Discovering the root cause of impediments using Lean Software Development and Data Science

EVERTON LUIZ DE RESENDE LUCAS, Federal University of São Paulo
FLÁVIA CRISTINA MARTINS QUEIROZ MARIANO, Federal University of São Paulo
LUIZ EDUARDO GALVÃO MARTINS, Federal University of São Paulo

As a software developer with more than 20 years of experience, I witnessed and went through some awkward situations. I had to create unuseful documentation because my boss wanted it. I saw teams defining story points to a set of user stories and complaining that some user stories were more complicated than they thought at the next retrospective. I've heard about teams creating interfaces with extra features solely because the designer thought it was helpful to the customer. I witnessed teams with a WIP of more than 200, thinking things were fine.

Sometimes it's hard for professionals to see situations like these as waste, barriers or impediments. The research I'm developing in the master's degree program aims to help developers to be more productive by removing unnecessary impediments. I used Data Science and the concept of waste to do so. I created software that implements a methodology for detecting waste and impediments within project management database software issues. The methodology is called FAIR: Flow Assessment and Impediment Recognition framework. It was possible to quantify wastes during the research and present waste trends to teams and managers. The presentation of waste trends helped teams focus on what matters the most. And it eased the root cause analysis. This report aims to demonstrate the root cause analysis process, some discoveries I came across during the application of the methodology and challenges faced while the research was developed and applied.

1. INTRODUCTION

The first time I had contact with agile methodologies was in 2008. Since then, I have learned a lot about how to create products using an approach which differed from traditional approaches such as those found in the PMBOK [Vargas]. I learned that product development is less about the scope triangle (scope-cost-time → quality) and more about moving in small steps towards the discovery of what matters to the customer.

As I learned more about Agile, I began to think deeply about the word **impediment**, defining it as something that happens in a team, which prevents work from being done, and needs to be addressed as soon as possible. Moving further in my studies, I discovered Lean Software Development and the concept of **wastes**. It was an interesting concept, because it strongly relates to many of the problems that happen in software development. And I became very interested in discovering the root cause of the waste to avoid the repetition of the waste.

As I became more proficient in agile methodologies (not only Scrum), I started to notice teams treating wastes as normal occurrences. So, I decided to dig deeper into some ways to help teams visualize wastes and impediments. I started a master's degree and my research aims to join Data Science and Lean Software Development. The purpose is to help developers become more productive by presenting waste and information about the value flow. This experience report tells the story behind the scenes and some results that were possible to achieve.

Author's address: Everton Luiz de Resende Lucas, Avenida Cesare Mansueto Giulio Lattes, 1201, São José dos Campos - SP, Brazil; email: evtlucas@gmail.com
Second author's address: Flávia Cristina Martins Queiroz Mariano, Avenida Cesare Mansueto Giulio Lattes, 1201, São José dos Campos - SP, Brazil; email: flaviaqz@gmail.com
Third author's address: Luiz Eduardo Galvão Martins, Avenida Cesare Mansueto Giulio Lattes, 1201, São José dos Campos - SP, Brazil; email: legmartins@unifesp.br.

2. MY STORY

My first contact with software development happened when I was a youth. My father was an electronic technician, and sometimes, during the '80s and '90s, he got a computer to fix and make some extra money. After my father fixed the computer, I could play some games and write some small programs in Basic.

My vocational studies were in Agriculture. But computers came to life after that. I started to work as a computer technician in a company in which my father was a partner. After a couple of years, I reconnected to the software development world. When I started as a software developer, I faced my biggest professional challenge ever, which was the creation of supermarket cashier software. This software needed to connect to different types of equipment and had several concurrent routines to perform its tasks.

I heard about agile methodologies in 2008. But I learned about waterfall and iterative, incremental software development processes before I got some awareness about agile. I tried to use these process models but with no success. The experience of building the cashier software brought me the first insights into impediments and waste. Clients usually ordered lots of things to be delivered without prioritization. The product I developed had extra features. I had limited software development expertise, and I was the only developer using Delphi in the company. I felt under a lot of pressure to produce results.

The years 2011-2012 were also tough. The Brazilian Government created new ways that companies must follow to determine their tax payments. It involved software that collected information about buying and selling, and sending this information to the Government. In this case, mistakes could even lead to problems of millions of Brazilian Reals in magnitude. This was the first opportunity I had to use Test-Driven Development in a real project. The software ran smoothly and with minimal problems. I understood Agile Development was an interesting way to create software. I left this company to join to Brazilian Air Force.

