There’s Something About Lean
A Case Study

Laura Bocock
Faculty of Computer and Mathematical Sciences
The University of Waikato
Hamilton, New Zealand
lb49@students.waikato.ac.nz

Angela Martin
Faculty of Computer and Mathematical Sciences
The University of Waikato
Hamilton, New Zealand
angela@cs.waikato.ac.nz

Abstract—One of the ideas gaining significant interest at the moment is “how can we apply Lean manufacturing principles to software development?” There is however, limited information on how Lean is being used by industry practitioners and if the principles and practices are as effective as current interest levels suggest it might be. We have used a case study to explore the practicalities of how one high-performing, open source team, has adopted Lean practices. We found that the existing meritocratic decision-making culture of the team under study appears to have greatly assisted the team’s application of Lean principles to their software development processes.

Keywords – lean, agile, open source, distributed software development, meritocracy.

I. INTRODUCTION

Industry practitioners are often at the forefront of our discipline, breaking new ground and finding ways to develop software better and faster. One of the ideas gaining significant interest at the moment is: how can we apply Lean manufacturing principles to software development? The last three years has seen a widespread rise in the software community’s interest in Lean, as evidenced by the new Lean conferences appearing in both North America and Europe. There is, however, limited information on how industry practitioners are using Lean and if the principles and practices are as effective as current interest levels suggest it might be. Our research is beginning to explore the practicalities of how Lean is being applied in industry teams, and this paper outlines some of our initial findings.

In the next section we will introduce Lean. In the third section we will describe our research method. Next we will describe the results of our research, including a description of the impact the existing company culture had on the adoption of Lean principles. We will conclude the paper with a section that brings together and integrates the earlier sections as well as discussing the implications these results have to practice and theory.

II. AN INTRODUCTION TO LEAN

The Lean story starts with Kiichiro Toyoda who started the Toyota car manufacturing company. The Poppendiecks [23] explain the evolution of Lean from its start in Toyota manufacturing lines through to its recent adoption within the IT industry. Toyoda had the vision of a company that could do two things: firstly, deliver products ‘just-in-time’, being flexible to the market demand, and secondly give workers responsibility and empower them to make changes and ensure quality [23]. After Toyoda’s death in 1952 Taiichi Ohno continued to research and develop a system based on Toyoda’s vision. In 1962, Ohno with the help of Shigeo Shingo, successfully implemented the Toyota Production System across the entire Toyota company.

The Toyota Production System enabled Toyota to be competitive against the American mass production methods. Mass production was unsuitable for the Japanese market where there was small demand and higher costs [26]. Instead Ohno designed a system that revolved around the customer; ensuring the customer was receiving value at every production step. Ohno insisted on eliminating any production steps that did not deliver value to the customer. Ohno also wanted to reduce inventory and deliver to the customer immediately after an order [2]; derived from Toyoda’s ‘just-in-time’ vision. Ohno created a set of principles that were applied to the production processes, but these principles had a much wider influence, including shaping Toyota’s business approach and company values. Following the publishing of The Machine That Changed the World in 1990, the Toyota Production System became known as Lean.

The Poppendiecks [22-24] have explored the application of Lean to software development. The Poppendiecks describe the seven principles of Lean and how these principles can be used within software development projects based on their personal experiences and research.

Principle 1: Eliminate Waste. In order to ensure that customers are receiving value and products are being delivered as fast as possible, it is necessary to eliminate waste in the production process. To accomplish successful elimination of waste in software development teams the team must first define ‘value’ within their environment. The team then identifies activities in their development process that do not provide value (e.g. task switching). Finally, the team decides how these activities could be reduced or eliminated.

Principle 2: Build Quality In. This principle focuses on the need to prevent defects and create a system that responds immediately to problems. Continuous integration and test driven development are at the forefront of software development techniques used to ensure quality is built into the system. These techniques show defects quickly and allow
the developers to assess and fix defects as they are introduced into the software, rather than relying on testing in the final stages of the development process.

**Principle 3: Create Knowledge.** This principle highlights the importance of teams learning collectively and generating new knowledge together. Feedback is a core element of learning. Software development environments should provide teams with the opportunity to improve and learn throughout the process [22]. Software development team members should provide feedback and have access to both customer and technical feedback throughout the development process.

