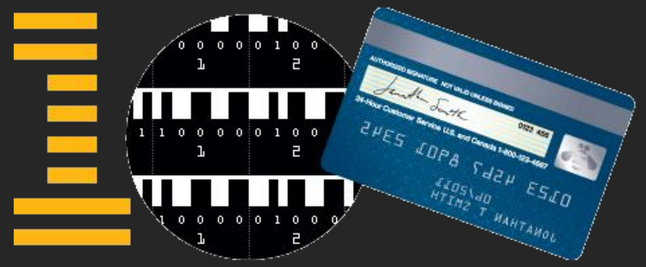
“Zero Waste” Engineering Practices

Suzanne Livingston
IBM Sustainability Software
Vice President Engineering

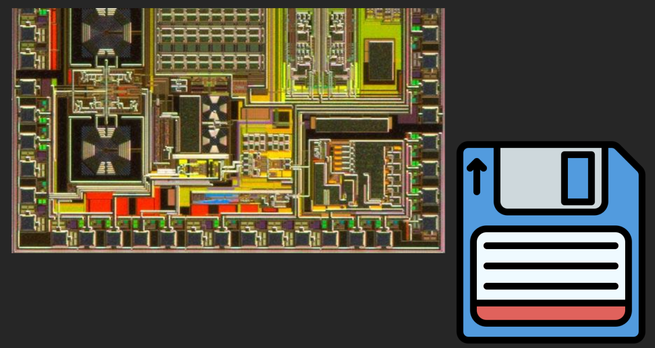
100+ Years
Investments in Research & Engineering

1880s Punchcards

1952 UPC Code

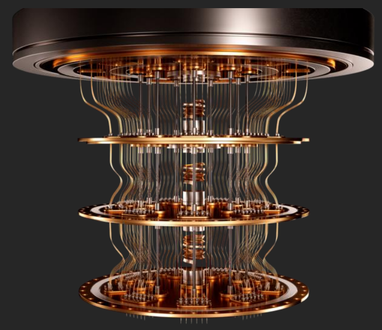


1969 Magnetic Stripe



1970s-80s Silicon Germanium Chips & Floppy Disks

1981 Laser Surgery



2019 Quantum

Today’s IBM (a sampling)

Red Hat

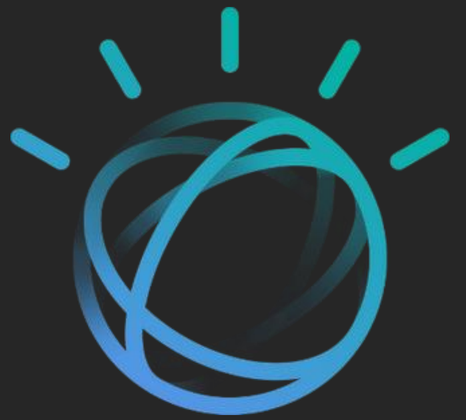
Maximo

SPSS Analytics

WebSphere

QRadar

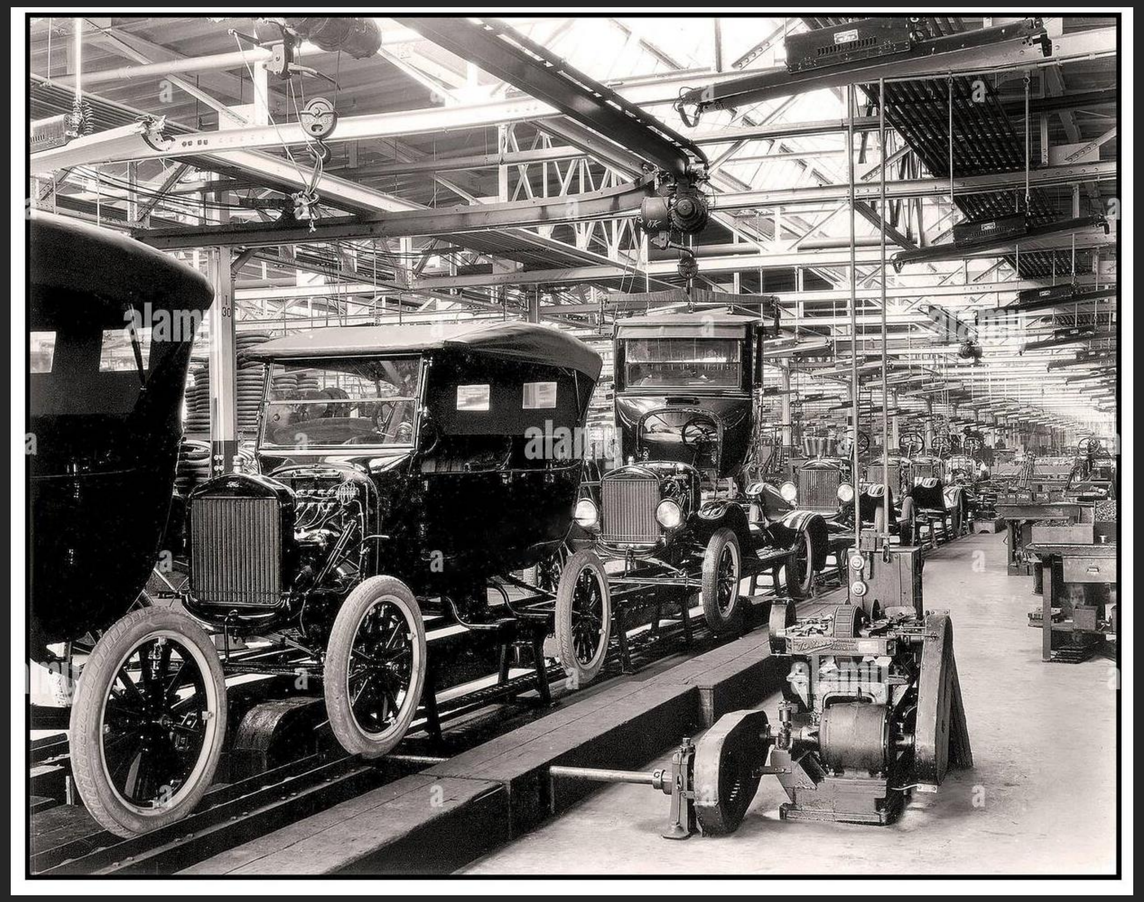
System Z



Watson

- “We can’t build a Tesla on a Model-T production line.”

