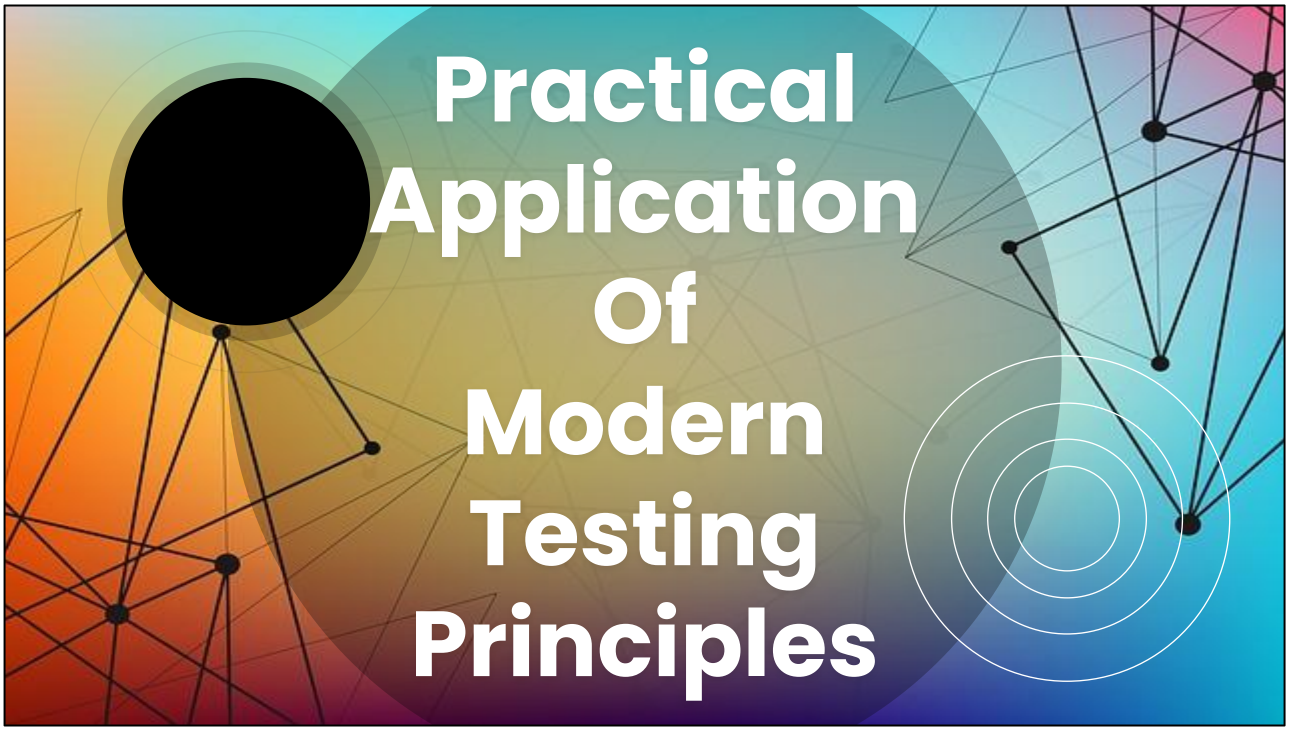
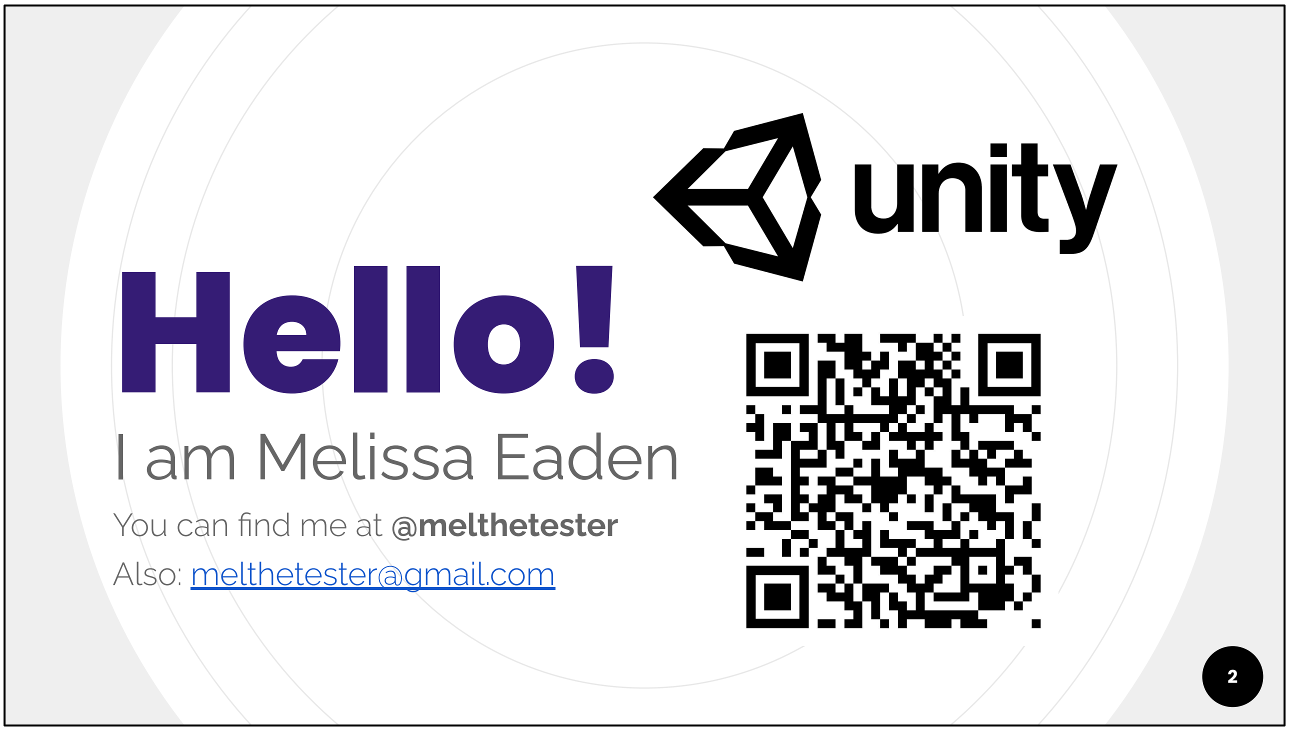


Practical Application Of Modern Testing Principles

https://www.ministryoftesting.com/dojo/lessons/modern-testing-principles

https://www.angryweasel.com/ABTesting/modern-testing-principles/



Hello!

I am Melissa Eaden

You can find me at **@melthetester**

Also: melthetester@gmail.com

About Me

Warning!

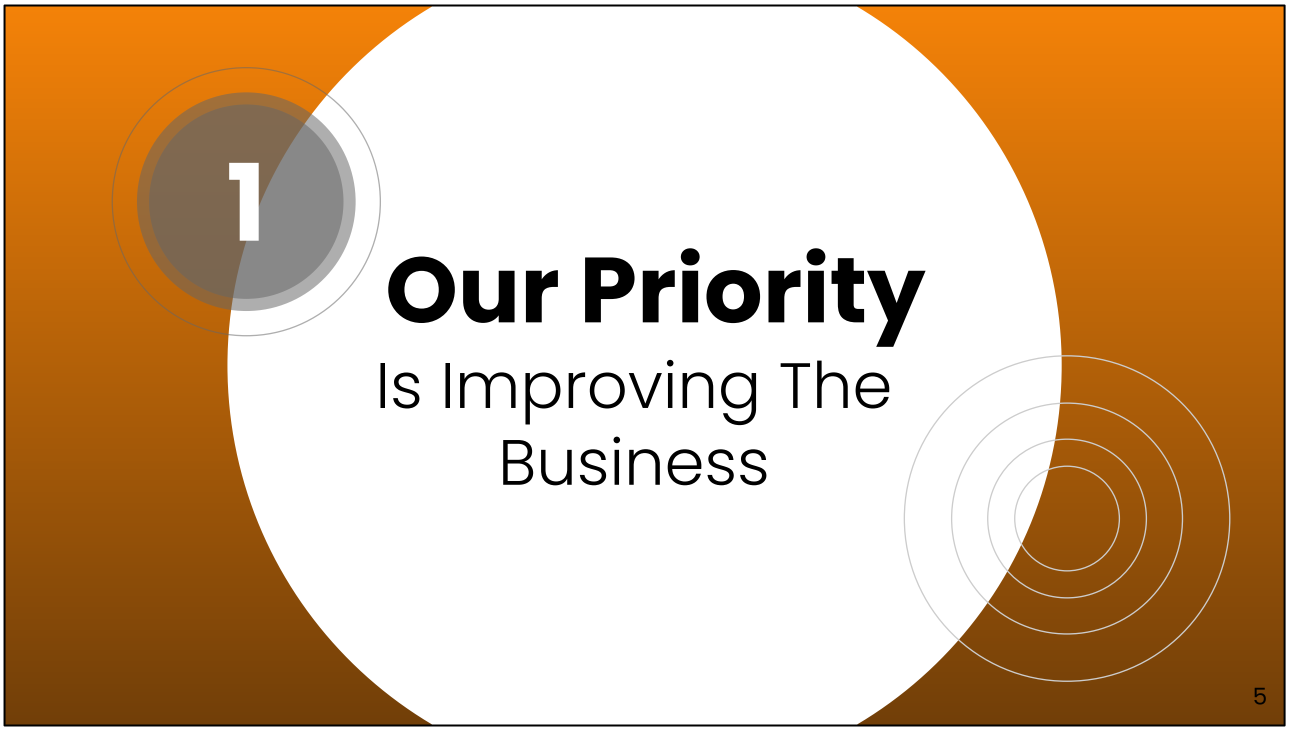
WARNING! The information contained here is not going to transform you into a lean, mean development team, at least not over night. This is going to take work, and progress towards goals set by a TEAM of folks dedicated to doing quality work.

Disclaimer

4

DISCLAIMER! The Modern Testing Principles are not my creation. Alan Page and Brent Jenson have been talking about these principles for a few years. I've written an article about them myself, and I have listened to AB Testing Podcast which is a wealth of information. I am here to impart what I've learned and what you can learn yourself. I can imagine that a lot of you have already been doing some of the things I'm going to talk about today. These are all ideas about what you can focus on, but ultimately, you'll need to do what works for your team, and your company.

