The Lean English Podcast

Episode 4: Agile Project Management

1. Episode 4: Agile Project Management

Welcome to the fourth episode of The Lean English Podcast. My name is Tom and I will be your host. Thank you for joining me. This is the last episode in a 4-part series on Lean, Six Sigma and Agile.

If this is your first time joining us, here's a quick backstory. Before I moved to France and became an English professor, I worked as a Quality Manager and Lean Six Sigma consultant for many years. For variety, I use podcasts to learn French vocabulary and pronunciation. So, as a supplement to the other language sessions I offer my students, I thought a podcast would be nice, particularly if it included other valuable information. The goal of this podcast is to provide comprehensible input at a slow pace for non-native speakers who are also looking to improve their business or their careers. If you are interested in learning the tools and philosophies of the most profitable businesses in the world, you are in the right place.

2. In case you are one of those people who likes to listen to things out of order, and instead of starting with episode 1 you started with episode 4, here is a recap of the episodes thus far.

The first episode was a high-level introduction to my company, the podcast and process improvement methodologies. I provide English language training for companies and individuals, in person and online. That training is focused on helping my students receive a language certification by passing the TOEIC exam. But the difference is, I use business lessons as the material for the course. If you work in an international company that uses English and need

help solving productivity and quality problems, give us a call. We will tailor a training program fitted to your level of language and technical expertise. And if you are in France, we are Qualiopi certified, so any of our courses can be paid for by your CPF.

Then we talked about the history and the main contributors of Lean, Six Sigma and Agile's development. In the early 1900's, we saw an introduction to scientific business management that changed how companies do business. Engineers like Frederick Taylor, Henry Ford, Walter Shewhart and Sakichi Toyoda developed ideas that would be the foundations of Lean, Six Sigma and Agile. They saw how the old way of business was detrimental to quality and productivity, and management needed to change. Six Sigma was developed by engineers at Motorola in the 1980's to provide more statistical rigor to problem solving. And we will see in this episode how, in the early 2000's, Agile used Lean's principles to offer a better way to develop software.

After our history lesson, we listed the similarities and differences between the methodologies. Depending on your business, the technical expertise of your team and the complexity of your problem, you may choose one method over another. It is important to get this decision right because business leaders rarely get a second chance to change how the company works.

For the second episode, we focused solely on Lean. Lean is a collection of philosophies and tools that developed over a century of innovation and collaboration. We see throughout the decades that new principles and values are added as new problems arise. Sakichi Toyoda's initial work with automatic looms inspires the Lean

principle *Jidoka*, the idea that defects should be easy to identify and work should stop until the problem is resolved. Kiichiro Toyoda followed in his father's footsteps and started the Toyota Motor Company. Lean is synonymous with Toyota motor company's production system called TPS because many of the principles were developed by Toyota.

I suspect one of the reasons it was labeled "Lean" is because they try to identify and remove waste from all the company's processes. Waste is essentially any activity that does not add value to the customer.

We covered the three types of Lean improvements; Kaizen, Kaizen Blitz and Kaikaku. They offer a range of options from small, daily improvements to radical change.

One of the benefits of Lean is that the problem-solving tools are easily understood and universally adapted to any business. Everyone in the company gets engaged in improving their processes and the high tide raises all boats. This means that everyone in the company benefits from using Lean problem-solving tools.

During the third episode, we highlighted the structure and phases of Six Sigma. Six Sigma is a project management method that uses statistical analysis to reduce the variation in processes. As processes and products become more technically advanced, slight variations can have an enormous impact on quality and customer satisfaction. These can only be detected by using highly sensitive measurement equipment and powerful statistical tools.

Many practitioners use both Lean and Six Sigma tools during their projects as they complement each other. For example, Six Sigma uses the same ideas that Lean highlights; value is customer driven, solutions must address the root cause of the problem and removing waste is essential for faster product delivery. Also, many Lean tools like *Heijunka*, 5S and Visual Management are used during Six Sigma's Improve phase.

Six Sigma uses 5 phases to reduce variation and create value; Define, Measure, Analyze, Improve and Control. This toll gated project management structure means they must show their completed work to the Project Champion before being allowed to move forward.

