Microsoft's DevOps Journey

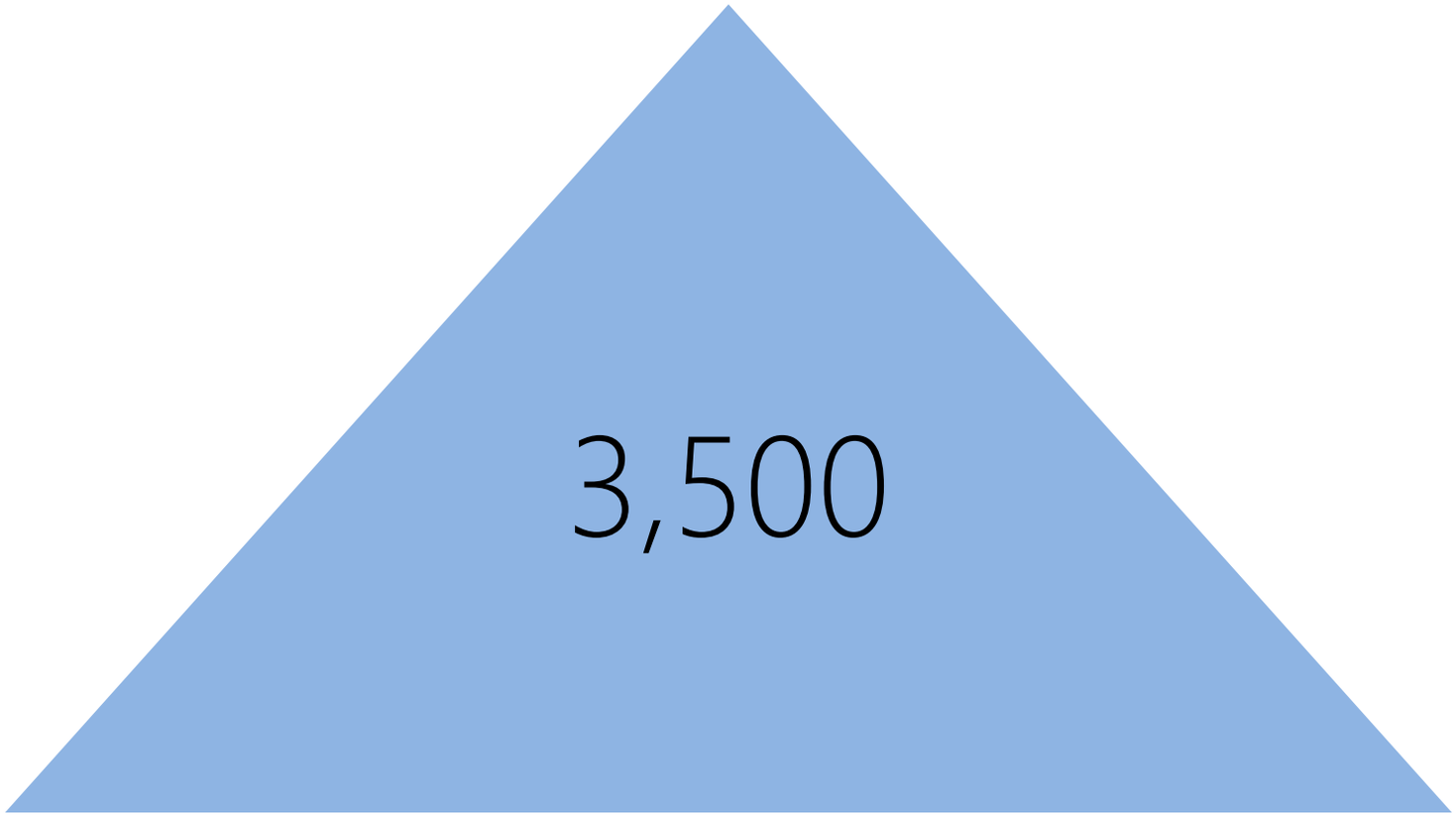
Paul Hacker
DevOps Architect – DevOps CAT
pahacker@microsoft.com