Podcasts 81-88 specifically talk about MTP - https://www.angryweasel.com/ABTesting/



1

Our Priority

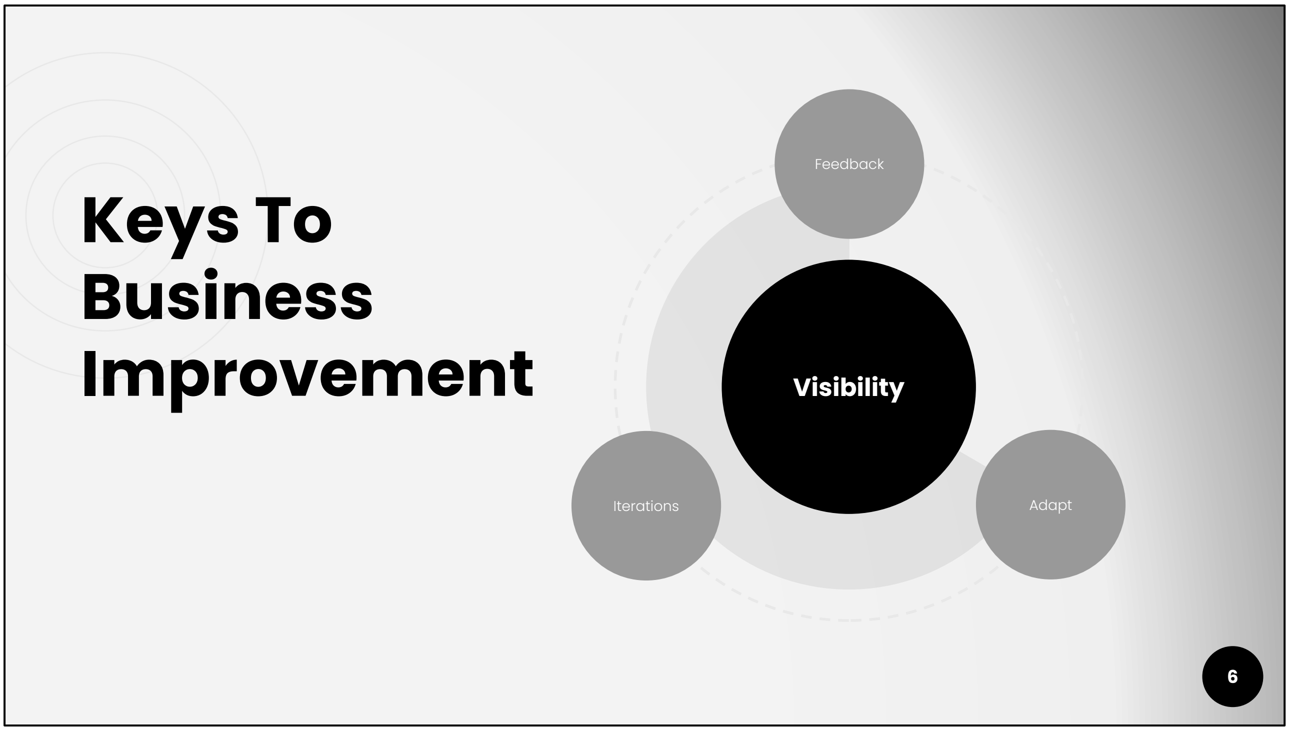
Is Improving The Business

There are three main tasks that can help you achieve this principle.

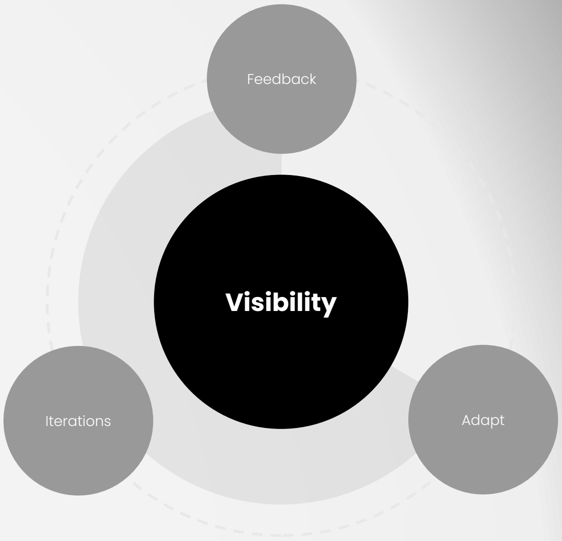
Build in feedback loops everywhere.

Adaption is survival.

Iteration is the key to the first two.



Keys To Business Improvement

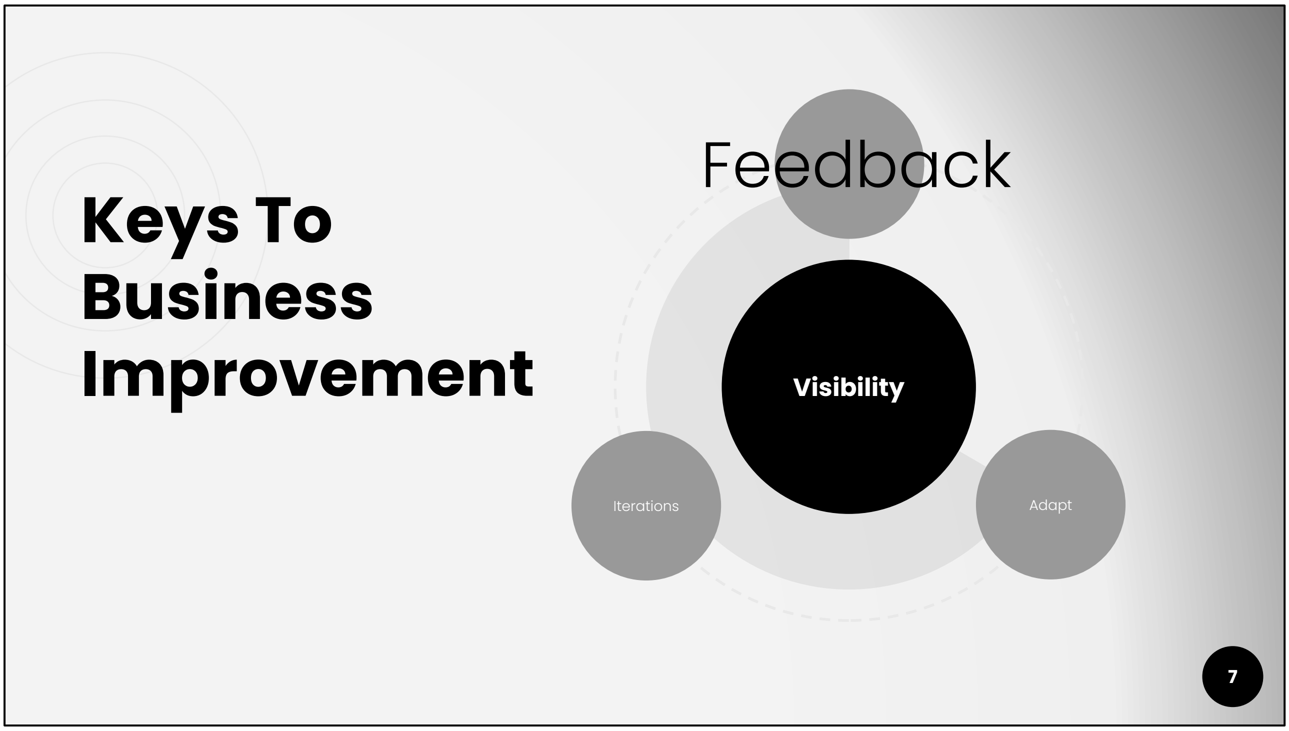


There are three main tasks that can help you achieve this principle.

Build in feedback loops everywhere.

Adaption is survival.

Iteration is the key to the first two.



Keys To Business Improvement

Feedback
Visibility
Iterations
Adapt
7

Building Feedback Loops

I'm not only talking about doing retros and having team meetings. I'm talking about getting customer feedback and stakeholder feedback, and feedback from other teams or feedback out to other teams. Even feedback from the software itself.

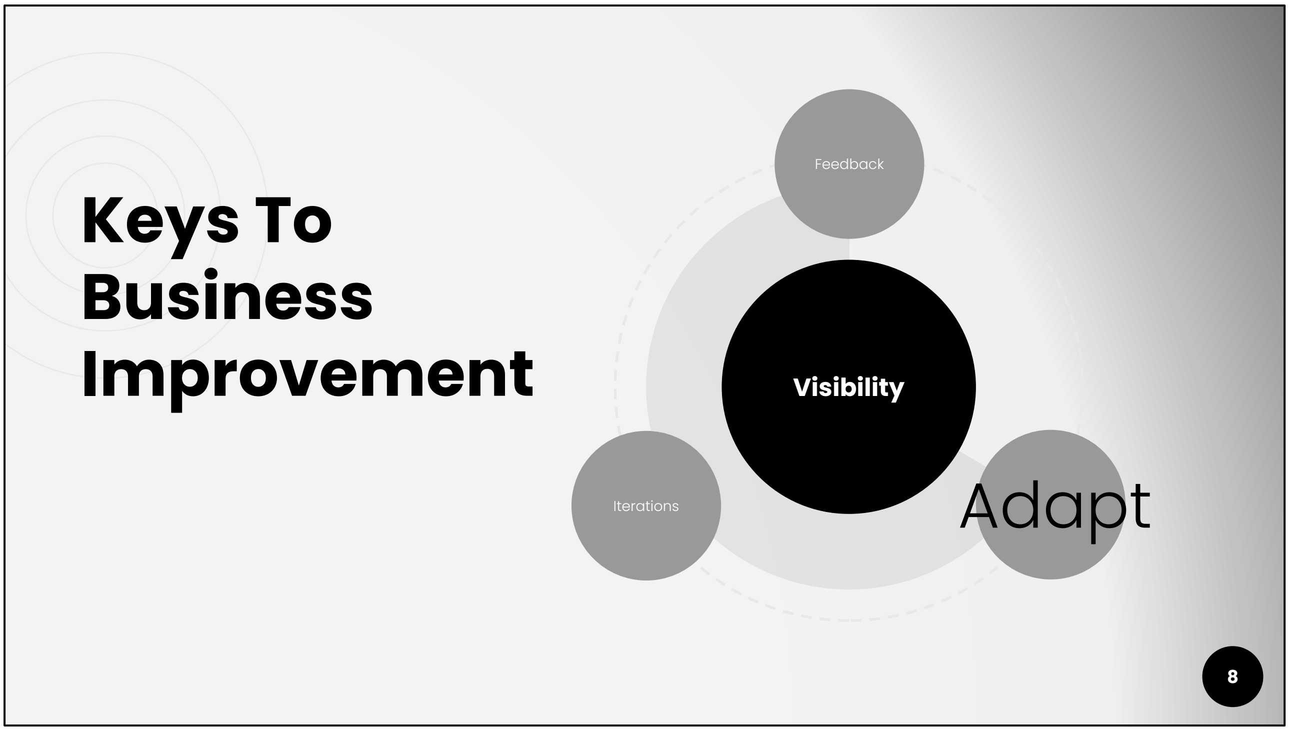
To do this, you have to have ways for people to give and receive feedback.

If you are thinking some surveys cover this section and you are all good, you would be at the beginning of what you could get. We are in the information age. Information and Context are Royalty. Treat them as such. It's worth it to develop open lines of communication, to cross train people, to instrument your software within the legal limits of the law. The information is there to be able to help your customer. Your customer is the only one that is going to be able to give you the ultimate feedback about whether something is working for them or not. That information needs to get to the teams involved. If you can get the software to give you information about what your customers are doing to it, that gives you another point of context.

All of these things can give you a clear picture of the problem you are trying to solve for the customer.

