IVAR JACOBSON INTERNATIONAL

Creating **winning** teams.

Creating a Culture of Collaboration & Understanding:

Applying SAFe DevOps Principles & Practices

Keith de Mendonca

London Meetup 20th Oct 2019.

Hosted by - **WorldPay**

www.ivarjacobson.com

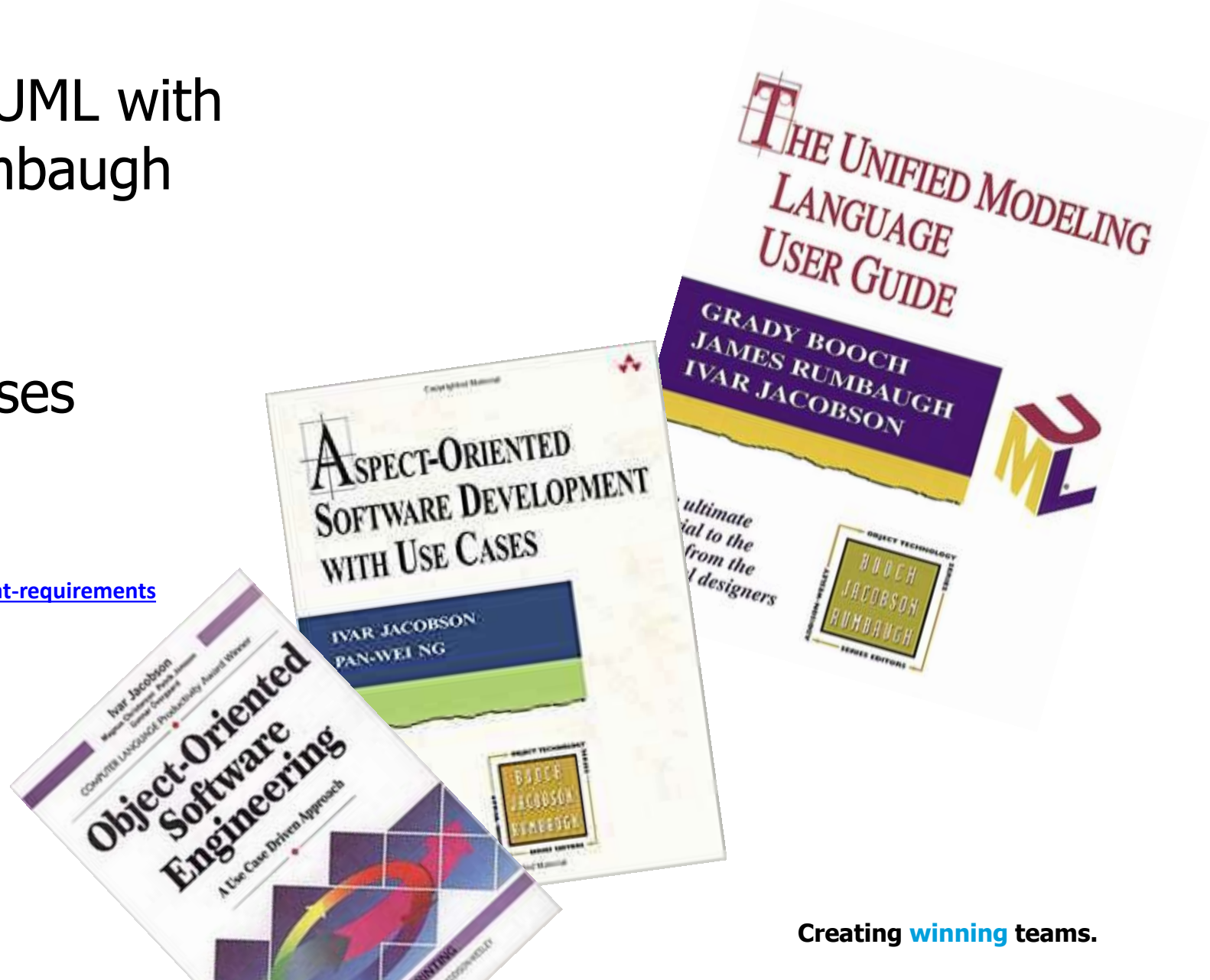
Contents

- Introduction to DevOps (in SAFe)
- The SAFe DevOps Health Radar
- DevOps and Culture

About Ivar Jacobson…

- Ivar Jacobson developed UML with Grady Booch and Jim Rumbaugh
  - The three amigos!
- Ivar also invented Use-Cases
  - Use-Cases 2.0 can be used by Agile teams

https://www.ivarjacobson.com/services/agile-development-requirements



About us… SAFe Gold Partner

- Our company helps large organisations to achieve greater software agility and so deliver better business results.
- We have helped:
  - Ericsson
  - Deutsche Bank
  - SimCorp
  - Fujitsu
  - Nordea Bank
  - UK National Health Service
  - Dutch National Railways
  - *and many, many more!*

https://www.scaledagileframework.com/case-studies

About me

- Principal Consultant at Ivar Jacobson International Ltd.

- 20+ years in software technical roles.
- Lived and worked in UK, India, China

- Part of a team who implemented feature toggling/single codeline management in 1000 person software organisation (2004).

About you

Why did you decide to come to this Meetup?

What is your experience of DevOps ?

Have you attended the SAFe DevOps course ?

Development + **Op**erations = **DevOps**

"Imagine a world where product owners, development, QA, IT operations, and InfoSec engineers work together,

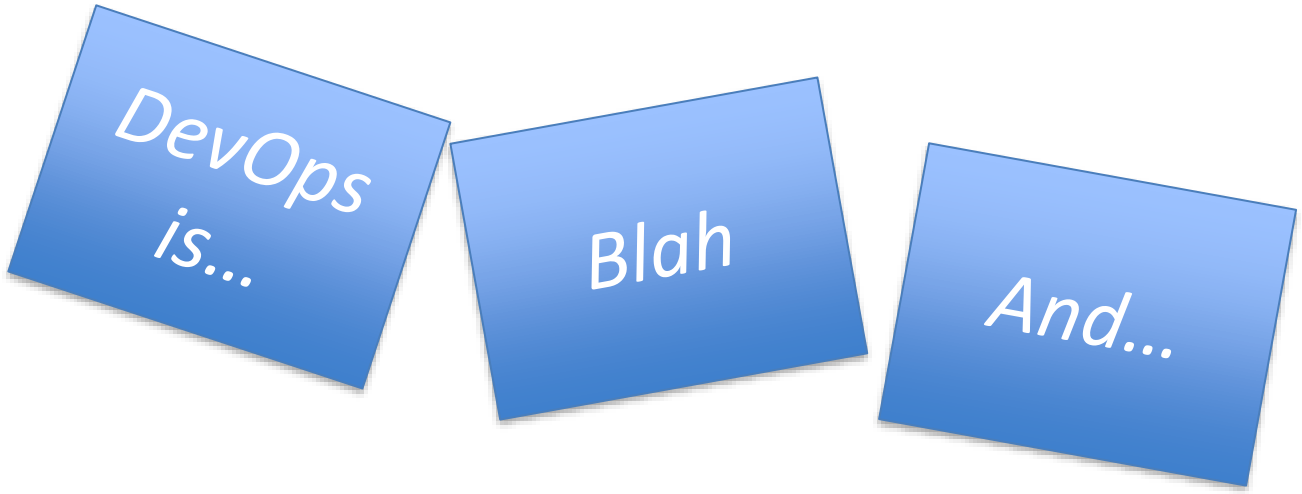
…not only to help each other, but also to ensure that the overall organisation succeeds.

By working toward a common goal, they enable the fast flow of planned work into production … while achieving world class stability, reliability, availability, and security"

From "The DevOps Handbook"
(Kim, Humble, Debois, Willis; 2016)

What does DevOps mean to you?

- Write down what you think is **one** key word or **one** short phrase that best describe "DevOps"
- Please write it on one Post-It note.



Where "Ops" = systems engineers, system administrators, operations staff, release engineers, DBAs, network engineers, security professionals,…

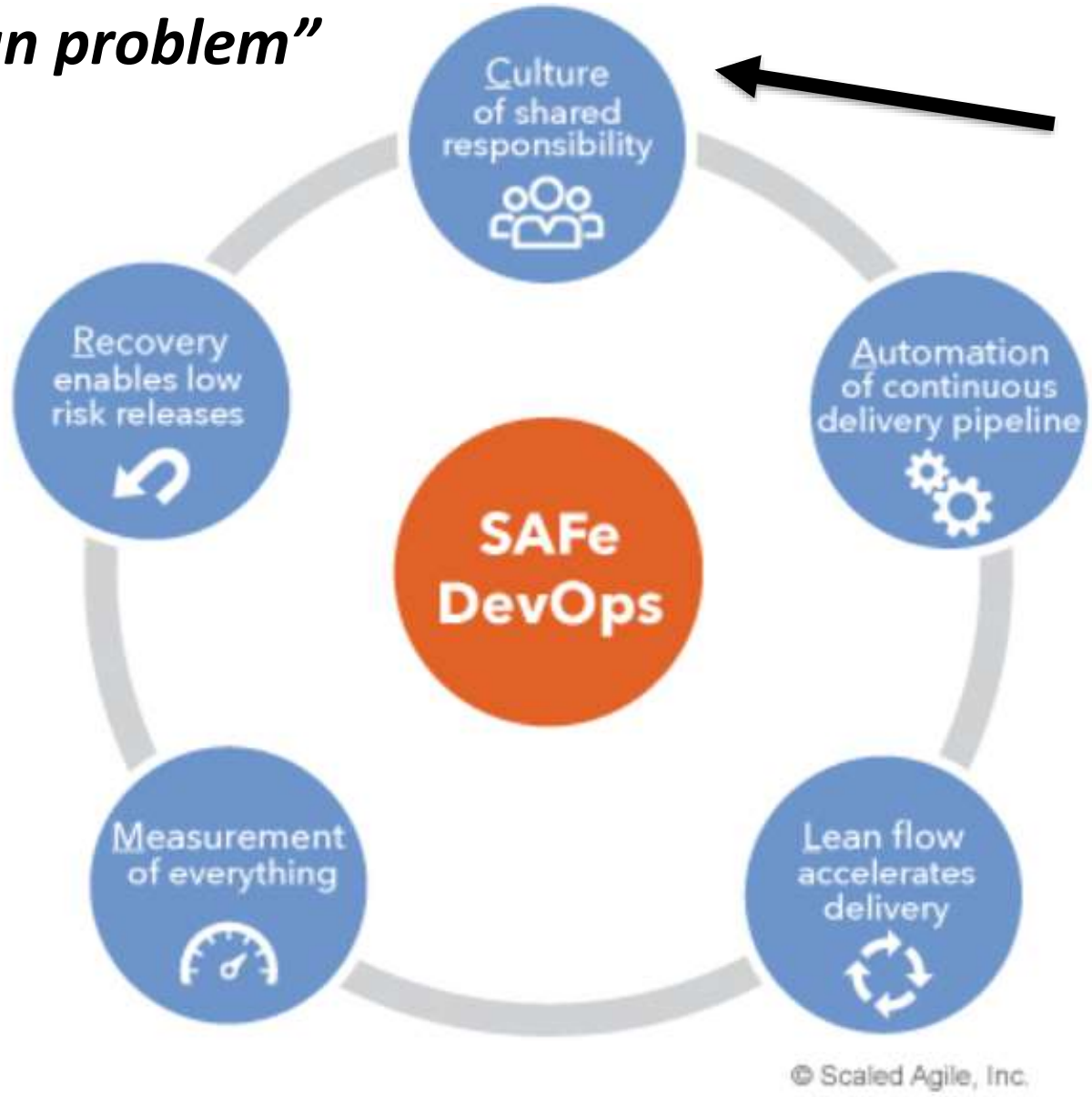
https://theagileadmin.com/what-is-devops/

The SAFe CALMR Approach to DevOps

CAMS… 2012
CALMS.. 2015
CALMR… 2018

A Culture of Shared Responsibility

"DevOps is a human problem"
– Patrick DeBois



As technical people we often focus on Tools and Automation in DevOps…

But we wont be successful unless we change the way we actually work, and change the way we all work together…

We must change our organisation's culture.

Automation of continuous delivery pipeline

Automate build, test, integration, deployment and release activities.

Accelerate build, integration & testing to generate faster feedback.
Automate deployment and release to reduce human errors.
Automate our ability to re-generate our pipeline machinery if it fails.