**Principle 4: Defer Commitment.** In order to be responsive, ensure quality, and eliminate waste, teams are asked to wait as long as possible before making irreversible decisions. Teams “should try to make most decisions reversible” [23]. When decisions are made to be reversible it will allow them to be changed easily if necessary. When it is not possible to make a decision reversible it is suggested that these decisions should be made at the “last responsible moment” [23]. Although this moment will vary due to circumstances and environments, it is expected that teams evaluate when the latest responsible moment for each decision is. Leaving decisions as late as possible will allow the decision makers to be more informed when the time comes to make a decision. However, leaving crucial decisions too long can harm the overall project.

**Principle 5: Deliver Fast.** Toyoda’s vision included a system that responded and delivered to customers immediately [23]. In software this principle often results in teams developing software in short time-boxed cycles that provide structures for feedback, reflection and delivery of working software.

**Principle 6: Respect People.** Teams are at the heart of software development, so the working environment should enable the team to build trust, respect one another, and collaborate on problems. The Poppendiecks [23] would go as far as to say that “if you implement only one principle – respect for people – you will position the people in your organization to discover and implement the remaining Lean principles” [23].

**Principle 7: Optimize The Whole.** This final principle aims to encourage people to understand the system as a whole. Instead of solving problems that arise at the surface level teams should investigate problems and ensure they are fixing the root cause of the problems [23].

As software development is continually evolving, the Poppendiecks emphasize that “there is no such thing as a ‘best’ practice; practices must take context into account” [22]. The seven principles allow teams to effectively focus their development efforts and incorporate different practices into their development processes. Lean in the software development industry provides principles that are applicable to changing environments. The principles focus on ensuring stability and controlling that which can be controlled, to the best of the teams’ ability.

### III. METHOD

Information Systems Development (ISD) methodology researchers [14, 21] have expressed a growing concern that existing ISD methods do not meet the needs of today’s business and software development environments. Studies [21] in this area have begun to explore practices in natural settings in order to begin to address these issues. Given this trend, we have used a case study [7] to begin to explore how industry practitioners are using Lean principles and practices on software development projects.

We asked leading Lean Software Development experts to recommend teams for our research study. One advantage of this selection approach is that it helped to ensure that we targeted a project that is perceived as practicing Lean well. This paper reports our initial findings from one team, a highly distributed development team working in the open source domain. We describe the team in more detail in section four.

To ensure we obtained a full understanding of the impact of Lean across the whole team we interviewed the Project Lead, the Product Strategist, the Technical Architect, a Team Lead and six Engineers, that is ten interviews in total. We used semi-structured in-depth one-on-one interviews as well as observations to collect the data for this research. All interviews were taped and later transcribed in detail. The interviewees were asked to validate both the transcriptions of the interview and the interpreted findings. The project observations were used to support both the interview process and the resulting findings.

The data analysis process was iterative and incremental. We created short summary labels for each significant line of the transcripts. This process is called open coding. We used comparative data analysis to compare and contrast the open codes, to develop the substantive codes (an abstracted category that fits multiple open codes). We also wrote theoretical memos that relate the codes, drawing out the potential ideas or concepts held within the data. This memoing process assisted us to focus on the core categories and the inter-relationships between the categories. We initially coded the transcripts within the margins and used index cards for the memos, as recommended by Glaser [15]. The term theoretical saturation is used to denote when to stop this process, that is, when no or very few new codes are discovered within the data. Thus, the data collection and analysis phase stopped as theoretical saturation was reached.

In the sections that follow, we identify the theory as it emerged from our analysis of the data. We use a number of quotes from the interviews to illustrate our findings; names have been avoided and we have given the team the pseudo name Apollo to preserve anonymity.

### IV. APOLLO

This case study looks into one successful open source software development project, Apollo. Apollo, unlike most open-source projects, has a dedicated paid core development team. The Apollo team is distributed worldwide with engineers all working from home and in almost all of the world time zones.
In the following sub-sections we will describe this team and their processes. We will see how this team is structured, who is in this team, and what processes they use in their software development process. During our data analysis the following areas were uncovered as essential to this high performing team: decision making, respect, process changes, measuring, and learning opportunities. The following sections will describe each of these in turn.

A. Who are the team?

The core team has thirty members that are spread across five sub-teams; each sub-team has a ‘team lead’. The company has recently restructured the team to also include an overall project lead, technical architect and product strategist. The team consists of well-respected software engineers. All members of the team (including managers and product specialists) develop code regularly; they are all practicing engineers. They respect each other’s opinions and ideas, and work together to solve problems and improve their processes. The Apollo team are all experienced and have proven their knowledge and ability before joining the team, in-fact many of them were head-hunted to join Apollo. This created a culture where everybody values each other and the work they do. The abilities of team members are known and rewarded.