--SVP IBM Software



Chapter 1
Clarity on what we are building

▸1

- Product alignment

- What are the top things we should be building?

▸2

- Backlog transparency

- What is each squad working on right now and next?

▸3

- Clarifying requirements

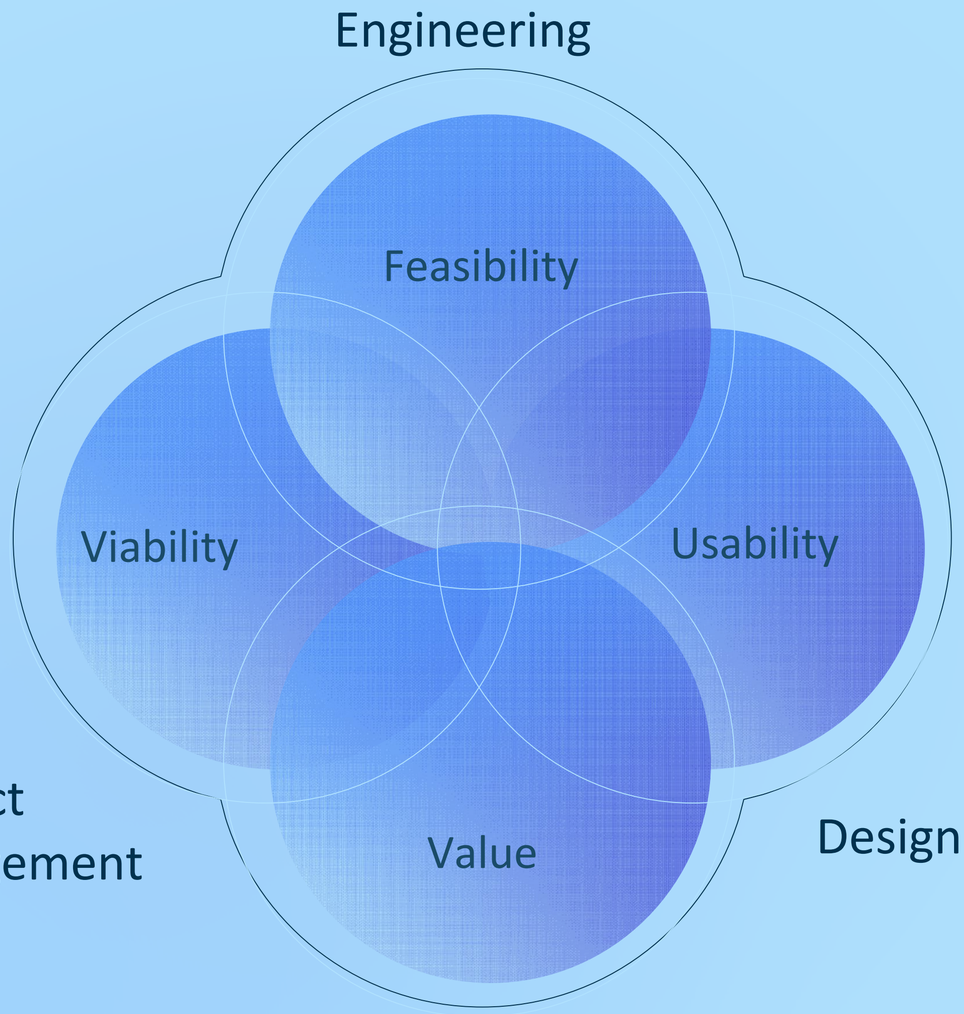
- Stakeholder expectation vs developer reality

▸4

- Discipline in our practices

- Reviewed?
- Automated?
- Tested?
- Accepted?

ALIGNMENT



40% - Keep the Lights On

- Currency
- Security
- CD: SRE/Operator maturity, test automation
- Engineering Transformation
- [redacted] SaaS operation

22% - [redacted]

15% [redacted]

7% - Other new Features

4% - [redacted]

3% - [redacted]

3% [redacted]

2% - [redacted]

2% - [redacted]

1% - [redacted]

1% - [redacted]

Do
opp

[redacted]

Scale

getting value?