Each phase has its own goals and requirements. And as the project progresses, the team learns about the problem, identifies the root cause, creates and tests which solution works best and controls the new process once it is put in place.

The purpose of the extensive statistical analysis is to prove a direct link between our causes and effects, and to test which solution provides the greatest benefit. It can be difficult to learn it all, but projects can provide significant improvements if performed correctly.

3. This episode in two parts will concentrate on Agile project management, and particularly the style called Scrum. As we will see, there are many different forms of Agile, each with its pros and cons. But I chose Scrum because it is the easiest to understand for new learners. Also, it incorporates many of the same tools as the other disciplines. Just to cover the basics, we will briefly talk about

Waterfall, the Agile Manifesto which gives us the 4 Values and 12 Principles of Agile, the 3 Pillars of Scrum, the Scrum project lifecycle and the Planning, Review and Retrospective meetings, as well as the team roles and responsibilities. As with the previous episodes, this is a high-level review of Agile and is not intended to be a guide for implementation. We plan to cover these topics and tools in more detail in future episodes.

4. But if you have listened to my previous episodes, you understand how I love to start with background information to put the material into context. This slide covers Waterfall Project Management. It is called Waterfall because it looks kind of like water flowing down multiple steps. But in this case, work flows from one stage to the next only after leadership determines that sufficient progress has been made, much like a Six Sigma project.

For example, the work of requirements gathering involves many meetings with the management team and the customer to determine what exactly needs to be created. Traditionally, the company negotiates the contract on the work that is being requested. They must be sure that everything is accounted for so the price can be as accurate as possible. Everything is extensively documented because any dispute or legal action will be judged by what was put in writing. Once requirements are gathered and the contract has been negotiated and signed, then the team can begin the project work. At this point, the customer leaves everything in the capable hands of the developers and waits for the call saying everything is ready to deploy. They offer very little support or additional information because any changes to the requirements

must be approved by a committee, further delaying the project and creating conflict between the developer and the client.

Waterfall works well for traditional projects like buildings because the customer would know exactly what they want, the company would know exactly how to build it and nothing much would change from contract to delivery. Unless you are building something exotic, there are not many unknowns when it comes to designing or acquiring the raw materials needed to build a house. Unfortunately, Waterfall doesn't work for software development because the market for software changes so rapidly that the customer's idea they are trying to sell may be obsolete or redundant before all the requirements are compiled. Merely documenting the requirements and all the points during the negotiation can take weeks or months. By then it is too late. Also, customers often do not have all the details about what they want, and developers will be creating something that has never been developed before. Software development is full of unknowns.

5. To combat this, in 2001, 17 software development experts wrote and signed the *Manifesto for Agile Software Development*, also known as *The Agile Manifesto*. This one-page letter is considered the key reference for all Agile practices. The goal was to create a document to show companies and developers that there is an easier, more inclusive way of delighting customers. It outlines 4 values and 12 principles.

On the right side of the slide are the 4 values. We quickly see that the new way of working is valued above the old ways that were

carried over from the traditional Waterfall style of project management. We prefer individuals and interactions over processes and tools. Throughout Agile is the promotion of open communication and teamwork. Individual developers add value to the product; processes and tools do not. Agile believes that the individual developer will find a way to produce a high-quality product, regardless of the tools they use.

Working software over comprehensive documentation. Waterfall project management is famous for documenting requirements, creating plans, and formalizing schedules. In the eyes of the customer, this is all non-value added. The customer is paying for working software, not management documents.

Customer collaboration over contract negotiation. One of the reasons Agile teams can get started so quickly is because the customer is part of the team and provides feedback throughout the project. They don't have to spend months on requirements gathering because they know the requirements will likely change as things progress and they can just ask the customer for feedback. Contract negotiations prolong the pre-project work and make an adversary out of the customer.

Responding to change over following a plan. Agile must be adaptive, so they created an iterative cycle of product development that we will talk about later. Plans take a long time to create, and once they are fixed, are very difficult to change. Unfortunately, technology changes so rapidly that fixed plans become obsolete very soon.

