Lessons Learned

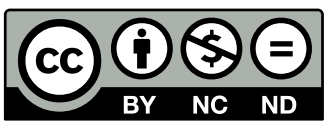
from 100+ (Simulated)

Agile Transitions

Elisabeth Hendrickson

If you like this eBook, come visit us at Agilistry Studio in Pleasanton, CA for our immersive, experience-based training classes.

www.agilistry.com



Overview

The WordCount simulation revolves around a fictional company, WordCount, Inc. As you might expect, WordCount, Inc. makes a system that counts words.

The participants in the simulation work for the fictional company as Product Managers, Testers, Developers, an Interoffice Mail Courier, or (in a rather non-traditional role) Computers.

A facilitator plays the role of Customer requesting features from the Product Managers at the fictional company. The Customer explains that she runs a children's book factory where she is having a terrible time with inventory management of words in the books.

"I have way too many 'zebra's' lying around," she explains, "and not nearly enough 'the's.'"

Product Managers

Product Managers define the product, interact with the Customer, and write requirements.

They have an initial set of four requirements on blue index cards.

In addition, they have a memo from their predecessor:

Subject: WordCount Road Map

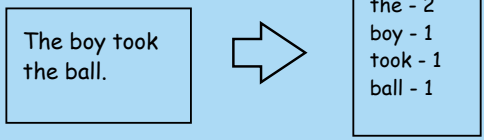
The customer meeting went great! They're looking forward to the delivery of Release 0 with the basic word counting features. As soon as they accept it, they'll pay $1000!

Next steps: the customer wants to use WordCount to compile a list of the most popular words across a set of documents. To do this, we need at least two additional features:

1. The results need to be sorted in descending order of word count so it's easy to determine the most popular words. I already wrote a requirement card for this.

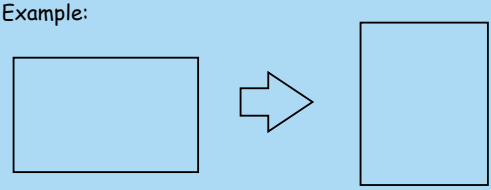
2. The results need to be compiled across multiple inputs, not just a single document.

WordCount Requirements: Release 0
Given a sentence or phrase in the English language, provide a count of how many times each word was used. Count should be case insensitive and ignore punctuation. Example:

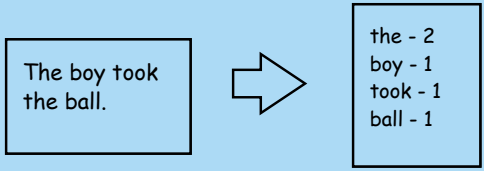


WordCount Requirements: Release 0
Given a blank input card, return a blank output card.

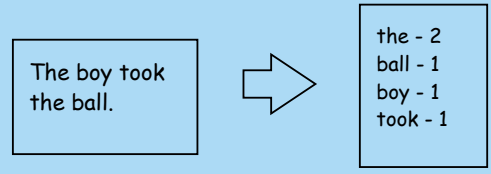
Example:



WordCount Requirements: Release 1
Return word count results in descending order of the count of occurrences. Example:



WordCount Requirements: Release 1
Sort word count results alphabetically within sets of the same occurrence count. Example:



Developers turn the requirements into executable instructions (“code”) on green index cards. When the code is ready, they install it on the Computer by placing it in the Code Installation Tray.

There is an existing code base in the Developers’ kit that is identical to the code already installed on the Computer:

In addition, the Developers have a copy of the Release 0 and Release 1 requirements.

WordCount Code
Version 0

1. Pick up a White card containing the Input from the Input tray.

WordCount Code
Version 0

2. Pick up a White card from the available stack of Output cards.

WordCount Code
Version 0

3. Execute the instructions on the following cards for each word on the Input card…

WordCount Code
Version 0

3A. If the word is not already written on the White Output card, write the word on the White Output card with an initial count of 1.

WordCount Code
Version 0

3B. If the word is already written on the White Output card, increment the count for the word on the White Output card by 1.