Many of the interviews also emphasized that the globally distributed nature of the team enables them to access a large talent base. This is perceived as highly valuable, and allows the project to have contributions and ideas from people all over the globe.

“So the first thing about [Apollo] that really impressed me when I got there was that it was unlike any company I’d worked before… all of the people here were people I completely respected technically and professionally” (Participant 2)

“When we needed a [software architecture style] implementation right we went out and hired the guy who wrote the… book… our talent is amazing because it’s not limited by what city you live in” (Participant 7)

The engineers in this team are passionate, driven and open minded. The team is aware of how these characteristics help shape their processes and the project.

“We all absolutely love what we do... ...This is my job, this is what I want to do, this is what I love to do, I'm going to make it better, I'm going to make it a more pleasant experience for me” (Participant 7)

“[I] have been loving it because it’s the first time I’ve got a job where I think the product is worthwhile” (Participant 5)

B. How does this team work together?

Yearly goals are set for the project in a meeting with the project lead, technical architect, product strategist and team leads in conjunction with the company founder. Apollo releases their software product every four weeks. Each release works as a “milestone” towards the yearly goals for the product. The “milestones” are created with contributions from all team members. The team creates a yearly outline that enables them to plan releases and sprints, and build in predictability. It is important for the team to ensure that they build in ‘slack’ within the release cycles. The planning weeks are at the beginning of each cycle. In parallel to the planning activities the remainder of the team completes unpolished or unfinished work. Development is driven from a backlog of features and bugs.

“[Apollo] is not the kind of project where we plan like, okay, month one, month two, month three, month four, we’re going to do this. We have sort of a general set of goals that we’re trying to hit for the year, and we have to sort of give and take to get to those goals. Many things we learn along the way and so we only really set about maybe a month or two out what we’re going, what the specific priority for that month is” (Participant 4)

“Often stuff comes up that you haven’t expected, so we try to build in slack into, you know, we try not to over-schedule, we try to pack down the schedule quite a bit.” (Participant 2)

The team recently uncovered an issue in their planning process. They found that all of the teams were moving forward but not always in directions that complemented each other. The team has placed more effort into creating an integral product plan in their planning process to address this. The recent creation of the product strategist role gives emphasis to this change, and the intention of the team to optimize the whole, and consider product strategy a core part of their engineering process.

In the following sections we will explore in more detail the practices the team uses on day-to-day basis.

1) Pre-Implementation Call

An engineer will typically start a piece of work with a pre-implementation teleconference call with another engineer. The pre-implementation call is used to discuss their planned approach to this piece of work whether it is feature creation, bug fixing or a specific problem. This call helps the team to learn about new tactics, code that possibly could be re-used and design solutions:

“Often you can think oh that’s really simple I’ll just do this... and then check with someone else about it and it’s actually not that simple or it’s completely up the wrong way, up the wrong path” (Participant 5)

“I could [go] off in a crazy tangent... trying to build the kitchen sink and not mean to” (Participant 7)

“It might be someone who knows less because it is also a way of spreading the knowledge around” (Participant 6)

“They both learn from each other... the end result the quality is much higher because they had to defend their approach to the other” (Participant 1)

This technique was inspired by Lean principles: incorporating others knowledge to help ensure quality and re-use existing code to eliminate waste. The Apollo team
identified the excessive number of changes prompted by the code review process, and realized many could be caught much earlier in the process. The pre-implementation calls also provide engineers with an avenue to share targeted knowledge with each other just as it is needed.

2) Reviews.
Code Reviews are also used by this team to help them share knowledge and ensure quality. A code review will typically consist of the reviewing engineers reading the submitted code and discussing it with the submitting engineer. The engineers collaborate to improve both the design and the code. This technique is not new and is perceived as an effective one that stops bugs entering the code base and shares knowledge concerning the existing code base and standards. An additional benefit of code reviews is that engineers are exposed to more of the code base within the product.

Engineers spend one day a week where they perform the code reviewer role. This role ensures that the other engineers are provided with rapid feedback, as the code reviewer has this as their top priority for that day. The introduction of the code reviewer role reduced the delays engineers used to experience when they needed to wait for others to be available to perform a code review. That is, it reduced waste.

“Reviews like this … are possibly the best thing that has happened to me professionally. I have been recommending it to other companies as well.” (Participant 6)

A mentoring process was used to introduce code reviews. This process paired experienced code reviewers with learners until the experienced reviewer was satisfied the learner was looking for the right things and asking the right questions. The learner was then graduated and allowed to complete code reviews alone.

