



Stephen Manley
Manager of Application Development

Certified Agile Leadership II

Agile Approaches to Change
Experience Report



Evolve Agility

© 2023 Evolve Agility, All Rights Reserved.

Agile Approaches to Change: The Avengers

Why The Avengers?

When we started the transformation from waterfall to agile methods for software development, I had a big job to do. There were half a dozen teams to work with, and everyone needed hands-on guidance and assistance. So, when the supervisor of our Oracle eBusiness Suite (OEBS) team told me that OEBS as a technology really wasn't suited to agile development because of its tight coupling with the database and the way changes had to be implemented, I believed her and let that group go its own way while I focused on our JAVA development teams.

Two years into the Division's transformation, that supervisor and her team were ready to try agile after all, but with some fairly strong reservations about Scrum or Kanban. As a consequence, they developed their own methods of working that they called Scrum, and really believed were Agile. However, they began to express dissatisfaction with some of their own behaviors and felt unable to change through inspection and adaptation. This, coupled with my own observations of team behaviors that were clearly Scrum anti-patterns, led me to choose The Avengers as my target for change through the CAL2 program.

Team-driven vs. leader-driven change

It is a foundational tenet of Agile that Motivated Individuals working in Self-Organizing and Managing teams provide better outcomes. There are many studies over the past several decades that show that self-management, and the personal belief that someone is a respected, contributing member of a team doing work she is proud of, is a key driver of engagement. Therefore, it is vital to the success of any effort toward changing a team's behaviors that the team be included in conversations about observed behaviors and outcomes and planning for possible solutions.

This becomes more of a challenge if the team does not recognize the need to change itself. If that is the case, management needs to lead conversations with the team in such a way that they come to recognize that:

- The team's process is different from other teams, and from Agile/Scrum norms
- These unique processes should at least be thoroughly examined
- They may be the cause of at least some of the team's challenges

With that understanding, the team can engage in Experiment planning for future change.

Understanding when change is needed

- Intro discussion
- Leadership Agility Outcomes Interviews

When I first met with The Avengers in the CAL2 context, I explained a little about my course of study, and asked the team's permission to work with them to help change behaviors or processes that they would recognize as being problematic. They agreed to work with me, and we spent half an hour or so brainstorming problems that they saw with the way the team, and team members, were behaving. After identifying a few challenges, I told the team that I would be meeting with each one individually to continue the conversation, and we finished the conversation.

I held Leadership Agility Outcomes Interviews with each developer on the team, and the scrum master. After a quick introduction and reminder about the conversation we had before as a team, I walked them through the Leadership Agility Outcomes Interview form, and we filled it out together. Here is a sample from one of the conversations that was held:

Name: [REDACTED]	Role: Developer	Team: Avengers
-------------------------	------------------------	-----------------------

Tell me about a time when practicing Agile was challenging, inconvenient, or frustrating.

Sometimes the stories are not getting completed within the given time; sometimes they take a lot longer to complete than the team thought; there can be technical issues, or can require a LOT of testing because of the volume of data being dealt with.

Sometimes the team and the product owner may not get all of the requirements when the story is refined; new requirements come out during UAT.

When I develop some stories and it's ready to test, the tester may not be ready for it because he's testing everything. So the story waits for tester availability, and could be running out of time in the sprint.

What were you trying to get done?

[Job statement = verb + object of the verb + context clarifier]
 [Example: Listen ... to music ... when cooking]

complete stories within the estimated time
Identify all requirements prior to starting work on a story
test stories as soon as development is completed

What were your desired outcomes – how were you trying to improve the situation?

[Outcome statement = direction of improvement + unit of measure + object of control + context clarifier]
 [Example: Minimize ... frequency of ... undesirable songs being played ... when listening to music]

Importance	Outcome Statement	Current Satisfaction
8	decrease the number of things not taken into account during estimating	9
8	minimize the number of changed requirements after completing work	6
9	reduce the number of days stories wait to be tested	5

Importance: On a scale of 1-10 – How important is it for you to meet this outcome?
Current Satisfaction: On a scale of 1-10 – Degree of current satisfaction, as the way things are today.