WordCount Code
Version 0

3C. If you cannot determine how to count the word, write the word on a Purple Catastrophic Error card.

WordCount Code
Version 0

3D. If there are more words remaining to be processed on the Input card, return to 3A. Otherwise, continue to instruction 4.

WordCount Code
Version 0

4. If a Catastrophic Error occurred, place the Purple Catastrophic Error card in the Output tray. Otherwise place the White Output card in the Output tray.

Testers design and execute test cases and report bugs to development. They have an initial set of 3 test cases.

The Testers also have a copy of the Release 0 and Release 1 requirements so they can design additional tests.

Testers execute test cases against the WordCount system by submitting input on white index cards to the Computer's Input Tray. They then receive output, also on white index cards, in the Computer's Output Tray. The Testers are supposed to compare the expected output from the yellow test case card to the actual output the Computer produced. If there is a discrepancy, the Testers capture a bug on a red card.

Test ID: 1

Objective: Verify system can count a single word

Input	Expected output
a	a - 1

Test ID: 2

Objective: Verify system can count words in a sentence

Input	Expected output
Go for a walk.	go - 1 for - 1 a - 1 walk - 1

Test ID: 3

Objective: Verify system returns a blank white card when Input card is blank

Input	Expected output

People playing the role of Computer execute a set of instructions, the "code" for the WordCount system, in order to turn input into output. It's a slightly strange role since it requires people to channel their inner robot.

In addition to the existing code base, the Computer has instructions that define a word as "a set of characters separated from other characters by a space."

Before any of the other participants have a chance to submit input cards to the Computer, the facilitator runs a test to calibrate the Computer, ensure the Computer will behave in a consistent and predictable manner, and verify that the people playing the role are interpreting the code literally.

Sometimes the people playing the Computer channel a little too much of their human intelligence into the role, and instead of counting "ball." as a single word with 5 characters, they intuit the writer's intentions by counting it as just "ball" without the ending period.

Of course, not all input can be processed. If the Computer cannot process the input, for example if someone puts a yellow test case card into the Input Tray, it throws a catastrophic error on a purple index card.

Interoffice Mail Courier

The Interoffice Mail Courier delivers messages and artifacts between groups.

In choosing roles it's common for participants to express concern that the Interoffice Mail Courier will be overwhelmed with the volume of messages sent between the groups. One participant was so concerned about this that he recruited an assistant, so there were two mail couriers.

However, participants never send all that many messages. They're too busy doing their own jobs to communicate with others. So the time we had two mail couriers, both of them were bored stiff.

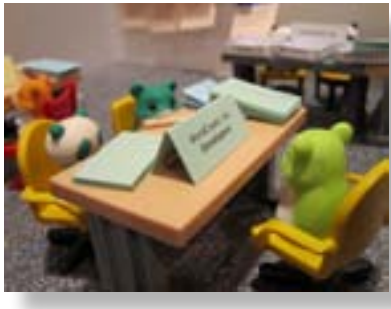
Observers

Observers watch interactions and take notes of things they saw and heard that they think may be of interest to the rest of the participants. They are specifically asked to focus on objective data rather than subjective judgments. The role of observer is intended to help participants see the big picture, not as a coaching or critiquing role.

This seems to be particularly difficult for some people. During debriefs we sometimes have to interrupt observer reports to remind observers that comments like, "It was really bad that you didn't communicate…" are not helpful to the participants.

Throughout the simulation, we work in 15-minute time boxed rounds. After each round, we pause to reflect on what's going on at WordCount, Inc. and adjust the working agreements.