Microsoft

Introductions

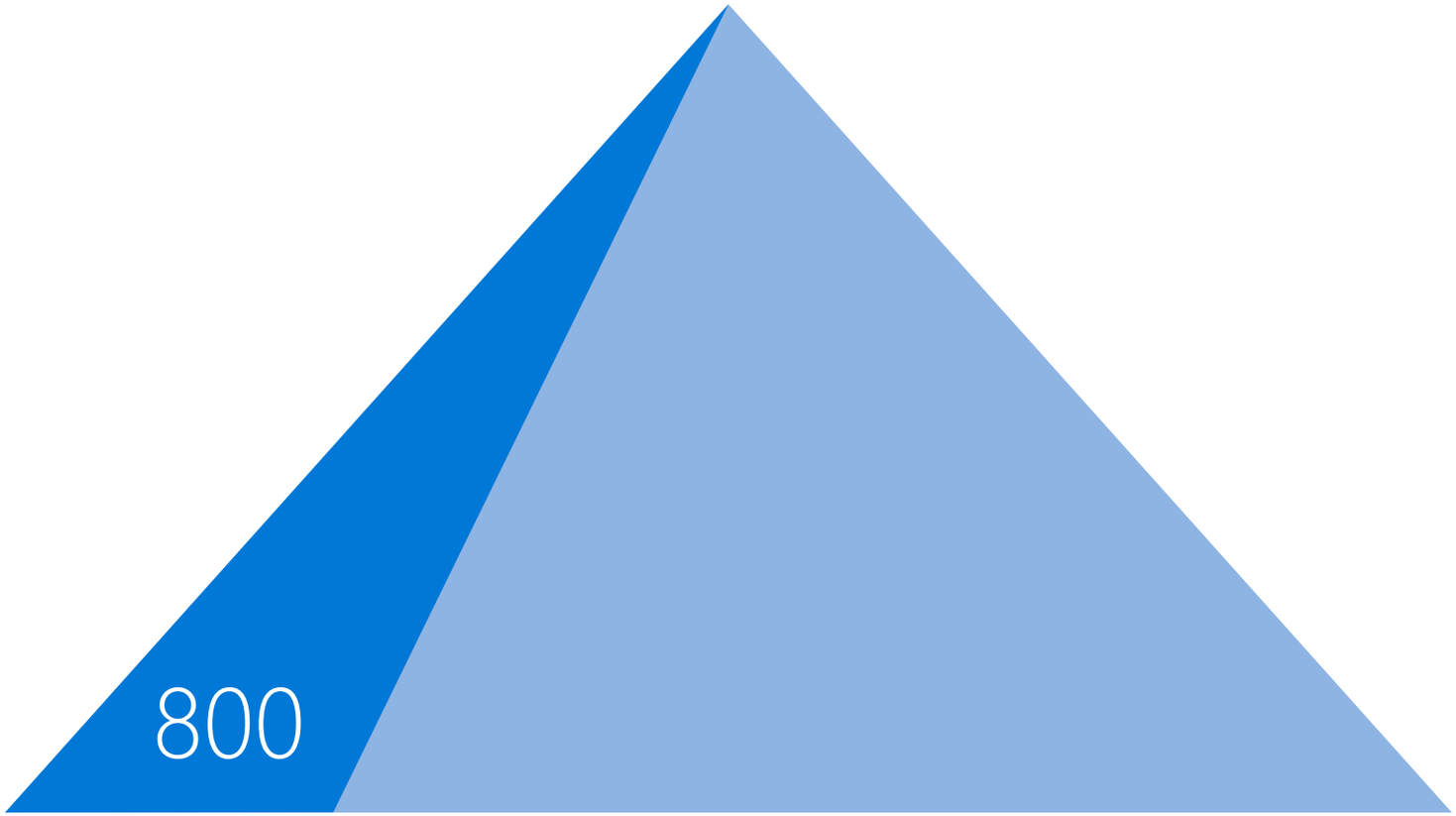
Paul Hacker

Came to Microsoft in November 2017
DevOps Architect - DevOps Customer Advisory Team (CAT)
Microsoft MVP in Dev Tools 14 years
Conference speaker