Crucial Conversations:
https://www.goodreads.com/book/show/15014.Crucial_Conversations
Critical Feedback:

https://www.thoughtworks.com/en-ca/insights/blog/5-ways-faster-and-more-effective-feedback



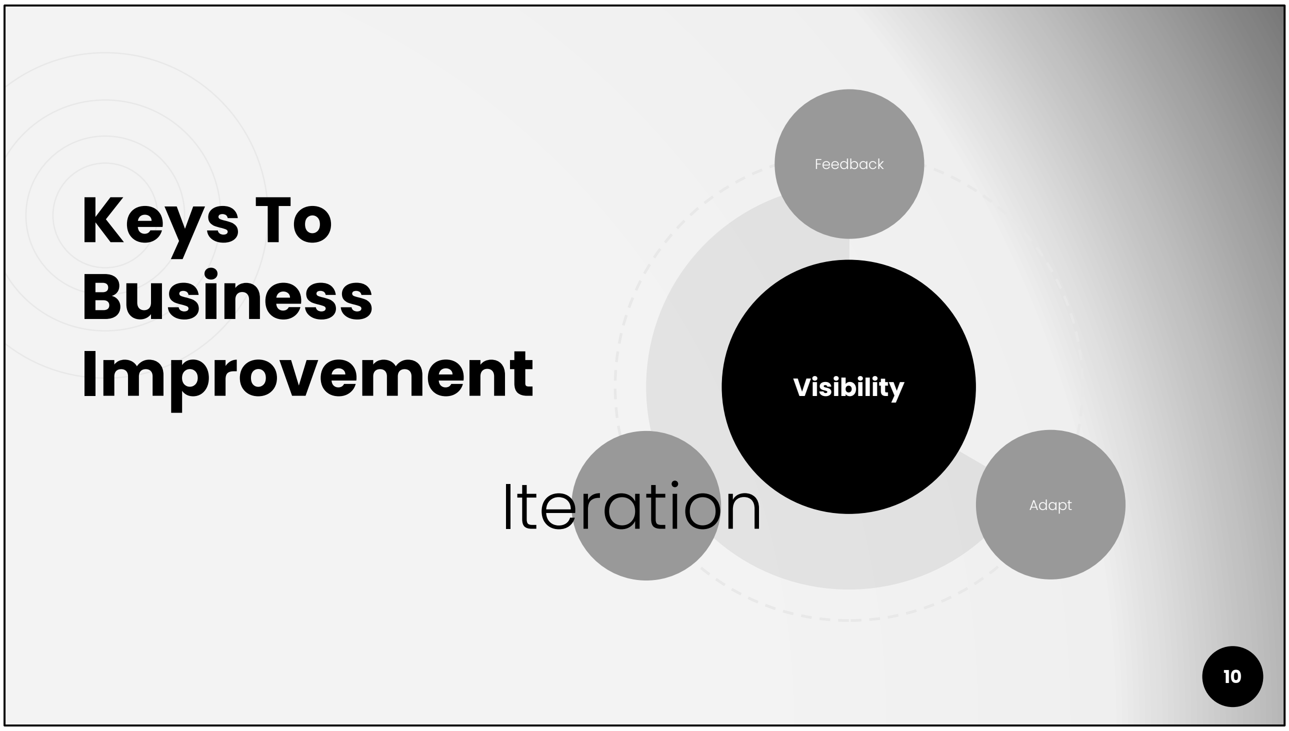
Keys To Business Improvement

Feedback
Visibility
Iterations
Adapt

Adaption isn't only a Biology Term
What good is all that feedback if you aren't doing anything with it. Adaption is the key to using that feedback. The faster you can adapt to feedback coming in, and the information it gives you, the faster your team can solve customer problems. Or the faster you understand whether adaption is necessary – if it’s business critical, or a dead end, the better off you business will be.

The systems in which you handle your feedback can create a bottleneck to adaptation. This slows down anything you might come up with and experiment and try to solve the customer(s) problems, especially if what you have created is a whole ecosystem to solve problems, that whole ecosystem comes with its own problems as well because we are humans and we make software.

However, we are one of the fastest adapting species on the planet. We can make or break a system, any system really. By using the feedback we get, and experimenting through failures to reach a really good solution, is all part of adaptation.



Keys To Business Improvement

Feedback
Visibility
Iteration
Adapt

Pivot on Iterations

Feedback and Adaption are all dependent on Iterations. It's not only a Scrum or Agile term, it is also a term you could apply to how you do develop an idea, build software, or how that software is delivered to your customer.

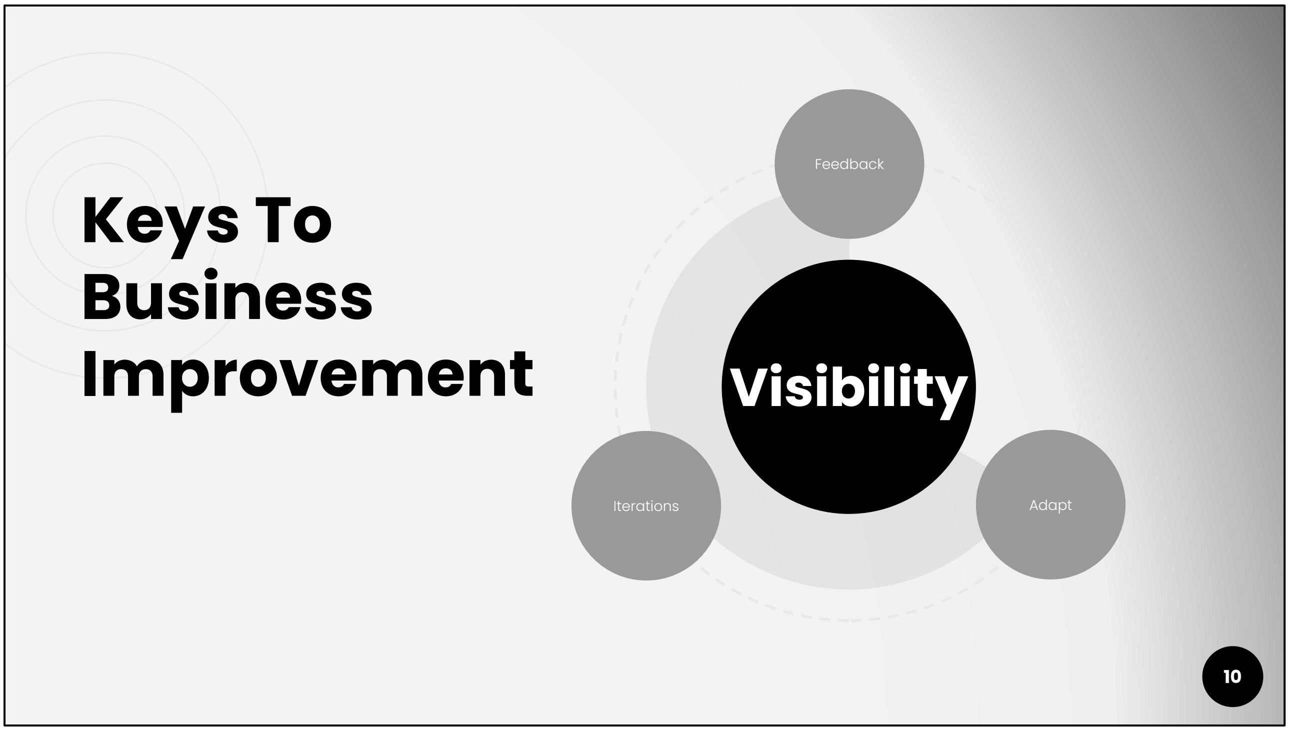
To get to some of the fastest customer feedback response times, you have to build in systems that can handle what you need, to scale. To do this over time, to deliver an idea to the customer and get feedback fast enough to adapt to the next changes or the next data driven idea, your teams have to employ a combination of things from automation, people skills, and data from customers, which might only be the tip of the software iceberg you are trying to resolve. There are probably other things that could mix into this like tools and infrastructure, but starting with development and customers is key.

Things that can block being able to do faster iterations are tech debt, lack of customer information/data, lack of visibility into the development process, lack of visibility into testing and how it is integrated into the development process. DORA metrics

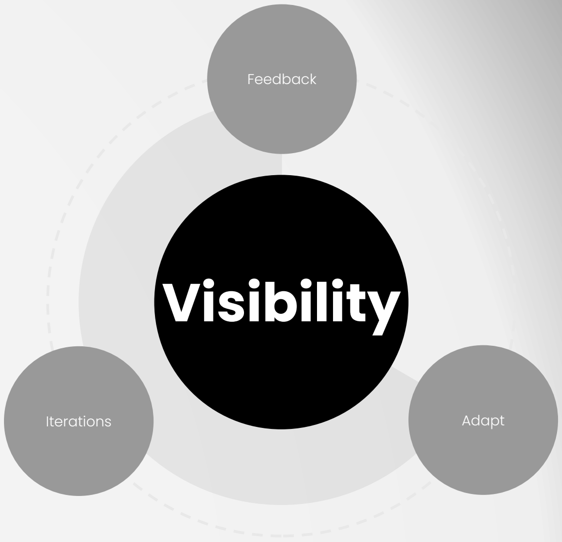
Accelerate: https://www.goodreads.com/en/book/show/35747076-accelerate

Defect backlogs:
https://testingandmoviesandstuff.blogspot.com/2022/03/the-defect-expiration-date.html

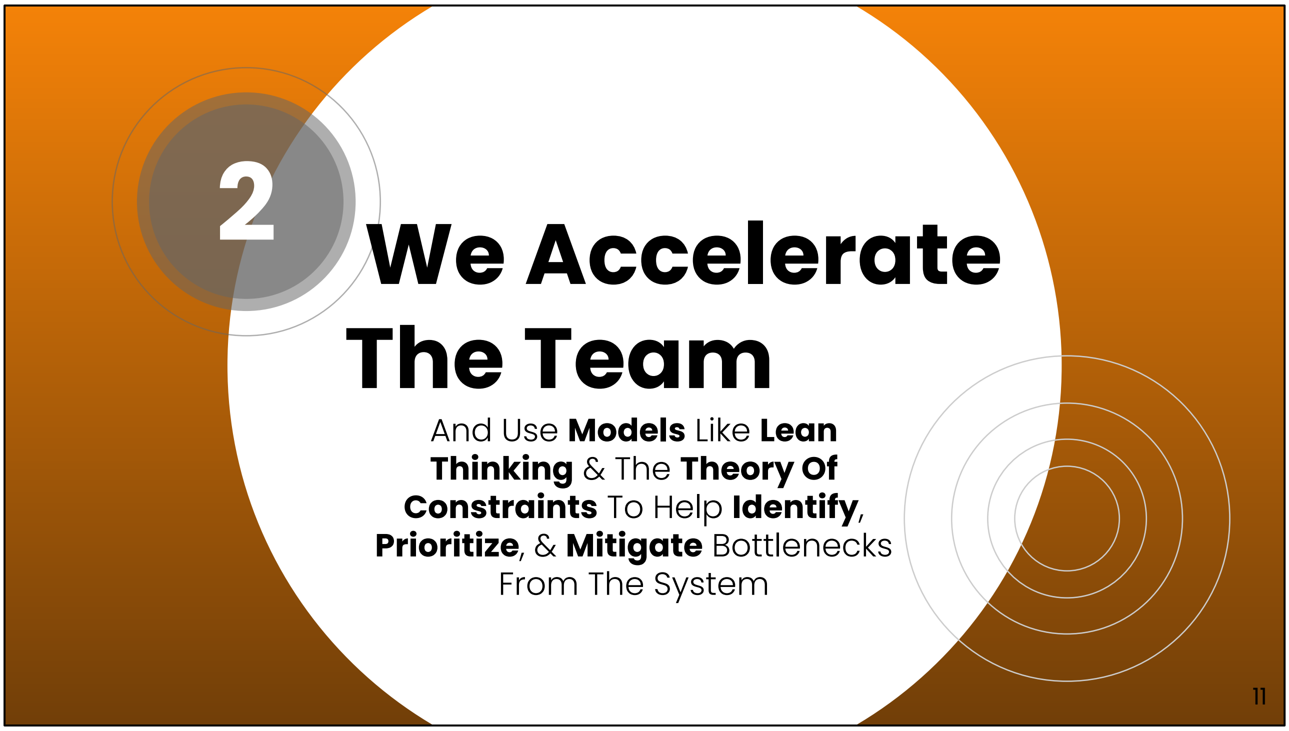
YOU can NOT really bake process in at the end. You might be able to do a bubble gum/duck tape quality patch, but it's still a patch. That means you have tech debt to deal with, which can slow iterations because you have to stop and do the work to make the patch not a patch but a real fix that actually improves the system.