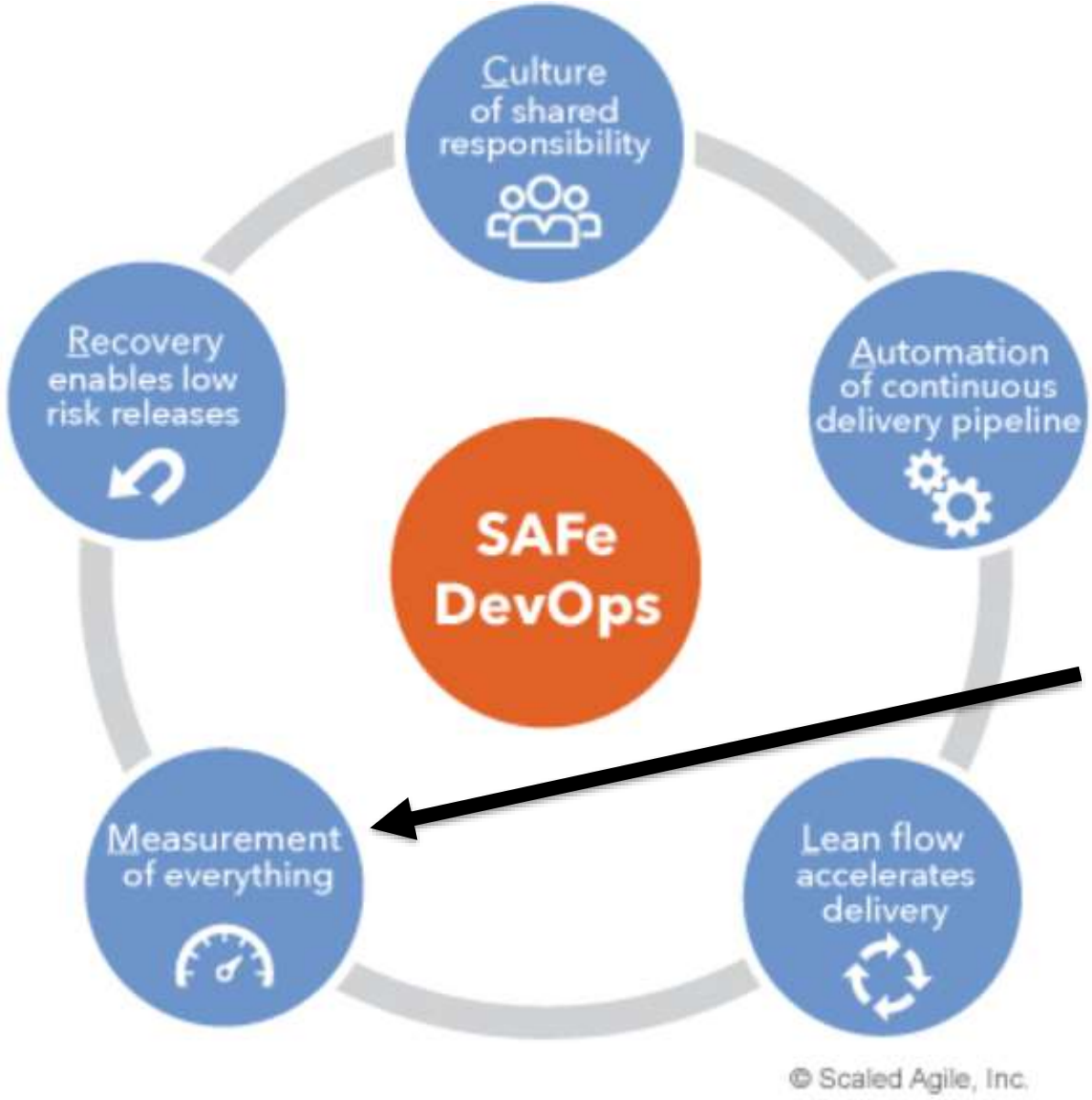
Lean Flow accelerates delivery

Make small, frequent continuous incremental changes to the system; avoid stop-start delays and waste.

Avoid “big bang” integrations with all the stress and delays that go with them.

Always have the most up-to-date software available for business to release when they choose to do so.

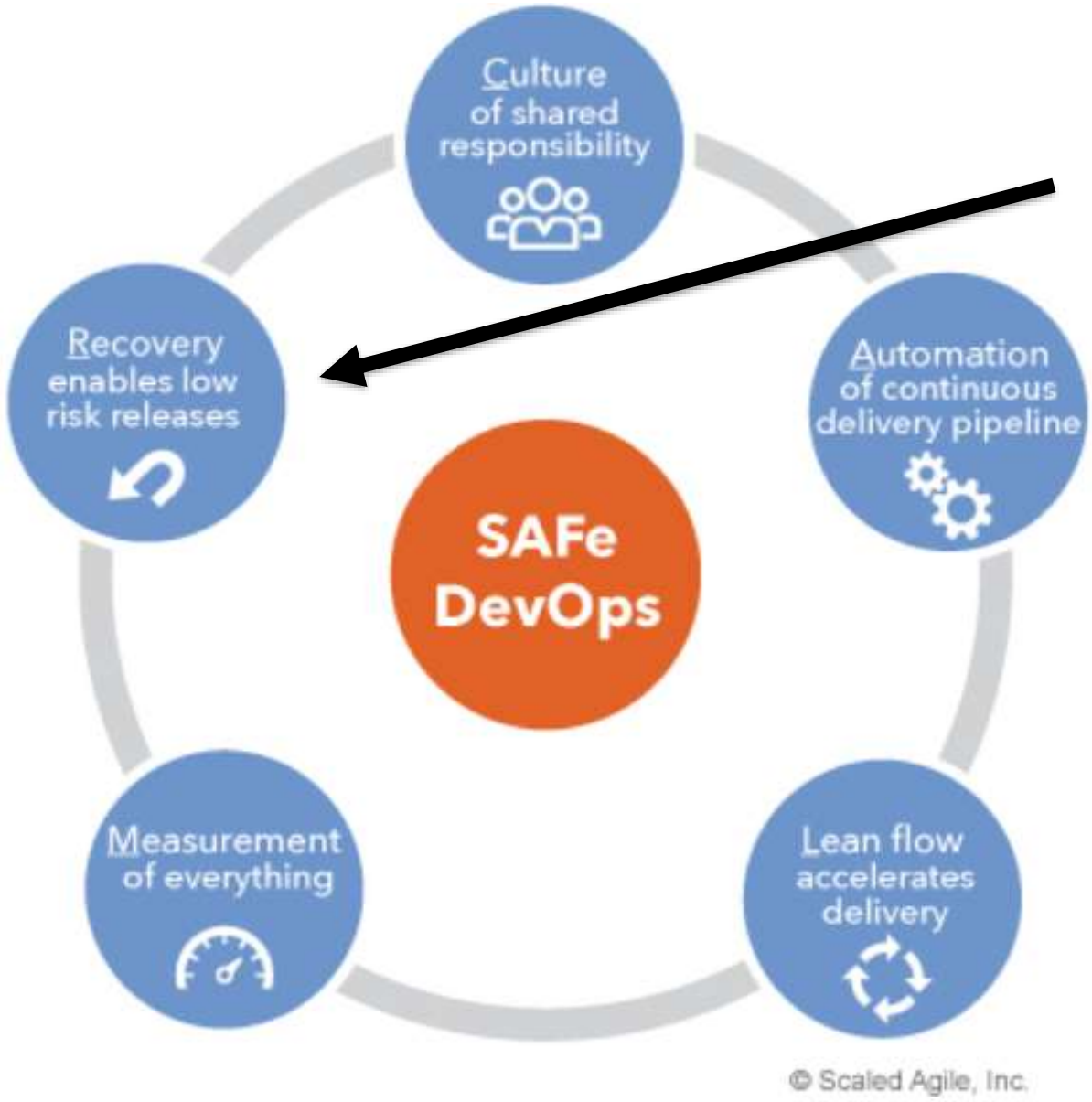
Measurement of everything



Let's be scientific…

Measure the quality of software, the speed it moves through each stage of the pipeline, the status of the live system and the business results we are achieving.
Data will help us pro-actively catch issues quickly and will direct us to the next improvement that will further accelerate delivery.

Recovery enables low risk releases



As we accelerate delivery of new business value by x10 or x50 or x440 we will inevitably make mistakes from time to time.

Automate our ability to revert back to a good known state or to repair quickly and safely with little additional risk, so we can be confident of making rapid deliveries.

Mean Time To Recover (MTTR)

The loooong journey to DevOps and beyond

https://goo.gl/images/JaUCpe

- "Agile" - *faster delivery of "just enough functionality with fast customer feedback"* 1990s
- "DevOps" – *"Oops the Ops guys see all this rapid change as more risk to our money-making live system"* 2009
- "DevSecOps" – *"Oops the security guys/regulators are scared of all this rapid change"* 2015
- "DevArchSecHRFinBizCusOps" ?! ☺ 2099

The speed which can result from Agile practices applies pressure to our elicitation, development, testing, delivery and support practices.

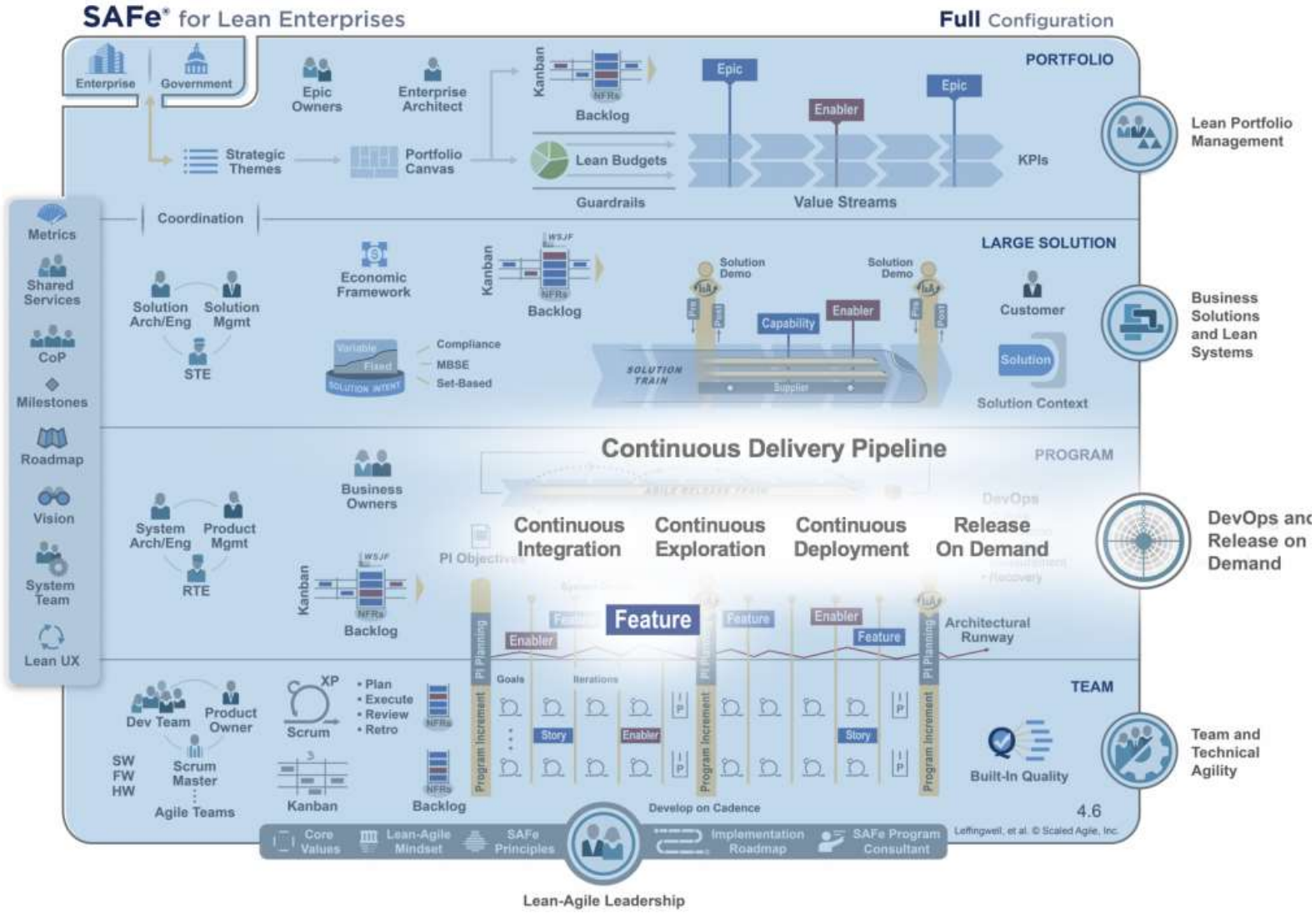
We can't ignore these issues if we want to deliver high quality software at ever-increasing rates.