The team has recently noticed that the user interaction (UI) within the ApolloProduct needs improvement. To address this issue they have introduced a UI review process. The Apollo team modeled the UI review process on the code review process, and kept many of the lessons learnt with code reviews, including the use of a mentoring process.

3) Sprints
In an effort to combat the problems that occur when engineers have little face-to-face time, engineers join together in ‘sprints’. This practice comes from the open source practice of sprinting, and is quite different to a sprint in Scrum. Apollo sprints are structured to allow the team to develop software as a co-located team, have discussions on plans and decisions, and form bonds. It is also important for the team to train and learn from one another and use techniques such as pair-programming, whiteboard visualization and lightning talks to increase knowledge sharing and team bonding. The length of a sprint is typically one to two weeks. The team always stays together in one location, typically a hotel, increasing the opportunities to both work together and socialize together.

“The sprints have to be for technical and product reasons but they are also simply to get people in a room and eating and talking together” (Participant 2)

“We try and sprint twice a year... what we actually do at the sprints kind of varies. We have had planning sprints where we’re mostly just talking about what we want to do for the next six months” (Participant 7)

“The things that you talked about at the sprint are focal points things that stay very clear in your mind, the things that you can discuss easily, things you can refer to because you have met people and looked at the same piece of paper, just the fact that you looked at a white board” (Participant 6)

The Apollo team comes together at an “all hands” sprint at least once a year. The team also has a number of additional sprints each with a specific focus, these includes sprints for team leads and management; sub-teams; cross-team collaboration; and specific areas, such as the UI. Each sub-team aims to sprint together as a team twice a year.

4) Constant Communication
Open-source communities and distributed teams are particularly aware of the importance of communication. Agile and Lean methodologies value communication and constant feedback and discussion. This team uses a wide variety of communication tools to enable them to help one another, gain information from users, and keep up to date with system and team statuses.

“It is problematic not working together with people face to face... [In a distributed environment] insist on a lot of communication, to have voice calls… You hear if somebody’s frustrated when they talk” (Participant 3)

“I am part of the calls and the mailing lists and I’m in the loop all the time” (Participant 1)

“If I don't communicate I start to feel isolated ... It’s night and day with [Apollo] we have IRC, we’ve got instant messenger…We just have these levels of escalation that if it’s this big call, and you know if it’s this big just send an email” (Participant 7)

“I try to err on the side of as much communication as possible without being annoying” (Participant 4)

The team has a firm belief in constant communication, and emphasizes the importance of constant communication between the team members and the community. The culture within this team is encouraging and built on knowing that someone is always there for you, to help you however you need it. Engineers spend large amounts of time in chat systems and are constantly available for both their team and their larger community. When someone needs help they often know who the best person to contact is. However, due to the distributed nature of their work environment, sometimes that person is unavailable. In this situation the team responds by opening all doors of communication to get the help that is needed.

“Sometimes you will know of someone who's kind of an expert in a specific area and you’ll try to get them,
but often you’ll just ask a question and see if anyone replies…If you get stuck you get help. You go and try to find out…I think easier than the non-distributor team…you know somebody is going to be there 24 hours a day” (Participant 3)

“We really encourage people if they [get] stuck, even just slightly like I’m not sure what to happen next, just talk to someone about it and it’s okay to do that” (Participant 2)

“If I get stuck there’s always someone to help me, maybe not specifically on my team but there always someone” (Participant 7)

5) Personal Communication

Although this team has regular meetings, they cannot rely on face-to-face communication. They are aware of the personal contact they miss due to the distribution of their team. To combat this, the Apollo team ensures they have both technical and personal discussions with each other. During sprints there is a large amount of face-to-face interaction. All of the sprints emphasize spending time getting to know each other personally, and forming bonds. The team members are all located in the same hotel, and are able to socialize constantly for the duration of the sprint. The team work, eat, and drink together and have a casual and relaxed atmosphere.