Viable

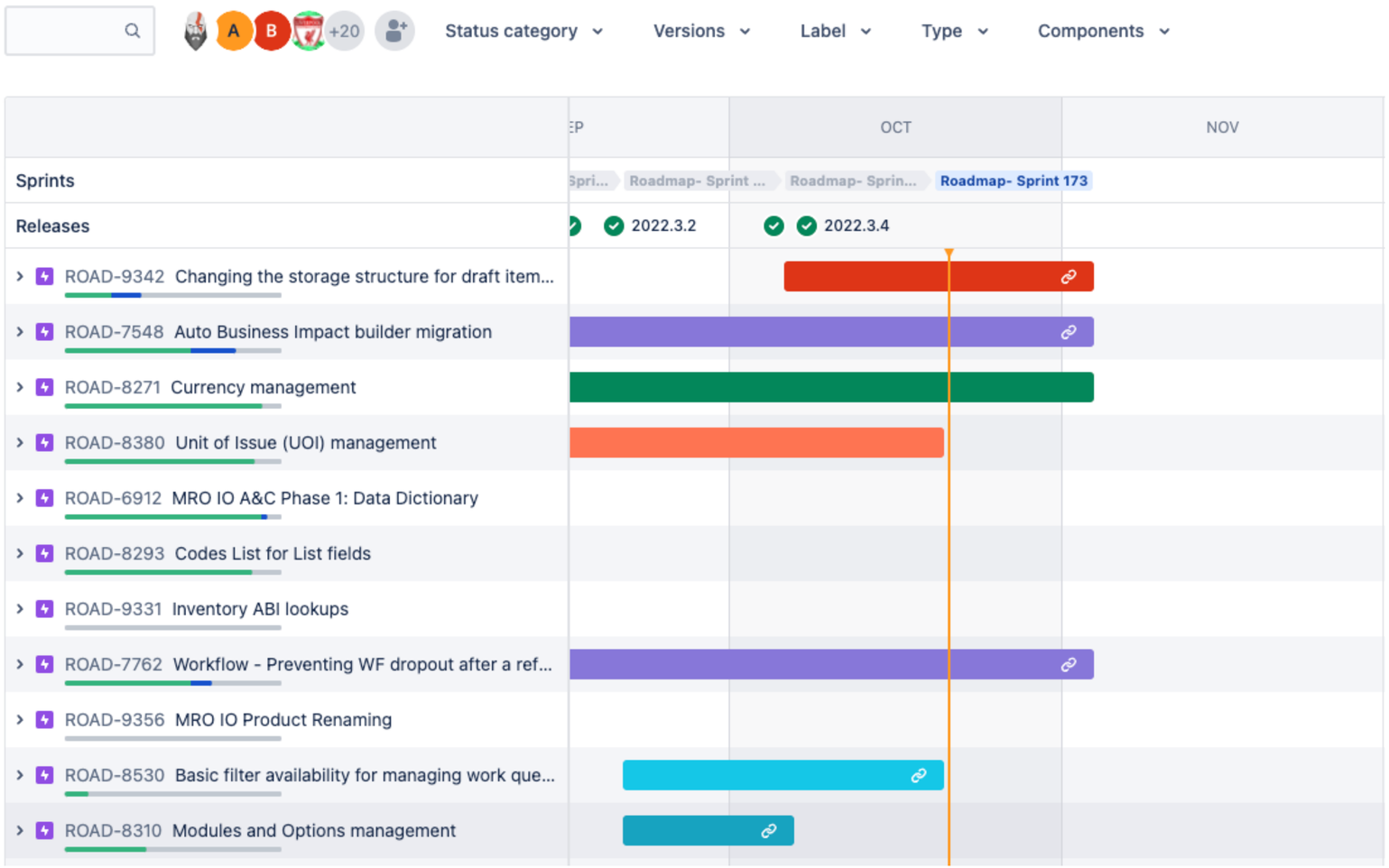
Experience
Functional
Non-functional

Revenue
Adoption
New logos
Margin

CLARIFY REQUIREMENTS

Epic ownership
- Joint ownership – PM, Dev, Design
- Understanding requirement
- Create technical design and docs
- Write and refine stories
- Form story acceptance criteria

Formally define
- Definition of ready
- Definition of done
- Roles & Responsibilities as a developer
- Roles & Responsibilities as a reviewer
- Quality, security & automation included



Retrospectives & Continuous Improvement

What did we learn?

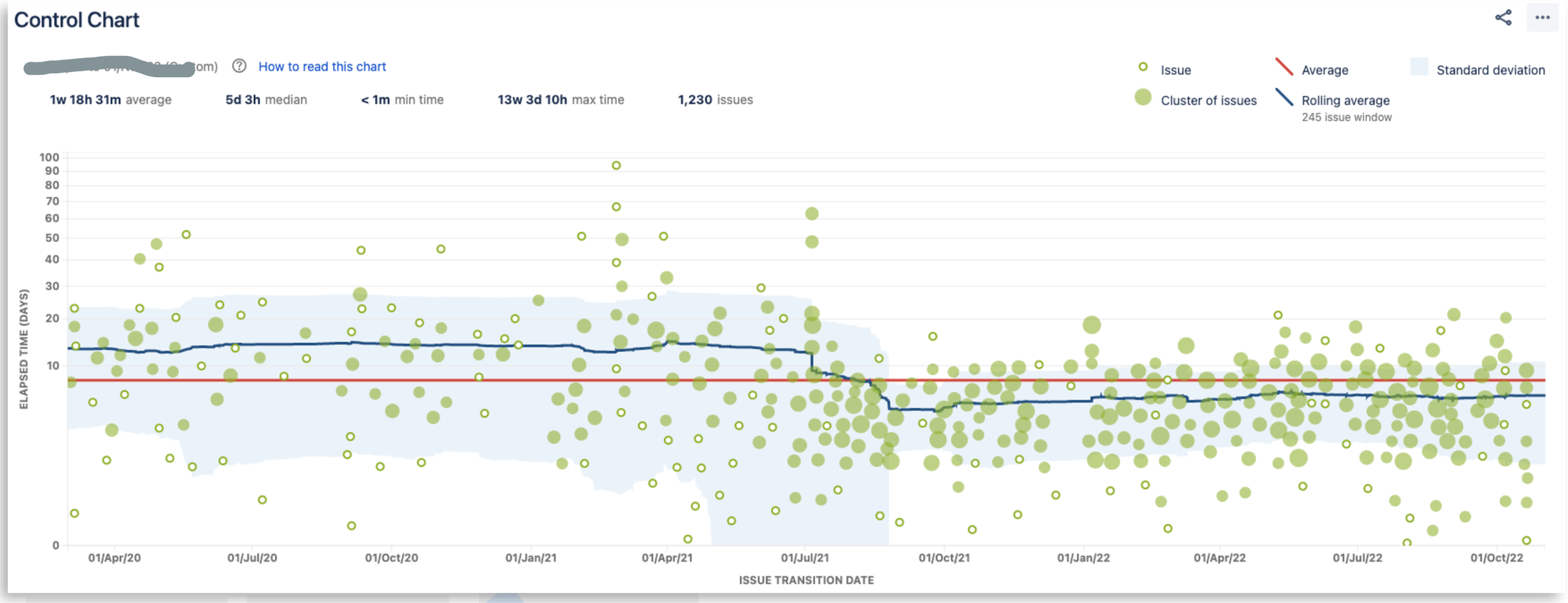
- Improved estimates in story points
- Backlog refinement helped the whole team to understand the coming work

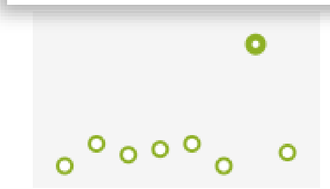
What do we need to improve?