SAFe v4.6

Scaled Agile Framework (SAFe) is a knowledge base of proven, integrated principles and practices for Lean, Agile, and DevOps.

SAFe is the world's leading framework for enterprise agility. It is updated regularly.

There are other agile scaling methods: e.g. LeSS, Nexus, Scaled Scrum, Disciplines Agile…



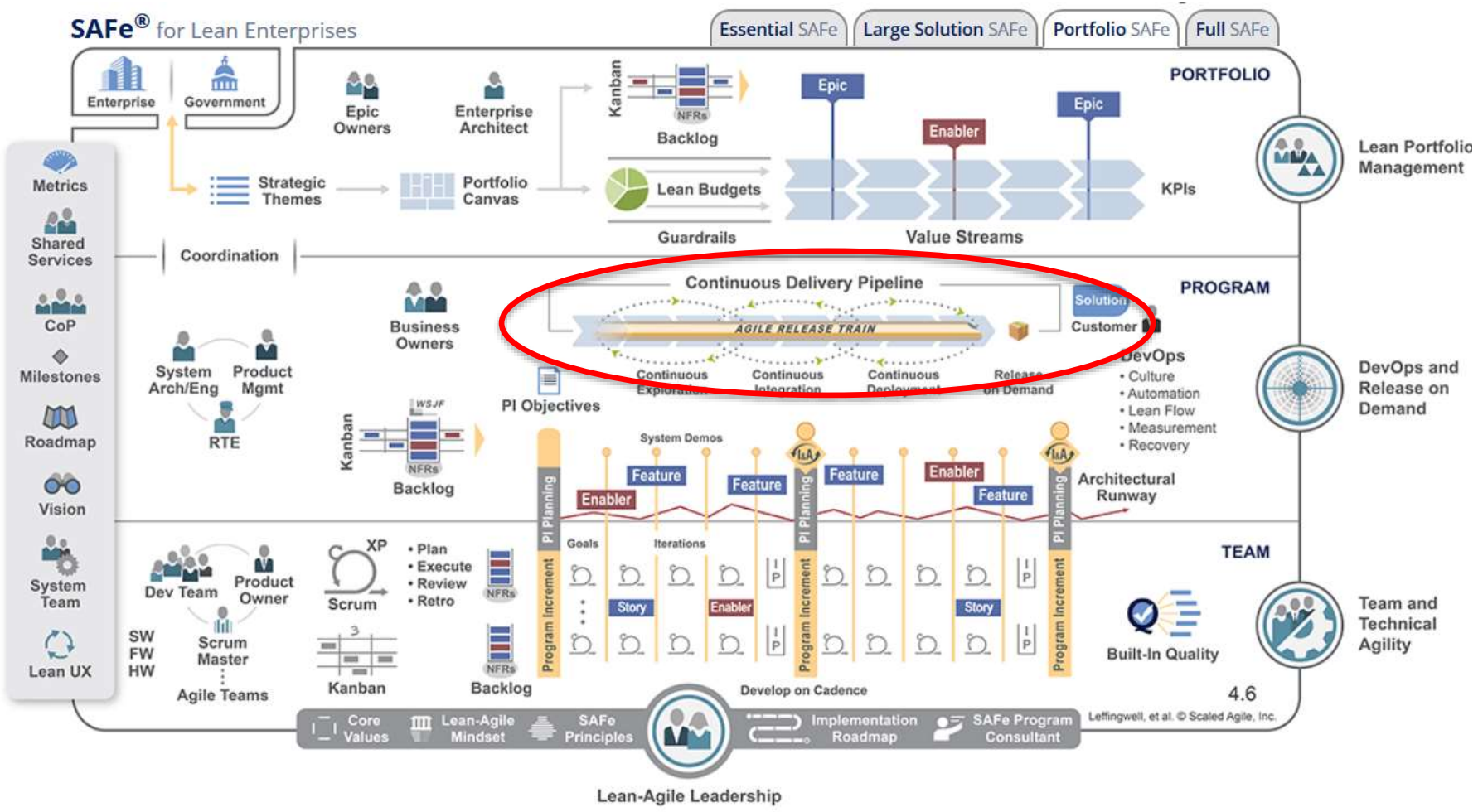
IVAR JACOBSON INTERNATIONAL

Creating winning teams.

SAFe v4.6

DevOps is everybody's responsibility on each SAFe Agile Release Train.

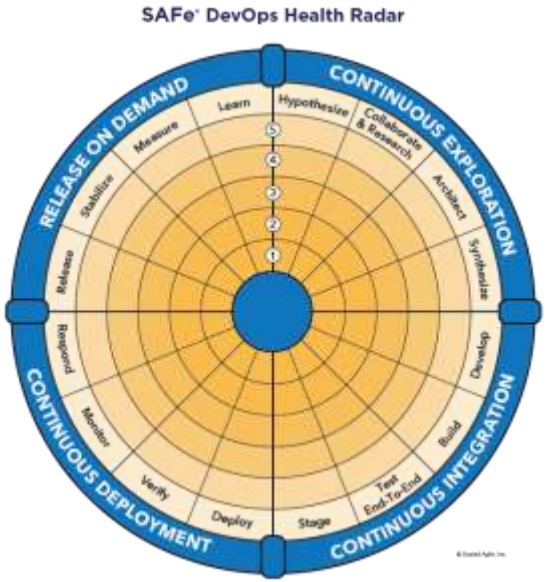
It is not the job of a single team.
We need to get better collaboration happening between our existing teams – aligned to delivering more value quickly and safely to our customers



IVAR JACOBSON INTERNATIONAL

Creating winning teams.

The SAFe DevOps Health Radar

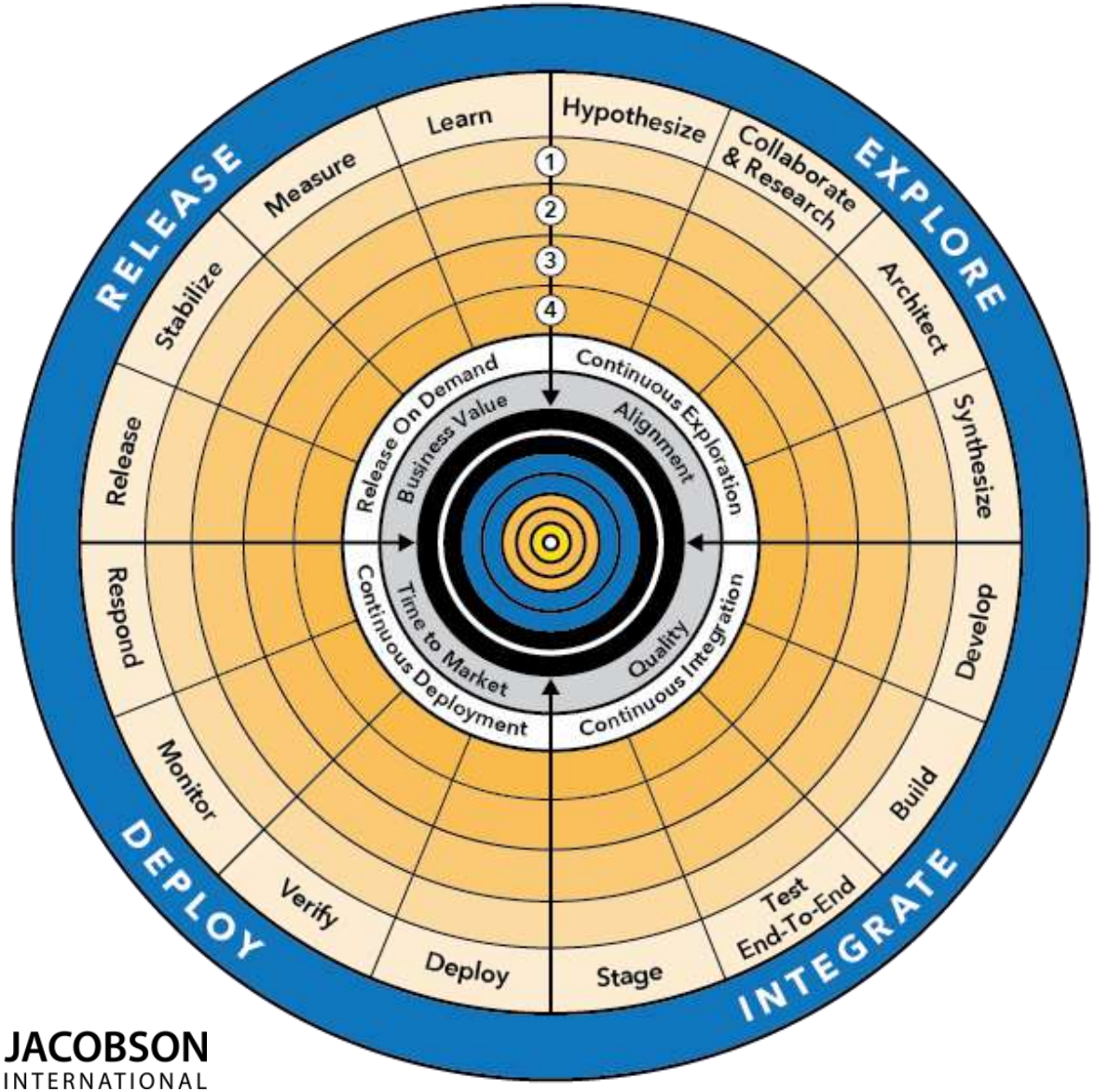


- SAFe created a DevOps health radar in 2018 for organisations to assess their current capability against specific phases in the software development lifecycle.

- There are two free versions available:

1. An excel version of the radar can be downloaded from the "Metrics" section of https://scaledagileframework.com/metrics
2. An online assessment is available at: https://agilityhealthradar.com/safe-devops-assessment/

The SAFe DevOps Health Radar



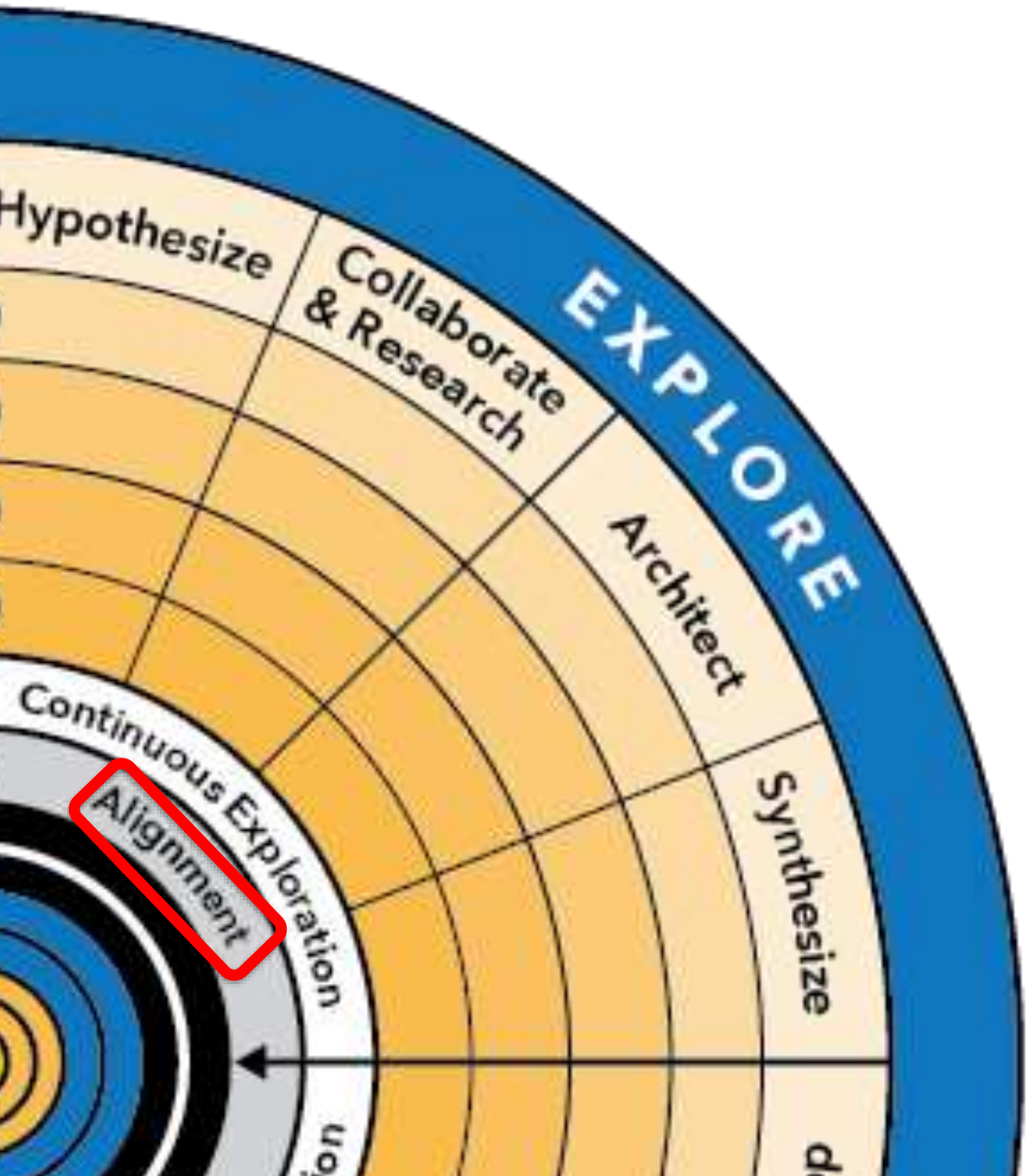
We want to create a steady "Lean" flow of valuable new requirements flowing through our *explore-build-test-deploy-release* cycle as quickly and as sustainably as possible.

i.e. We would like to make a complete 360 degree cycle around the DevOps health radar as quickly as possible.