6. Now that we know Agile's values, let's talk about the Agile Manifesto Principles and how they changed the way projects are managed.

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. Waterfall takes too long to develop software because the work only begins after contract negotiations, requirement documentation, project planning, team building, etc. And even after the work starts, the customer sees the product only after the product is completed. Agile remedies this by prioritizing working software, not project plans. The team starts building after they have just enough planning to complete a minimally viable product. This greatly shortens the development process.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. With Waterfall, the requirements are set in stone and do not change unless there is a significant request from the customer. And even then, the plan must be altered, which changes timelines, costs and expenses. All of this leads to more negotiation and more delays. Agile teams prefer to work on small chunks of requirements during a Sprint to adapt quickly when the customer changes their mind. This is critical because the computer and application marketplace is rapidly changing. What was cutting edge one year ago is now obsolete. So Agile teams can adapt to this change better than teams using Waterfall.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

This keeps the customer engaged in the project and allows the team to build a product that is continually tested. Agile teams incorporate quality control into their daily workload by using best practices like Test Driven Development and Continuous Integration. This ensures that the new code is free from bugs and integrates with the program that already exists.

7. Businesspeople and developers must work together daily throughout the project. As the developers are the only people in the company adding value to the product by writing high quality code, everyone else does what they can to help them throughout the project. Furthermore, if there is open communication between these groups, it is less likely that the sales team will make promises that the development team cannot keep.

Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done. In Agile, developers are given a lot of freedom to schedule and structure their work, but they are held responsible if goals are not achieved. Therefore, they must be motivated to complete their work without too much oversight.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. As requirements and expectations change quickly, there is no time to write emails, wait for a response and then take action. Action needs to be taken today, now. Also, there should be no ambiguity about the message. If we are communicating face to face, we can give feedback, respond to questions, come to an agreement and have a better understanding of what to do next.

8. Working software is the primary measure of progress. Again, all value is viewed from the customer's perspective. They don't care how many meetings you have or if your leadership team is up to date. They want tangible evidence that the developers are going to deliver a working product on time. And the best way to do that is to show them working software as it is created.

Agile processes promote sustainable development. The development team should be able to maintain a constant pace of work indefinitely. This aligns with the Lean principle of *Heijunka*, leveling the workload. There should never be a rush to complete work or a scramble to meet a deadline as this creates unnecessary stress and defects. Steady development also helps the team predict their future performance. If the team alternates between working overtime and doing nothing, it will be difficult for them to make accurate promises to the customer.

Continuous attention to technical excellence and good design enhances agility. Agile offers many techniques, such as pair programming, to ensure the code is correct each day and defects are kept to a minimum. Good design goes hand in hand with our next principle, simplicity.

9. Simplicity – the art of maximizing the amount of work not done – is essential. Einstein said, "if you can't explain it simply, you don't understand it well enough." The more complexity in the code, the harder it is to understand, modify and debug.

The best architectures, requirements, and designs emerge from self-organizing teams. In Waterfall, the project manager determines the workload, but not so in Agile. Agile teams are given freedom to determine how to get the most work done. This knowledge is very valuable as it helps the team determine their own daily priorities and schedules for how to do the work. The more teams work together, the more they develop the processes that work best for them given their strengths and weaknesses.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. This aligns with the spirit of Kaizen, adopted from Lean. There are mandatory meetings after each sprint that provide a safe environment for discussing what went well and what did not go well. This is another use of face-to-face communication in Agile.

10. Here is the Scrum Project Lifecycle. When the project starts, the Product Owner works with the customer to create the product backlog. This is the list of all the major functionality and features the customer needs for their product or application. A few things are special about the product backlog. In the beginning it is not completely refined, but the team starts when they have just enough documentation to get to work.

The backlog is written from the viewpoint of the end user in the form of User Stories. User stories are formed like this: As an X, I need Y to help me do Z. As a web designer, I need a dashboard to help me see the work progress for all the webpages on the site at once. This format allows enough flexibility for the team to determine the best way to develop that functionality.

Then the Product Owner shares the backlog with the scrum master and team. Depending on the size of the project, the backlog is prioritized into Releases or large portions of the whole project with similar functionality. This Release Backlog helps the team prioritize the work of the most important functionality first.