Working in the Brazilian Air Force (FAB) brought the opportunity to start a master's degree course at the Federal University of São Paulo, São José dos Campos. I took classes in Innovation, Entrepreneurship, and Statistics. The Statistics classes led to my first experiment, which involved software development: a software development analysis about issues and the underlying reasons for them (whether the issue was stuck or taking longer than the time of sprint, how long it was stuck, and why they are stuck). The experiment was made using issue information like the task's date of creation, date of the beginning, date of the end, complexity, and priority. The results showed there were complex issues with high priority that led to defects in the product. The results of the experiment led to the development of my master's research.

Moving further in my professional experience, I noticed other evidence of wastes. Once I discovered I wasted my time reading an outdated manual. I stumbled upon the problem with the manual after studying it for hours and gave the customer the wrong directions. The wrong directions were based upon the obsolete manual. So, I decided to research this theme and help other developers avoid this type of situation.

Over the years, I have learned about Lean Software Development [Popendieck], Toyota Production System [Jones], and the Agile Manifesto. The definition of value always helped me to focus on what matters the most, and waste became something to fight against. However, I witnessed retrospectives that didn't figure out the problem. I've heard teams complaining they set wrong points to a feature for more than two years, but there was neither root cause analysis nor any awareness that the situation needed resolution.

I started the master's degree program by taking statistics classes, and I realized that even basic statistics could help professionals understand what's going on within an environment. Statistics and agile metrics like lead time, work in progress, and throughput working together could even be powerful tools. During the same experiment, the team believed they were using Scrum. But it was a shock to the team when I presented tasks lasting longer than a couple of months. I noticed that the concepts of lead time and throughput were too abstract. However, talking to the team about defects and tasks taking longer than a Sprint were more concrete.

The master's research usually starts with the theoretical background. The program I joined required that students must perform a Systematic Literature Review (SLR). An SLR has a heavy process to guarantee the desired consistency to the research. In my case, I started with 667 papers, and I had to filter papers to perform information extraction from 40 selected papers. Performing an SLR almost drove me crazy, but I found out some interesting insights. The figure below demonstrates the paper's filtering process.

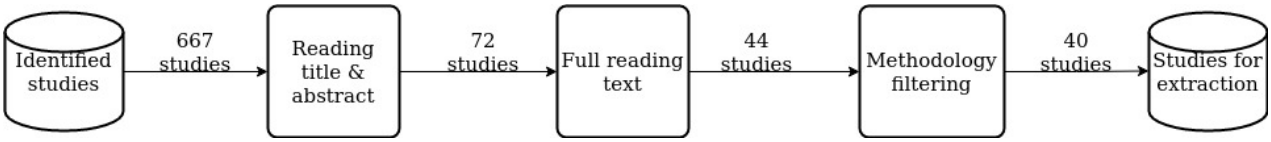

Figure 1: SLR paper extraction information process

I found papers with really sad histories. There was a paper that brought the following testimonies: “My software dev skills dropped off as I became more and more frustrated until I eventually closed it off and came back the next day to work on it” and "[the unhappiness] has left me feeling very stupid and as a result I have no leadership skills, no desire to participate and feel like I’m being forced to code to live as a kind of punishment" [].

The analysis of the papers has shown that professionals usually link waste to factors like rework, bottlenecks, overprocessing, extra features, and work in process. Something changed in my mind when I read that waste could be considered anything that creates obstacles to the value flow [Power]. I found papers with comments and reflections on miscommunication, requirements misunderstanding, unmet human potential, cognitive load, and toxic environments. It blew my mind.

Sometimes, I’ve worked in toxic environments. I’ve learned that toxic environments are made by toxic people. Sometimes toxic people have too much power in their hands. I had toxic bosses who screamed at people when things went wrong rather than have conversations about the expected level of service. This environment was very bad for those who needed concentration, support and trust to do their job.

On the other hand, I knew stories about pseudo-servant leaders. I classify pseudo-servant leaders as those who claim themselves as agile professionals, and even though they carry lots of certifications, all they do are funny retrospectives, and act in opportunistic ways; for example, they foster practices like Story Points to promote themselves in front of their bosses, telling teams are more productive because they are doing more points each sprint. This is something I call subliminal toxicity.

There was a paper that expressed how difficult it is to talk about problems, impediments, and waste in western culture [Power]. The paper recommended talking about enabling flow. I also learned that it is important to talk about stuck issues not only to figure out those issues, but to understand what is behind them.