Amazon report they can do this in 24 hours!

IVAR JACOBSON INTERNATIONAL

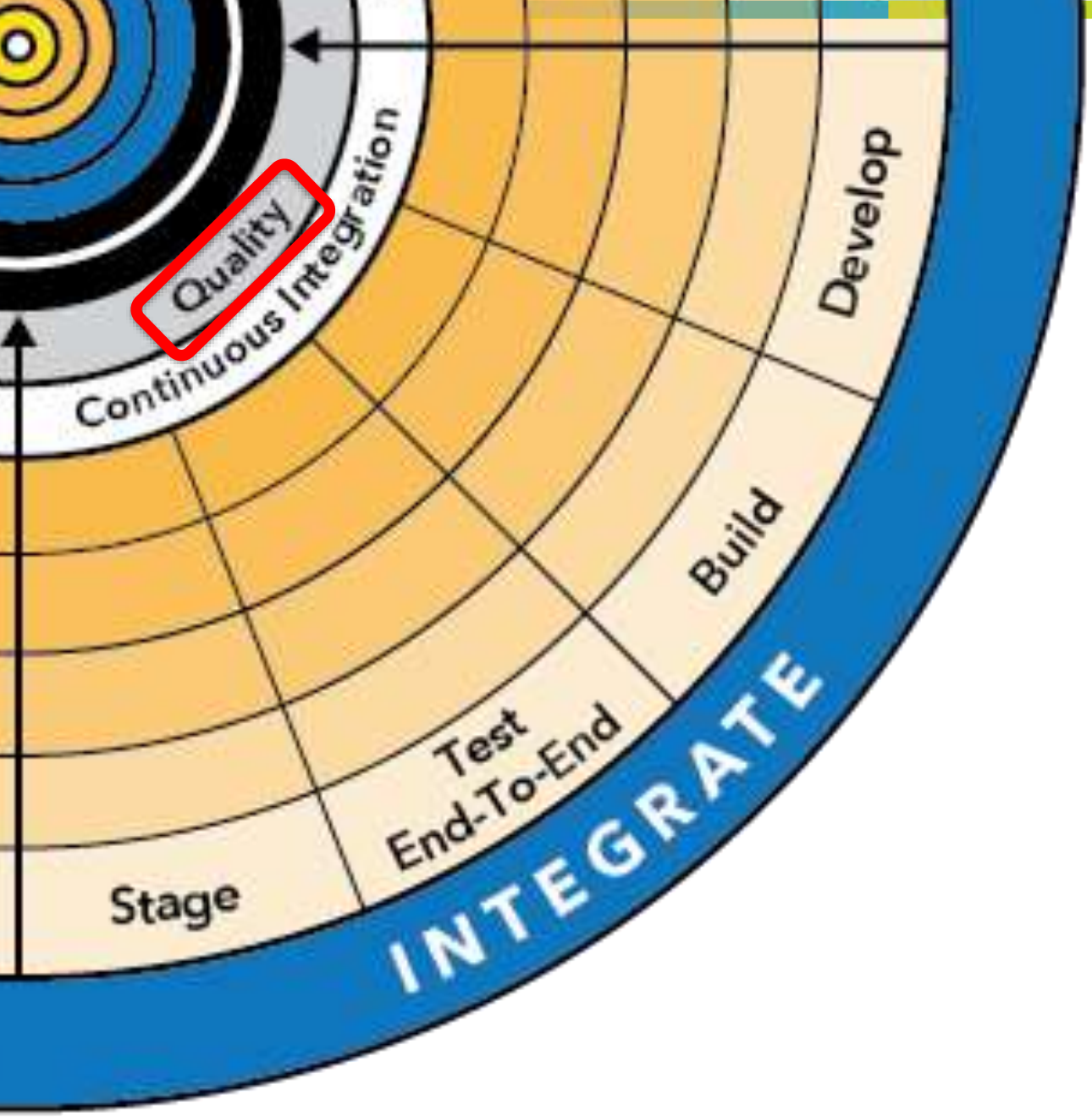
Creating winning teams.



1. The *Explore* quadrant

Why? We want to ensure close **Alignment** with our customer's needs with **continuous exploration** of the market and possible solutions.

Treat each requirement as a **Hypothesis** to test. **Collaborate** with customers and stakeholders; conduct **Research**.
Architect a suitable technical solution with the right non-functional requirements (e.g. security).
Synthesize all of these insights into a prioritised list of Features



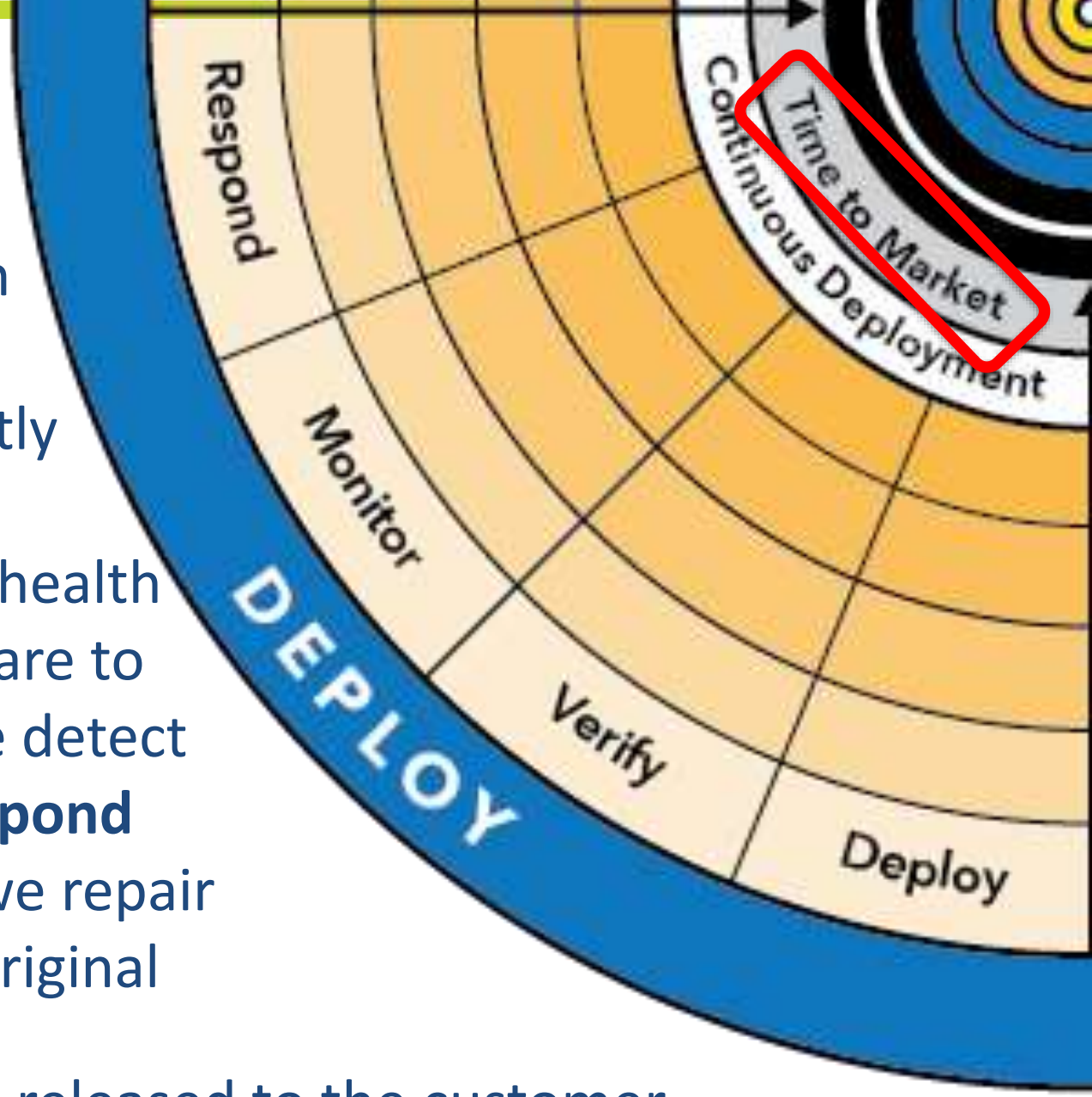
2. The *Integrate* Quadrant

Why? We want to create high **Quality** software with **Continuous Integration** of small incremental changes to our software system.

We capture design inputs from product owner, **Develop**ers, test specialists and technical SMEs so we can **Build** small increments of working software. We frequently integrate all our code together and run **End-to-End tests** to confirm our design assumptions. We **Stage** the software in a production-like environment.

3. The *Deploy* quadrant

Why? We want **Continuous Deployment** of our working Software System so we can significantly reduce our **Time To Market**.
We **Deploy** our software changes frequently and **Verify** its quality in a production-like environment. We constantly **Monitor** the health of the deployment pipeline and the software to give us high confidence of its quality. If we detect any issues in the deployed version we **Respond** quickly to revert to an earlier version, or we repair and redeploy the problem quickly in the original codeline.

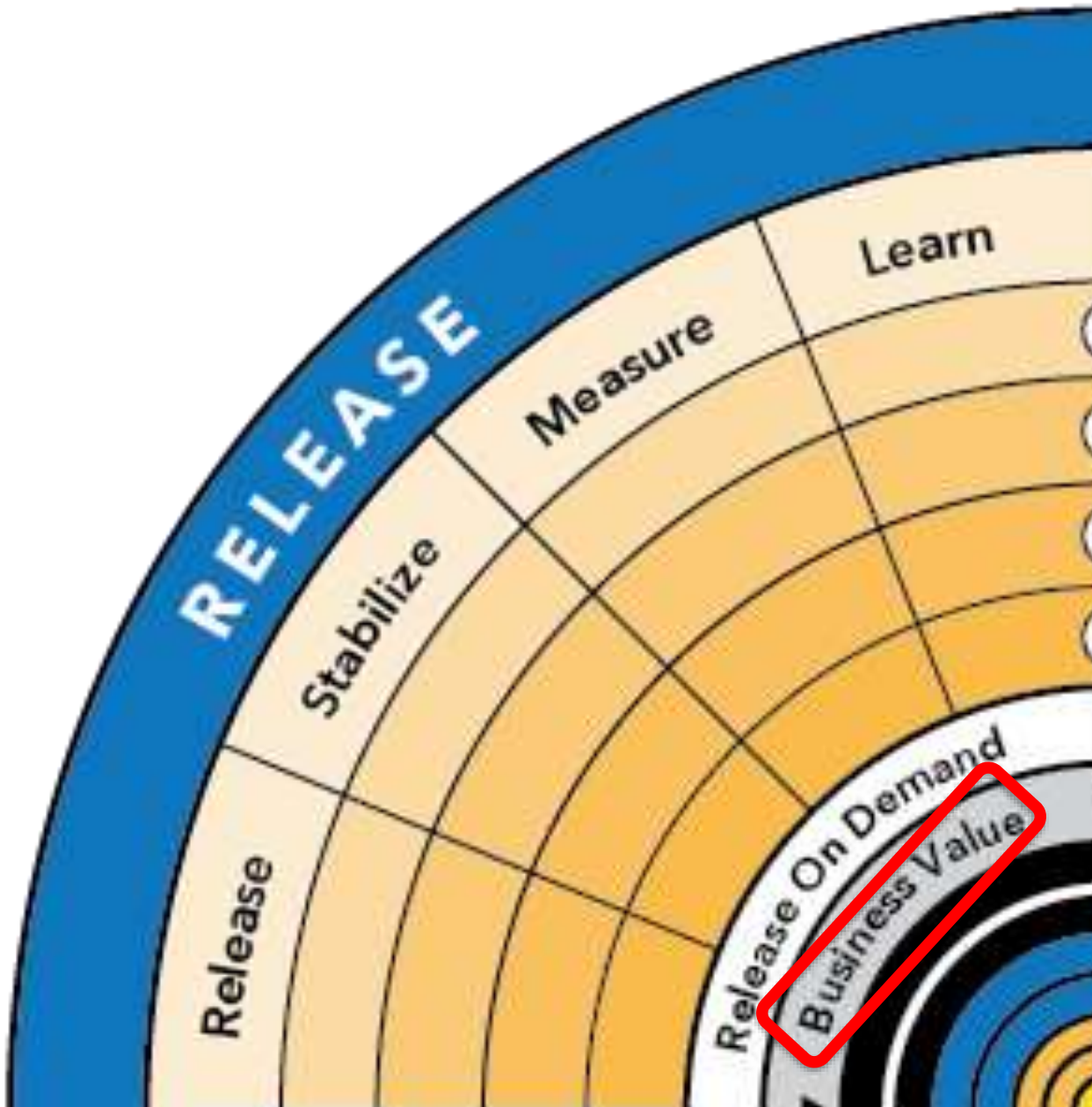


NB Deployed software is not yet released to the customer.