“I think when you can’t meet people face to face you actually have to put more emphasis on communication to make sure you don’t miss anything” (Participant 3)

“You have to be able to go out and have some beers with your co-workers…it’s good to have that interaction and it’s not strictly like a work relationship I guess” (Participant 7)

“I think it’s incredibly important just to be able to build those relationships…like we were talking the other week as in lets plan to make sure we have an evening making pizza together or something just to make sure we’re doing things together to learn a bit more about each other and try and, you know, the stuff you do everyday when you’re working in an office and it collects over six months, trying to squeeze all that into a week. ” (Participant 5)

Engineers make time to engage with each other outside of sprints too:

“Once a week I usually have a one on one [with my manager] where we don’t talk about work at all just…personal problems…it’s basically a gripe fest” (Participant 7)

“Something I’ve found quite useful lately is meeting up more often locally with people in the company. So two weeks ago I flew… to work with one of the guys in my team just to be able to spend two days together and… not having a formal sprint but just being able to get together and work together. Other times I’ll go into Berlin and work with another guy who’s not in my team, doesn’t do anything related to me but is in the company and who I know and just a social contact as well as we’ll get together and sit and work together for the day” (Participant 5)

6) Open Communication

Apollo is constantly evaluating its processes and values, and all members of the team are able to instigate discussions on problems and ideas for solutions. All team members are asked for input to make decisions on direction, value and changes to the process and product.

“It’s literally just an open discussion… ... “It’s very free and open environment you just say I’m not comfortable with this direction because of this” (Participant 4)

“We are very open to new ideas…We have times where it’s just open to the floor” (Participant 7)

In this section we have investigated the practices of the Apollo team. The practices of pre-implementation calls and reviews have assisted the team in creating knowledge, eliminating waste and constantly improving their processes. The use of sprints and open communication practices also assist Apollo to create knowledge where team members help each other to build quality in.

C. How does this Team Make Decisions?

The Apollo teams’ culture creates a safe environment. This culture is important when making decisions and all team members have the opportunity to explain their views, and understand and learn from others backgrounds and experience. In the distributed environment many of the decisions the team makes are done in calls with sub-team members or one-on-ones with team leads. All decisions are discussed by the team, from those related to what should be in the next release to those concerning process experiments or changes. Sprints are also used as a time to plan and discuss goals and milestones, as well as evaluate processes. Those who cannot attend the sprints are represented by other team members, and are consulted and kept in the loop. In the sprint we observed, one absent team member video conferred in for the majority of the sprint.

The Apollo team self-identify as a meritocracy. The decision-making process is a meritocracy of ideas. All ideas are valued equally and assessed on their merits, regardless of where the idea has originated. This team describes the decision-making to be consensus driven and involves everybody, without it being a top-down or bottom-up management approach. Often decisions end up incorporating ideas from all team members. Given this approach all decisions are accepted and viewed as a step forward for the project.

“This is not a democracy its not one person one vote it’s... a meritocracy ...” (Participant 6)

“I’ve not seen anything where it was like top-down, handed you know, go and make it so and no discussion or consensus... I'm quite comfortable with
what we’re doing now after a week of sitting in a room with twelve guys ironing it out” (Participant 4)

“The variation will be discussed on our mailing list, so that everyone can contribute to the process and probably in the end maybe like in a forum of all the team…A small forum would try to kind of really identify the problems, then there’ll be some kind of discussion that’s really open to everyone in the team and everyone in the community and then to again have it to a smaller and higher up forum for action and changes” (Participant 3)

Decisions on the overall product and the work involved appear to be talked about through all levels in the company. Engineers were aware of the job to be completed as part of their sub-team milestone.

“Around this time we’ll also kind of decide what's going into the next milestone. It would be mostly the responsibility of the lead developer with everyone” (Participant 3)

“We usually try and find a good middle ground for what we need to do...It’s kind of like a discussion more than it is an assignment I think” (Participant 7)

One concern the team has recently identified is that all of the big picture details, such as product yearly goals, are not communicated effectively to engineers. The team is currently experimenting with process changes that will improve the big picture and product strategy communication within the team.

It is unclear if the Apollo team is a ‘true’ meritocracy. The values of meritocracy, however, are apparent within the team, particularly with regards to their decision-making processes. Each idea or decision is debated on merit.

D. How do they improve their processes?

The Apollo team improves their processes relentlessly, constantly reflecting and changing. Although the Apollo team perceive themselves to be Lean and Agile, it is important that this team were already doing this before the team formally adopted these methods. The open environment established by the team culture allows any one to suggest a change and begin experimentation. This team is not biased towards methods or specific practices. The engineers believe in trials and are driven to improve their development methods and tools. Team members are constantly looking for places in their process that will enable them to increase both the quality and quantity of code they provide. Typically each feature will have a cover sheet that asks the engineer to reflect and provide feedback on how the process could be improved. During team calls and sprints the team reflects on bottlenecks and problems they have faced, and discuss options for change. Ideas concerning process improvements are circulated throughout the team and experiments are run.