Keys To Business Improvement



You could do any of the last three ideas I mentioned but without making them visible to the audiences that are participating, it will be half as effective.



2

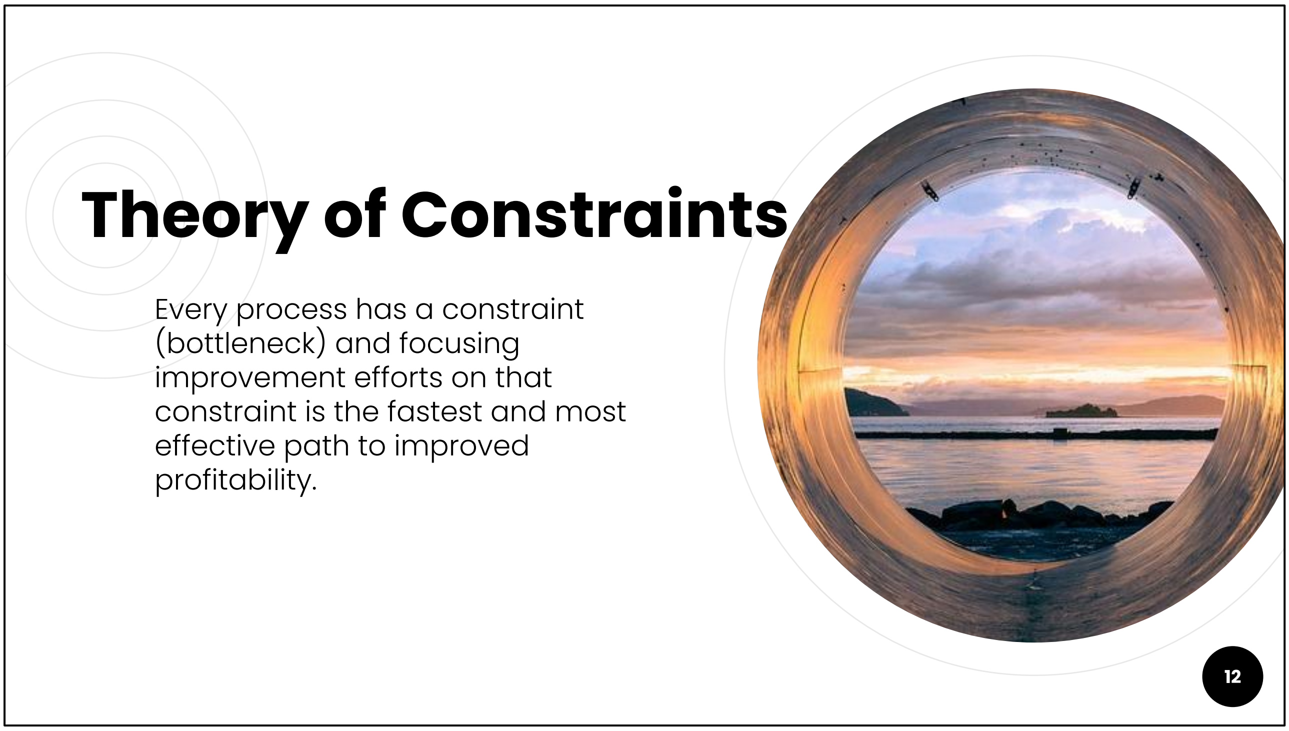
We Accelerate The Team

And Use **Models** Like **Lean Thinking** & The **Theory Of Constraints** To Help **Identify**, **Prioritize**, & **Mitigate** Bottlenecks From The System

Simply put, your software does not live in a vacuum. When your software hits production and is being used by customers, you'll hit all the unknown unknowns that could possibly be and will ever be, that you won't ever find on a local or staging environment.

Two things are important here to understand and utilize:

- Theory of Constraints
- Systems Thinking



Theory of Constraints

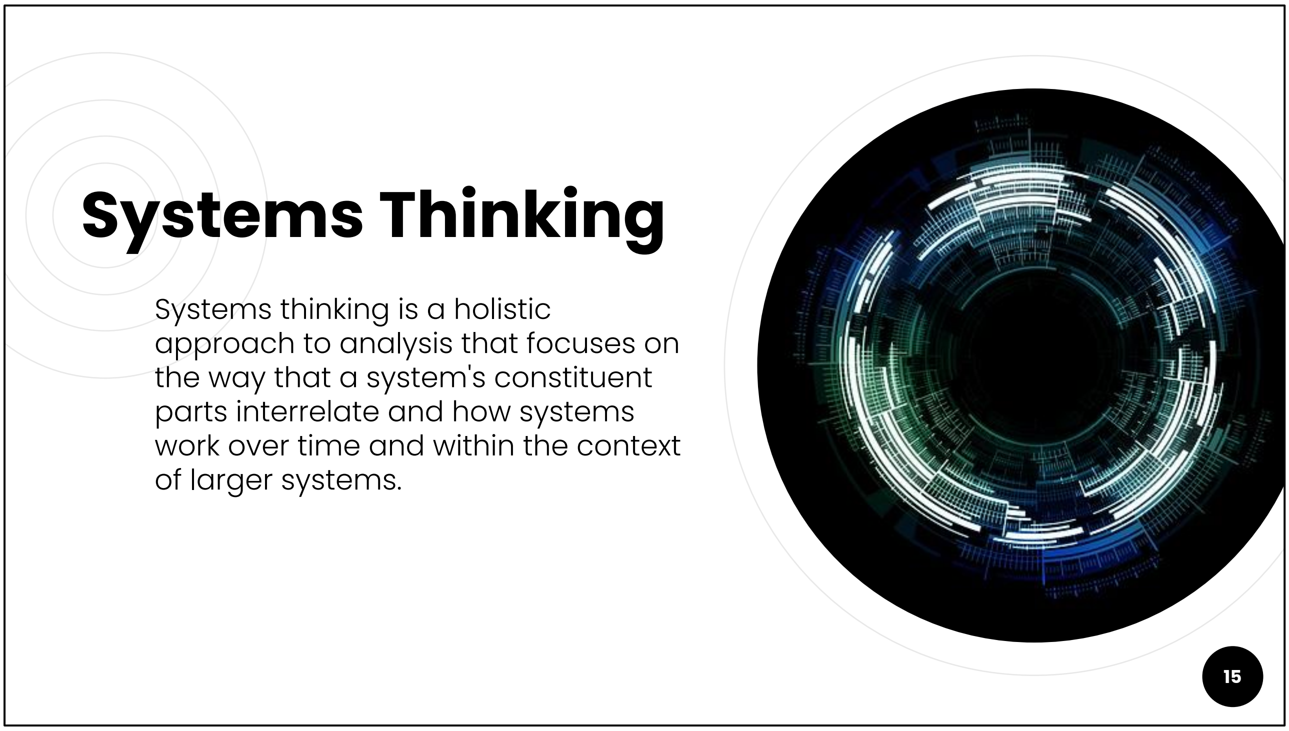
Every process has a constraint (bottleneck) and focusing improvement efforts on that constraint is the fastest and most effective path to improved profitability.

https://www.leanproduction.com/theory-of-constraints.html

The Theory of Constraints is a methodology for identifying the most important limiting factor

The best way to deal with constraints is to find the biggest, toughest one to deal with, and then tackle that head on. Get everyone working on removing the constraint, whatever it is, even if it's only offering moral support to the team trying to work through the problem.

(VP is proud of the number of tests run - removed the constraint of running a massive number of tests/ critical defects are still coming up - why? What was the real constraint? Are we testing the right thing?)



Systems Thinking

Systems thinking is a holistic approach to analysis that focuses on the way that a system's constituent parts interrelate and how systems work over time and within the context of larger systems.

https://thesystemsthinker.com/introduction-to-systems-thinking/

https://searchcio.techtarget.com/definition/systems-thinking

(General description)

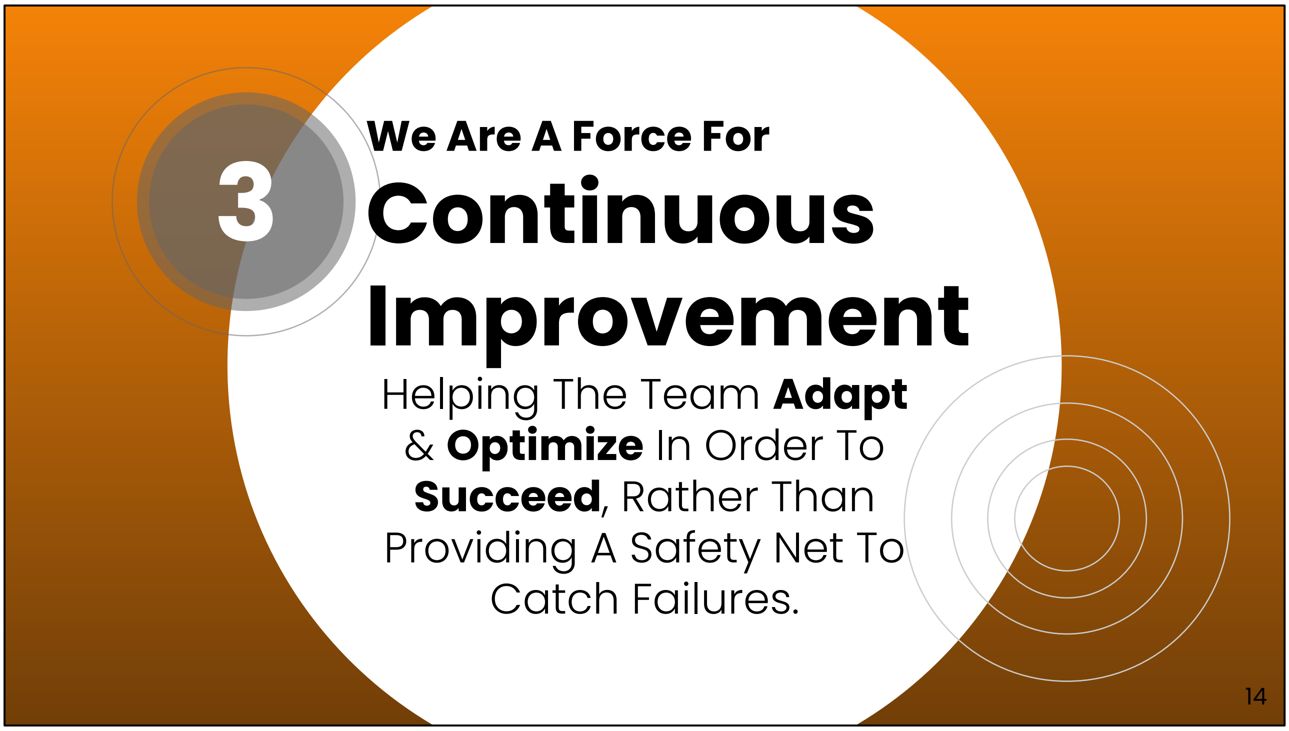
You have to understand the system or systems you are working with to understand where the bottlenecks are located. This is where we balance iterations and adaptation, by understanding what the system can handle and how often we can introduce change into that system. This is why understanding where the bottlenecks are and where fast, light, changes can be better, if your system has been adapted to handle things like this. Your system of change could be the bottleneck. The code could be the bottleneck, or the lack of different kinds of testing. Your system is going to tell you immediately what it needs to move faster because the minute you try to move faster, you'll hit a massive wall of something which won't let you pass.

There might be a temptation to brutally remove a constraint, whether that's people or processes or tools. Often it's a combination of things not one particular thing that slows a system down. Alan and Brent often say it's the difference between removing a band-aid or a tourniquet.