The Developer Division at Microsoft

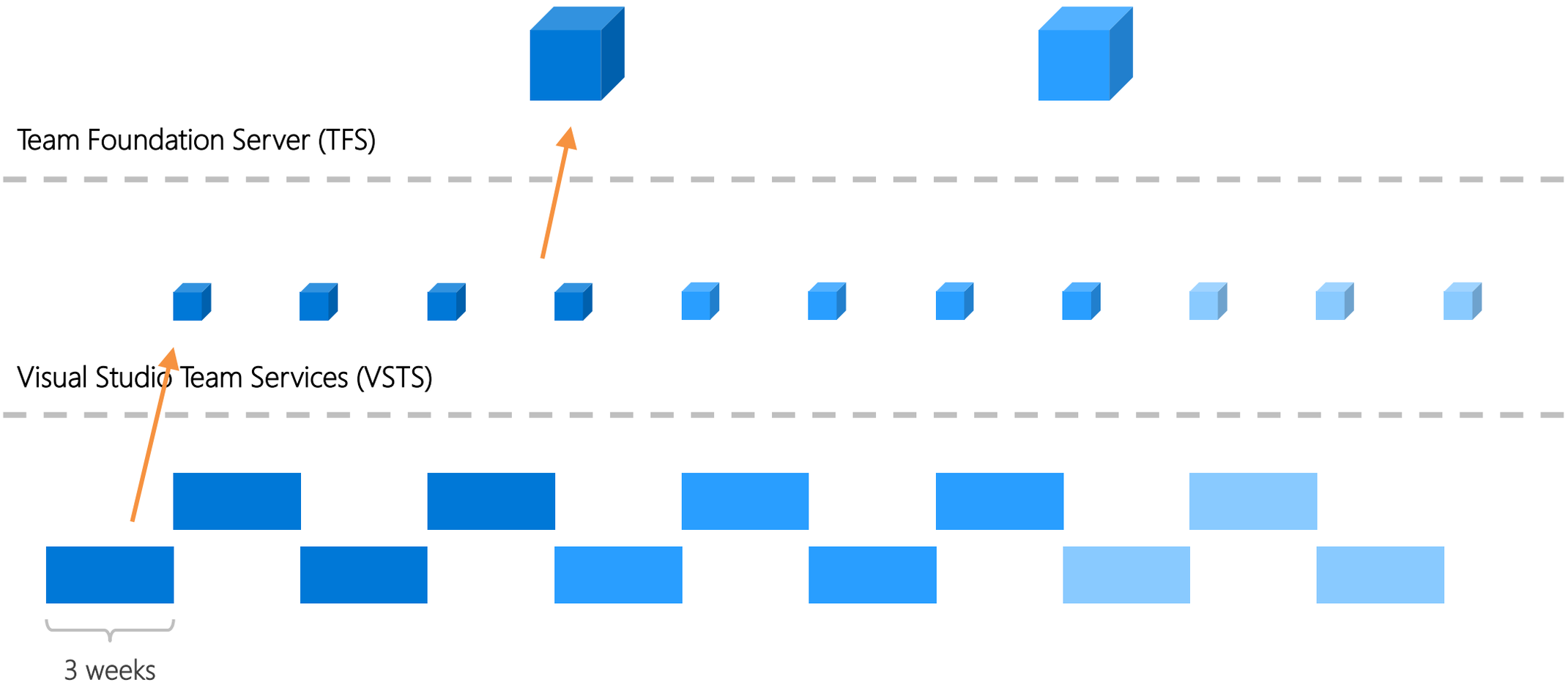
Microsoft



The VSTS team… spread out across 40 feature teams

Microsoft

How do we work?