“We do like the notion that our process is continually improved ... as a team always looking at the process and seeing what could be better” (Participant 6)

“That's one of the great things about [Apollo Org] its kind of this organization where as soon as you see a problem just say and we'll figure out what to do about it” (Participant 7)

“Someone will have an idea about how this process could be improved so they'll send an email out to our, our internal list saying request for comments, here's what I’m thinking… everyone will comment on it and they’ll say yip lets run this as an experiment...

...Trying to ensure that people are being reflective and constantly evaluating how things are being done and how we can improve it, great let’s put an experiment and try it...We pick the cherries from the different processes...You’ve just got to adapt and try and find ways to make the best of the situation” (Participant 5)

“We believe in experiments because we believe in science...We’re experimenting with this style of doing things which maybe it will work maybe won’t” (Participant 2)

“It’s not a cookie cutter approach in Lean” (Participant 10)

The team did experiment with one Lean tool in this space, value stream mapping. A value stream map is a tool used by teams to visualize their processes [22]. A value stream map shows each step in a process, the time it takes to move through each step, and the time between. The goal of the map is to highlight potential waste. The next step in value stream mapping is to improve the process, focusing on ways to reduce or eliminate the waste. The Apollo team used this tool to improve their code review process:

“The value stream mapping exercise that we did there was useful. One of the things it pointed out was that round trips for us are extremely expensive” (Participant 2)

Interestingly enough, although most participants relayed to us the story of the one time they had used a value stream map, most were not aware of using this technique again (years later). One participant’s comments might help shed some light on that:

“I thought that’s the problem I’ve always had with value stream maps is that sometimes they seem not to offer more value than just kind of what intuitively you could say about the process... I didn’t need this process to tell me that not working on something for half a year is a little bit funny” (Participant 3)

Although Lean emphasizes the importance of measurement in order to access the impact of change this team does not regularly track or measure processes. Engineers are aware that items they are working on may be years old and there is no way of tracking how long it has taken for a bug to be found and fixed:

“[The bug] could be from three years ago” (Participant 5).

The evaluation, reflection and experimentation already done by the Apollo team suggest the results of measuring and tracking are often found without formal processes.
E. Learning

The Apollo team has a passion for learning, they all believe their processes enable them to continue to learn, and often saw joining the Apollo team as a great learning experience. This passion for learning is evident in their practices of cross team collaboration, constant reading and communication, and process improvements, including pre-implementation calls, reviews, commenting, blogs and lighting talks at team meetings.

“We're really keen to learn” (Participant 2)

“The thing that I like the most I guess, I love learning. Learning, it's a passion for me and I think in [Apollo] there's so many opportunities to learn” (Participant 5)

The team emphasizes the importance of creating (shared) knowledge, learning from each other, and constantly improving their practices.

F. How do the Apollo team practices cause problems?

The Apollo team has developed a set of practices that help to ensure constant visibility and responsiveness. However, in this section we will explore the “thorns” that exist in the “bed of roses”. In particular we shall highlight the interruptions, and the impact these interruptions have on the engineers. We will also examine the causes of task switching and unfinished work. Finally, we will express how the team processes factor into causing these problems, and what the team has done to reduce the negative impacts of their “thorns”.

7) Interruptions

Many things in the Apollo team’s environment can cause interruptions. The team have to structure their day and tasks around necessary communication. Engineers discussed how practices, such as pre-implementation calls, reviews, and IRC communication, could be interruptive and frustrating:

“Interruptions can be a killer, if you get interrupted all day you may get nothing done. A different way of looking at it is you get lots of stuff done just not the one thing you were working on.” (Participant 6)

DeMarco and Lister [10] and others [2, 23] emphasize how interruptions can compromise a developer’s ability to remain in a state of flow. Engineers are aware of getting into a state of flow and often not being able to sustain this for a long period of time. The majority of interruptions within this case study are a result of direct communication. The team is aware of the waste that can be generated due to interruptions, and they do their best to streamline this process. For example, daily team status calls are at a set time each day, and are time boxed to 15-20 minutes.

The Apollo team are constantly aware of their common goal, and they all have an attitude that any improvements are an improvement overall. This attitude facilitates the team to prioritize supporting others, and ensuring project and team health. Communications that support the whole of the project are not seen as interruptions because of the value that is gained by having them. This value includes the necessity for communication in a distributed environment, and the opportunity for team members to share knowledge and ideas.

Other interruptions are often due to supporting the community and other team members. The impact of the review process is detailed in the following sub-section.