When the simulation begins, the participants must follow a pre-defined set of working agreements that define responsibilities and constrain communication paths:



1. Each team sits in its own workspace (Developers with Developers, Testers with Testers, etc.).

2. Communication between Developers, Testers, and Product Managers occurs only through Interoffice Mail.

3. Testers send a Test Status Report to Product Managers at least once during each round.

WordCount Test Status Report

Number of Test Cases in Test Suite _____

Test Execution Status:

Version	# Cases Executed	# Pass	# Fail

Bug Counts:

Number of Bugs Found In This Round _____

Number of Bugs Sent to Developers _____

Number of Fixed Bugs to be Verified _____

Number of Bugs Closed This Round _____

Progress Notes:

Development Status Report

Requirements:

Number of Requirements Received this Round _____

Number Implemented _____

Code & Deployment Metrics:

Current Installed Version _____

Number of Cards in Installed Code _____

Bug Counts:

Number of Bugs Received this Round _____

Number of Bugs Fixed This Round _____

Progress Notes:

4. Developers send a Development Status Report to Product Managers at least once during each round.

5. Product Managers send Requirements to Developers and Testers through Interoffice Mail.

6. Developers deliver new Code to the Computer by placing it in the Code Installation tray.

7. Only Developers may create or modify Code.

8. Testers must document all Test Cases to be executed on Test Case cards.

9. Testers send Bug Cards to the Development team.

10. All bugs must be documented and tracked on Bug Cards.

11. Only the Product Managers may interact with the Customer (played by the facilitator).

These working agreements are only in place for Round 1. After that, participants are allowed, and encouraged, to change them in order to increase WordCount, Inc's efficiency and effectiveness at delivering value to the Customer.

After each 15-minute round we pause to reflect on the current state of things and improve the process.

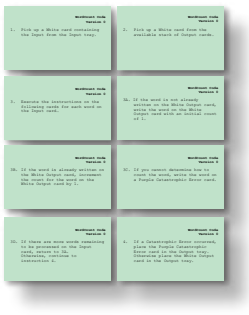
We start each of these reflect-and-adapt session by gathering data:

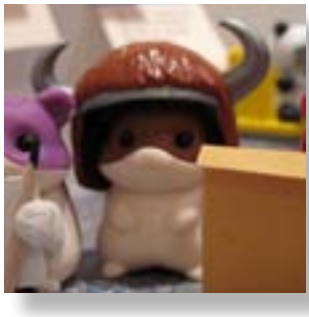
1. The facilitator calls for a one-word reaction from each participant. The resulting list of words provides a snapshot view of how the preceding round went.
2. If there are status reports, we hear their contents.
3. The Observers report from their notes.
4. Everyone regardless of their role is encouraged to contribute their own observations.

As we gather data we also consider questions and try to unravel mysteries: “How did that happen?” “Where did that requirement come from?” “Why is that test failing?”

We also consider how well the current working agreements are supporting Agile in terms of Communication, Collaboration, Feedback, Visibility, Alignment on Shared Goals, Reduce Waste, Focus on Value, and Delivering Early & Often.

Once all the data is out on the table, the facilitator turns control over to the participants who then discuss how they wish to work together in the next round. The participants can decide to change as many or as few of the working agreements as they wish.





This is a silly simulation. The “code” is on 3x5 index cards executed by people pretending to be Computers. The Customer often wears a Viking helmet. Even the basic premise—that the customer needs to keep track of an inventory of words for Children’s books—is ridiculous.

Despite all the silliness, the challenges are incredibly real. We see teams spin their wheels in the simulation for the same reasons they do in the real world: because they can’t get a handle on the real requirements, because team members try to be responsible to the point of irresponsibility, because the code is a convoluted mess, because they lack any sense of version control, etc.

This is the power of the simulation: it creates the same conditions that cause organizations grief, but within the context of a microcosm where we really can tease out all the underlying miscommunications and misalignments, disentangle them, and get the team back on track.

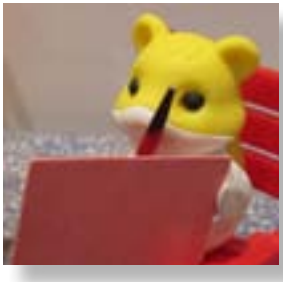
Just as the challenges are all real, the insights that teams have while working through the simulation are equally real, and powerful lessons they can take with them back to the real world where Computers don’t actually talk back.

As the first 15-minute round begins, participants are inevitably completely focused on their particular role. The room is usually very quiet during this first round, and it's rare for the teams to send more than a handful of messages.