- Story writing needed improvement
- Epic refinement enabled us to see the gaps
- Average number of tickets created is higher than previous sprint, as an indicator of quality

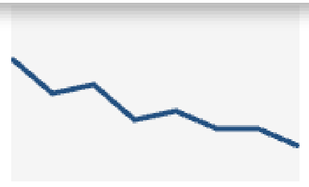
	What we are doing well in [redacted]	
1.	Team coordination is great, e.g devops squad is very responsive to resolve issues.	
2.		
	What we need improve continuously	**Suggested Actions**
1.	Although we reviewed any stop-ship feature altogether when we start working on [redacted], couple requirements came late into the game with high attention such as FIPS, eSig for Mobile etc.	• Identify the stop ship feature with Product manager and circle back to the broader team in the beginning of [redacted] cycle. • If there any change on the requirement, it should be circled back to the team as well for awareness (SoS meeting is a good platform for such communication).
2.	Early program need more close collaboration especially from PM side	• [redacted] is looking beta program for [redacted] specifically. • [redacted] is under the scope in [redacted] → It has dependency on [redacted] core, [redacted] • *This early program should be a check item when we do the new release planning (as requirement).*

Why do this

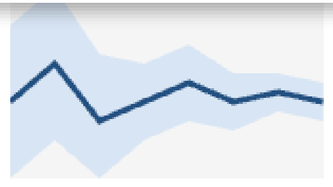




Visibility
See outliers and investigate their cause to reduce them in the future.



Efficiency
Decreasing rolling average indicates process improvements and increased throughput.

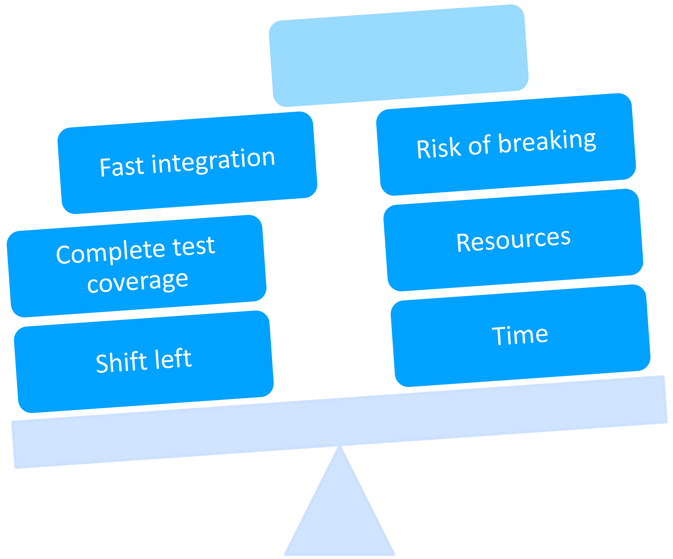


Predictability
Narrow standard deviation through process improvements to improve predictability of cycle time.

Chapter 2
Automation

- 10%
  - Decrease in keep-the-lights-on activities
- 50%
  - Development time savings from automation
- 80%
  - Fewer security vulnerabilities with security automation

Automation from scratch



Shift left

Less problems

Build, unit test

Build common framework
Feature unit test framework
Team's own build with unit test process

Smallest footprint, fast turn around, most shift left and economical

Team pipelines

Typically, Jenkins driven, integration at app or team level

Smaller footprint, application functional specific environmental variations can be more economically covered here

Integration pipelines

Tekton pipelines and travis, pipelines

System level integration, large footprint, longer turn around, cover system integration variations

Release candidate pipeline

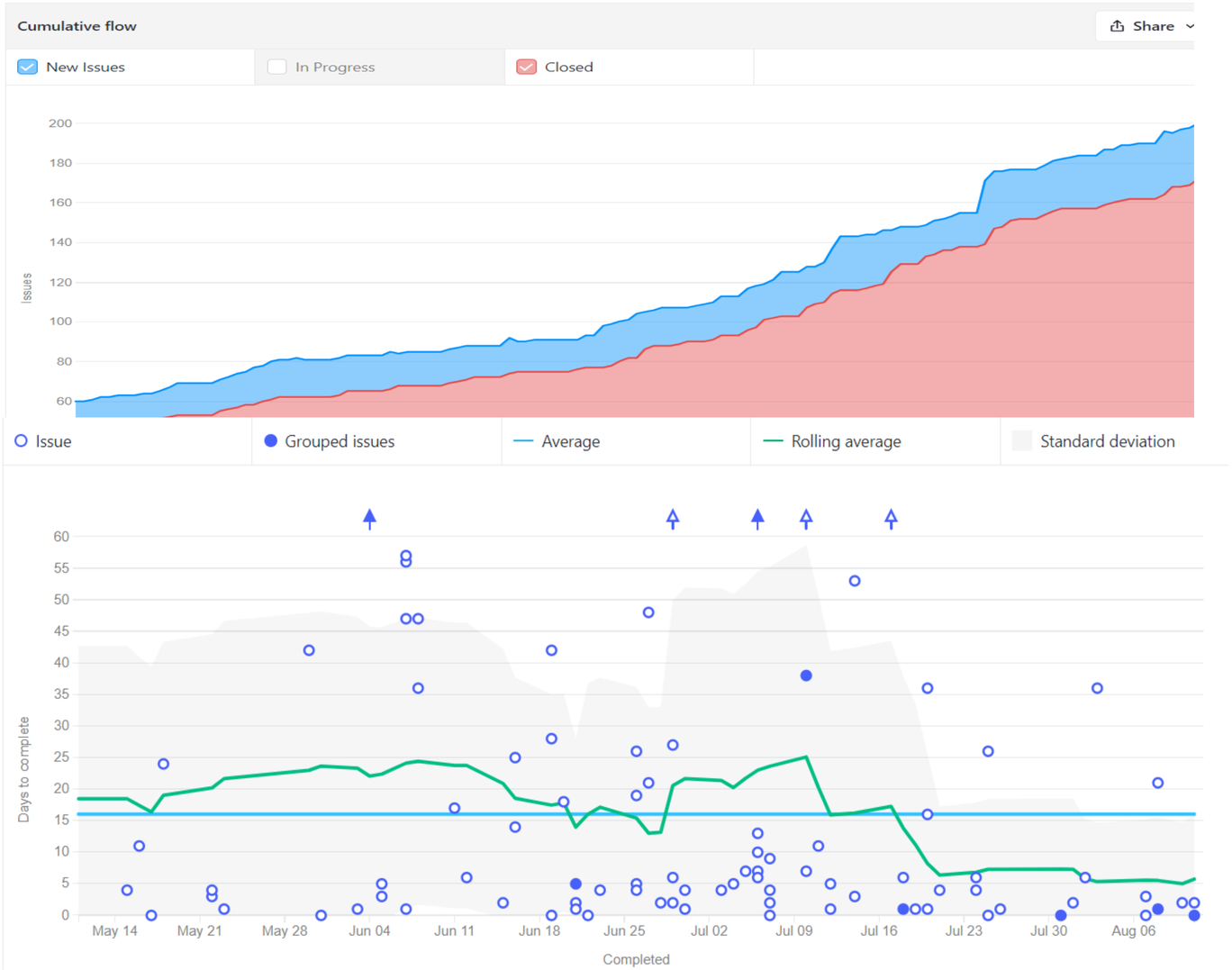
Tekton pipelines and travis, long running and more