The examples people like to point out, one Alan likes to point out especially is when Yahoo fired all of their testers. They found a constraint and they brutally removed it. They have been trying to recover ever since. They understood that they had a constraint but instead of studying it and understanding what might happen by shifting certain types of testing to developers, they removed a large part of their system knowledge base along with anyone that knew anything about how the system was tested. If they had done a more gradual change, helped foster adaptation for everyone whether they moved to specialized testing on particular teams, or worked in product support, or created a test coach role to help developers learn and take on new skills, it might have been a different story for Yahoo.

A great example of how to do this correctly comes out of Microsoft actually, or at least one division. Other divisions have done similar things like Yahoo. The Cloud Infrastructure org chose to make changes over two years, and worked with everyone on teams to figure out how they could remove blockers and constraints, and how they could move team members to other places which can use their perspective talents, or offer training to help them into another role. When a company plans for adaptation using iterations, they can achieve really cool things.

https://www.agilealliance.org/resources/sessions/microsoft-devops-transformation-donovan-brown/



3

We Are A Force For

Continuous Improvement

Helping The Team **Adapt** & **Optimize** In Order To **Succeed**, Rather Than Providing A Safety Net To Catch Failures.

Testing is often labeled a bottleneck. It's often why that department gets cut first, mostly because their value is little understood to the system and often seen as an impediment to adaptation.

Testing should be a force for improvement not a safety net.

Practical application of MTP#3

- Be visible to be valuable
- Critical Thinking

Visibility Is Valuable

Siloed Knowledge Is A Value KIller

Visibility is Valuable - Silos are only good for grain

When companies don't understand what testing is and does, that very lack of understanding can often mean testing and testers are sacrificed for speed to market. While getting things to the customer is important to feedback and adaptation, there is no point unless it's done with quality in mind, by looking at risks and how the system is performing with and without testing.

If you are on a team where you don't understand how testing works, find out.

If you are a tester and you aren't talking to your developers about how and what you are testing and why, change that.

If you are a company and you don't understand the folks doing the testing, whether that's developers or testers or both, figure that out. Make it visible. DEMO things to the company at large. Make quality part of the culture, not just in the development organization, in every part of the organization.

Make constraints visible. Make successes visible. Make failures visible.

All this information leads to understanding the system, and adaptation. This is when people in an organization will come up with their best ideas. They might not work, but if you give them the chance to see if they can work or not, then you're getting valuable feedback about that failure.

Instead of telling someone no, to an idea, what would happen if you gave them a sandbox and let them play in it and show it off whether it works or not? This is data you can get that can lead to another iteration which could lead to the adaptation, which can lead to amazing things.

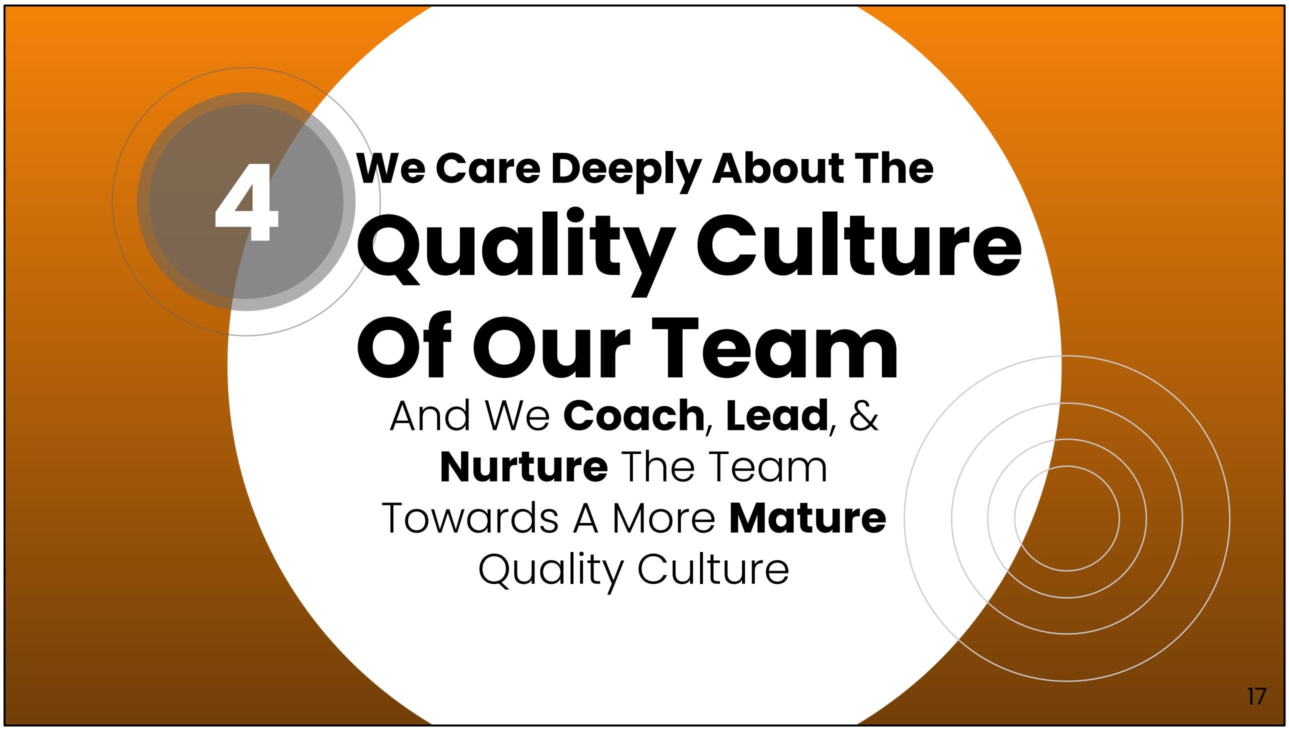
Critical Thinking

Thinking Fast & Slow

Critical Thinking
(description: the objective analysis and evaluation of an issue in order to form a judgment.)

There are a ton of things out there about critical thinking. The first one that always leaps to mind is Thinking Fast & Slow by Daniel Khanman

Sometimes people are so obsessed with solving a problem, they sometimes lose sight of what the problem was to begin with, and the fact that the customer can only determine quality. (The tester/developer/engineering team is not the customer proxy, they are sometimes an advocate, but we'll get into that more in principle #5)



4

We Care Deeply About The

Quality Culture Of Our Team

And We **Coach**, **Lead**, & **Nurture** The Team Towards A More **Mature** Quality Culture

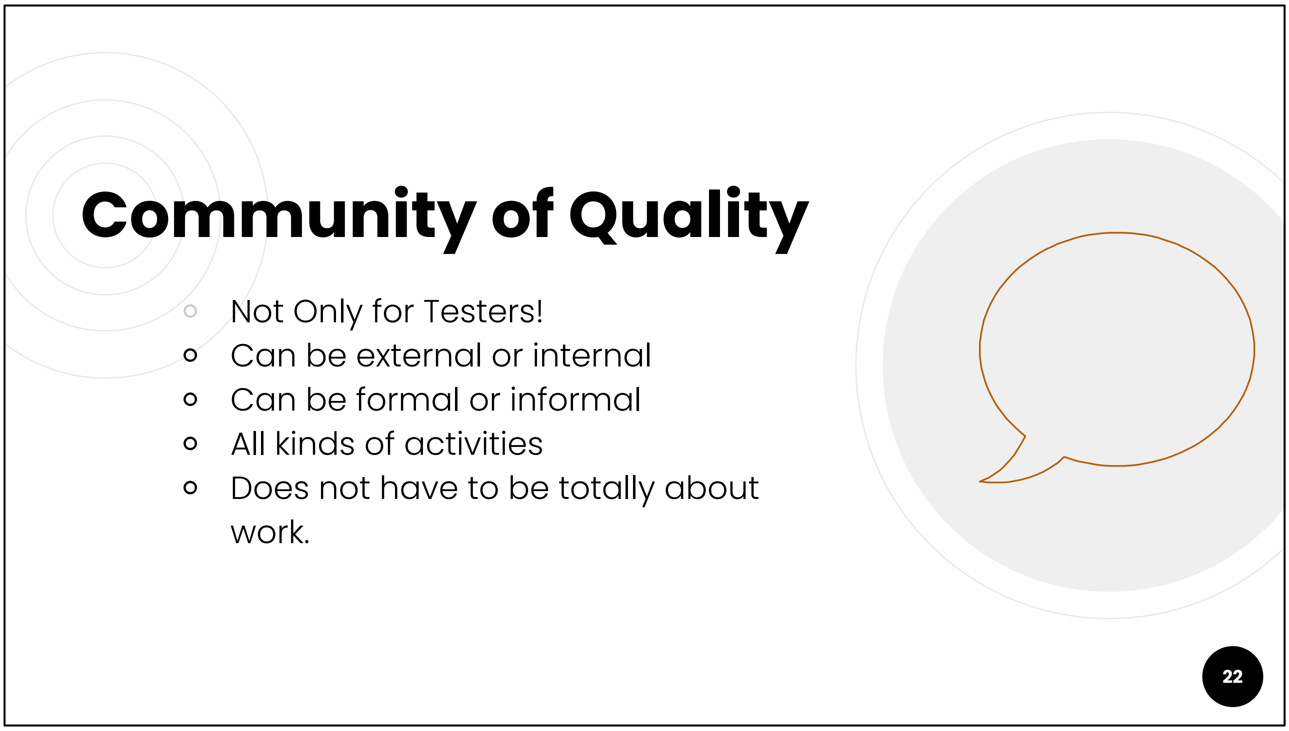
17

- Creating a community of Quality – this is a whole organization thing, not just the development team. (if you exclude your product/program/management folks, you're excluding the glue that can be a driving force for quality)

- Create a culture of Quality – this is where sharing, being visible about problems and ideas can lead to fast feedback and iteration/innovations.

- Growing Leader/Leadership – needs to understand that they are integral in creating an environment in which quality is fostered while balancing it with business needs. This is often where risk based approach to prioritization helps. Visibility of the risks can create better prioritization and reduce firefighting or reactive approaches which creates more stability.

  Ex: knowing when to get serious about performance and load testing vs waiting until there is a large scale failure then try to fit that kind of testing into the system while it's under stress.



Community of Quality

- Not Only for Testers!
- Can be external or internal
- Can be formal or informal
- All kinds of activities
- Does not have to be totally about work.

Community of Quality
Hint: It's not only for testers

Creating leaders and instilling leadership into a community. That community is important to create, build and maintain. This is what carries an organization forward. Communities can be external or internal, or both.

Some of the most successful I've seen are Ministry of Testing and Test Automation University. They center around testing, but these communities are not only for testers. Both of these are addressing the software development community needs and desires to bring quality into whatever they are working on, whether that is hiring the right candidates, or having everyone on a team learn about different kinds of testing. Because testing isn't your job focus doesn't mean that these communities and others like them are not for you. I'm also involved with Women Who Code. It's another community that focuses on helping women in technical roles achieve goals, learn, and adapt to job market changes.

Teams and organizations can create a community of quality in a company as well. This could mean having groups of people attend a conference like this one and presenting information and new ideas to the organization. It could mean running a book club, it could mean doing lunch and learns where everyone can learn something about the app or supporting tools the organization uses to develop software.