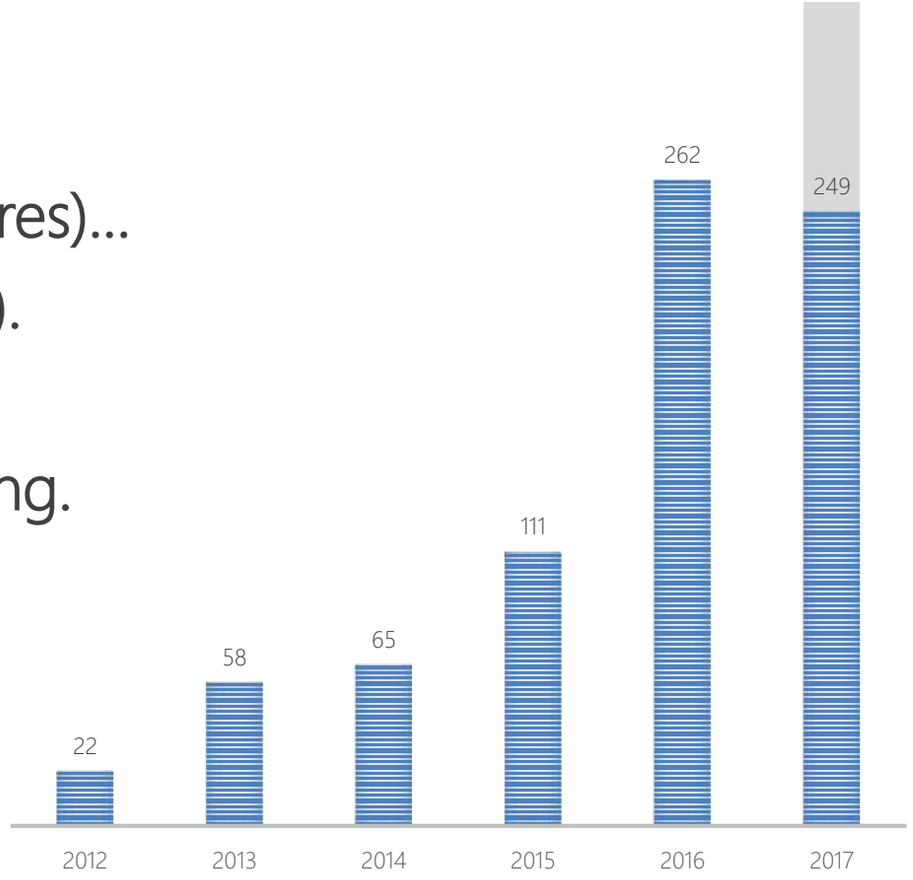


Microsoft

Features Delivered per Year

We are delivering value to customers and an increased velocity.

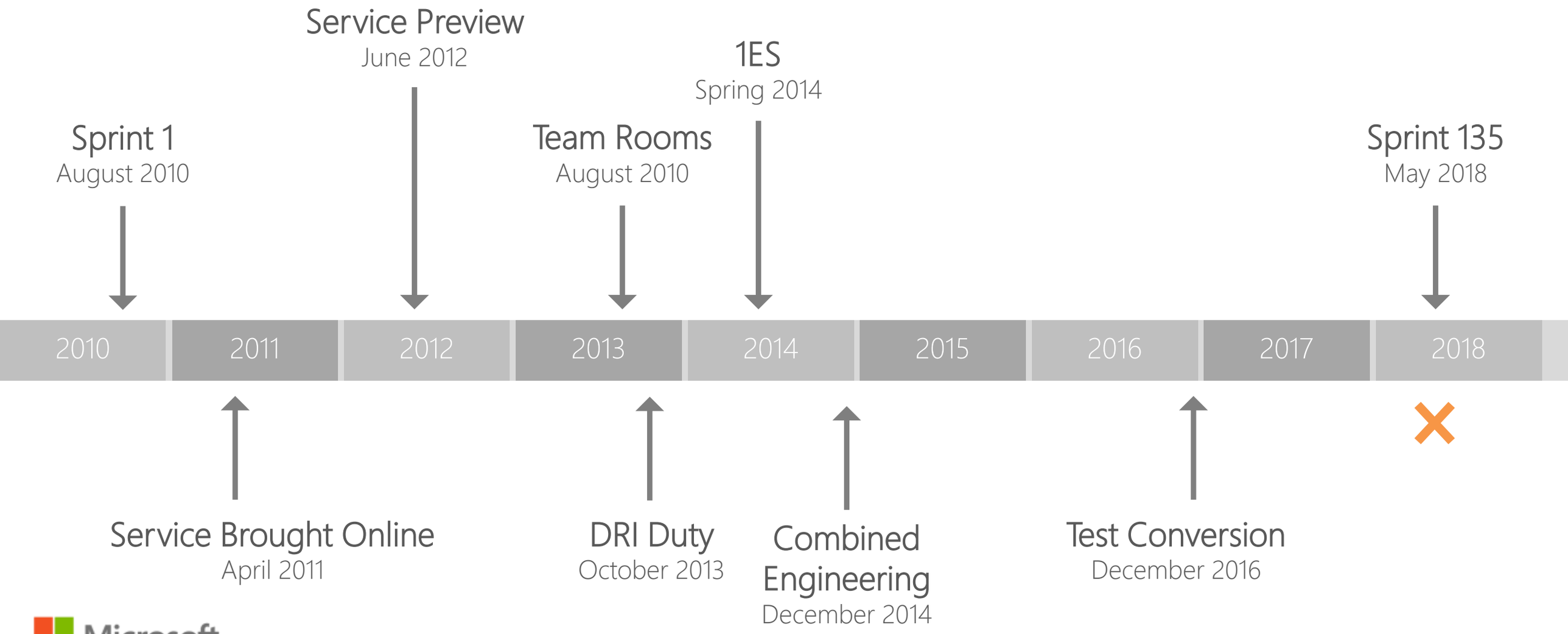
- More features in the 2016 calendar year (262 features)…
- Than the previous 4 years combined (256 features).
- 346 features in the 2017 calendar year.
- ~211 features in the 2018 calendar year and counting.