They then divide the releases into iterations of manageable work, or sprints, 2-4 weeks long. The team selects which tasks to work on during each sprint. Here the sprint backlog user stories are further

broken down into manageable tasks that the team works on during the day. This principle aligns with Lean's idea of One-Piece Flow; only one task is worked on at a time until it is completed, tested and integrated.

During the sprint, there are many important meetings. Each day there is a stand-up meeting where the members share with the team and the scrum master what they plan to do. The sprint review happens after the sprint. It is when the functionality is demonstrated for the customer. This is when the team shows what they have or have not completed. The customer's feedback from the sprint review is then used by the product owner to reprioritize the backlog and give the team a better sense of what the customer wants.

After the sprint review, the sprint retrospective is time for the team to identify ways to improve their performance. And then the process repeats until the work is completed for a release. And then the release cycle repeats until the customer is happy with what has been produced and accepts delivery.

11. This is a high-level view of an Agile release cycle. Beginning with the customer requirements, the team prioritizes the items from the product backlog to plan the sprint, develop, test and deploy the Minimally Viable Product and review it with the customer. They use the customer's feedback to update the product backlog, and the cycle starts all over again. This sprint cycle repeats many times until

the customer accepts the Minimum Marketable Feature at the end of release.

We will discuss the differences between the Minimally Viable Product and Minimum Marketable Feature in part 2 of the Agile episode. But for right now the Minimally Viable Product is a proof of concept, a small piece of functionality that can be created in one Sprint. The Minimum Marketable Feature is functionality that can stand on its own as a product or application the customer can use.

12. And here is the lifecycle for releasing the entire product. There may be multiple releases before the final product is delivered to the customer and the project can end. However, regardless of the number of releases, the team still follows the same pattern for every release; prioritize the most important functionality according to the client, create an MVP during a Sprint, release the MMF, and repeat until all the functionality has been delivered.

13. Now we will start talking about the specifics of Scrum. These are the three pillars of Scrum. Being transparent with the customer, leadership and the team members is the first. Transparency means there is nowhere to hide on an Agile project. The team must demonstrate at the end of the sprint in front of the leadership team and the customer whether they have fulfilled their work pledge. Teams work in shared spaces and update information radiators for everyone to monitor their progress.

Everyone agrees to open testing of the code they produce, and work is inspected continually during the sprint. Team members are open to finding errors rather than covering them up. Agile companies understand that mistakes happen, so self-diagnosis is looked upon favorably.

Adaptation creates a mindset that helps the team move forward without sentiment for the past. Technology develops quickly, so adapting to the market demand is necessary for survival. Agile teams embrace change and know this ability gives them an advantage over traditional development. This is all possible because they only commit to a Sprint's worth of work at a time.

14. The Sprint Planning meeting is scheduled before very Sprint, when the development team, scrum master and product owner select and prioritize which functionality will be developed. The meeting is time-boxed to be 1 to 2 hours per week of Sprint. If your sprints are 3 weeks long, expect a 3-to-6-hour meeting. Agile uses timeboxing to limit the time the team spends in meetings. The Definition of Done must be clarified so the team understands exactly what they need to do. The team chooses how much they can accomplish and must pledge to complete the work they select.

15. The Daily Stand-up meeting is conducted before each workday. To limit interference, it is only attended by the Scrum Master and the development team. To promote work transparency, each team member gives 3 updates: the tasks they completed yesterday, the

scheduled tasks for today and the issues that require Scrum Master support. The Scrum Master uses the information to update information radiators and calculate the team's velocity; the rate at which they complete work. The meeting is time-boxed to 15 minutes. If larger issues arise from the meeting, they are discussed offline.

16. The Sprint review is held after every Sprint to show the customer the workable software that the team created. It is attended by the Customer, Product Owner, Scrum Master, Development Team and other potential stakeholders and sponsors. It is time-boxed at 1 hour for every week of sprint. This meeting serves several purposes. It holds the team responsible for the work they agreed to create. It shows the customers that they are valued and a part of the team. It offers the Product Owner an opportunity to communicate face to face with the customer and receive the feedback necessary to further refine and groom the product backlog.