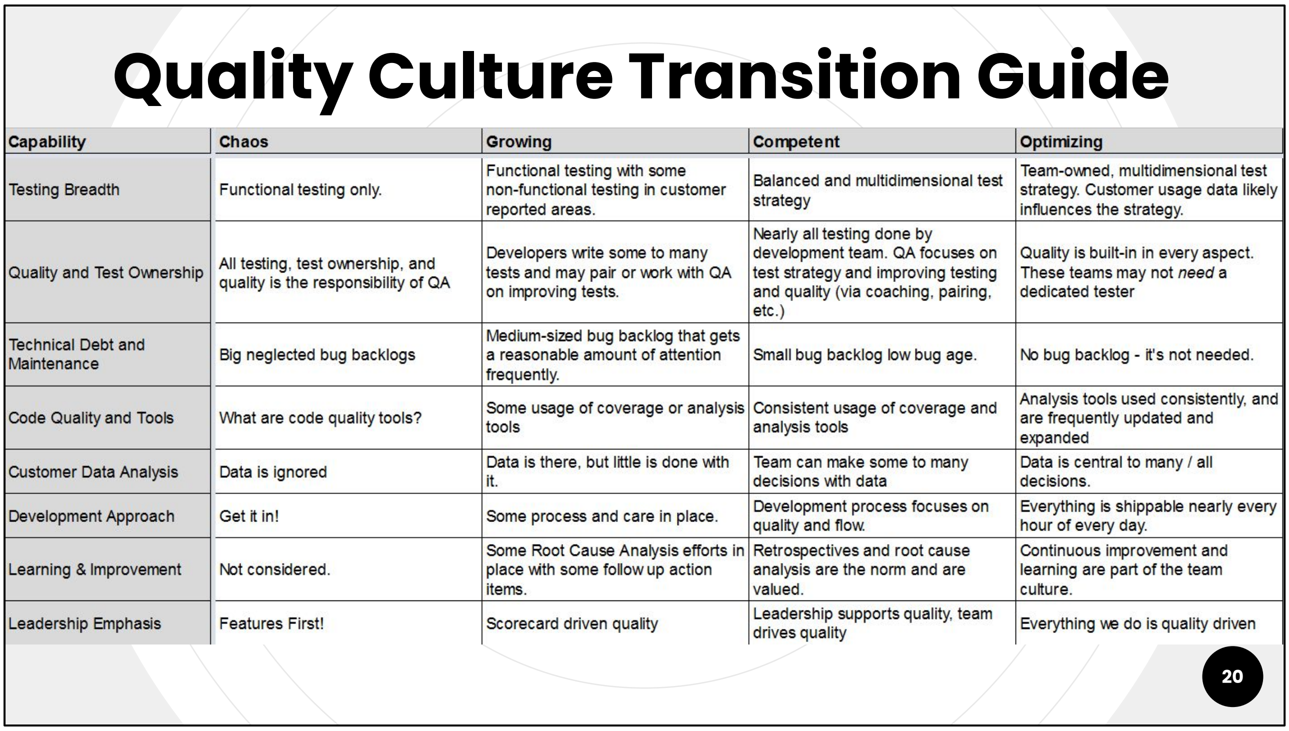
As long as communities are inclusive and helpful, then they are worth the time investment, whether it's attending and/or maybe organizing. You are furthering the goal of creating shippable quality.

Culture of Quality

- Whole Team Involved
- Pairing
- Mentoring
- Leadership
- Working towards a Mature Quality Mindset.

Culture of Quality or Quality Culture - can be applied at any level of the company. It only takes leadership and management to be supportive of it. It also takes leadership that understands
Quality to help other leaders be supportive of the processes. IE Quality Coaches, SDETs, Test Engineers, Developers, Program/Product, Management - anyone that gets it that can be an advocate leader can also help other leaders understand why this kind of culture is necessary to the success of the business.

There is a model for measuring where your team is at on the quality mindset scale. Alan came up with it and it's a work in progress. Not that not all models are correct, they are only models. You might even adapt his example based on the goals your team wants to achieve.



Quality Culture Transition Guide

Capability	Chaos	Growing	Competent	Optimizing
Testing Breadth	Functional testing only.	Functional testing with some non-functional testing in customer reported areas.	Balanced and multidimensional test strategy	Team-owned, multidimensional test strategy. Customer usage data likely influences the strategy.
Quality and Test Ownership	All testing, test ownership, and quality is the responsibility of QA	Developers write some to many tests and may pair or work with QA on improving tests.	Nearly all testing done by development team. QA focuses on test strategy and improving testing and quality (via coaching, pairing, etc.)	Quality is built-in in every aspect. These teams may not *need* a dedicated tester
Technical Debt and Maintenance	Big neglected bug backlogs	Medium-sized bug backlog that gets a reasonable amount of attention frequently.	Small bug backlog low bug age.	No bug backlog - it's not needed.
Code Quality and Tools	What are code quality tools?	Some usage of coverage or analysis tools	Consistent usage of coverage and analysis tools	Analysis tools used consistently, and are frequently updated and expanded
Customer Data Analysis	Data is ignored	Data is there, but little is done with it.	Team can make some to many decisions with data	Data is central to many / all decisions.
Development Approach	Get it in!	Some process and care in place.	Development process focuses on quality and flow.	Everything is shippable nearly every hour of every day.
Learning & Improvement	Not considered.	Some Root Cause Analysis efforts in place with some follow up action items.	Retrospectives and root cause analysis are the norm and are valued.	Continuous improvement and learning are part of the team culture.
Leadership Emphasis	Features First!	Scorecard driven quality	Leadership supports quality, team drives quality	Everything we do is quality driven

Quality Culture Transition Guide

While this isn't a maturity model, it can lead to mature development practices. This model is more about setting goals and getting to goals. If you know where your current team sits with a particular factor, knowing that there is a goal that can be reached is nice. This takes time. You can't leapfrog through some of these goals. And these goals might not even fit what you want your Quality Culture to achieve.

The tendency is to create a checklist out of this - tick the boxes, gives the warm fuzzy feeling. The problem with checklists is that over time, the discussions stop happening and only the work gets done. It reduces things down to needing to be done vs asking why they need to be done (directly relates to #2/#3).

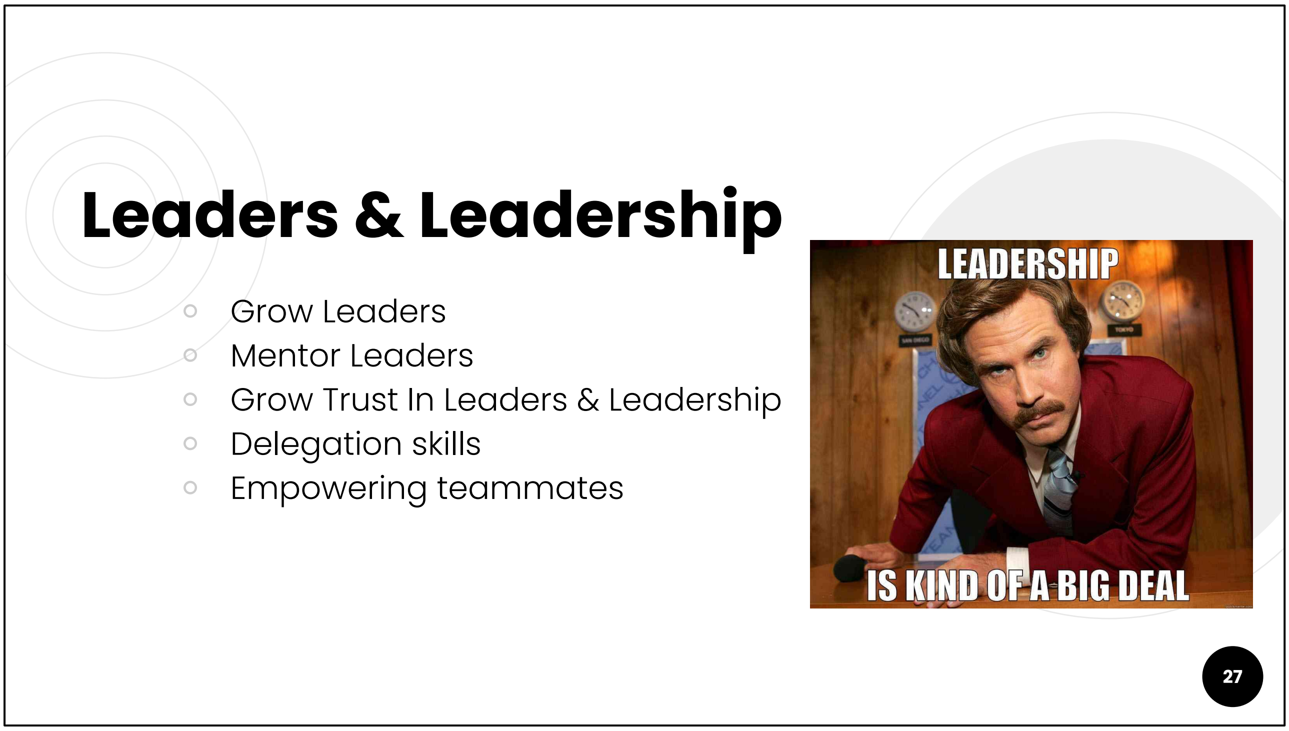
ADKAR Model

- Awareness
- Desire
- Knowledge
- Ability
- Reinforcement

ADKAR
(description)
You don't have to use the mindset model. Instead, you could use the ADKAR model to find goals and break them into incremental steps, which can be done in iteration, gathering feedback, and then adapting to that feedback.

https://www.prosci.com/adkar/adkar-model

https://www.deviantart.com/empirestripsback/art/admiral-ackbar-meme-part-3-312973398



Leaders & Leadership

- Grow Leaders
- Mentor Leaders
- Grow Trust In Leaders & Leadership
- Delegation skills
- Empowering teammates

Leaders and Leadership

A community and culture need leaders

To make sure you have leaders into the future, which is part of building a community or a culture, you'll need the following:

- A way to grow leaders
- A way to mentor leaders
- Growing trust in the leadership/leaders you build.

Episode 85 has a lot of great information about leadership and growing leaders. It talks about getting permission to lead, and knowing the difference between delegation and empowerment. It's a large part of trying stuff, learning from the failures and successes. Doing small experiments that can add up to a big change over time.

One quote that Alan uses often:

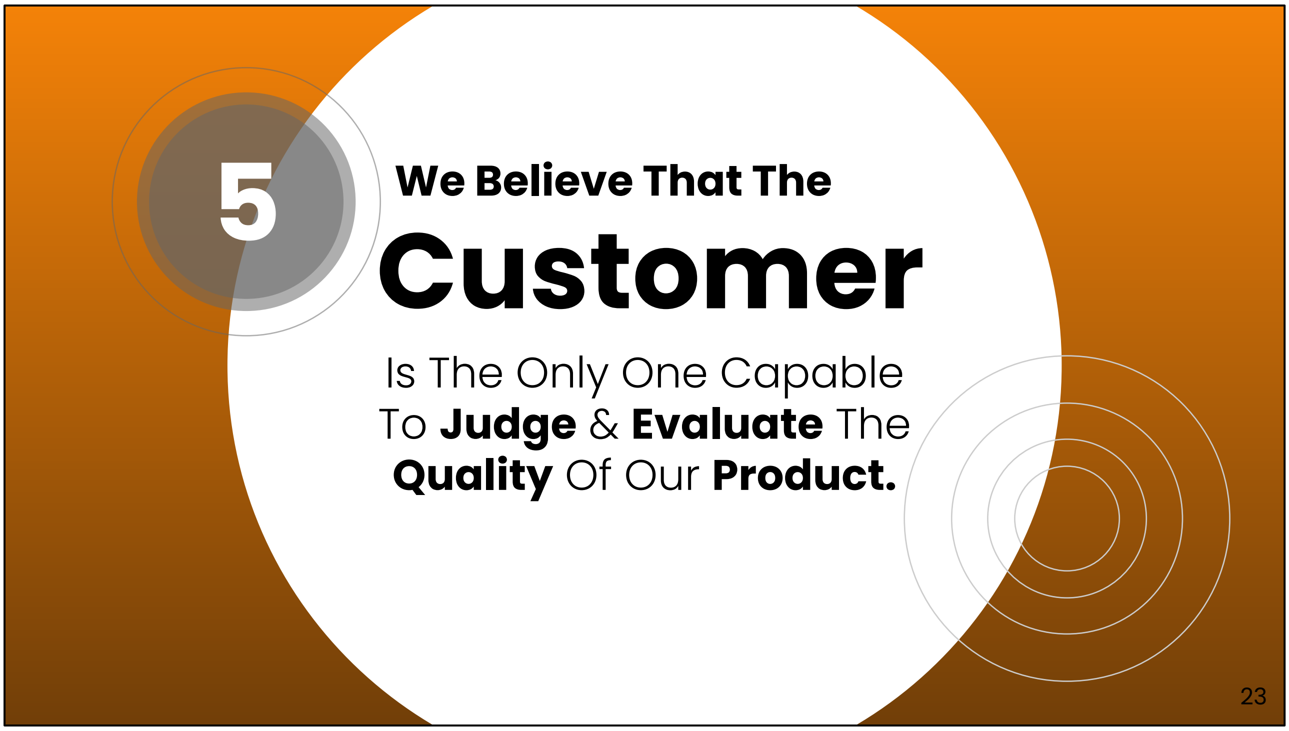
"Leadership is disappointing people at a rate they can absorb." - **Marty Linsky**

“Leadership is disappointing people at a rate they can tolerate.” - Alan Page

Celebrate the things leading you /your organization in a direction you want to go, make them visible. Make the failures visible so others can learn from that failure and try something different.