After the submission of the SLR, I initiated the creation of a methodology and the development of software that detects waste and calculates the value flow situation. After some experiments, I decided to call the methodology **FAIR: Flow Assessment and Impediment Recognition framework**. It was not possible to use Scrum during the research due to the master’s program and supervisors’ dynamics. So, I decided to use a Kanban approach. I created a Trello board with enough information that gave me context while I switched between writing papers and developing software. The backlog was prioritized and, when I had review meetings with my main supervisor, I added or updated backlog items. To give some context to the project, I wrote the following product vision statement:

> For software development leaders and teams
> Whose development flow must be fluid and continuous,
> The MUDAbuster
> Is a value flow’s situation evaluator and a waste detector
> That assesses value flow, detects waste, and identifies project impediment trends
> Unlike other products, which generally present only metrics
> This product numerically demonstrates the health of the value flow, and preserves its history, which improves the ability of a team to remove impediments to its progress

MUDAbuster is the current name of the project. The word MUDA comes from the Japanese term which means waste. I haven’t yet decided if it is a good name. I learned the way the vision statement was written from a method called Lean Inception [Caroli]. Lean Inception was created by a Brazilian guy called Paulo Caroli. Along with the vision of the product, it has four statements called what the product is, what isn’t, what the product does, and what the product doesn’t. These statements were useful for dealing with trade-offs during the project development.

Initially, there was no company interested in my research. So, I decided to develop a simulator to create simulated issues. Those simulated issues’ duration followed a Gaussian distribution, a constant time (all tasks

with same duration), and random time. The simulator had an internal clock and "developers" that implemented the issues. The issues created by the simulator were useful to start the development and the software's first tests. Those issues were created in a json file and then they were imported to a MongoDB database. Based on those tests, it was possible to show the Mudabuster's capabilities based on stored data within its database.

The first version of the software allowed the test of FAIR along with the presentation of the software to a company. The company agreed to join the research, but the pandemic changed the way the company ran, and almost all the employees went back to their hometowns. So, the collaboration wasn't possible.

Meanwhile, I received the SLR rejection, and it disappointed me. So, for a couple of months, I worked on FAIR, the software, and fixing the SLR based on the comments made by the reviewers. On the other hand, my wife and I were trying to have a baby for the last 5 years, and we went through in-vitro fertilization treatment. After two unsuccessful attempts, we knew we had 1 viable embryo. My wife went through the implantation process, and we got pregnant!

I knew I had something valuable to other developers on my hands, so I decided to leave my job to have free time to become fully dedicated to my master's research. I gave a presentation about the software to two other companies. Both companies agreed to join the research, but one company was faster than the other one to give answers and schedule meetings. Simultaneously, my wife and I decided to come back to our home town, but we decided to have a layover in the middle of the path. We stayed for 6 months in Curitiba.

Before our daughter's birth, I worked a lot making tests, updates and having meetings with the first company. Those tests showed me a broader perspective about how to present and discover the root cause of existing wastes within that environment. Among all the research questions answered in the SLR, there was one that could fit the main objective of the research: Is there some relationship among wastes? I analyzed each paper to extract information, and each one presented a set of wastes. I used Spearman's rank correlation to analyze the likelihood of waste occurrences. This likelihood contributed to helping me to discover the root cause of the wastes.

My daughter was born, and something called routine went out the window. Meanwhile, I discovered I could code whilst my wife breastfed my daughter at 4 A.M. At that time, I realized that the first version of FAIR didn't match reality. It was necessary to implement something more realistic, and I needed to add another waste to the calculation. The first version dealt only with delay waste.

Unfortunately, due to scientific journal policies, I can't explain the whole algorithm here. But I can give a spoiler: the algorithm involves descriptive and non-parametric statistics, vector calculus, and trigonometry. When I added defects into the algorithm, I stumbled upon a framework for considering additional forms of waste. By presenting the value flow situation, taking into account deliveries and waste trends, managers and teams could focus more clearly on improvement opportunities without undue pressure.

After the second version development, I restarted the evaluation meetings with the companies with new insights. The software gained new features and possibilities. The calculus of the value flow situation was brand new and the graphic representation provided by the MUDAbuster demonstrated something much more realistic than before.