Release pipeline

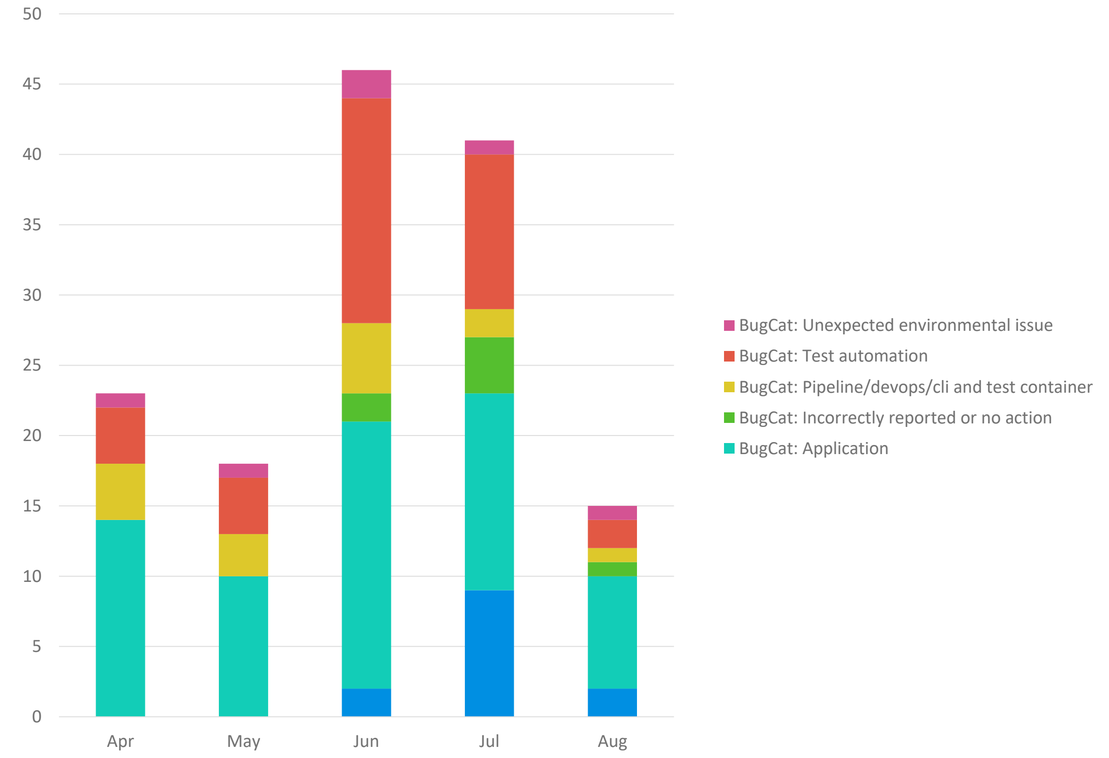
Tekton pipelines and travis

Final system validation for release and release process automation

Automation Payoff



Total bugs 200, Application bugs: 94

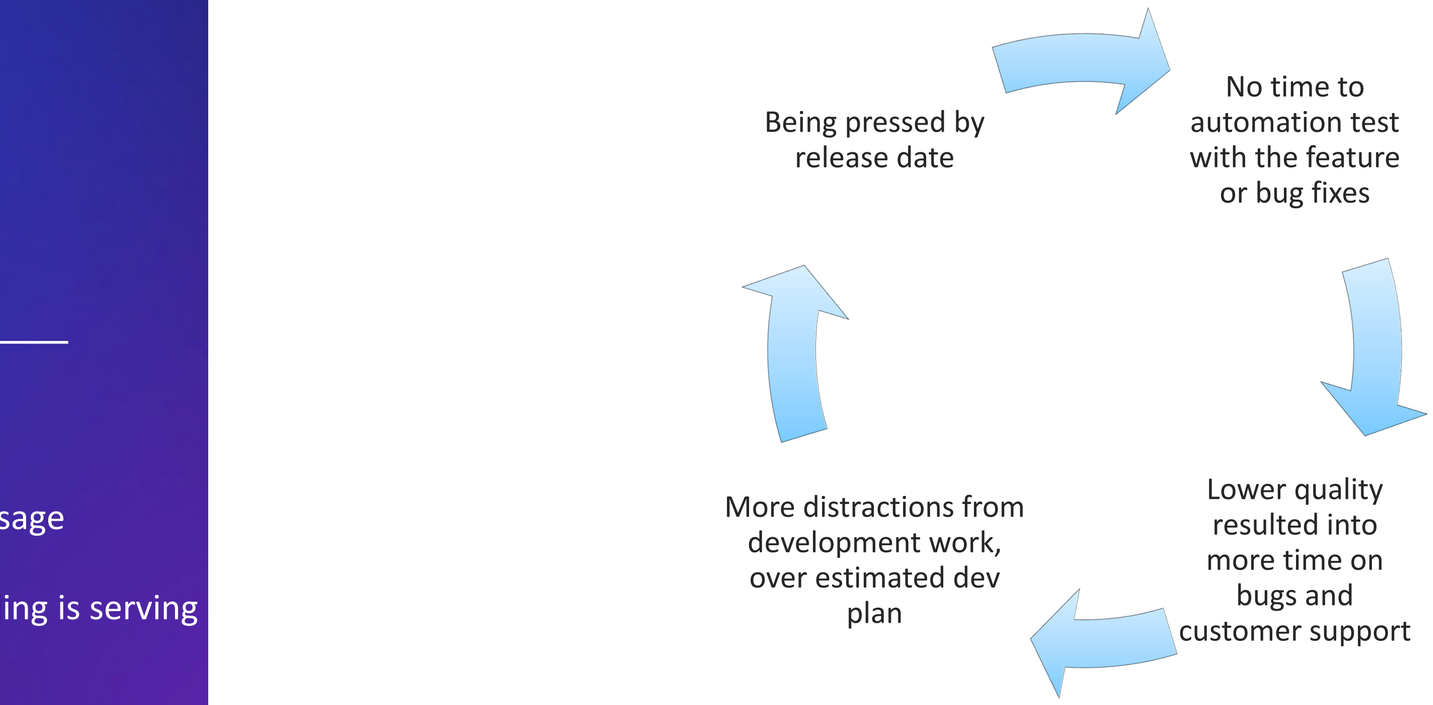


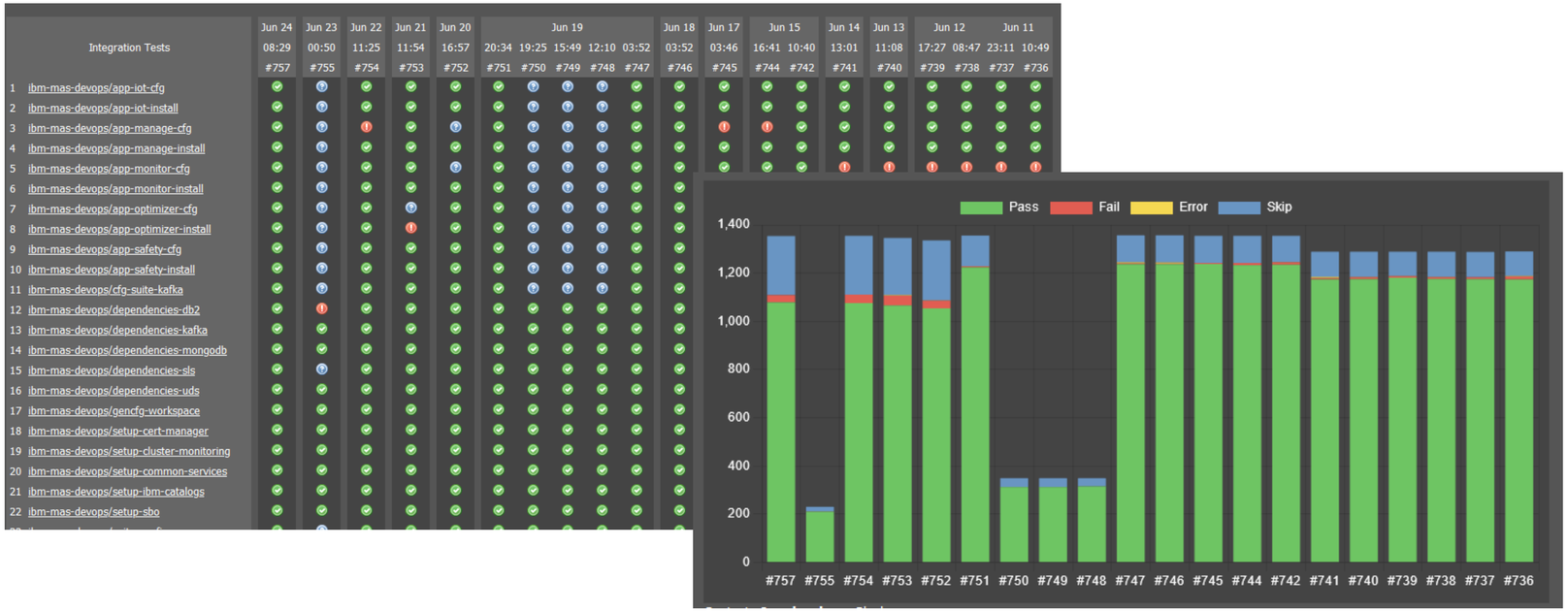
CONTINUOUS ENGINEERING

Mindset shift - A feature is not just code
- Understand the big picture - persona, usage scenarios
- Ask questions of how what we are building is serving the consumer/stakeholder/user

Development practice shift
- Test driven development
- Functional and technical design at every level
- Dev and test being one
- Squads oversee writing their own tests and deployment out to production