Occasionally a manager will mistake the silence for productivity. One manager, playing the role of Observer, commented during the first round: "See, this is how it should be. Look at how everyone is concentrating so hard! They're getting lots done."

It's all an illusion. In reality the isolated teams are spinning their wheels and creating unnecessary work for themselves by acting on assumptions rather than clarifying the state of the system.

Usually the Product Managers talk to the Customer during this first round, but they usually cannot offer any timeline for giving a demonstration of the WordCount system. Sometimes they are so concerned with reading the requirements on the blue cards and the memo that their predecessor left behind that they spend a scant few minutes with the actual Customer.



Testers often begin cranking away on designing test cases based on the initial requirements. Sometimes they get so carried away with creating test cases that they forget to execute any tests. So by the end of the first round the testers often have no greater insight into the state of the existing code base than they had in the beginning.

In the meantime the Developers are so eager to get to coding that they almost never execute any tests on the Computer. Sometimes they don't even realize that the code base they have is already installed on the Computer and they're stunned when they get bug reports from the Testers.

With little to no input, the Computers are sometimes so bored that they invent things to do like randomly throwing errors, making odd beeping noises, or pretending there's a screensaver running.

It should come as no surprise that no one ever ships anything in the first round.

But then the participants get the chance to change the rules.

When participants have the opportunity to change the rules for themselves, we start to see what can happen in organizations transitioning from traditional command-and-control structures to self-organized teams.

Sometimes the group struggles with self-organization. Some members look to the facilitator for answers. "Can we do away with the Interoffice Mail Courier role? Would that be OK?"

The facilitator tells them to decide what they want, not ask permission.

Lesson: if teams are supposed to self-organize, don't undermine their authority by telling them how to self-organize.

Soon the group starts making decisions. Items on the working agreement posters are crossed out. New agreements are added. Some working agreement posters are torn up entirely. Some groups change the rules just a little, others change just about everything.

When we first started running this simulation we facilitated the changing of the working agreements. But having the participants take control turned out to be much better. Participants feel a sense of ownership over their process when they drive all the decisions.

There's an important lesson here for companies in transition: telling a team that their self-organizing and then imposing someone on the team who tells them how to go about self-organizing undermines the team. The team won't feel ownership over their decisions, and thus may not abide by them.

In round 2, teams usually have done away with the Interoffice Mail Courier. Participants are free to get up and move around, and talk with whoever they need to.

The result usually is no better than round 1. Out of over 100 teams that have gone through the WordCount simulation, maybe 2 have shipped in the second round.

It seems that just solving the communication problems of the first round is not enough to make the team fire on all cylinders.

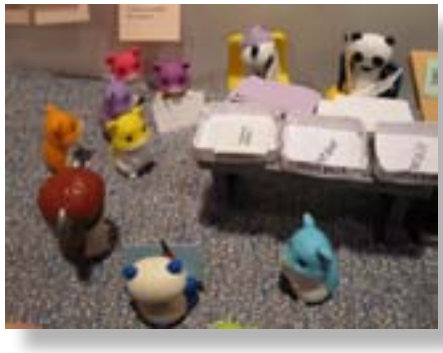
Lesson: Any transition results in chaos.

The lesson here for organizations in transition is that no matter how good the change to Agile is, it will be accompanied by a period of chaos during which it may seem that things are worse than before.

There is no way to avoid the chaos. The best we can do is acknowledge that it exists and give teams time to adjust to the new way of working.

In response to the chaos, sometimes someone on the team (usually a Product Manager) will declare, "We have to get things under control!" This declaration is often accompanied by a call for a Project Manager or someone who will tell everyone else what to do.

This never goes well.



Lesson: a cry for "we have to get things under control!" isn't necessarily about control. It may be a cry for visibility.

That's because the real problem is not a lack of centralized control but rather a lack of visibility. Centralized control won't solve the visibility problem. It might solve the alignment problem, but at the cost of bottlenecking everything through a single individual.