17. The Sprint Retrospective is conducted after every Sprint Review and is an opportunity for the team to talk about the Sprint. The meeting is time-boxed to be 45 minutes per week of Sprint. It is attended by the Scrum Master, Development Team and possibly an External Facilitator. The focus is to make issues visible so they can identify improvement opportunities. They all share their opinion of what went well, what went poorly, and what the team needs to do to improve the next Sprint. The team takes ownership of their performance and seeks help where they need it. This represents

Agile's commitment to the Lean principle of continuous improvement.

18. Agile asks leadership and team members to take specific roles during the project. Traditional project management roles are not applicable here, so training and the right mindset must be reinforced and incentivized by a supportive culture. For the Product Owner, they work for the development company, but they represent the demands of the customer. They must be able to drive the customer's vision, requirements and priorities. They spend most of their time grooming the Product Backlog and elaborating the requirements, so they are easy to understand by the development team.

The Scrum Master is responsible to the team, and works to help them improve their performance. This involves reducing the interference from leadership and providing training when necessary. They act as Servant Leaders, removing obstacles so the team can perform at their best.

The Development Team is best when they are a small group of five to nine engineers who are all able to perform different tasks. They are self-organized, closely linked and often paired. They must work together, in the same space and on the same code, so there must be a strong culture of open communication and collaboration for the project to be a success.

19. Well, this looks like a good place to stop. We will complete our brief Agile course in part two. As always, if you have any questions or need help with your language or business training, send me an email and I will reach out to you soon. Thanks so much for listening. Until next time, be good and I'll talk to you later.

1. Part 2 of Episode 4: Agile Project Management

Welcome back to the Lean English Podcast. I am your host Tom, and this is the second part of our episode on Agile. If you are just joining us, in part 1 we covered the differences between traditional Waterfall project management and Agile. We talked about the Agile Manifesto's 4 Values and 12 Principles, the Sprint and Release cycles and the basics of Scrum. So, check that out if you are interested.

2. For today, we will finish our discussion of Scrum by outlining each roles' duties. Then we will highlight other types of Agile, like Xtreme Programming and Lean Kanban, before we introduce you to Information Radiators, the Product Backlog, and discuss how companies should prioritize the services they provide. Before we end, we will define the Minimally Viable Product and how it differs from the Minimum Marketable Feature. So, we have a lot to cover and not much time to do it. Let's go.

3. As I said before, the Product Owner is the customer's representative inside the company. They provide the vision of the product. They gain consensus from the team on the best way to deliver value as soon as possible. They ensure the Product Backlog

is created and is elaborated for the team's understanding. This means they can put the team in the customer's shoes and show them the value in the features. They set reasonable priorities and expectations for each Release and Sprint. While they do not get involved in day-to-day development operations, they set the goals that need to be hit. They collaborate with and work through the Scrum Master to help the team understand the market implications and delivery window. Sometimes a release needs to be synchronized with other events, and any delays would be detrimental to the client. The Product Owner defines the parameters for the Minimally Viable Product and helps the team understand the Definition of Done.

4. The Scrum Master is trained to be an expert in team management, not necessarily in software development. Their role is described as a Servant Leader because they are responsible for enabling the team and maximizing their output. They need to be humble and let the team get the glory without letting ego get in the way. They need to be collaborative and not competitive with the team, looking for win-win situations. They must be committed to the team even if they are not full-time Scrum Masters. Scrum Masters have no formal authority, so they must find other ways to be influential. They must negotiate and leverage their influence to get the things they need. Although they are not technical experts or coders, they are experts in Scrum and understand the technical environment.

5. Unlike other project management teams, an Agile Development Team has much more control over their work. But with that control comes added responsibility.

First, they must commit to delivering during the Sprint and reject the desire to gold-plate the work. They need to be comfortable receiving feedback and sometimes criticism from the client during the Sprint Review and putting their opinions aside.

Second, the team suggests which stories to work during the Sprint Planning Meeting, provides time estimates of each User Story, and finds the simplest ways to develop functionality.

Third, they must remain present with the team. Co-locating with the team improves osmotic communication and develops team camaraderie.