https://www.visualstudio.com/en-us/articles/news/features-timeline

Microsoft

The Journey



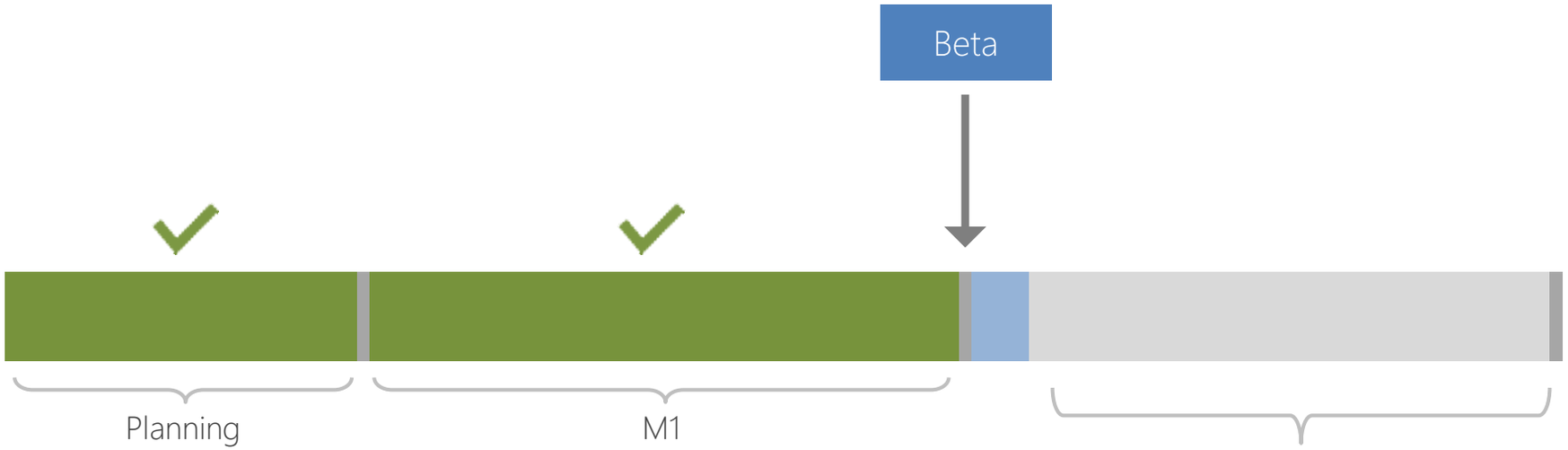
Microsoft

What did it look like before?

Before

The OLD way

Customer feedback – we should change the way a feature works. We didn't get it *quite* right…