In one case the team called for someone to be in the role of Project Manager. After much arm twisting, the real-world Project Manager took the role. At the end of the round, she announced loudly to the entire room: "I QUIT!"

She meant that she quit the role of Project Manager in the simulation. But her frustrations in the simulation were very related to her frustrations in real life. She quit her real life job in frustration just a couple months later.

The key to getting past the feeling that everything is out of control is to recognize that the call for centralized authority often stems from feeling of helplessness. There is no one place to find out what's going on. The different teams are not aligned in their activities or even their purpose. So centralized management control seems like a reasonable solution.

The same thing sometimes happens in the real world. In response to the chaos of change, management will step in and take control away from the team.

So the lesson here is that a cry to get things "under control" is usually a symptom of a lack of visibility. Give management the visibility they need, and the demands to get things under control usually evaporate.

On a related note, teams often spend an inordinate amount of time during round 2 meeting. Sometimes they'll schedule meeting after meeting during the short 15 minutes. One team established a 6-minute "standup" at the beginning of the round and a 4-minute "checkin" meeting at the end. That left them just 5 minutes to get real work done.

The meetings are almost never effective. "What did you do yesterday? What are you planning to do today? And what's getting in your way?" are pointless questions when you're in the middle of a 15 minute round.

But the participants know they need to communicate better so they hold meetings.

Out in the real world this happens too. One participant shared his feelings during a debrief: "I hate Agile. We used to have a one hour status meeting once a week on Fridays. Now with Agile we have a 45 minute status meeting every single day."

This is a sign that the teams are not using meetings for coordination but rather are using them to disseminate information. The result is a mind-numbing waste of time.

The solution is to recognize that meetings solve a specific type of problem related to communication and coordination. But the problem that the teams face in round 2 is usually that no one has any idea what's going on elsewhere. They need visibility more than communication.

Lesson: Never use a communication solution to address a visibility problem.

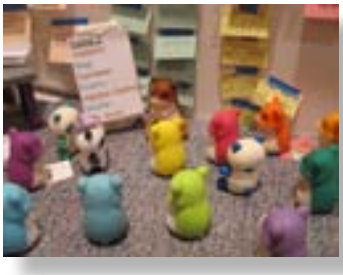
By round 3 the team has usually started figuring out how to work together effectively. Developers have usually learned how to get feedback from the Computer about the extent to which they said what they intended. Testers have learned how to narrow the focus of their testing to the specific quality criteria that the Customer cares about. Product Managers have learned to get examples from the Customer.

The biggest change in round 3 is often related to the physical space. Tables are pushed closer together and all pretense of separate work areas melts away. Just moving tables a few inches can result in a dramatic increase in collaboration between the roles.

In one group, the team mentioned during a debrief that there was an ongoing debate within the organization about co-location. Some people were resisting the team room concept because they liked having their own personal space.

Lesson: small changes in layout can have a huge impact.

After the third round one of the biggest advocates of private work spaces abruptly announced, "I'm convinced. I'll move into the team room." He had been able to see how proximity led to greater effectiveness.



Sometimes the energy in the room during the third round changes not because of huge sweeping changes in layout but rather tiny changes. Moving a flipchart just 12 inches can have a huge impact.

The lesson here is that physical layout matters. It's important to get team members close, and even small changes—removing a seemingly insignificant barrier or providing just a little more shared space—can have a big effect.

Typically by the fourth round the participants are working as a cohesive unit. They've created visibility by putting all the artifacts in a central location, often on a wall. And they have usually found ways to convey a huge amount of information at a glance so the Product Managers know when a new feature is ready to demonstrate to the Customer.

In one case the transformation was astounding.

The team had been in chaos for the prior two rounds and prepping for every Customer demo involved a great deal of drama: discussions, questions, arguments.