However, life was not a bed of roses. I faced many problems during the development of the research. I envisioned that the software would read issues from project management software like, for instance, Trello and Jira. I developed a module which reads issues from Trello. But I realized that companies were afraid to provide their api keys. So, I built some scripts that import csv files into the database. The second challenge was a lack of comparison with other software tools, based on the novelty of the way the value flow's being assessed. There are tools demonstrating metrics such as lead time, work in progress, throughput, and cumulative flow diagram. An example is GetNave (http://getnave.com), which presents this and other charts. There were clues that we were on the right way. However, it is formal research, and the formal assessment is being carried out at this moment.

The first meeting with one of the companies happened on October 8, 2020. During this meeting, it was possible to detect the designer was creating unnecessary capricious adjustments, based on the analysis of deliveries and duration trends made by MUDABuster. The first recommendation was the creation of a definition of ready to avoid overprocessing.

The company under analysis didn't allow me to reach the team. Even though only the technical leader was allowed to participate in the research, it was possible to detect many possibilities for improvement. This company kept agile metrics, but those metrics were used to do forecasting and not for improvement. The company had not taken the opportunity to use the metrics to foster quality. The company acted according to

my recommendations, and they were carrying out their process improvement on their own, but they were missing some points like not having a definition of ready.

It was possible to have one meeting with a second company, and I got a file to process and demonstrated how the entire capability analysis works. The team was suffering and stuck because of unresolved issues which persisted for more than 40 days. Using the MUDABuster, I found out changes in priorities were the root cause of unresolved issues. And those issues caused a lack of commitment by the team.

Today, MUDABuster presents an overview about the flow, information about deliveries, task duration, and defects. Figure 2 shows the flow situation of the company under analysis.

Figure 2: Gauge indicating the company's flow situation

The flow situation is determined by deliveries and wastes trends. MUDABuster demonstrates this information with charts such as those in Figure 3. The wastes trend is increasing more than deliveries trend in the company at this moment. In this case, the deliveries trend has a trend to increase from 1 to 3 per day in the next 10 days, while wastes trend has a trend to increase from approximately 15 to almost 50 within the same period. Deliveries and wastes trend information is combined to create the gauge present in Figure 1.

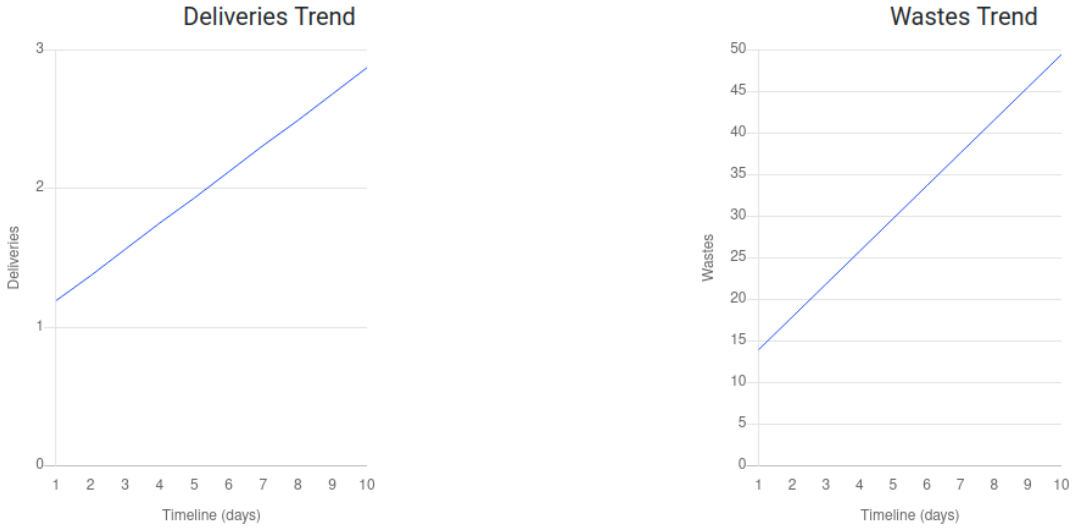

Figure 3: Deliveries and waste trends

An example of delivery information as presented by the MUDABuster is in Figure 3. The deliveries chart represents a hypothetical trend of deliveries behavior in the next 10 days. The wastes trend chart represents a hypothetical amount of wastes that could happen in the next 10 days. The table presents some descriptive statistics about weekly deliveries. These statistics aim to describe the deliveries' behavior, along with the chart.