Fourth, Agile prefers Feature Teams over Component Teams. This means hiring developers who are good at everything is preferable to hiring developers who are great at only one thing. In Agile, a feature is an end-to-end functionality built by cross-functional teams, while a component is a modular, self-contained piece of code performing a specific function within a larger system. Feature teams are critical to Scrum success because they don't just focus on how one component works; they focus on how the component they are developing functions inside the whole system.

6. We have talked enough about Scrum. Now, let's find out a little about other types of Agile. While they all follow the values and principles of the Agile Manifesto, they choose to highlight different

tools and techniques to give Agile users more flexibility when dealing with certain requirements.

Xtreme programming provides a strong emphasis on the technical aspects of software development. There are too many practices to name here, so we will cover them in our next slide. Suffice it to say that other versions of Agile gladly borrow them for their teams.

Feature Driven Development focuses on delivering functionality to the client using the Domain Object Model. They use 5 activities to accomplish this: Develop overall model, build the feature list, plan by feature, design by feature and build by feature.

Lean Kanban blends Lean's philosophy and the Kanban pull system of work in progress (WIP) with Agile principles. Projects with a lot of uncertainty will have a Product Backlog that changes frequently. To be more responsive, Lean Kanban constantly updates the Work In Progress so the team can choose new tasks as soon as they finish their current work. This way they do not have to wait for the end of a Sprint to change the project's direction.

Crystal is a lightweight Agile approach that emphasizes team collaboration and limited documentation. Projects are color coded for size and risk. Clear and yellow for small projects with low risk; maroon and blue for large projects with high risk.

7. These are some of the best practices developed for Xtreme Programming that are adopted throughout Agile.

We talked a little about Test Driven Development already and how it accelerates software development by writing an automated unit test before coding begins. This way, developers can write the code with the test in mind.

Continuous Integration; as new code is written, it is immediately adopted into the whole for automated tests and to learn what functionality is still needed. Colored lights are used as *Andon* lights to show if the integration of the new code is successful or not. When the code is broken, the red light is illuminated. When the code integrates seamlessly, the green light shines.

In Pair Programming, two programmers write the same code; one writes, the other performs quality checks and suggests improvements. This helps junior developers learn better ways to write and design code from senior developers, reinforcing the team's camaraderie and communication skills.

Refactoring continuously improves code by eliminating redundancy and unnecessary functions and increasing code coherency. The more frequently refactoring is performed, the fewer problems the team must solve in the future and the easier the code is to maintain.

All these principles promote coding standards, when the team shares best practice formats and styles for writing code. As team

members learn from one another, their ability to solve problems grows.

8. Here is an example of a Lean Kanban board. The columns are labeled To Do, tasks that are waiting to be started, Doing, tasks that are currently In Work, and Done. The rows hold the tasks per project that move from left to right depending on their status. As I mentioned earlier, Lean Kanban Works best when there is a lot of change and uncertainty. The Product Owner can add tasks to the To Do column as requirements become available. This allows the team to always have work to do, reducing the time the team spends away from work/writing code. At the end of a Sprint, for example, the team may have completed all the work they committed to and will have nothing to do. And as an information radiator for leadership updates, provide a Kanban or WIP board to show the team's progress.

9. So, what is an information radiator? you ask. It is any chart or graph that displays information about the team's progress in an easy-to-understand format. As Transparency is a pillar of Agile, Information Radiators push the team to accomplish all the work they committed to during the Sprint. The pressure is on to meet their goal. As a waste reduction tool, these graphs allow the team to show their progress without interference from Leadership. Remember, update meetings are Non-Value-Added. Daily and weekly status reports to higher-ups are no longer needed as anyone can understand the team's progress in a matter of seconds.

10. The most popular type of information radiator is probably the burndown charts. They provide a daily update to the work that the team is performing. Here we see that the team has 120 units of work to complete for the 4-week sprint. The yellow columns that appear just above the date indicate how much work the team accomplished on that day. The orange line indicates how much work is remaining. The blue line shows the amount of work that should remain if the team is completing tasks at a steady pace over the sprint. Any columns that appear to go into the negative indicate the amount of rework that the team must accomplish. You can see a corresponding rise in the orange line for those dates. This chart provides enough information to satisfy the desire for management to get an update on the team's progress.