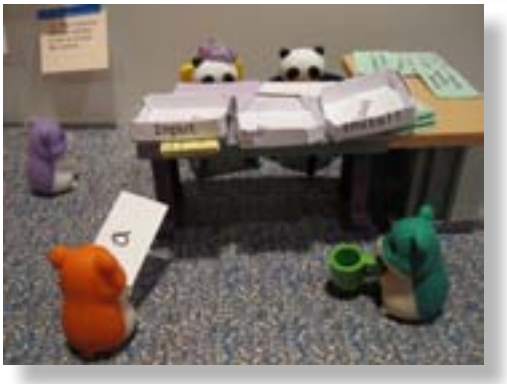
In the last round, the Product Manager didn't even have to ask "is the feature ready?" He just looked at the wall with the acceptance tests. When he first looked, he noticed that one of the tests had a red sticker on it, so he stood back and waited, letting the team fix the issue. A moment later all the tests had green stickers on them. The Product Manager calmly walked over to the Customer and invited her to the demo. The Customer was delighted with the feature and paid on the spot.

The power of tests to convey such critical information is sometimes lost on teams that see tests as being all about bugs. Such teams usually struggle harder to establish a solid shared understanding of the full scope of each feature. They accuse the Customer of indulging in scope creep (even though the Customer in this simulation is completely consistent throughout about expectations for what the system should and should not do).

Lesson: tests are an alignment tool

The ultimate lesson that this simulation teaches is that tests can be a tool for aligning various stakeholders. Tests aren't just a mechanism for finding bugs. They're clear, concrete statements about expectations that are either true, or not. Given a set of tests that delineate the scope of a feature, a team can gain a sense of momentum and avoid all the hand-wringing and drama around requirements.

The essence of the test-driven mindset is to begin with the end in mind. Given that the entire simulation takes place on 3x5 index cards, it is surprising how well WordCount provides deep lessons in the power of tests to shape the outcome of a development effort.



Lesson: Failure stems from dysfunctional systems, not individuals or teams.

There are any number of teams that have struggled throughout the simulation. A few (a bare handful) even failed to ship a single feature that the customer would buy.

When this happens, it's never because of a simple screw up. The problems are systemic.

One team that failed to ship blamed the Customer. "The Customer kept changing her mind!" they complained. This was not true: the simulation is difficult enough with a Customer who is clear about his requirements and consistent in his requests. So we play the Customer role straight. We don't waffle or trick participants by claiming to want one thing then purchasing something different.

The team saw ambiguity because they hadn't listened to what the Customer wanted. They built features on assumptions, and blamed the Customer for not wanting what they built.

The same team also blamed the Facilitator. "It's all a trick," they said. "The whole thing is a trick."

That same team had failed to ship anything in real life too, and they used the same excuses in the real world that they used in the simulation.

And why were they experiencing these problems both in the simulation and in real life?

There is no one single reason. In the real world, the team was dealing with a host of issues: the perception that they were being set up to fail by a controlling management team; self-imposed pressures that interfered with truly listening to the business needs; and an organizational structure that made true collaboration almost impossible.

In short, the team's problems stemmed from deep-rooted issues far outside the team's sphere of control or influence.

Certainly the team's victim mentality interfered with their ability to succeed in the simulation. That same victim mentality no doubt contributed to their real world challenges. But to blame the team for failure in either the simulation or the real world is to ignore the influence of the system.

The WordCount simulation creates a microcosm that reflects real world tensions and forces in an Agile adoption. Thus the patterns we see in teams that struggle or succeed wildly also reflect real world patterns.

Teams that struggle in the simulation tend to hold tight to roles, silos, and separate

work areas. They usually have everything in progress and nothing done. They frequently waste time trying to coordinate through meetings instead of creating visibility. And they almost always fail to engage effectively with the customer.

By contrast teams that succeed wildly drive development with customer-provided examples. They seek customer feedback early and often, running demos as soon as they have anything to show even if it isn't done yet. These teams also usually demonstrate a sense of ownership over their working agreements and physical environment, and re-shape both fearlessly.

In the end, the teams that succeed in WordCount typically end up re-inventing key technical Agile engineering practices around feedback and visibility such as index-card equivalents for continuous integration, ATDD/TDD, and automated regression testing.

In the end, it's amazing how well a simulation that runs on paper can capture the intricacies of software development.