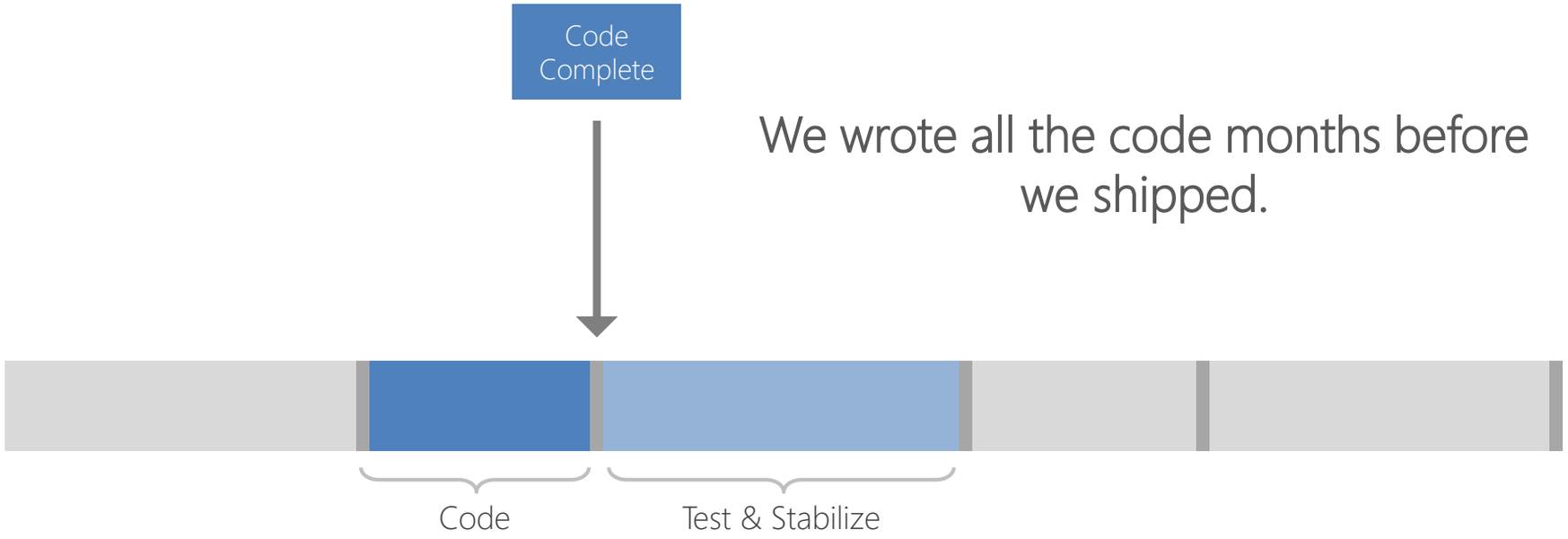


… but we're booked solid already.

Microsoft

Before

The OLD way



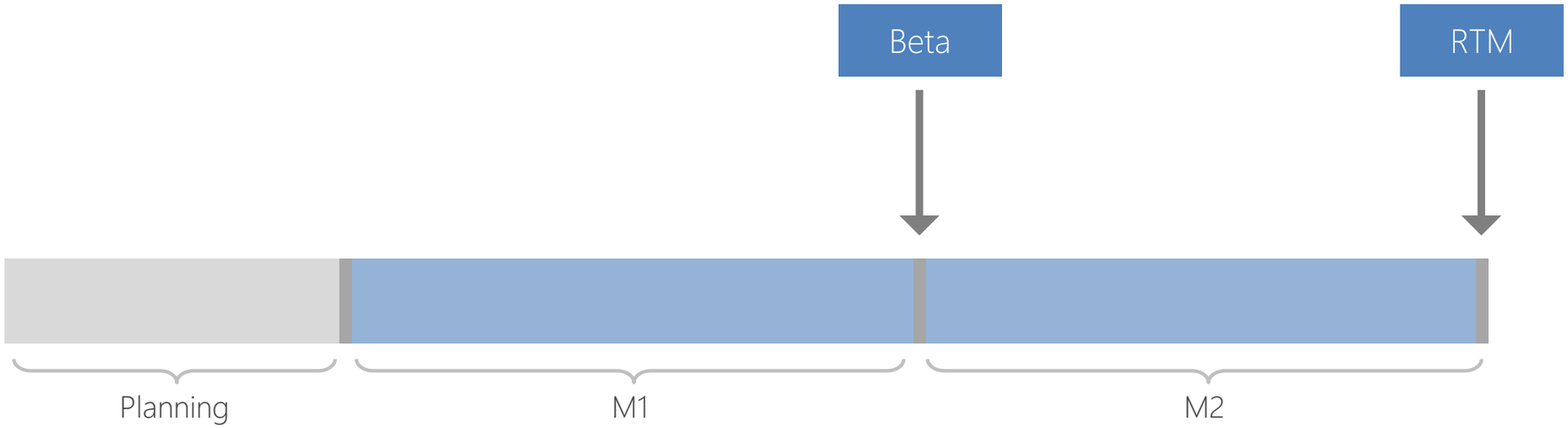
We wrote all the code months before we shipped.