4. The *Release* On-Demand quadrant

Why? We want to be able to **Release on Demand** whenever our customer or business needs it, so we can deliver higher levels of **Business Value** to our customers.

After each **Release** we confirm that our change has further **Stabilize**d the system. We **Measure** the system health and business impact of the change – we use this feedback to **Learn** how to deliver even better value in the future.



IVAR JACOBSON INTERNATIONAL

Score each dimension as… sit–crawl-walk-run–fly!

Dimension	Sub-dimension	Dimension / Question
Continuous Exploration	Hypothesize	Hypothesizing entails expressing a business idea (epic) in terms of the business value it is expected to deliver. This hypothesis is then implemented as a Minimum Viable Product (MVP) through the Continuous Delivery Pipeline and evaluated for accuracy when released. Please rate your team's ability to translate business ideas into clear and measurable epic hypothesis statements. Sit (1-2): Ideas are vague or not defined. Crawl (3-4): Ideas are defined (e.g., as epics) but do not include hypothesis statements. Walk (5-6): Some ideas are expressed as hypothesis statements with measurable outcomes. Run (7-8): Most ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs. Fly (9-10): All ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs.
		Collaborate and Research involves Product Managers working directly with end-users, stakeholders and subject matter experts to understand customers' needs and identify specific business outcomes and associated metrics to guide solution delivery. Please rate your team's ability to collaborate with customer-side and IT-side experts to define Minimum Marketable Features (MMF) in support of the hypothesis. Sit (1-2): Product Management roles and responsibilities are not defined or followed. Crawl (3-4): Product Management creates requirements in large batches with little customer or development collaboration. Walk (5-6): Product Management collaborates with business-side or development-side experts, but not both, when defining requirements.

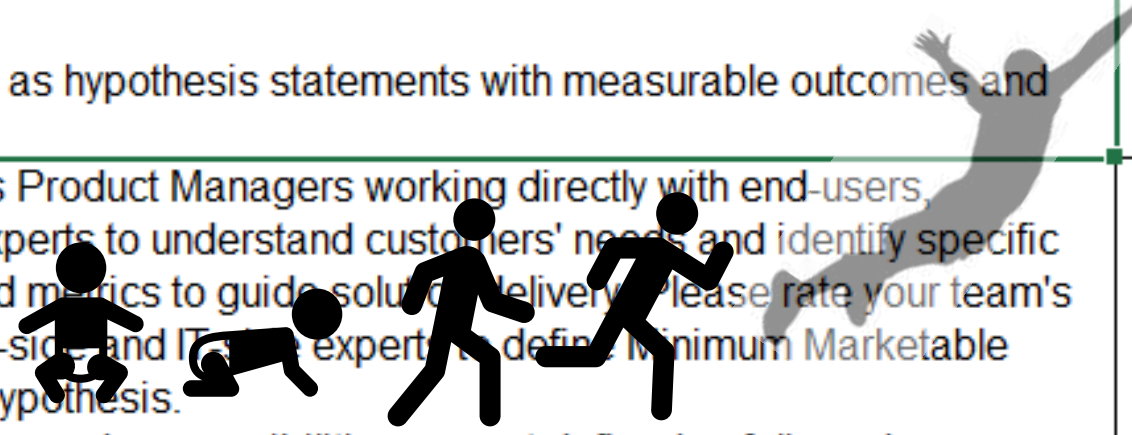
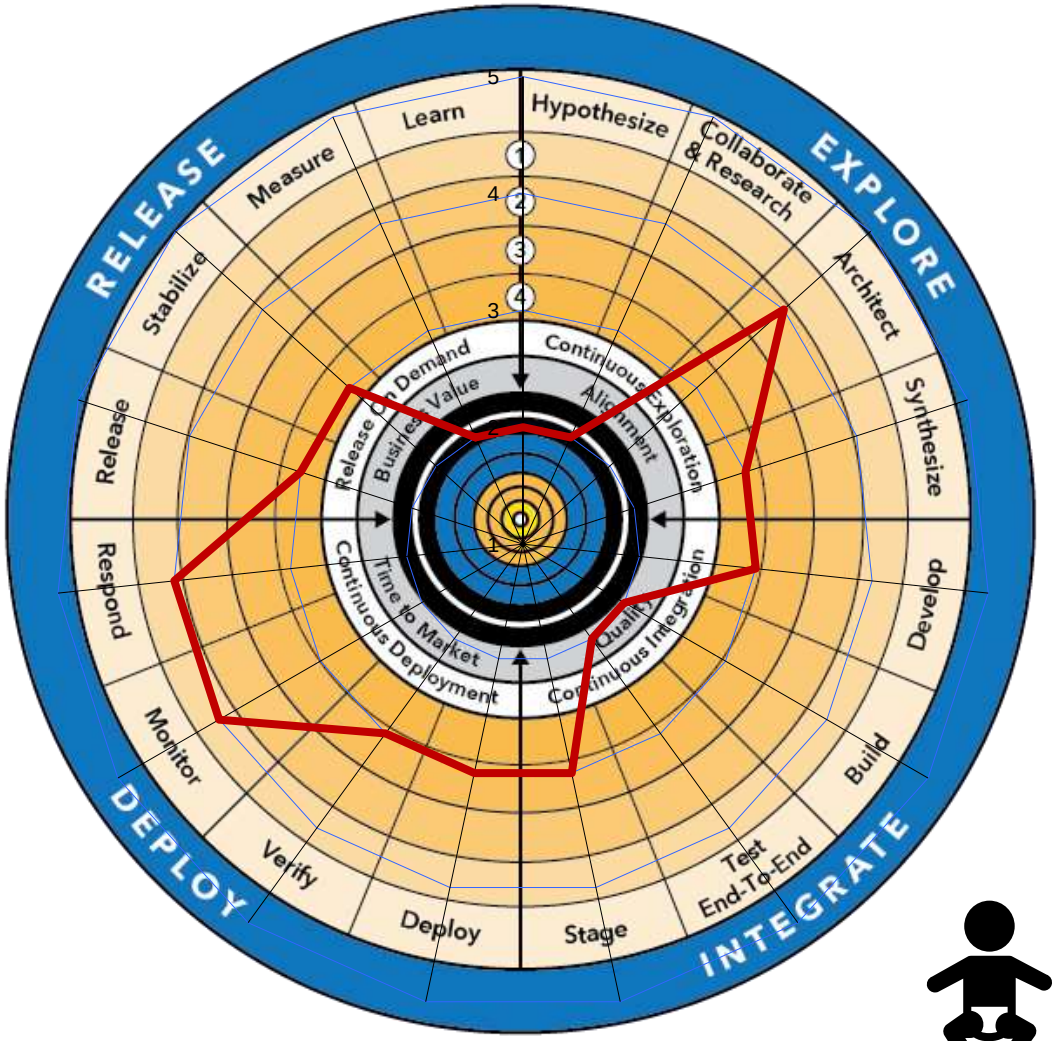
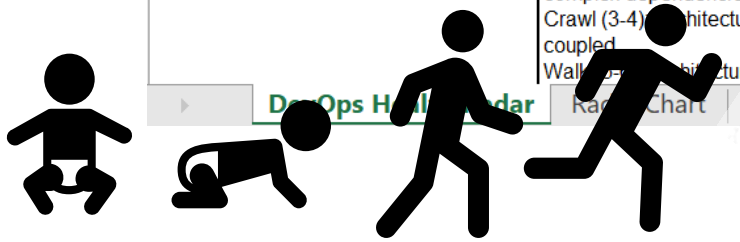


Chart the results on the Radar



Dimension	Sub-dimension	Dimension / Question	Score	Rating
Continuous Exploration	Hypothesize	Hypothesizing entails expressing a business idea (epic) in terms of the business value it is expected to deliver. This hypothesis is then implemented as a Minimum Viable Product (MVP) through the Continuous Delivery Pipeline and evaluated for accuracy when released. Please rate your team's ability to translate business ideas into clear and measurable epic hypothesis statements. Sit (1-2): Ideas are vague or not defined. Crawl (3-4): Ideas are defined (e.g., as epics) but do not include hypothesis statements. Walk (5-6): Some ideas are expressed as hypothesis statements with measurable outcomes. Run (7-8): Most ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs. Fly (9-10): All ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs.	3.0	2
Continuous Exploration	Collaborate & Research	Collaborate and Research involves Product Managers working directly with end-users, stakeholders and subject matter experts to understand customers' needs and identify specific business outcomes and associated metrics to guide solution delivery. Please rate your team's ability to collaborate with customer-side and IT-side experts to define Minimum Marketable Features (MMF) in support of the hypothesis. Sit (1-2): Product Management roles and responsibilities are not defined or followed. Crawl (3-4): Product Management creates requirements in large batches with little customer or development collaboration. Walk (5-6): Product Management collaborates with business-side or development-side experts, but not both, when defining requirements. Run (7-8): Product Management regularly collaborates with business-side, development-side, and operation-side experts but does not define Minimum Marketable Features. Fly (9-10): Product Management always collaborates with business-side, development-side, and operation-side experts and defines Minimum Marketable Features.	3.0	2
		Architecting for continuous delivery involves applying ""just enough"" intentional architecture to assure policy compliance without sacrificing product development flow, to ensure solutions are loosely coupled and to continuously pay down technical debt. Please rate your team's effectiveness at architecting for continuous delivery. Sit (1-2): Architecture is monolithic and fragile; it is difficult to change and involves managing complex dependencies across many components and systems. Crawl (3-4): [illegible]hitecture is predominantly monolithic but some applications/systems are loosely coupled. Wal[illegible]tures is most decoupled but doesn't allow Release on Demand.		