I asked each team member for three challenges they had noticed and would like to see the team address. Here is a summary of all interviews:

Role	Team	Topic Area (tags)	Outcome Statement	Importance	Satisfaction	Opportunity Score	Job to be done
Scrum Master	Avengers	Norms	decrease the cycle time for stories in the sprint backlog	9	4	14	Team members to pull the next story from the backlog instead of waiting to be told
Developer	Avengers	DOR	Increase the amount of documentation available for research	9	4	14	Complete analysis without bothering other members of the team
Developer	Avengers	Norms	reduce the number of days stories wait to be tested	9	5	13	test stories as soon as development is completed
Scrum Master	Avengers	DOR	Increase the accuracy of the estimate for each story	8	5	11	Accurately or completely Estimate stories during backlog refinement
Sr. Developer	Avengers	Process	minimize the number of unplanned tasks that come in during the sprint	8	5	11	Manage unplanned work that comes in during the sprint
Developer	Avengers	DOR	increase the number of acceptance criteria for each story	9	7	11	Create stories with appropriate, thorough acceptance criteria
Scrum Master	Avengers	DOR	Increase the number of stories the team understands before refinement events	7	4	10	Analyze stories before backlog refinement
Sr. Developer	Avengers	Process	decrease the number of interruptions during the sprint	8	6	10	Focus on the most important task until it is finished
Developer	Avengers	DOR	minimize the number of changed requirements after completing work	8	6	10	Identify all requirements prior to starting work on a story
Developer	Avengers	Process	Increase the flexibility of the team's planned work for the sprint	8	7	9	Complete support issues without disrupting the plan for the sprint
Developer	Avengers	DOD	Increase the amount of functional documentation completed for any new	8	7	9	Complete stories with proper documentation included
Scrum Master	Avengers	Process	Increase the time available for UAT before the end of the sprint	6	4	8	Complete UAT by the end of the sprint
Sr. Developer	Avengers	Norms	Increase the number of stories that developers claim without being told	7	6	8	Developers to take responsibility for claiming and completing stories
Developer	Avengers	Process	decrease the number of things not taken into account during estimating	8	9	8	complete stories within the estimated time
Developer	Avengers	Process	Increase the technical understanding of P&F employees so they can solve some of	7	8	7	Complete stories during the sprint

I went into the interview process with some trepidation – I was more than half convinced that the team would tell me that they didn't want my help, didn't think they had any challenges to work on, and didn't have the time to spend with me. Thankfully, none of those turned out to be the case. This was one of the early "learning moments" for me in this course. I had spent time ahead of meeting with the team convincing myself that I was in for a real battle with them. What I found was a team that, far from being smug and self-satisfied, had a healthy dose of self-awareness:

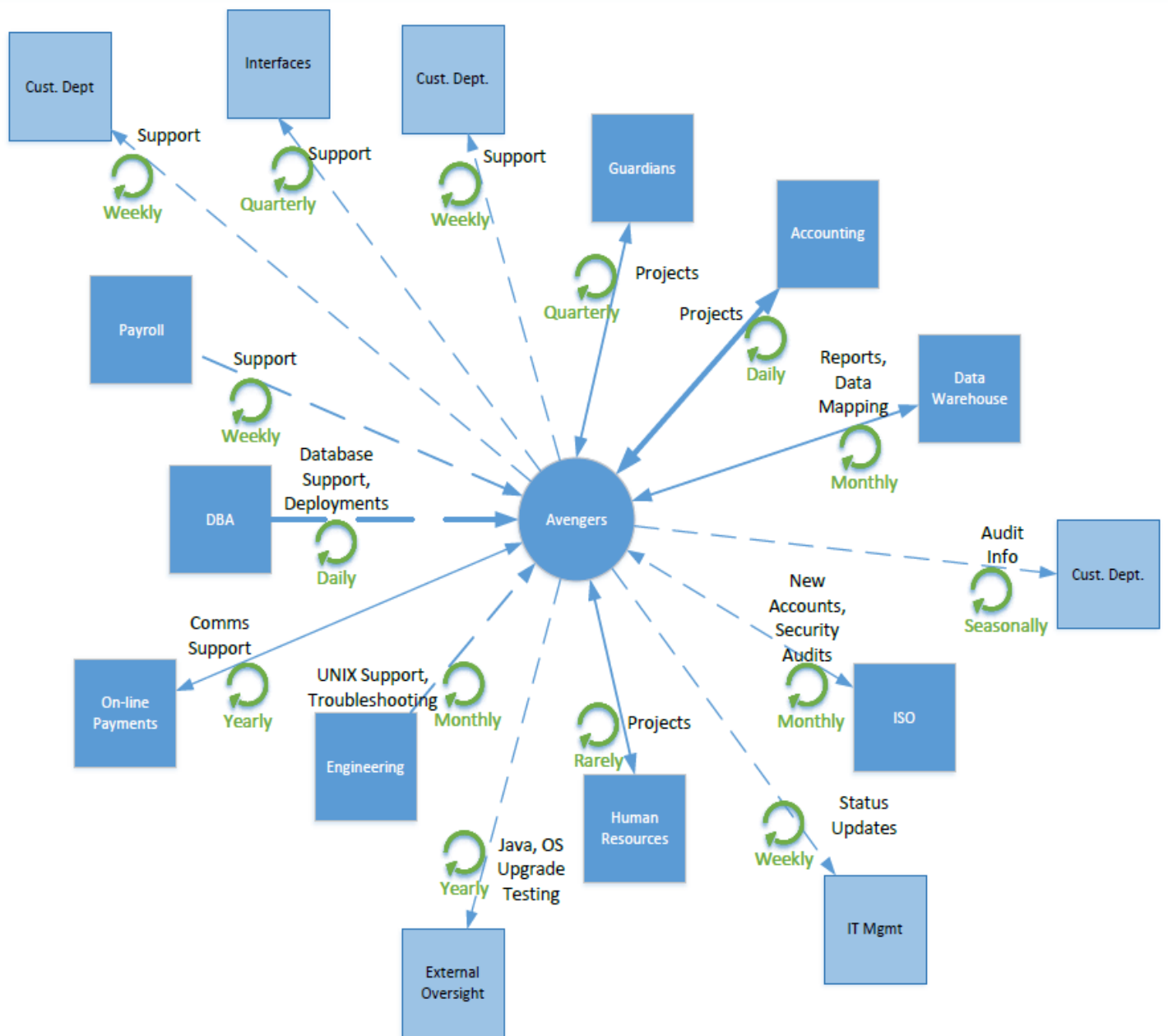
- They knew there were challenges they needed to address
- They knew they could be better
- They were able to identify specific behaviors that were causing them problems, and they agreed on what those were

What they didn't know was how to make things better.