8) Reviews and Multi-Tasking

The review process often leads to the need for engineers to multi-task:

“It forces you to start doing more than one thing...then it's the risk of basically you never finish it...If I put a patch up and then for the rest of the day can’t do anything about it because I’m waiting for someone to get around to it or the person who needs to review it is in Boston and they’re asleep and they won't be awake, that just really takes such energy out, it forces you to start doing more than one thing...it’s the risk of basically you never finish it, you never get around to finishing it because it was always a secondarily important thing that you were doing opportunistically.” (Participant 2)

Members of the team must wait for a review to be complete, during which time they often begin new features or bugs to keep their work flowing. Engineers overlap tasks in order to stay productive and focused. Additionally, the code review process took engineers away from their own work so that they could review someone else’s code. Once the on-call reviewer was introduced there was less interruption from other engineers asking for a review.

In order for the on-call reviewer role to be successful the Apollo team members all needed to become reviewers. Engineers expressed that this training process also increased interruptions. Some engineers reflected on times when they were aware of the queue for the on-call reviewer, so they would ask other engineers to review their code instead. Examples of this practice bring attention to the possible overload of reviews for one on-call reviewer. The team however, believed that while not perfect, the introduction of an on-call reviewer reduced the number of interruptions each engineer had.

9) Changing Priorities

During the interviews a number of engineers reflected on how the priorities of their tasks changed frequently. They explained that new high priority tasks could interrupt work already in progress. The constant changing of requests increased the amount of task switching and work-in-progress for an engineer:

“If [main customer] come and tell us that something is very urgent we're likely to say okay we'll work on it immediately and do other things later.” (Participant 3)

“Every once in a while I have to assert other things and drop everything I’m doing and fix things but for the most part I can stick to what I do. I do often end up being tech support... . . .Someone up the chain will say this absolutely needs to get done and it maybe that... at the end of NZ's day they haven't done it so
or I won’t be able to get to it, so can you please do this the first thing tomorrow.” (Participant 7)

The Apollo team was proud of their flexibility and responsiveness to changes in priorities. Typically one might expect a Lean team to start to measure the problem once it was recognized as a problem. The Apollo team, however, has not done that. The interviewees did not elaborate on the reasons for not implementing a measurement process, but at the moment it seems they value their perceived responsiveness as more important.

10) Getting Value from Interruptions.
Understanding value is vital when implementing Lean principles. In order to effectively identify and eliminate waste, teams must understand what it is that is providing value. In these sub-sections we have identified that interruptions were often a result from communication between team members. This communication is necessary for the practices of pre-implementation calls and reviews to be effective. The Apollo team recognized the disruptive nature of pre-implementation calls, reviews and general interaction. The addition of pre-implementation calls effectively reduced the time engineers spent unblocking one another and altering reviewed code. The time an engineer spends on pre-implementation calls, reviews and general help were viewed as positive and deliberate interactions. The Apollo team has optimized their processes to reduce the negative impact of interruptions.

Having multiple tasks in progress and incomplete work are recognized as areas of waste in software development [2, 22, 23]. Software developers, however, are often expected to regularly switch tasks and have many tasks in progress [20, 23]. Anderson [2] and the Poppendiecks [23] understand that task switching involves the developer mentally switching in order to work on different tasks. It is well known that software development is complex and involves a lot of concentration [2, 19, 23]. These authors also demonstrate how task switching often results in neither task being completed within the ideal time. Engineers in this case were expected to respond immediately to changes and switch focus. In the case of reviews, the engineer is forced to begin other tasks during the delay period as a result of the review process. In an effort to reduce the problems caused by the review process an on-call reviewer role was instigated. This role has provided a focus for fast feedback and review turnover. The Apollo team has recognized both the “waste” and “value” of interruptions and multi-tasking, and at the moment it appears engineers are willing to cope with a few “thorns” in order for the “roses to stay healthy”.

V. DISCUSSION

In this paper, we have provided a detailed explanation of who the Apollo team is, how they work together, make decisions, and improve their processes. These aspects of the team’s culture are crucial to understanding the high level of respect and openness the team exhibits. In this discussion we shall explore how this open and meritocratic decision-making culture has aided the team’s implementation of Lean. This section will focus on the concept of visibility within the team and their processes.