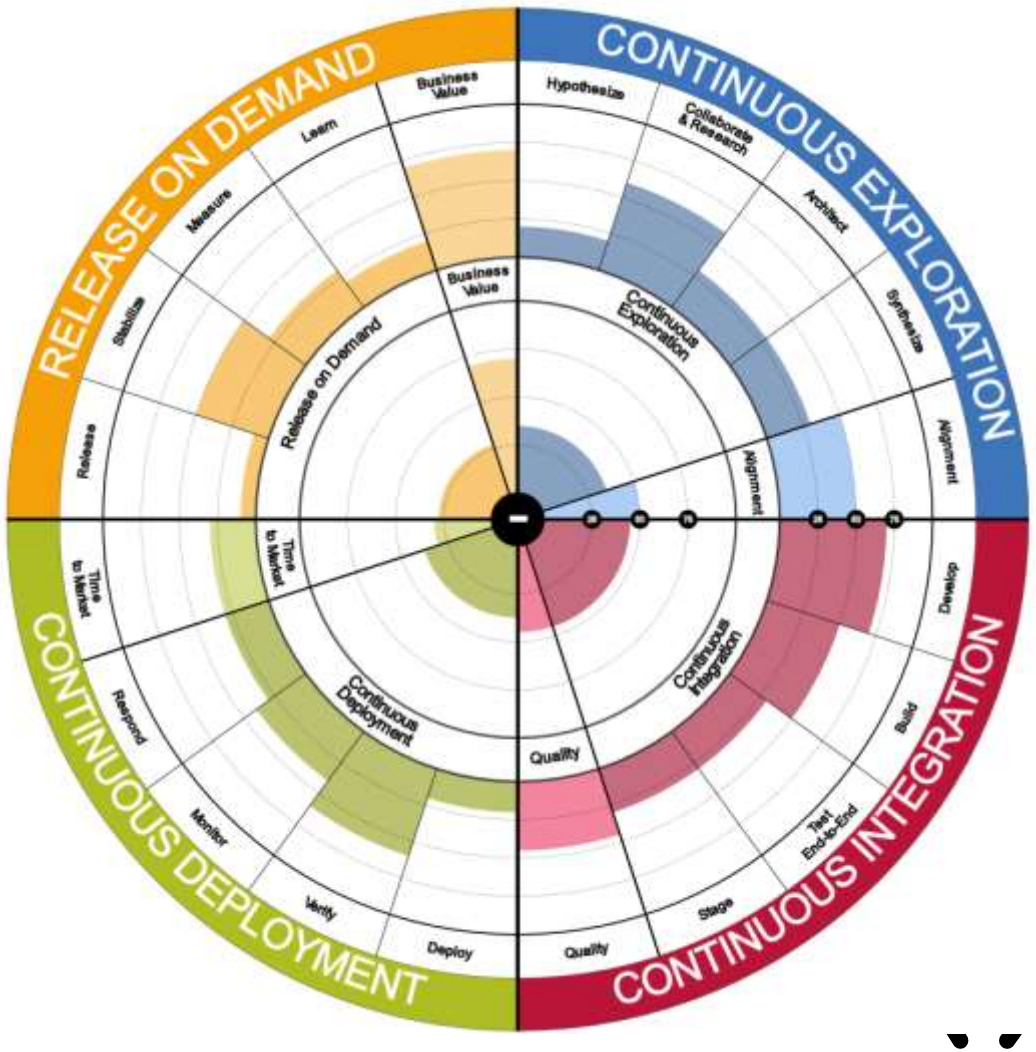


Scoring:
Sit (1-2), Crawl (3-4), Walk (5-6), Run (7-8), Fly (9-10)

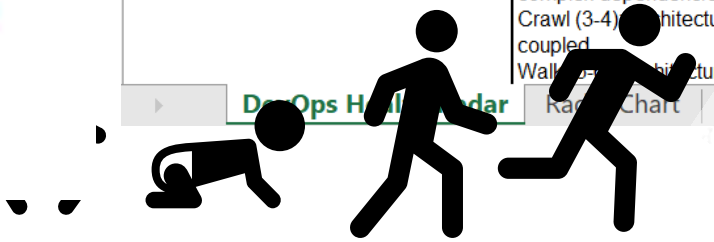
IVAR JACOBSON INTERNATIONAL

Creating winning teams.

Chart the results on the Radar (online version)



Dimension	Sub-dimension	Dimension / Question	Score	Rating
Continuous Exploration	Hypothesize	Hypothesizing entails expressing a business idea (epic) in terms of the business value it is expected to deliver. This hypothesis is then implemented as a Minimum Viable Product (MVP) through the Continuous Delivery Pipeline and evaluated for accuracy when released. Please rate your team's ability to translate business ideas into clear and measurable epic hypothesis statements. Sit (1-2): Ideas are vague or not defined. Crawl (3-4): Ideas are defined (e.g., as epics) but do not include hypothesis statements. Walk (5-6): Some ideas are expressed as hypothesis statements with measurable outcomes. Run (7-8): Most ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs. Fly (9-10): All ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs.	3.0	2
Continuous Exploration	Collaborate & Research	Collaborate and Research involves Product Managers working directly with end-users, stakeholders and subject matter experts to understand customers' needs and identify specific business outcomes and associated metrics to guide solution delivery. Please rate your team's ability to collaborate with customer-side and IT-side experts to define Minimum Marketable Features (MMF) in support of the hypothesis. Sit (1-2): Product Management roles and responsibilities are not defined or followed. Crawl (3-4): Product Management creates requirements in large batches with little customer or development collaboration. Walk (5-6): Product Management collaborates with business-side or development-side experts, but not both, when defining requirements. Run (7-8): Product Management regularly collaborates with business-side, development-side, and operation-side experts but does not define Minimum Marketable Features. Fly (9-10): Product Management always collaborates with business-side, development-side, and operation-side experts and defines Minimum Marketable Features.	3.0	2
		Architecting for continuous delivery involves applying ""just enough"" intentional architecture to assure policy compliance without sacrificing product development flow, to ensure solutions are loosely coupled and to continuously pay down technical debt. Please rate your team's effectiveness at architecting for continuous delivery. Sit (1-2): Architecture is monolithic and fragile; it is difficult to change and involves managing complex dependencies across many components and systems. Crawl (3-4): [illegible]hitecture is predominantly monolithic but some applications/systems are loosely coupled. Walk [illegible]tures is most decoupled but doesn't allow Release on Demand.		

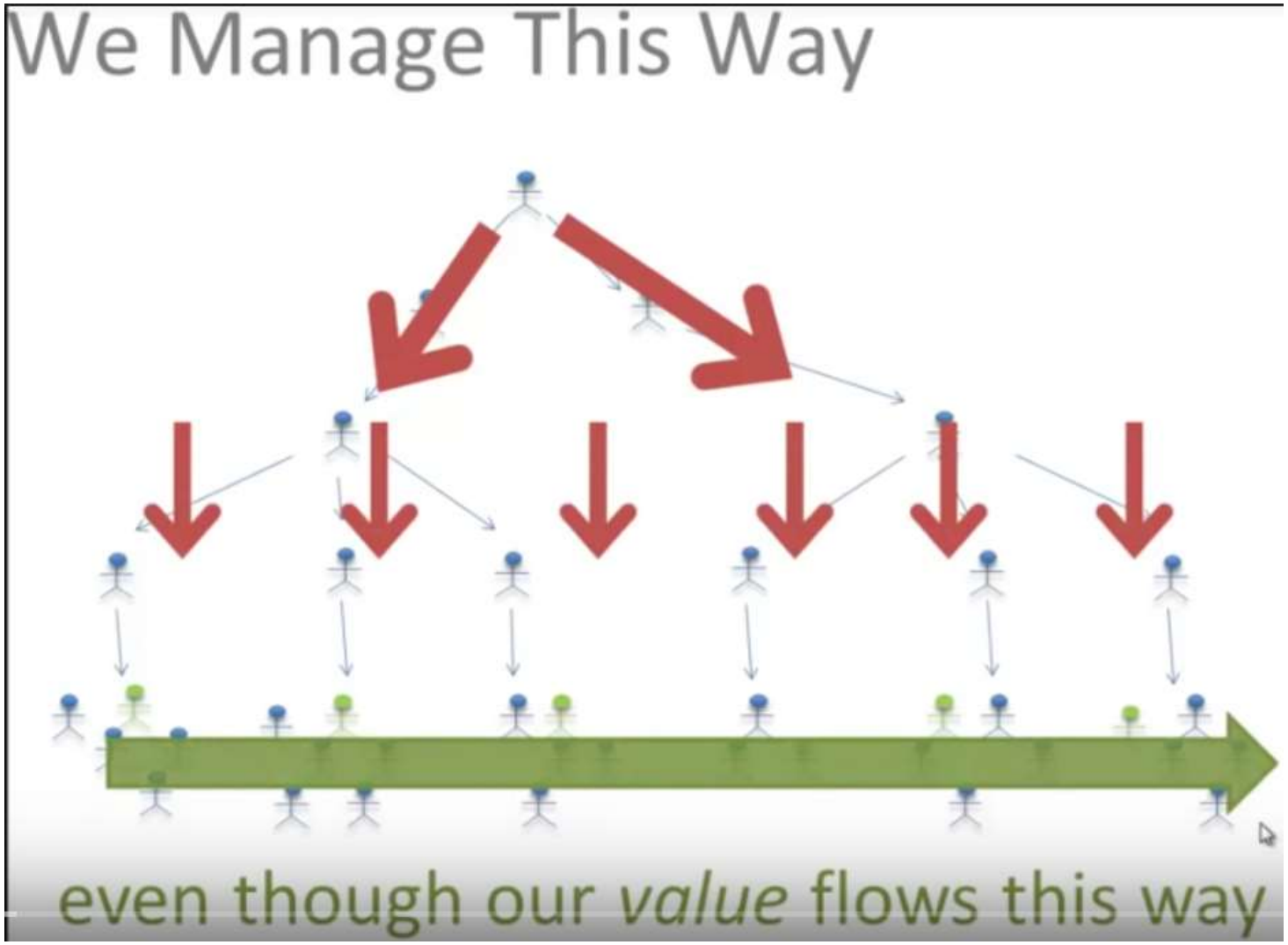


IVAR JACOBSON INTERNATIONAL

Scoring:
Sit (1-2), Crawl (3-4), Walk (5-6), Run (7-8), Fly (9-10)

Creating winning teams.

Org Structure and the flow of Value

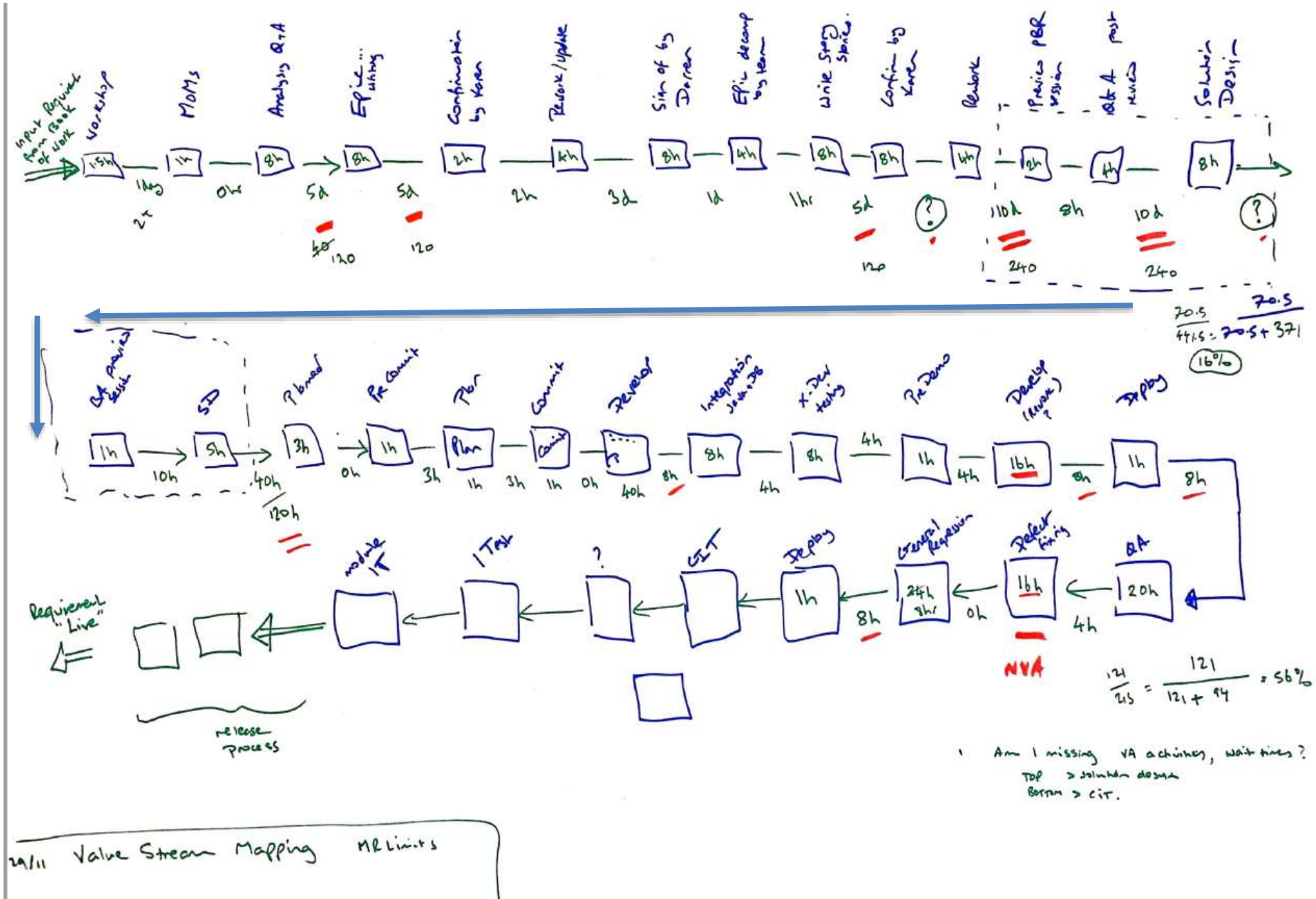


Al Shalloway Lightning talk
https://www.youtube.com/watch?v=EMzaP5VHdyE&feature=youtu.be

