-off from PO to Scrum team
    k. Tie-in with Additions #6-7

11. **Allowing requirements to mature without deadlines**
    a. The difference between real and arbitrary deadlines
    b. Why arbitrary deadlines are a problem
    c. Starting immature work items because of a deadline reduces chance of meeting deadline

d. How to work with real deadlines
e. Allowing requirements to mature to readiness saves a lot of time, how and why
f. Addition #8: Allow requirements to mature to readiness, without respect for deadlines

12. Competency 2 - Requirements maturation flow

a. How Process 2, 3, & 4 work together
b. Updated life of a requirement
c. Updated example flow of information and work in Scrum
d. Benefits of Competency 2 (before and after)
e. Tie-in with Addition #8

13. Conclusion

a. It’s all compatible with canonical Scrum
b. Recommended rollout plan
c. Final thoughts on requirements and Scrum
d. Advice for change & persuasion
e. Recommended next steps
f. Final gifts

Unlock Your Team's Success:

Say Goodbye to Sprint Failures and Welcome to Clear Requirements

In the competitive landscape of software development, the efficiency of your Scrum teams isn't just desirable—it's critical. Yet, too many teams find themselves mired in the quicksand of missed sprints, misunderstood requirements, and endless rework. This isn't just frustrating; it's a roadblock to your company's innovation and success. The root of the problem? A clear, comprehensive understanding and management of requirements.

Producore's Better Requirements for Scrum Training is not just another course. It's a transformative experience designed to target and eliminate the root causes of your Scrum team's challenges. Tailored for anyone involved in the Scrum process—from developers to executive management—this course illuminates the path to truly seamless and efficient project management.

Don't let another sprint go by mired in the same old challenges. Embrace the future with The Better Requirements for Scrum Training and bring the benefits of streamlined, efficient, and effective project management back to your organization.

Take the first step towards unlocking the full potential of your Scrum teams. Empower them with the knowledge, strategies, and practices that lead to success. Save your seat now and witness the transformation unfold.

Save Your Seat Now →

www.producore.com