(off-site story about devs trying to accomplish a feature, which had been tried before, but they didn't ask anyone nor did they tell anyone what they were doing. They could have saved three days of discovering finding the first failure, when they could have used the info to see if they could actually find the solution. Their approach to the problem might have changed if they had only talked about what they were doing. Still - they learned from the failure and when other ideas were in the works, they asked about them before getting started.)

http://www.quickmeme.com/p/3w2ga4



5

We Believe That The

Customer

Is The Only One Capable
To **Judge** & **Evaluate** The
Quality Of Our **Product.**

MTP#5

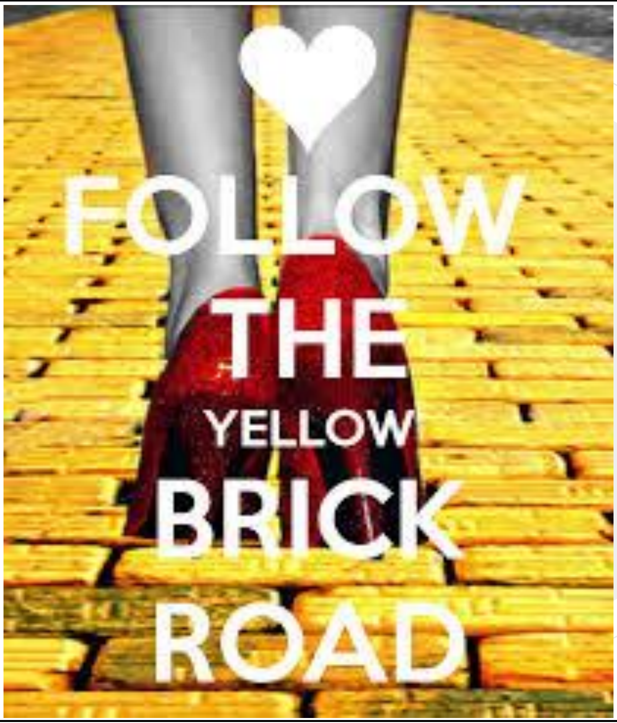
- What is the problem being solved?
- Data driven decisions
- Data driven development

Problem Solving is not the same as Problem Discovering - passive vs active

- What is the biggest problem customers face?
- Prioritize the problems
- Treat requirements as hypotheses. A hypothesis is always a question - what data do you have that can help confirm the hypothesis?
- What delivers the most value to the customer and solves the biggest customer problem? Focusing on the right problem - what's necessary and what's noise?
- Customer Value == Company Growth over Time

This should sound very familiar as it's borrowing from Agile pretty heavily. If you are working in an Agile environment, this is like turning that up to 11. If the team working on a particular part of a problem understand the whole customer problem being faced, they are more likely to organize and communicate with other teams working on the same problem. Your development teams are not mushrooms, even if sometimes office lighting suggests otherwise.

Customers Lead The Way...

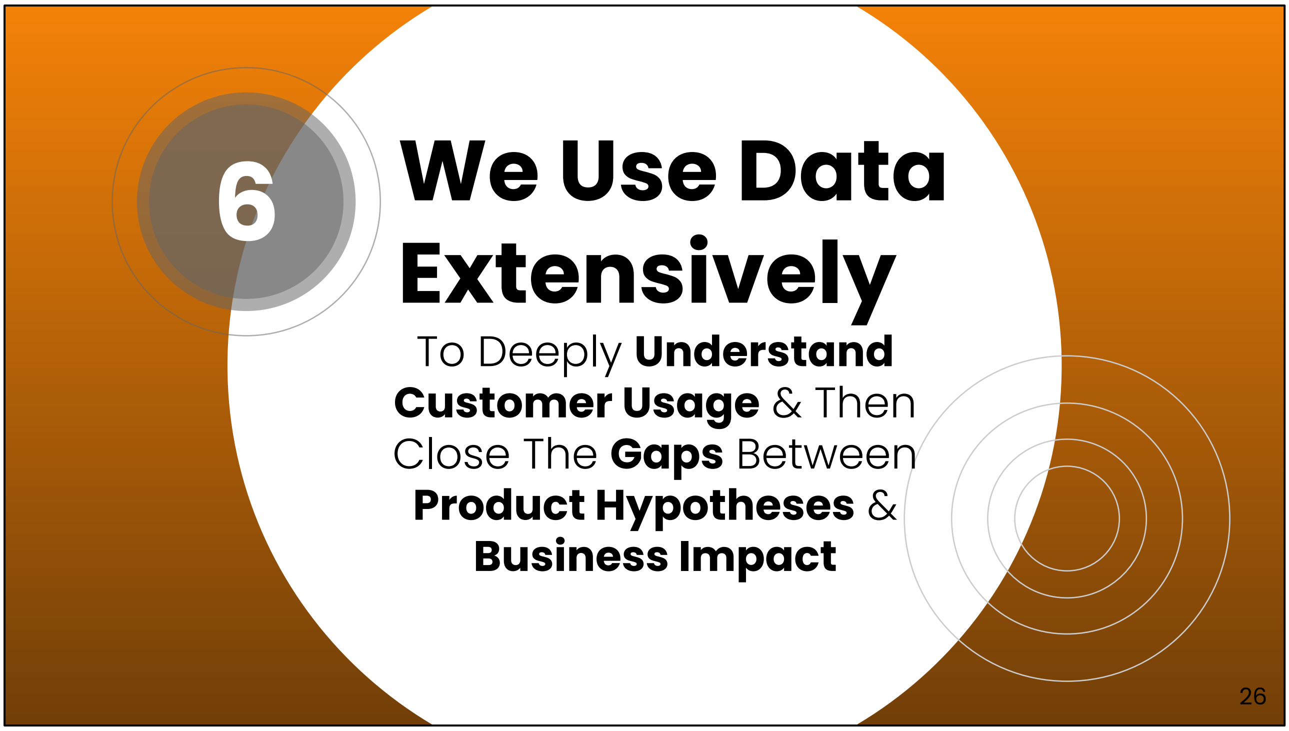


Customers Lead The Way

- Driving decisions with customer data and feedback create an understanding of the system and constraints inherent to that system.

- Customer Advocacy needs to be backed by data, communications, and market trends/research. You can't advocate in a vacuum. Hypotheses need to be grounded in data. The test of a hypothesis isn't the development team, it's customers in production.

- This is why developing a pipeline to delivery is just as important as the product itself



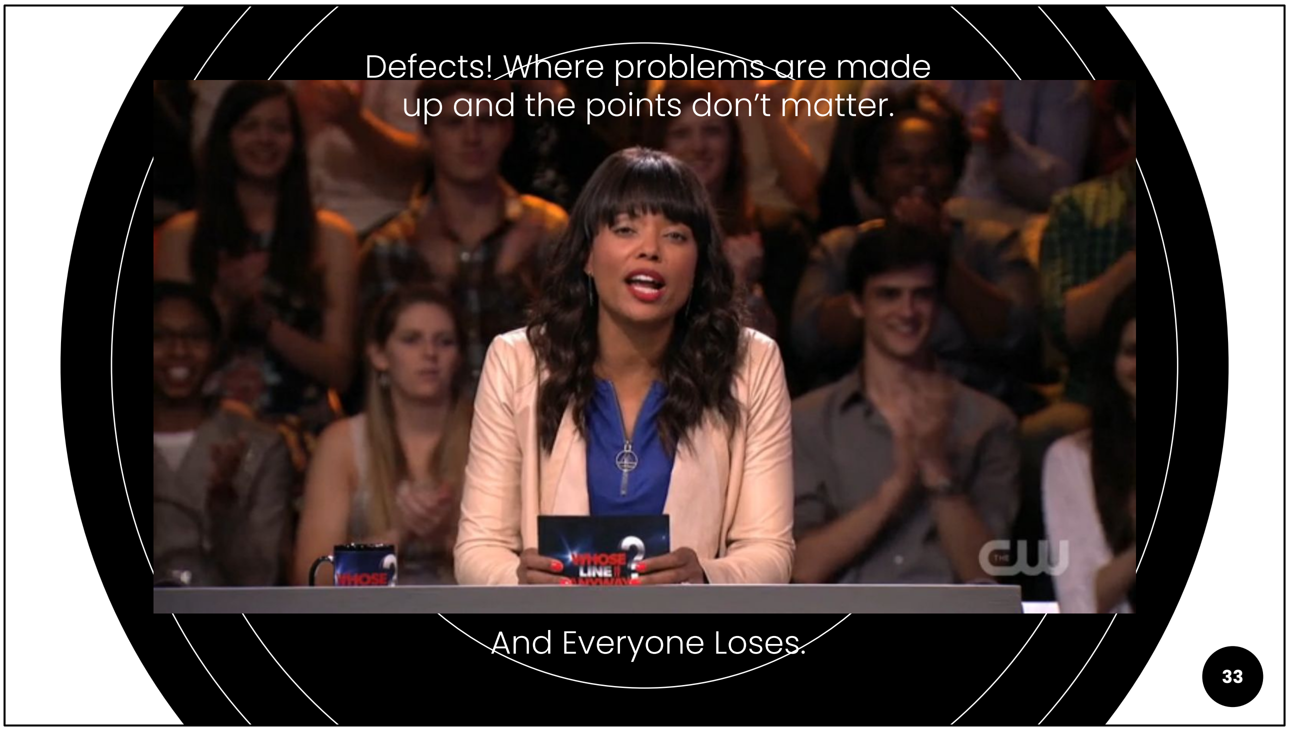
6

We Use Data Extensively

To Deeply **Understand Customer Usage** & Then Close The **Gaps** Between **Product Hypotheses** & **Business Impact**

MTP#6

Get the data that leads to shippable quality the customer can use. No customer is ever going to be happy that you solved 500 defects, if none of those defects actually solved a problem they were facing.



Defects! Where problems are made up and the points don't matter.

And Everyone Loses.