MUDABuster also uses the following resources to clarify information regarding the flow:

- Mean, standard deviation, mode, and percentiles applied to deliveries and task duration.
- Boxplot to demonstrate the behavior of duration between in-progress and closed tasks.
- Defect's mean time to failure and mean time to repair.
- Poisson's distribution applied to evaluate defect's behavior and support strategies to tackle defects.

I discovered how good it feels to create something useful for someone else. Given the difficulties I had to tackle during this journey, seeing people and companies improving their lives and getting rid of useless stuff is priceless.

Descriptive measures of weekly deliveries

	Deliveries
Mean ± Standard deviation	4,58 items ± 3,27 items
Mode	1 item
Minimum	1 item
25 %	2 items
50 %	3 items
75 %	6,50 items
Maximum	12 items

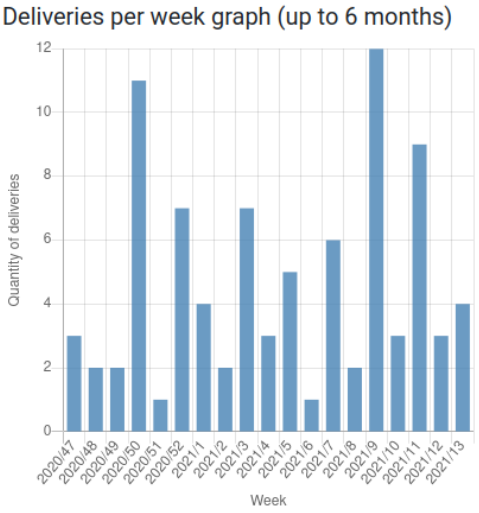

Figure 4: Deliveries information

I intend to release this software as open-source as soon as possible. I truly believe in sharing knowledge as a way of getting a better world.

3. WHAT WE LEARNED

During the course of my master's research, I've learned a lot. I've also given a lot of thought about what I have learned.

The first thing I asked myself was about the word success. The companies I stayed in touch with were successful companies, but some of them had people working overtime and suffered from low-quality code, defects, and miscommunication. What was the price of this success?

I learned there is a long journey for professionals to really understand what agile means. Dealing with low variability deliveries, small work items, thinking quality as something to protect the team against working overtime and unnecessary pressure, and having a clear vision about where the team must reach are what matters the most. After that, the methods come.

When I showed results without judgment and tried to understand the rationale behind the results, it seems there was a space to improve without looking for guilty people. I think it is important to create a space with psychological security that supports healthy growth.

4. ACKNOWLEDGEMENTS

Thanks to my wife, my daughter, and my dog for understanding the amount of time I was absorbed by this research. I've been by their side but sharing time with the mission of hunting waste. And the crazy times we lived together was an act of faith.

I'm deeply grateful to my parents. They gave me the life, the support, and the courage to face moments like the one me and my family are living today.

I like to thank all the support I got from my supervisors. My main supervisor is a Mathematician and PhD in Statistics. She was so brave to get into the wild software development world. My co-supervisor gave me so much freedom to navigate within the master's program jungle.

I'd like to mention companies that opened space to run the practical step of my research, but for anonymity reasons it was not possible. However, I'll be thankful forever.

And I'm grateful for my Shepherd, Susan Burk, for the support, insights, questions, and lovely conversations we had during the experience report writing process: *Thanks, Shepherd, we couldn't have done it without you!*

REFERENCES

[Caroli] Caroli, Paulo. *Lean Inception: How to Align People and Build the Right Product* 1st Edition. Editora Caroli. 2018.
[Graziotin] Graziotin, D. et al. What happens when software developers are (un)happy. Journal of Systems and Software. https://www.sciencedirect.com/science/article/pii/S0164121218300323?via%3Dihub
[Jones] Jones, D., Womack, J. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation* 2nd edition. Free Press, 2003.
[Poppendieck] Poppendieck, M., Poppendieck, T. *Implementing Lean Software Development: from concept to cash.* Addison-Wesley Professional. 1st edition, 2006.
[Power] Power, Ken and Conboy, Kieran. A Metric-Based Approach to Managing Architecture-Related Impediments in Product Development Flow: An Industry Case Study from Cisco. 2015 IEEE/ACM 2nd International Workshop on Software Architecture and Metrics. http://ieeexplore.ieee.org/document/7174844/
[Vargas] Vargas, Ricardo Viana. *Manual prático do plano de projeto: utilizando o PMBOK Guide* 3rd Edition. Brasport, 2014.