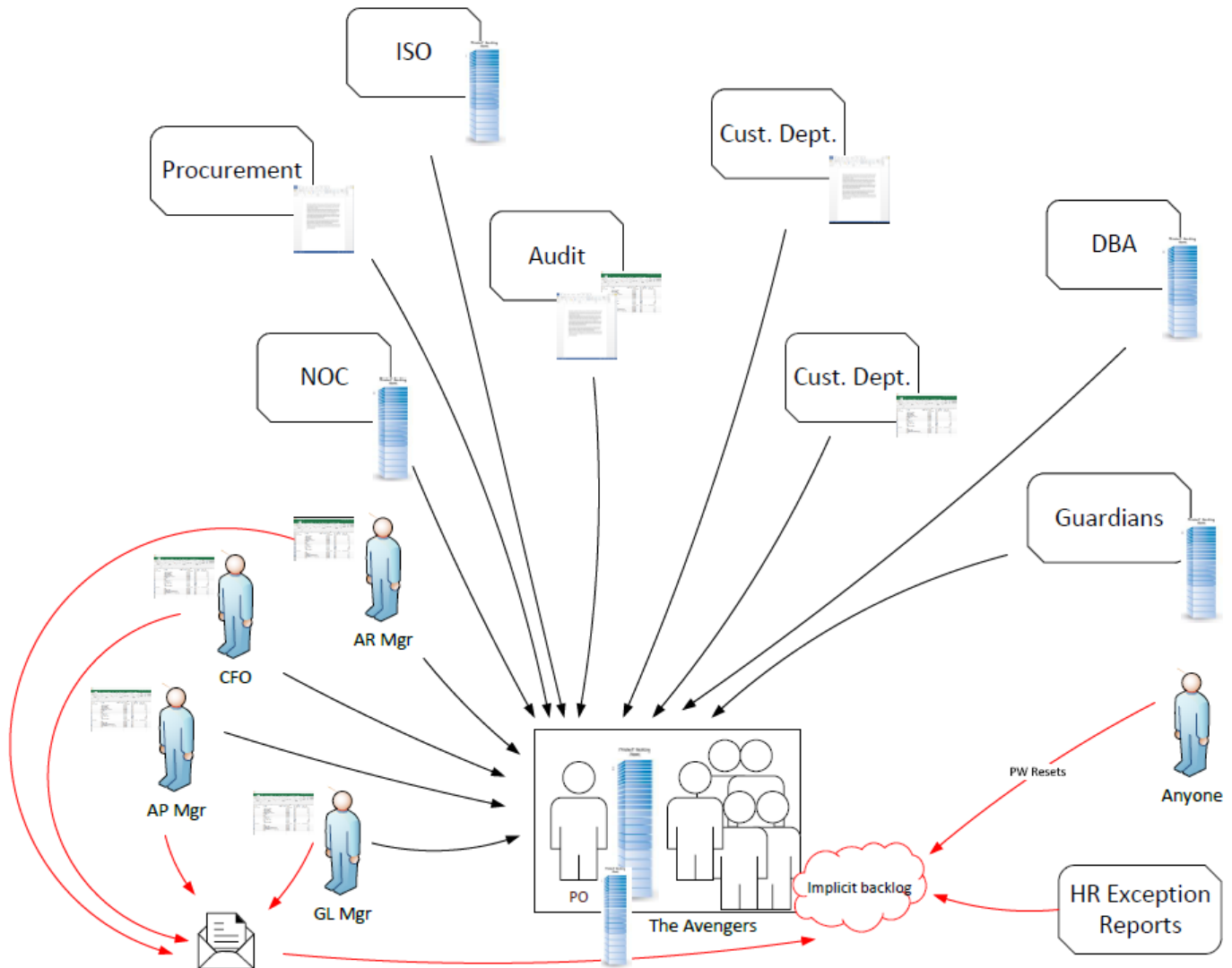
Mapping current state

After capturing the team's thoughts on what they felt should be changed, we completed two more team discussions wherein we completed a Team Interaction Map and a Team Backlog Map, which are shown here:

Team Interactions



Team Backlogs


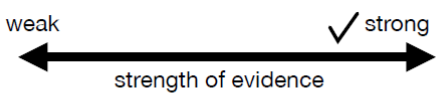


The team's response to these exercises was an understandable thought – “man, we're really busy!” My question to the team upon completion of these diagrams was, “Has this changed your mind about anything you think we should address, or do we need to add any other rows to our Jobs and Outcomes Summary?” The team decided to continue with the priorities they had already established.

The First Experiment

After completion of current state mapping, I met with the team to determine finally what our first experiment would be. We reviewed the Jobs and Outcomes Summary and agreed to tackle one of the two items with the highest Opportunity Score: completing more technical documentation so the individual team members could do more of their own research without interrupting each other. I completed an Experiment Canvas with the hypothesis that adding a requirement for completion of technical documentation to the Team's Definition of Done would improve this Outcome.

Here is the Experiment Canvas:

Title: <i>Concise, self-explanatory</i>		Date:	 Evolve Agility
Pre-Conditions <i>What is necessary to run this experiment?</i> <i>Baseline measures? (qualitative/quantitative)</i> 1. Team process and quality documents (Norms, DOR, DOD). 2. Count of technical documents now being created. 3. Evaluation of quality of documents now being created.	Falsifiable Hypothesis <i>IF ...Specific testable action...</i> <i>THENSpecific measurable outcome.....</i> If the requirement for the creation of technical documentation is included in team Definition of Done, the team will create more technical documentation.	Results/Post-Condition <i>Record qualitative/quantitative measurable outcomes</i> After four weeks of work with the updated Definition of Done, the team has completed two stories requiring technical documentation. The documentation was completed for each story.	
<hr style="border-top: 1px dashed black;"/> Assumption <i>We believe that ...</i> 1. The team follows all standards as set forth in the Definition of Ready and the Definition of Done.	<hr style="border-top: 1px dashed black;"/> Experiment Setup <i>What kind of experiment?</i> <i>What are you measuring?</i> <i>How are you measuring?</i> <i>How often?</i> 1. Update Definition of Done to include specific requirement for technical documentation to be completed before a story is "Done". 2. Review all stories completed in the coming sprint for technical documentation. 3. Count number of stories that have documentation and the number that don't.	<div style="text-align: center;">  </div> <hr style="border-top: 1px dashed black;"/> Conclusion <input checked="" type="checkbox"/> Validated <input type="checkbox"/> Invalidated <input type="checkbox"/> Inconclusive	
<small>This work by Dhaval Panchal is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-sa/4.0</small>			

I then reviewed the team's Definition of Done with an eye toward making the edit suggested by the experiment. However, I was surprised to discover that there were other significant challenges with this document.

Though each team is responsible for creating its own Definition of Done, there are some "universal standards" that have developed around this document, including the idea that it should represent, or define, the minimum work required to meet a team's agreed-upon quality level (emphasis on the word "minimum"), and that it is short enough to be readily understandable, memorable, and usable for every story. The Definition of Done that I found for The Avengers failed on all counts. It was a page and a half long, was not visible or used (or useable), and had not been referenced by the team in over two years. In fact, it already included the requirement for technical documentation to be completed for each story.

In response to this discovery, I met with the team again and we reworked their Definition of Done into a useable document. Several items in the original belong more in a Definition of Ready, if they belong anywhere; several others might belong in Team Norms; and others would fall into "that's just the way we work", rather than being codified anywhere. When we finished, the new Definition of Done went from a list of thirty items down to seven, including the requirement to complete technical documentation on any story which requires software development. The team agreed to begin referring to their new DoD on every story in their next sprint.

1st Month's Results

After a month with their new Definition of Done, the team reports the following results for our first experiment:

1. Two stories required technical documentation to be created. In addition, the new Definition of Done was referenced for every story that was completed in the Sprint
2. The team needs to learn to adjust estimates based on documentation requirements – as expected, stories took longer to complete; however, part of the extra time was creating a template for the new documentation, so they expect future stories to take less time
3. The team is creating a space in Confluence to hold these documents, and will create folders for each Oracle eBusiness Suite module over time

The team still thinks the documentation will be helpful to them as they do more development in the future. They plan to add onto these documents as future changes are made to the same components. They feel that the changes to process will help them to be more efficient with future development efforts. Even though this is a small sample, the team is happy with the result of the test.

Next Steps

With a positive first month's result, the team is prepared to continue using their new Definition of Done for upcoming sprints. This will give us ample opportunity over the upcoming weeks and months to verify that technical documentation is still being completed. As the library of technical documentation grows, the team will be able to verify that it is as helpful for future development as they expect it to be.

In the meanwhile, there are several other steps the team has agreed to take:

1. Review their other foundational documents – the Definition of Ready and Team Norms – with an eye toward optimizing them for regular use by the team
2. Begin the next experiment based on findings in the original Leadership Agility Outcomes Interviews; as we start to look to the future, the team will revisit their original Leadership Agility Outcomes results to verify whether they are still relevant, or if there is something else more pressing they would like to start on instead
3. Revisit and examine more closely the Team Interaction Map and Team Backlog Map that they created; there is almost surely valuable insight here beyond “man, we’re really busy.”

With early success for The Avengers, the team is committed to continuing their Inspect and Adapt journey. They are learning that they can improve, and that there are tools available to help them ... and I am learning new ways of engaging a team in an ongoing conversation that can build real consensus and momentum for positive outcomes.

Certified Agile Leadership II



Leadership Development Program

The CAL II accreditation is rigorous and requires evidence of validated practice in real-world situations.

This has been a featured case study from successful leaders who participated in our cohort. Our program ensures that leaders acquire practical skills to apply agile leadership knowledge in their organizations.

To join our cohort or to find out more, simply contact us info@evolveagility.com