11. Now that we know how work is performed and assigned, let's talk about how the Product Backlog is constructed since it is at the heart of Agile's requirements processing.

Agile teams need to be able to see the forest for the trees; that is to say, they should see how the User Story they are developing fits into the larger scheme of the product. Remember, User Stories are the main unit of functionality for the project; As an administrator I need a shared calendar to organize company meetings.

One way to do this is through Themes and Epics. A Theme is a set of related user stories that can be combined and treated as a single entity for ease of estimation and release planning. Epics are large user stories with low priority. They are too big to implement in a single iteration and therefore need to be disaggregated into smaller

user stories at some point. And User stories can be split if they are too large to handle in one iteration. Those stories in turn will be broken down into tasks, which are easier to estimate and complete.

For example, imagine you are developing an ERP system for a company. ERP stands for Enterprise Resource Planning. The theme may be; As a CEO, I need an ERP system to run my company efficiently. An Epic may be the part that allows the Operations Manager to run the supply chain, with a User Story being the piece that allows the inventory manager to keep track of all their parts.

12. Now that we know how the Product Backlog is constructed, let's discuss what the team needs to complete their work. As we discussed earlier, the Agile project begins with just enough information to get started and adapting to the customer's needs is one of Agile's pillars. The Sprint Review allows the Product Owner to complete the Definition of Done, an important technique that was brought over from Lean that defines the product requirements before the customer accepts delivery. In the beginning, the customers' ideas may be vague, but as they review the team's work, they refine their expectations and provide more detail. The Product Owner will define the final product, and the team must agree on the definition for each Sprint. All work must add value until the product meets this well-articulated definition. The only time that the team is allowed to "Goldplate" the product is after the minimally viable product has met the definition and the team has time to try to impress the client.

13. Now you ask, Tom, how does the team determine what will impress the client? Here are two tools that teams can use to brainstorm some ideas.

Expectation Confirmation Theory states that there are four types of customer experiences depending on their expectations. An experience can be negative or positive, depending on the outcome. Furthermore, the experience will lead to confirmation or non-confirmation.

For example, if you eat at the same restaurant and order the same thing every Friday, then you will have a positive confirmation experience if your meal is exactly what you expected and you have enjoyed it. Your expectation was confirmed. But if your meal is not what you expected and you did not enjoy it, it would be a negative non-confirmation experience. Your expectation was not confirmed. After telling a friend about your disappointment, they advise you to try a new restaurant, but you have low expectations. However, the meal is fantastic. You just had a positive non-confirmation experience because you got what you did not expect but really enjoyed it. The next week, you try another new restaurant. With low expectations, you order your meal and it is not delicious. So, you have a negative confirmation experience. Of these experiences, positive confirmation experiences give the customer what they expect; you meet the definition of done. Your customer is satisfied and will continue doing business with you. However, a positive non-confirmation experience delights the customer because they get

something they did not expect. This makes new customers loyal to you.

The Kano Model works in much the same way. It says there are 3 types of requirements. Basic needs are features fundamental to the product. All cars have seats, a steering wheel and an engine. But adding more of these features eventually reduces customer satisfaction. Your car only needs one steering wheel; two would be confusing. Then there are performance needs features. The more you get of these, the more satisfaction the customer has. Think of efficiency performance. The more miles you can drive per gallon of gas, the happier you are. Then there are the delighters. These are features in a car that the customer did not expect. Think of the first time you were able to use Bluetooth in your car. Or if you are old like me, the first time you drove a car with power windows. Creating Delighters is how we went from home phones to cell phones, and from cell phones to smartphones. Teams should work with the Product Owner to identify Delighters whenever possible so they can create positive non-confirmation experiences for their clients.

14. Now let's talk about two things that are confusing to some people; the Minimally Viable Product and the Minimum Marketable Feature. Developing a Minimally Viable Product is a way for the team to get important feedback from the customer. It is essentially an early model of the final product, very basic in terms of functionality, that tries to show the customer the direction in which the team is moving. It is not a piece of the whole, but a simple version of the whole, and helps the team complete the Build – Measure – Learn feedback loop.