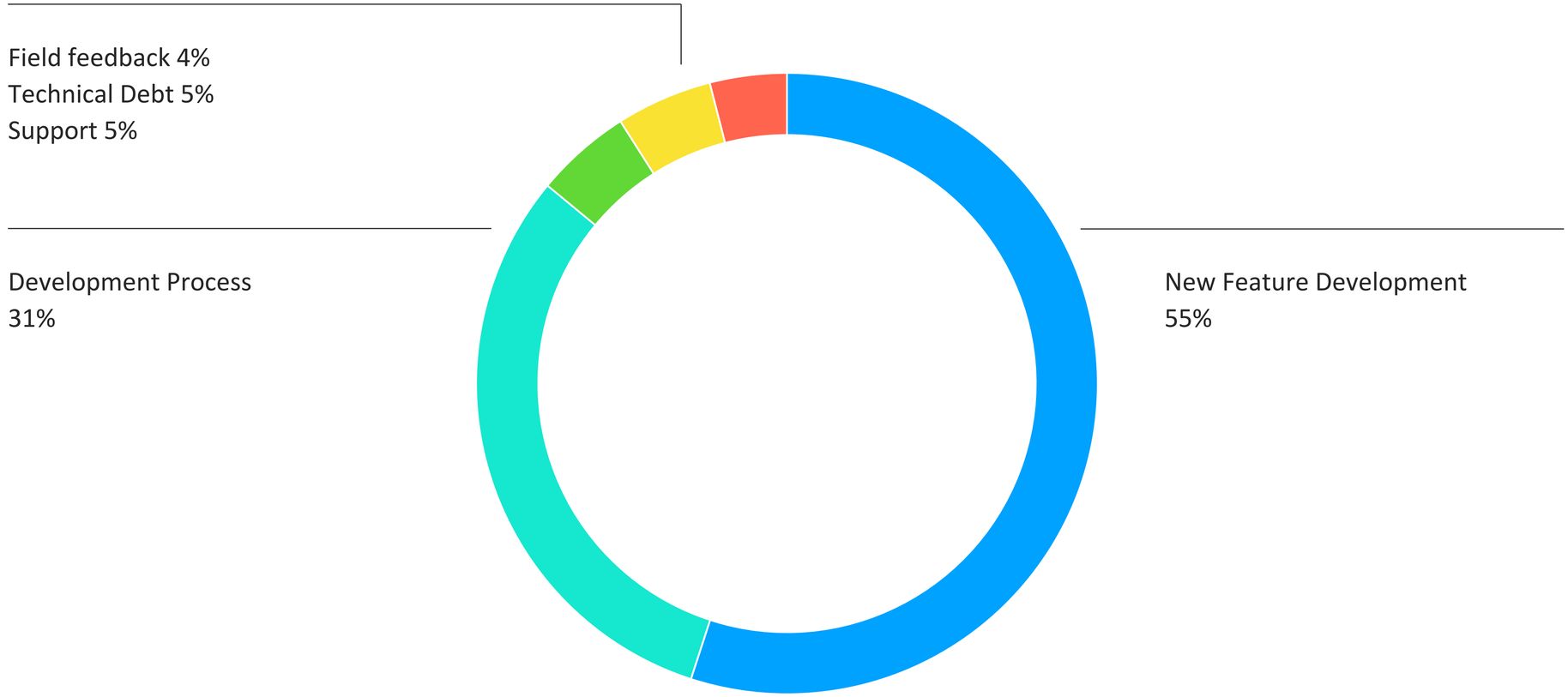
Smaller changes produce higher quality
- Reviewers/Committers refuse large PRs
- Shift as much of the security/compliance as left as possible i.e. Scan and report compliance on every PR & Merge build



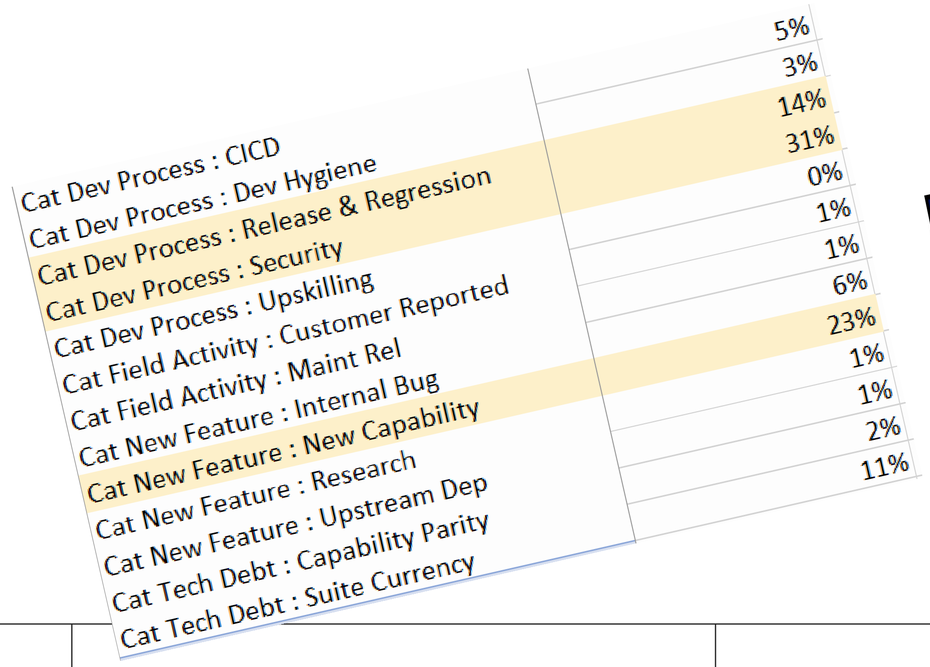


Chapter 3
Visibility

How much time is being spent on what types of stories in each development cycle



Chapter 3 Visibility



Cat Dev Process : CICD	5%
Cat Dev Process : Dev Hygiene	3%
Cat Dev Process : Release & Regression	14%
Cat Dev Process : Security	31%
Cat Dev Process : Upskilling	0%
Cat Field Activity : Customer Reported	1%
Cat Field Activity : Maint Rel	1%
Cat New Feature : Internal Bug	6%
Cat New Feature : New Capability	23%
Cat New Feature : Research	1%
Cat New Feature : Upstream Dep	1%
Cat Tech Debt : Capability Parity	2%
Cat Tech Debt : Suite Currency	11%

Sprint: Dates (2 wks)	Planned Pts	Completed Pts	Plan vs Comp. (gap)
S16: Sept 22 – Oct 05	120	118	-2, -2%
S17: Oct 06 – Oct 19	100	111	+11, +11%
S18: Oct 20 – Nov 02	90	103	+13, +14%
S19: Nov 03 – Nov 16	120	current	
S20: Nov 17 – Nov 30	100	NA	
S21: Dec 01 – Dec 14	120	NA	
Avg Velocity	103	110	