Work flows from hand to hand…

Al Shalloway Lightning talk
https://www.youtube.com/watch?v=EMzaP5VHdyE&feature=youtu.be

Example Value Stream Map

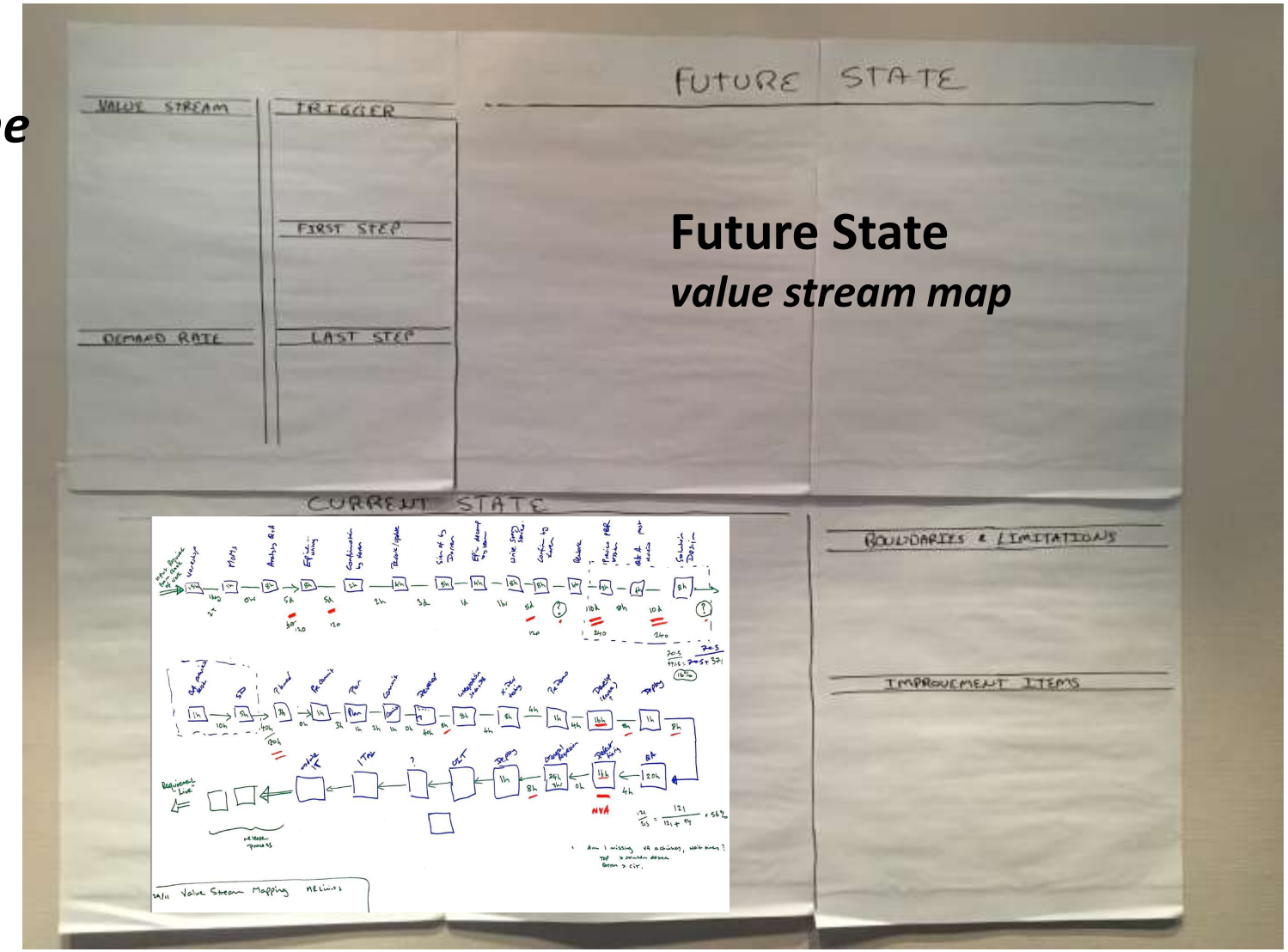


IVAR JACOBSON INTERNATIONAL

Map the Value Stream to identify Improvements

Value Stream *name*

- **Trigger**
- **First Step**
- **Last Step**
- **Demand Rate**



Future State
value stream map

Boundaries, Limitations
i.e. what can't you change?

Improvement Ideas

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

Map the Value Stream to identify Improvements

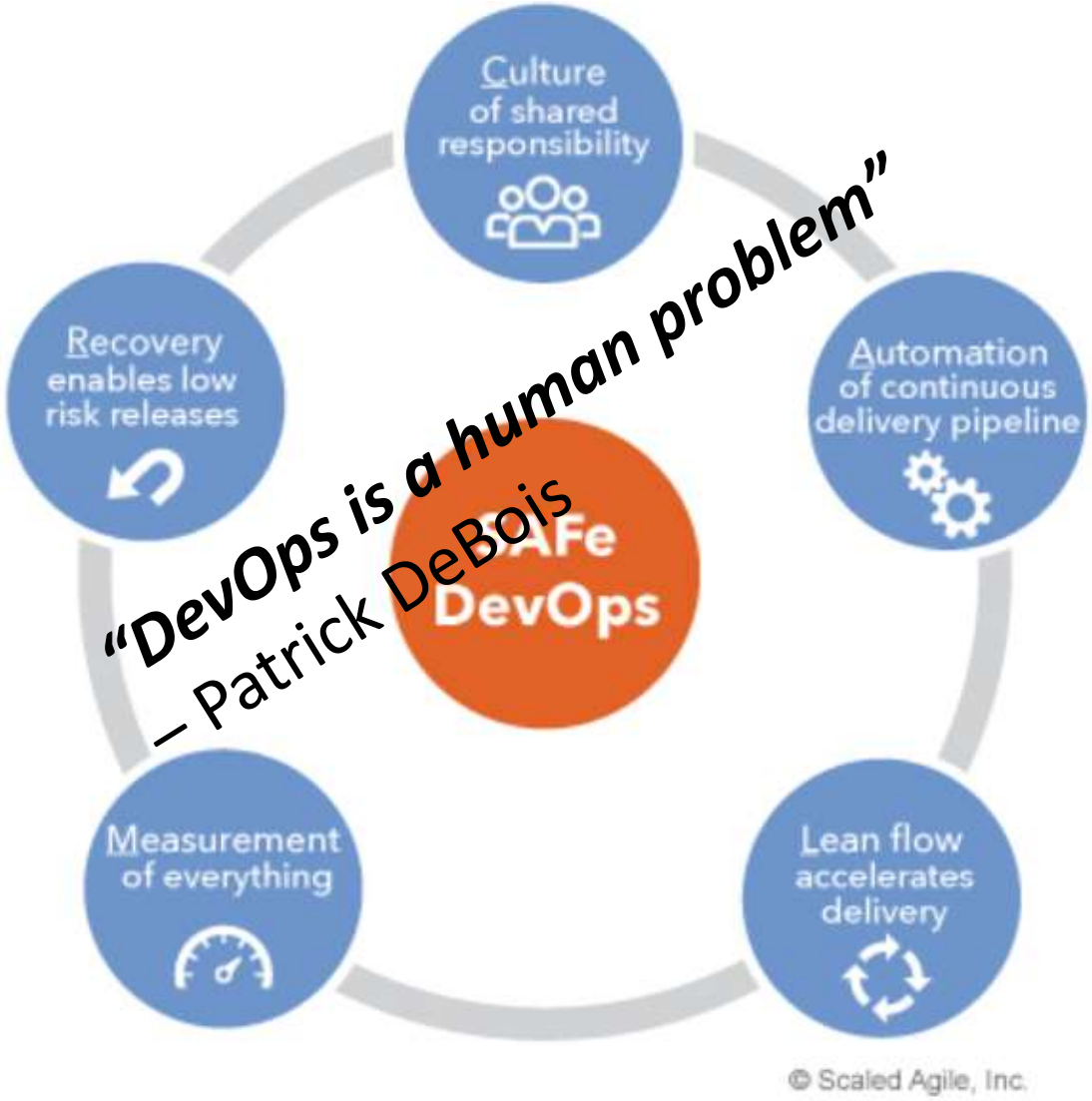
Identify all the key steps in your software development lifecycle, how long each activity takes, how long work typically waits before it moves to the next step, and what is the probability of finding errors when it moves to the next step.

Improvements could be:

- new practices to speed up work or remove errors,
- Reverse, remove or merge work steps
- Increase collaboration

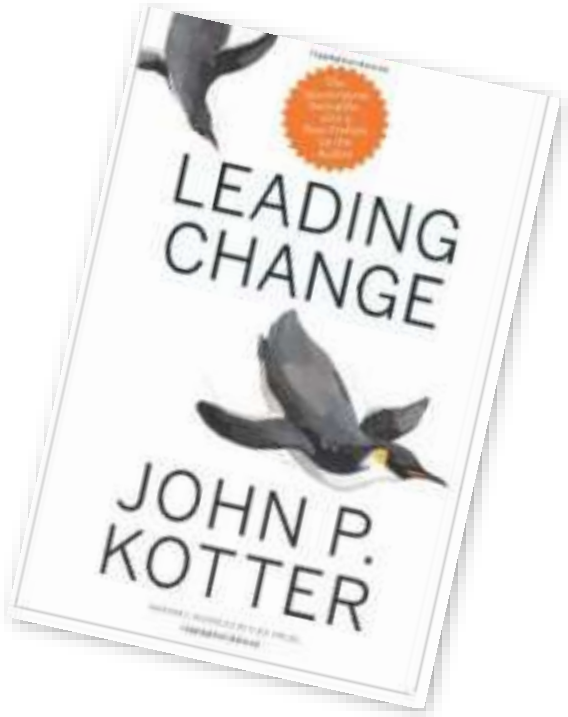
DevOps is a Cultural Change… and change is hard!

- To be successful we need to change our work practices and our culture…
- We should follow good practice to embed this change in our organisation.
- Kotter's model for change is a useful structure to follow…



Key Principles of Change Management

1. Establish a sense of urgency
2. Create a powerful guiding coalition
3. Develop the vision and strategy
4. Communicate the vision
5. Empower employees for broad-based action
6. Generate short-term wins
7. Consolidate gains and produce more wins
8. Anchor new approaches in the culture

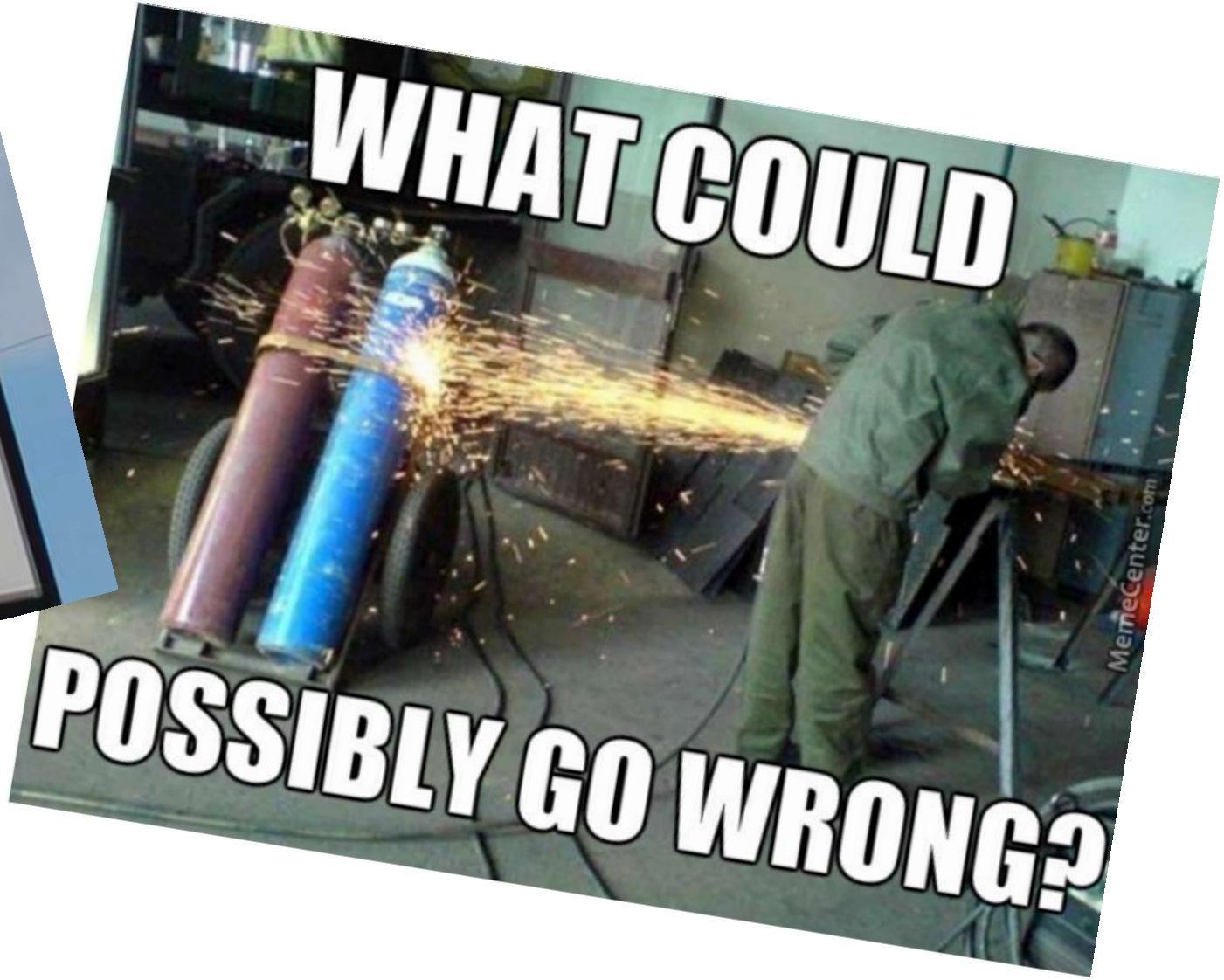


"To change the culture you have to change the organization."