Visibility of Decisions. The engineer’s involvement in decisions encourages an open and safe environment, and improves the team’s implementation of Lean. In open source and meritocratic societies decisions incorporate opinions from group members [8, 9]. Community members are encouraged to discuss ideas and make decisions openly. When exploring how decisions are made in the Apollo team, we found similarities to the open source and meritocratic experiences. In section 4.3 “How does this team make decisions?” we have illustrated how all the Apollo team members are involved in the decision-making processes. Engineers are given the opportunity to raise concerns, put forward ideas, and provide input into the direction of the product through their team leads and public discussions. Team leads are trusted by their team members to represent their views and concerns in planning sessions and discussions. Ideas are viewed on their merits and given equal consideration.

All of the engineers have influence over their processes and experiments. This influence reflects the Lean encouragement to allow workers to improve their own processes, as the workers are the ones who know the most about them [23]. Encouraging engineers to improve their processes enhances their passion and motivation for the overall project. Team members use the opportunity to provide feedback and ideas as another way to show other team members their skills and knowledge. This is an important step for members who are wanting to progress within the team. The team members expressed that making decisions is a collaborative effort and the outcomes of decisions are accepted, at least in part, because of the open decision making process. This section highlights how the decision making process provides the team with a platform to respect others, continuously improve processes, and eliminate waste together.

Visibility of Work. The communication practices of daily meetings, pre-implementation calls and reviews, highlight how the work produced by an engineer is constantly visible to other team members. In Lean software development an emphasis is put on improving and creating knowledge amongst team members [23]. Regular feedback loops and open communication are seen to provide learning opportunities for developers [22]. The knowledge gained through the communication practices described in this paper enables the team to build quality into the development cycle. Ensuring team member’s work is visible also provides other areas engineers are able to reflect on and investigate when attempting to eliminate waste from their processes.

Visibility of People. In a distributed development environment social communication can often be lost. Social interaction is perceived to increase the effectiveness of communication in general [3] as well as promote the team culture [11]. Open source and Lean researchers [3, 4, 17, 22, 25, 30] highlight the importance for teams to interact on a social level. In this paper we have illustrated that as a distributed team, Apollo did not automatically get the casual interaction that co-located teams experience. The team
identified the use of constant, personal and open communication practices to reduce the impact of distribution. These practices also enable the Apollo team to build trust, respect and friendships. The culture of Lean is founded on the respect for the workers, in the case of software development teams; members should be respected by one another and the organization [23]. It is clear that the Lean principle of respecting others is ingrained into the Apollo team culture. Sharing knowledge and experiences among team members also increases the knowledge base of team, which is important in Lean software development.

We have discovered that the multiple communication techniques used by the Apollo team, supported them in providing transparency of their work, and their processes to each other. Given there was high transparency in the Apollo team environment; the team easily adopted the principles of Lean. Transparency in these areas provides instant visibility of the current state of the project. Individuals and the team collaboratively, are able to evaluate and reflect easily on the work flow of members and the whole system. This ensures waste can be identified and the flow improved constantly.

It is worth considering the idea that the meritocratic decision-making within this team appears to have had significant impact on its ability to implement Lean principles. This case study has provided insights into possible small steps that could be taken by teams as a platform for Lean implementation. Providing visibility in all areas of software development could greatly assist teams in the adoption of Lean.

VI. CONCLUSION

In this study we set out to explore the practicalities of how Lean is being applied in industry software development teams. We have begun this research by investigating one team’s experiences implementing the Lean principles. Exploring the Apollo team’s processes and understanding the team’s culture has shown that software development teams can work successfully in a distributed, open source environment using Agile and Lean practices. They are aware of the difficulties in directly implementing practices from methodologies given their working environment, and choose to “cherry pick” those that suited them. It is important to note that members of this team did not create their processes with the intention of being Lean or Agile. Instead the values demonstrated are a consequence of the team culture and experience. It is worth considering the idea that the meritocratic nature of the team appears to have had significant impact on its ability to implement and enforce Lean principles.

Throughout this study we have noticed that there appears to be a mutually beneficial relationship between meritocratic decision-making culture and Lean software development. In this case the Apollo team are clear they already had a meritocratic decision-making culture prior to adopting Lean. However, we are not convinced that this relationship is one way only, and the relationship between the culture of an organization and their software development processes may be more like the quandary of the chicken or the egg: Which comes first?

VII. ACKNOWLEDGMENTS

Our thanks to the Apollo Team. A number of organizations have helped cover our travel expenses, these are: Adscale, NetValue and the Marsden Software Process Improvement (SPII) Research Fund. Finally, we would like to acknowledge the assistance of Scott Avery with four of the initial interviews and the transcription of those interviews.

VIII. REFERENCES