So, in this example where a woman starts with a scooter, then rides a bike, then a motorcycle and finally a car, we see a progression toward the final product. The customer starts with a basic requirement; I need a mode of transportation that is faster and easier than walking. They don't know all the details about what they will eventually need, but this is how it begins. The product owner takes that requirement to the team and they build a scooter. The scooter works perfectly well and fits the requirements the team received. But the customer tests it and, although she likes two wheels and a handlebar, she finds that she would like to sit down while travelling. With this feedback, the team returns to work and produces a bicycle. Again, the customer takes it for a spin and realizes that she likes being able to sit down, but she would like to go faster. The team returns to work and decides they need to add a gas engine to the frame. This time they build a motorcycle for the customer. The customer gets on and starts to ride and immediately crashes it. She likes going fast, but this leads her to believe she needs something more stable. She likes sitting down when she travels, she still likes going fast and feeling the wind in her hair and the bugs in her teeth, but for her safety, and the safety of everyone else on the road, they build her a convertible car. At each stage, the team offers a product that meets the basic needs of the customer but eventually create the final product using the Build – Measure – Learn feedback loop. And I hear you. Hey Tom, Build, Measure, Learn sounds a lot like Plan, Do, Check, Act that we learned when we talked about W. Edwards Deming and his influence on Lean.

Could this be another example of Lean's influence on Agile? Yes, you got it. You've cracked the code.

15. Creating a Minimum Marketable Feature is like creating the Minimally Viable Product in that the team is trying to deliver value to the customer as soon as possible.

However, the Minimum Marketable Feature is a completed product that is ready to be used or sold, not a model. It contains the smallest set of functionalities that provide the most value to the customer. The Release Backlog should contain the highest priority feature sets, so that the customer receives the most value as soon as possible. Eventually, the customer will receive a product that meets most of their requirements, and they may decide to take delivery of the product early. This helps them beat a competitor to the market and limit unnecessary work. Think of a product that you own that has useless functionality. For example, my washing machine has 12 features, but I only use two. All those extra features are worthless to me. I would prefer a washer that has fewer settings that costs less.

16. There are five steps to create the MMF.

First, determine what the MMF needs to be. You will eventually settle on a small set of features that are highly valuable to the project and will be the team's top priority. This is not usually the final product, but the most valuable first piece of the final product that can stand alone and be delivered early.

Second, introduce Slack. Slack is some additional time the team will need to deal with quality issues, scheduling conflicts and other problems. After your team has been working together for a while, you will be able to determine accurately their velocity or how much work they can accomplish in a sprint. After you calculate how long it will take to complete all the MMF tasks, add a little time to the end of the project.

Third, manage change. Take the feedback from the customer and reprioritize the features on the product backlog.

Fourth, improve the MMF. If everything goes according to plan, you will complete the MMF in the time you calculated, and you will be left with a high quality, viable product and the slack time. Rather than doing nothing, add some Delighters. The cost, time and quality will not change.

Fifth, the team can accommodate the New Features because of the slack that is built into the timeline. This allows the team to "under-promise and over-deliver." So, in our example from before of the customer who decided on a convertible car, the team will tell the customer they can build it in 6 months. But it will actually be built in 5 months. If the customer suddenly informs the team that she wants to take delivery as soon as possible, the team is in good shape and can just provide the minimum marketable feature (the base model of the car) in 5 months. But if not, the team should take the final month to add features that they know the customer would want.

Now, what should be avoided is rushing to create a low-quality product with extra features in 5 months and then try to fix all the bugs in 1 month. Take your time to create a high-quality product first and then add new features. And if you find there are some bugs, please do not try to cover them up with superficial improvements. No one wants a product that looks good but does not function properly.

17. Well, that wraps up our episode on Agile. A copy of the transcript for this episode is available on our website at LeanEnglishConsulting.com. Leave a comment below and hit the like button if enjoy this episode. And as always, if you find that you need more in-depth process improvement or language training, send me an email or give me a call.

Thanks for listening. Until next time, be good and I'll talk to you later