Microsoft

Before

The OLD way

We had a perfect schedule and knew exactly when it would be ready!

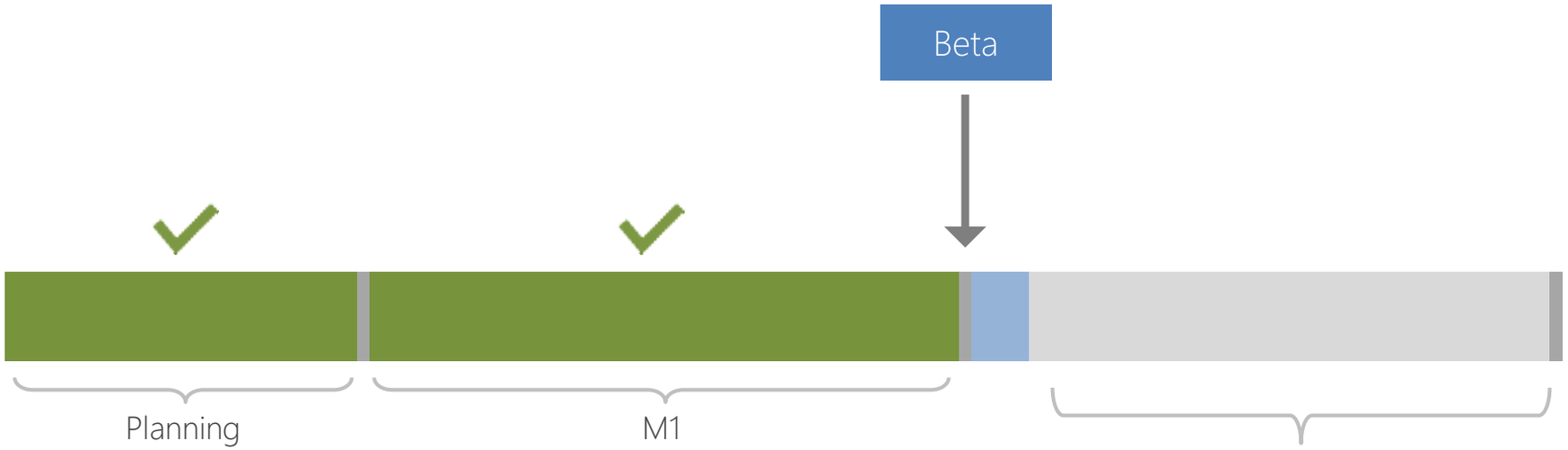


Microsoft

Before

The OLD way

Customer feedback – we should change the way a feature works. We didn't get it *quite* right…



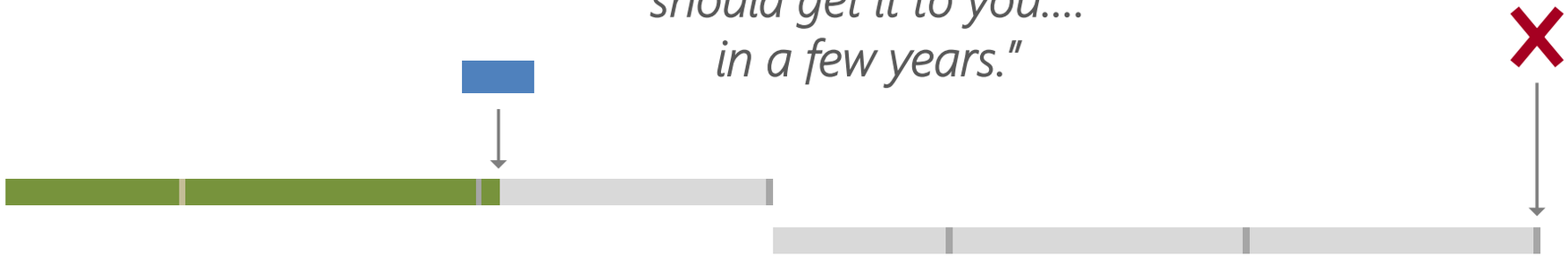
… but we're booked solid already.