▶1

- Planned <= Completed creates a healthier work environment

▶2

- Smaller, well-defined stories completed on time

▶3

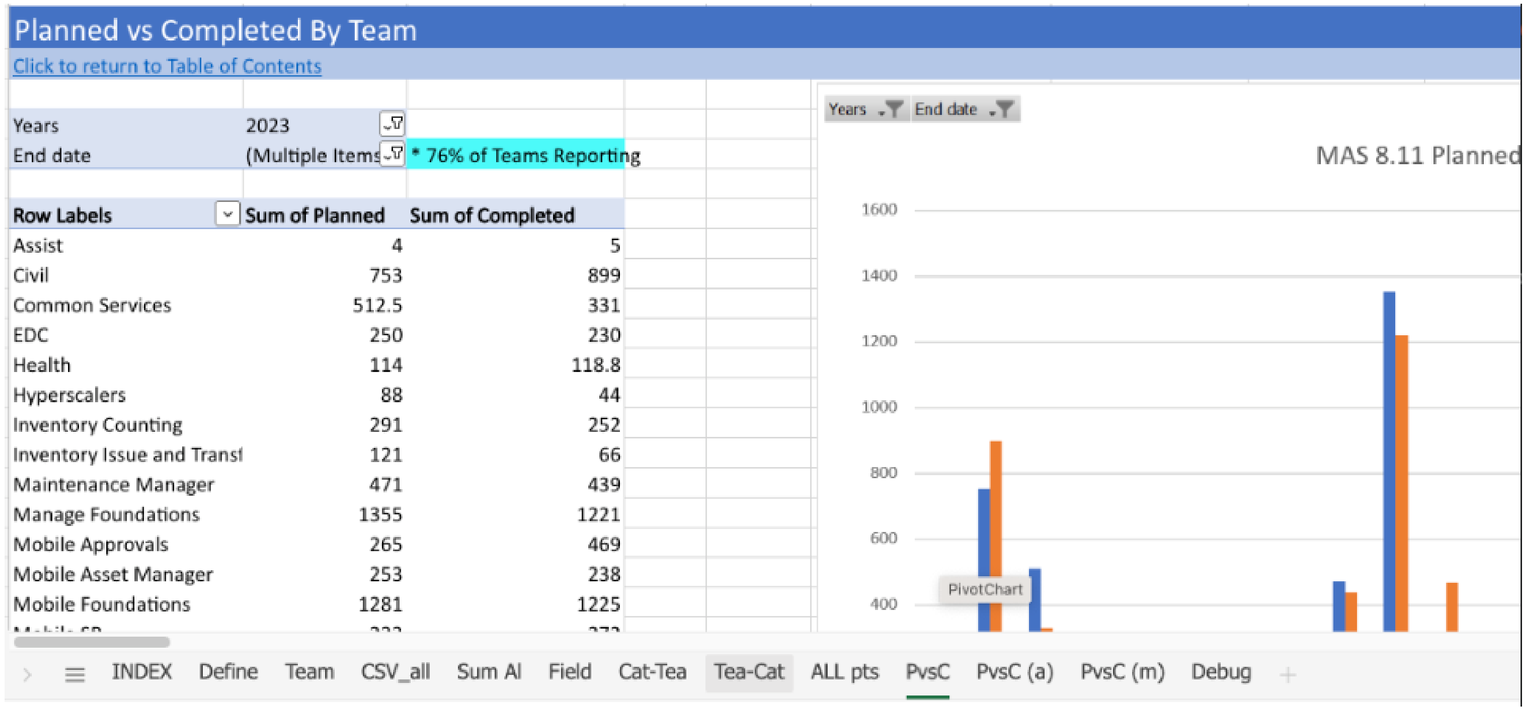
- Team velocity improved & new members ramping up faster

▶4

- Teams reviewing average velocity as part of sprint planning

END-OF-SPRINT REVIEW

- Demos of key changes
- How did our estimations hold up? If we were off, what got in our way? Could we have prevented distractions?
- What areas of concern or potential problem areas are being identified?
- What can we do better/differently to improve?
- What best practices have we used for areas that we're doing well in?



Planned vs Completed By Team

Click to return to Table of Contents

Years	2023	
End date	(Multiple Items	* 76% of Teams Reporting

Row Labels	Sum of Planned	Sum of Completed
Assist	4	5
Civil	753	899
Common Services	512.5	331
EDC	250	230
Health	114	118.8
Hyperscalers	88	44
Inventory Counting	291	252
Inventory Issue and Transf	121	66
Maintenance Manager	471	439
Manage Foundations	1355	1221
Mobile Approvals	265	469
Mobile Asset Manager	253	238
Mobile Foundations	1281	1225

Chapter 4 Operations

/IBM-engineering-for-sustainability

▶80%

- Average amount unused in overallocated infrastructure for seasonality

Environmental impact practices

How IBM designs software products today has a growing impact on the carbon footprint in data centers across the world. New techniques in large-scale computing and data use a significant amount of energy.

Software development

The typical software development process has changed drastically over the years, from longer delivery cycles of monolithic products to a continuous integration and continuous delivery (CICD) approach that is focused on finer-grained services, called microservices. However, the combination of these services still requires a level of architecture specification for their integrated operation.

Software architecture

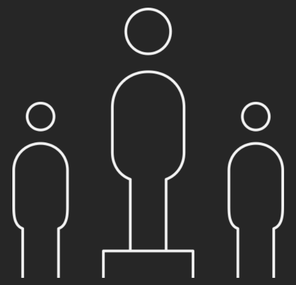
Well-architected software products meet design requirements while minimizing the overall infrastructure usage that corresponds to energy and carbon footprint. The software architecture process prioritizes the tradeoffs between sustainability and optimization requirements, which can range from speed-to-market, cost of engineering, user experience, availability, social impact, and sustainability. These requirements often conflict with each other. For example, you can improve availability through redundancy, but that can negatively impact energy usage. Also, you can improve speed-to-market but sacrifice usability or sustainability. Some conflicts can be improved or resolved by increasing the cost of engineering.

Considering the carbon footprint of running the software is a crucial part of designing and architecting it for environmental sustainability. Software architecture must be guided by a few key principles that affect carbon footprint.

↑110 TW

The Cambridge Center for Alternative Finance estimated that bitcoin alone consumes around 110 terawatt hours per year, about the same energy usage as the entire country of Sweden.

In Summary



Alignment

Avg Issue dropped
25 to 5 days

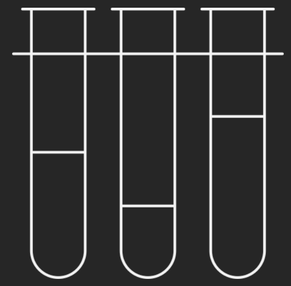
Predictability improved
1 Month+ to 1 Sprint

Automation

50% time back to invest

10% fewer KTLO activities

80% security improvement



Visibility

Estimation improvement

Transparent communication

Efficient prioritization

Continuously hunt & reduce waste