Key Principles of Change Management

1. Establish a sense of urgency – Competitors moving faster than us? Customer Complaints?
2. Create a powerful guiding coalition – A team of 5-10 people from Business, Dev, Test, Ops, Architects, Governance, management …
3. Develop the vision and strategy – Clear vision : what success will look like. Use Value Stream Mapping as a iterative process to identify and remove sources of biggest delays and errors.
4. Communicate the vision – Regular briefings during SAFe PI Planning events and elsewhere.
5. Empower employees for broad-based action – Provide training & coaching; give permission for teams to innovate; reserve time to implement improvements we identify with value stream mapping
6. Generate short-term wins – report impact of initial changes each PI. Celebrate! Faster? Better?
7. Consolidate gains and produce more wins – revise value stream regularly and act on new insights.
8. Anchor new approaches in the culture – Reinforce that our jobs now demand more collaboration. Focus on robots and automated processes that confirm quality to our satisfaction at each stage of the development process, and which reduce the possibility of human error.

So what could possibly go wrong?



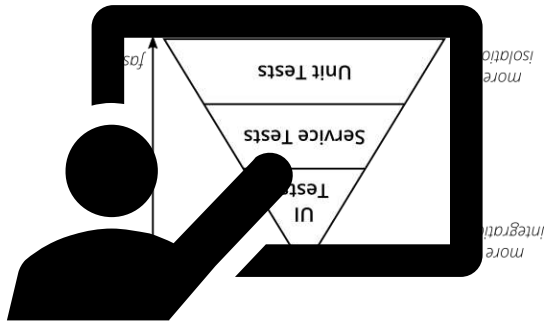
A few DevOps Anti Patterns - It ain't easy to change!

Here are a few anti patterns I have seen occur in organisations who truly want to become more agile and implement DevOps practices. This not an exhaustive list! (All are Test-centric)

2nd Class Citizen

The Waterfall

The Patch



Flaky Tests

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

DevOps Anti Patterns – "Waterfall Automation"

"We are agile but...

We will run a separate 'project' to automate all our tests, using a separate team, then deliver the results back into our system only at the end"

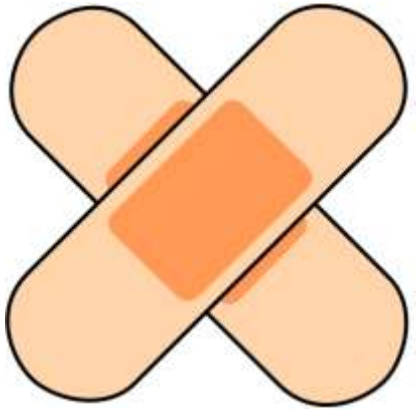
This is just a big batch of work with little or no valuable feedback generated until the end of the project. Its not agile! You will end up with a big "integration" and "testing" phase at the end of your test automation project.

- Automation is hard, so you need to experiment a little and then see what improvements you get, and what problems you need to solve.
- Don't just automate the existing manual tests; automated tests have different characteristics to manual tests and so need to be constructed differently.
- You may need to revise your whole Test Strategy, which will impact what sort of tests you actually will need in the future.

DevOps Anti Pattern – "The Team Patch"

"Oh DevOps sounds really important…

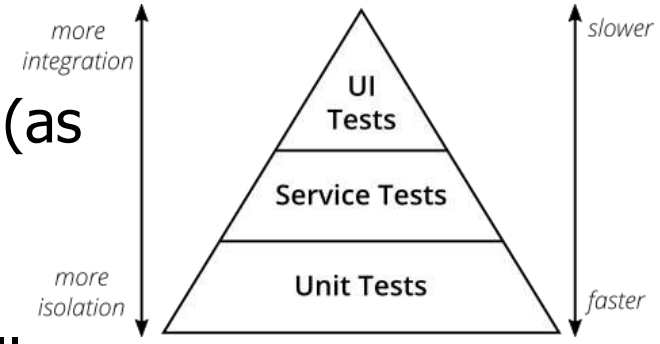
So let's create a new DevOps team who are now solely responsible for all of these changes in our organisation."



- Inserting a new team into the existing development lifecycle will just increase hand-offs and slow things down.
- We probably do need new skills. And we do need a team who will do the value stream mapping of our end-end processes.
- But we need to steadily change the way everyone is working and collaborating together. Don't try to patch things by keeping all the old ways and just adding a new team.
- We need to train everyone to collaborate, and not tell one team to "*do DevOps*".

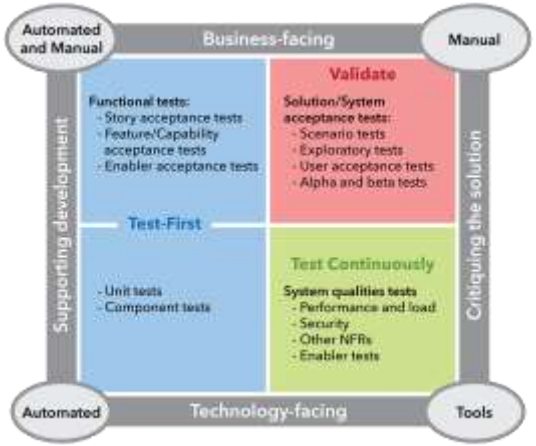
DevOps Anti Pattern – "Flaky Tests"

"We create mostly UI-based tests to show end-end functionality, so we just will automate all of those existing tests…



If the architecture of your system is monolithic and tightly coupled (as most legacy systems are), it is easier to write UI tests than to test internal components independently.

- But these UI tests take a long time to run, and are 'flaky' – they sometimes don't work, so you run them again and hope they will work next time!
- A new test strategy will reduce the number of these slow end-end tests, discard the untrustworthy tests, increase unit tests and component tests to create a "testing pyramid". These lower-level tests run more quickly, can run in parallel and are more reliable.
- The existing architecture may need to be refactored to clean up the modular structure of the system and expose clear APIs to test.
  - NB Not all UI tests are end-end tests; keep any UI tests that just unit test screens, views and controls.
  - Not all tests need to be automated.



DevOps Anti Patterns – Tests as 2nd class citizen

"We build quality in...

Unless it is test code, in which case we don't design it or use any quality standards"

Your tests are part of your system. Test code requires architectural oversight.

Poor test design results in tight coupling with test code… creating fragile tests Fragile tests means many tests break whenever you add new tests and evolve functionality, so you spend more time fixing tests than moving forwards.

Solution: Design tests and have a Test Architecture. Maintain loose coupling between test code and the system. Create tests which can initialise their own test data. Architects & designers must care about Tests!

"*As the tests get more specific, the production code gets more generic.*" ('Uncle' Bob Martin)

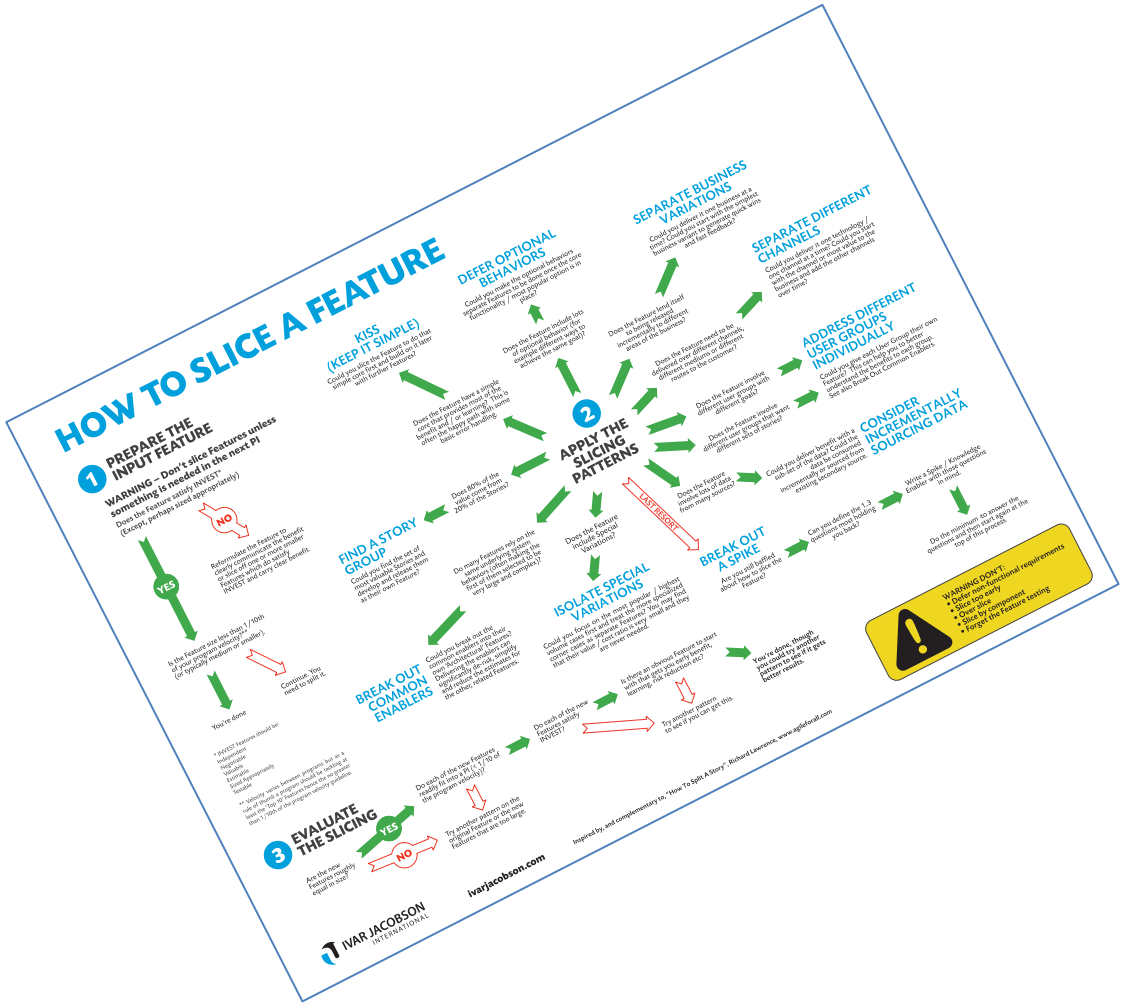
http://blog.cleancoder.com/uncle-bob/2017/03/03/TDD-Harms-Architecture.html

Summary

- Many of these anti-patterns are primarily caused by the **culture** of our organisation.
- New technical skills and competencies are definitely also required to accelerate deliveries - of course.
- It is often the organisational culture, silos and leadership attitudes which prevent us from collaborating fully.
  - E.g. Architects don't have any oversight over of test assets because its 'not important software'

"DevOps is a human problem"

– Patrick DeBois



Visit www.ivarjacobson.com for cards, posters, blogs, webinars etc…

Thank You WorldPay!

IVAR JACOBSON INTERNATIONAL

Creating winning teams.