"Defects! Where the points don't matter and everyone loses"

Root cause analysis: If we are focused on only the gross count of defects and not why the defects or where they are coming from, then teams are only dealing with the symptoms of a bigger problem. A systemic problem even.

What if we shifted to finding defects with telemetry and data instead of writing defects. What if we shipped code that might not be perfect, but is good enough – with ways to monitor it and watch what it does when it lands in production.

What if defects were discussed in the team and resolved in a half day of work rather than writing up a whole defect to track and plan around?

All of these options need to have things in place, systems and abilities such as feature flagging, and quick roll backs, or blue-green deployment, or flight rings to affect these things. If you don't have systems like these in place, you can't move fast and get the data you need to move faster. The customer can't give you the feedback if they never see the product or feature you are writing for them.

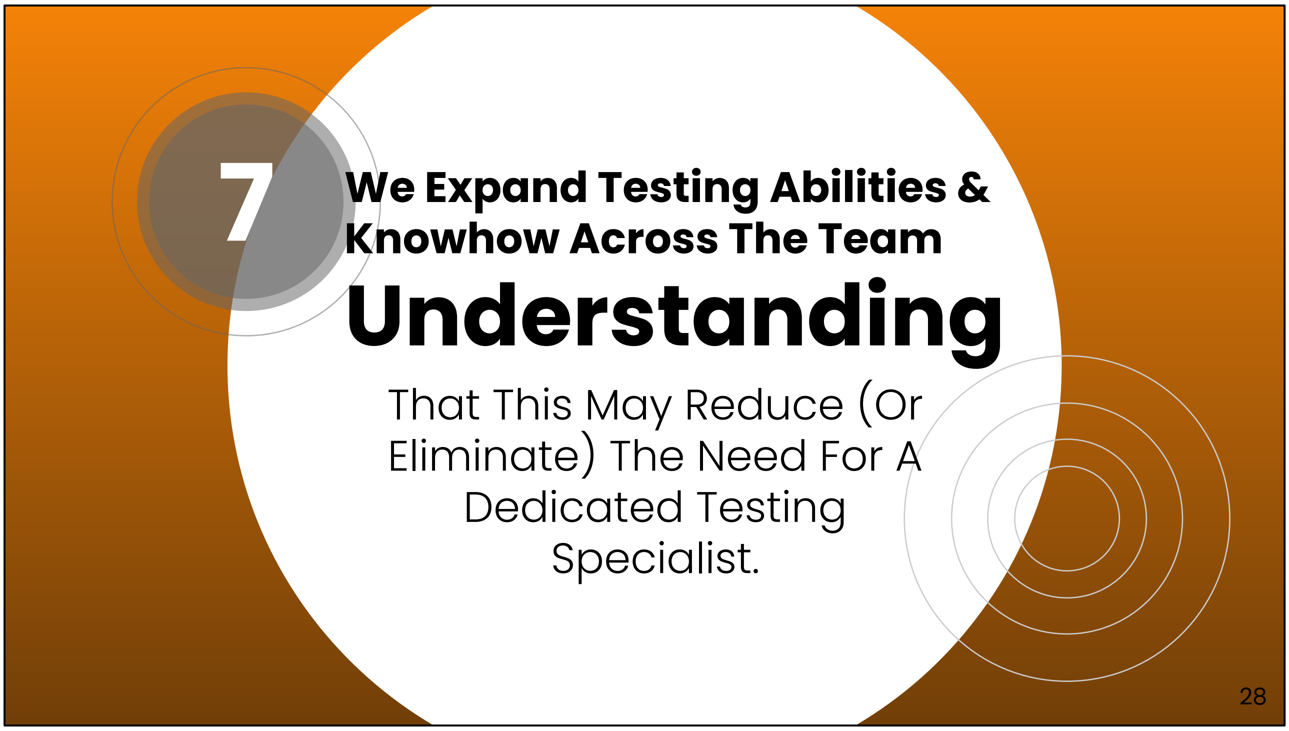
Data collected == team(s) focus

It's about time to solve (DORA) of actual customer problems.
It's about optimizing the feedback loop to be as fast, relizable, and early as possible.

Look for the gap in knowledge.

(Story about changing testing coverage to get faster feedback loops (subcutaneous testing)

https://www.honeycomb.io/



7

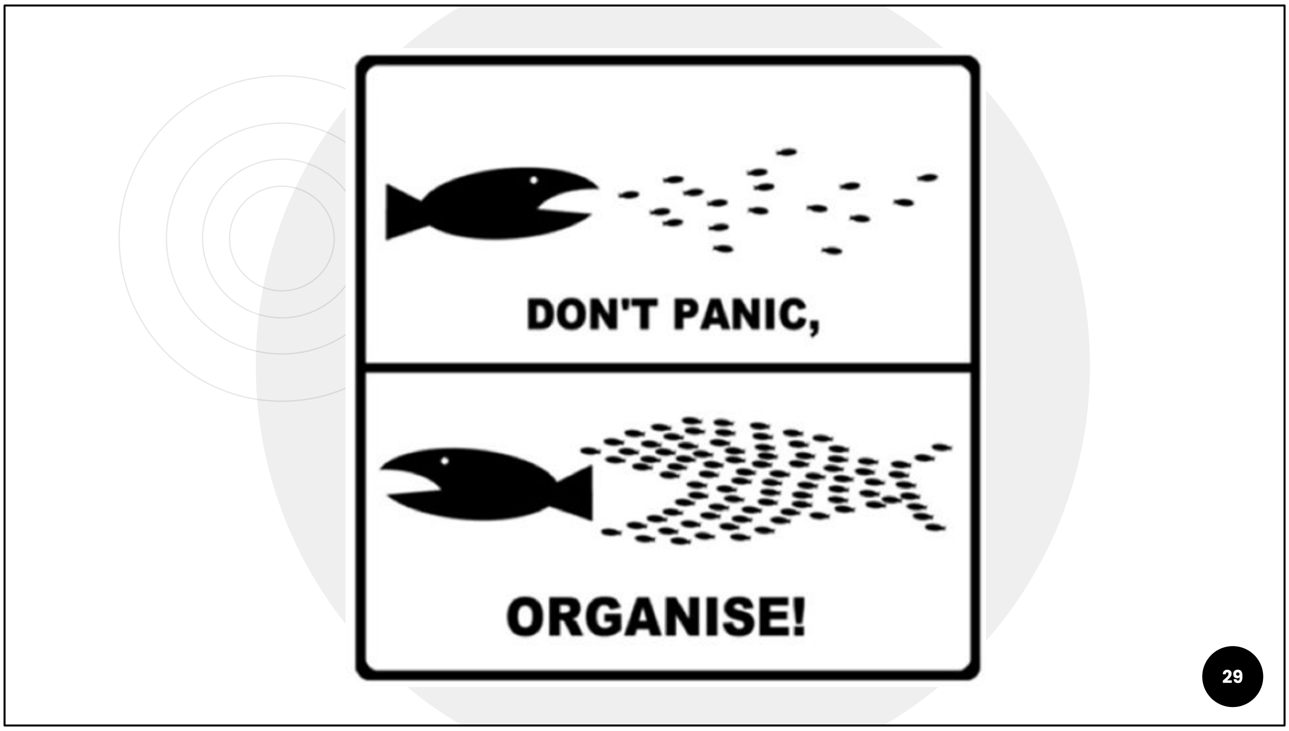
We Expand Testing Abilities & Knowhow Across The Team

Understanding

That This May Reduce (Or Eliminate) The Need For A Dedicated Testing Specialist.

MTP#7
(Description)

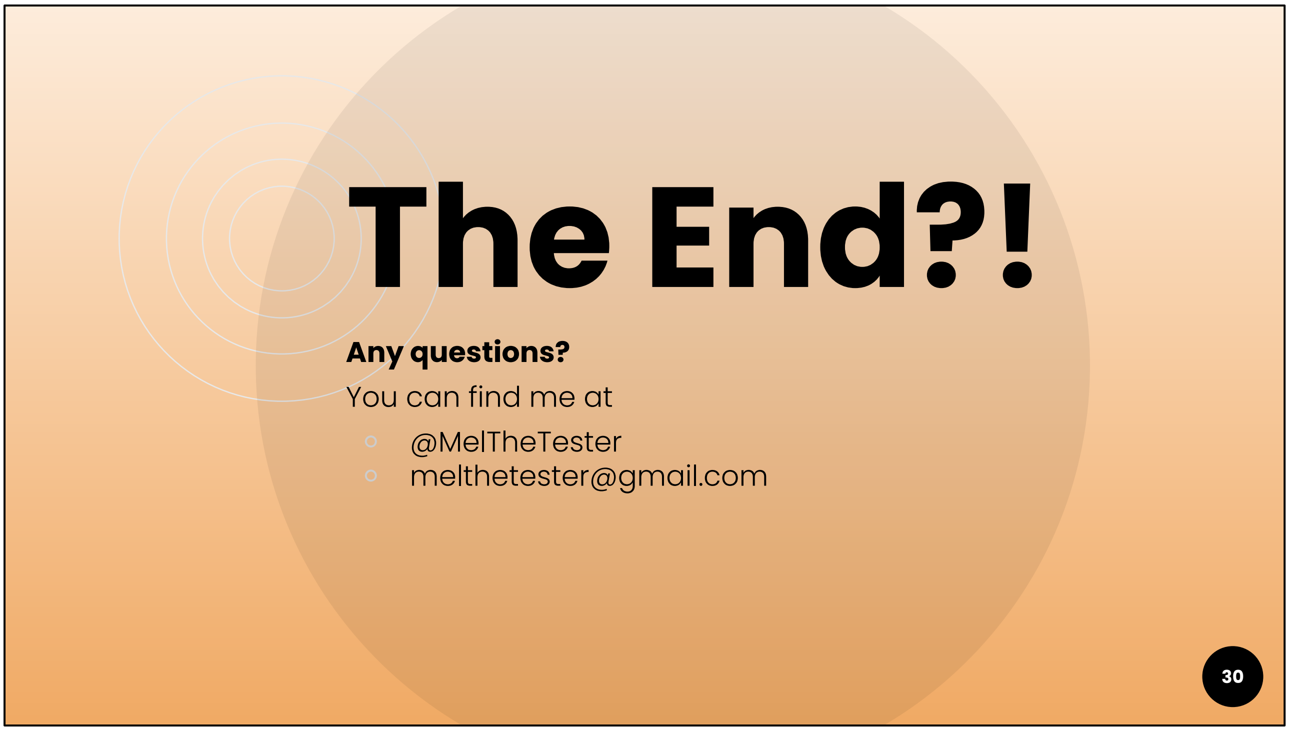
Accountability
Ownership
Whole Team Approach



Dealing with The Traditional Tester Role becoming Anachronistic

- Learn other things.
- Focusing on only functional testing limits the team to one small aspect of quality

Think of this – AI/ML are going to get to the point that those neural networks are going to start writing their own code. Some of that is already happening. These systems aren't going to be functionally wrong. They are going to have other errors, like biases, like incorrect data feeds, data collisions... it's also possible that these systems would interface with other systems and those interfaces would need additional testing. Testing is changing. It's not just websites and UI's – it's voice controls and touch interfaces. We have to think beyond the webpage. Everyone does. It's going to be a world where a Psychology degree is going to be more important than a CS degree. Humanities and Theology will be necessary. We'll need to understand art and sciences, not just code and CPUs.



The End?!

Any questions?

You can find me at

- @MelTheTester
- melthetester@gmail.com

30

Fear Not True Believer

This is not the end, it's only the beginning of the end. Keep learning, Adapting, and Progressing into Quality. You'll find that by doing that, you'll be ahead of any curve technology might throw your way.