Microsoft

Before

The OLD way

"Great feedback. Thanks! We'll take a look in planning for the next release. We should get it to you….
in a few years."

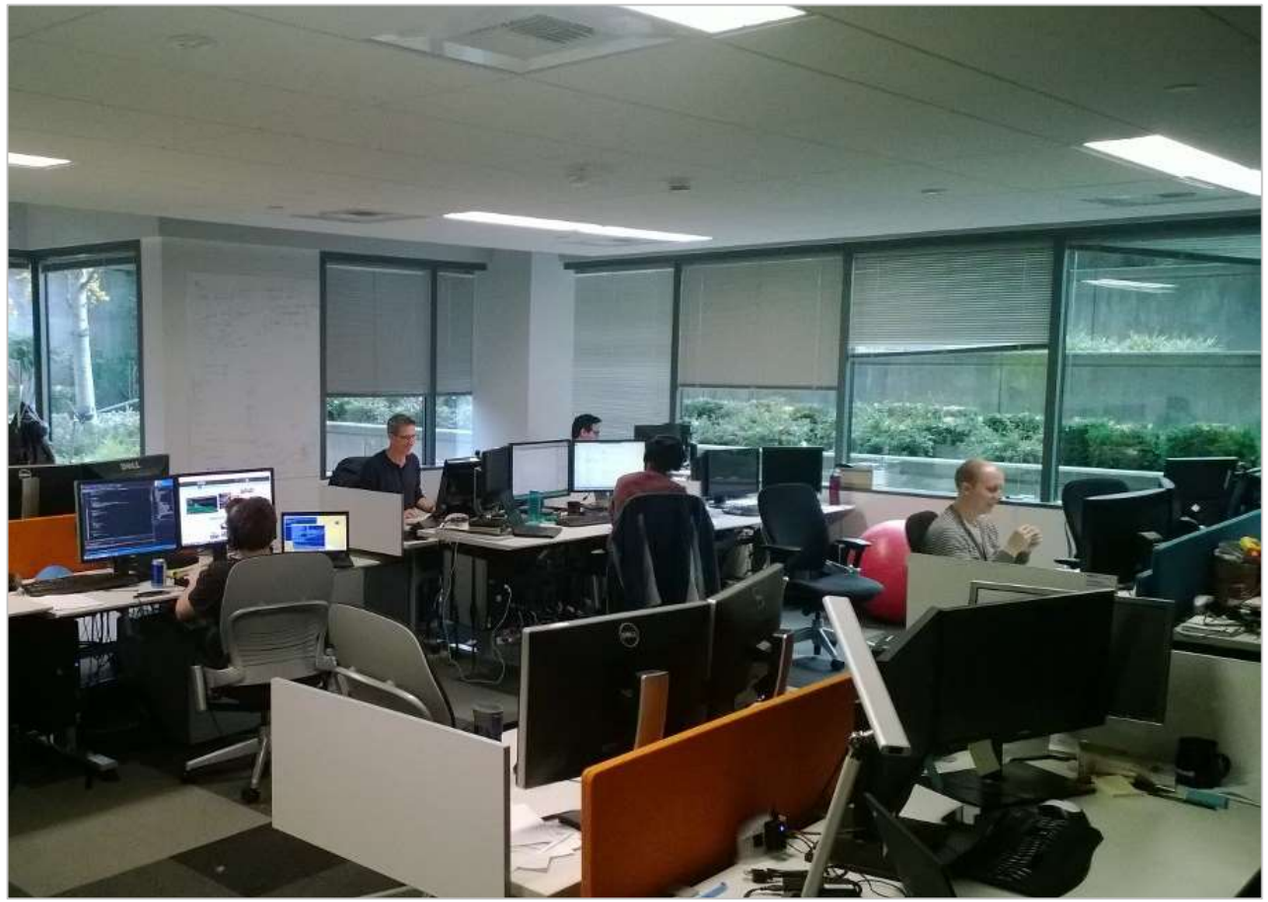


Microsoft

Cha Cha Cha Changes

Teams

Cross discipline
10-12 people
Self managing
Clear charter and goals
Intact for 12-18 months
Physical team rooms
Own features in production
Own deployment of features



Self Forming Teams

We have chosen to re-think the charter and make-up of our teams at strategic checkpoints. This happen every 18 months (or so).

The “Yellow Sticky” exercise:

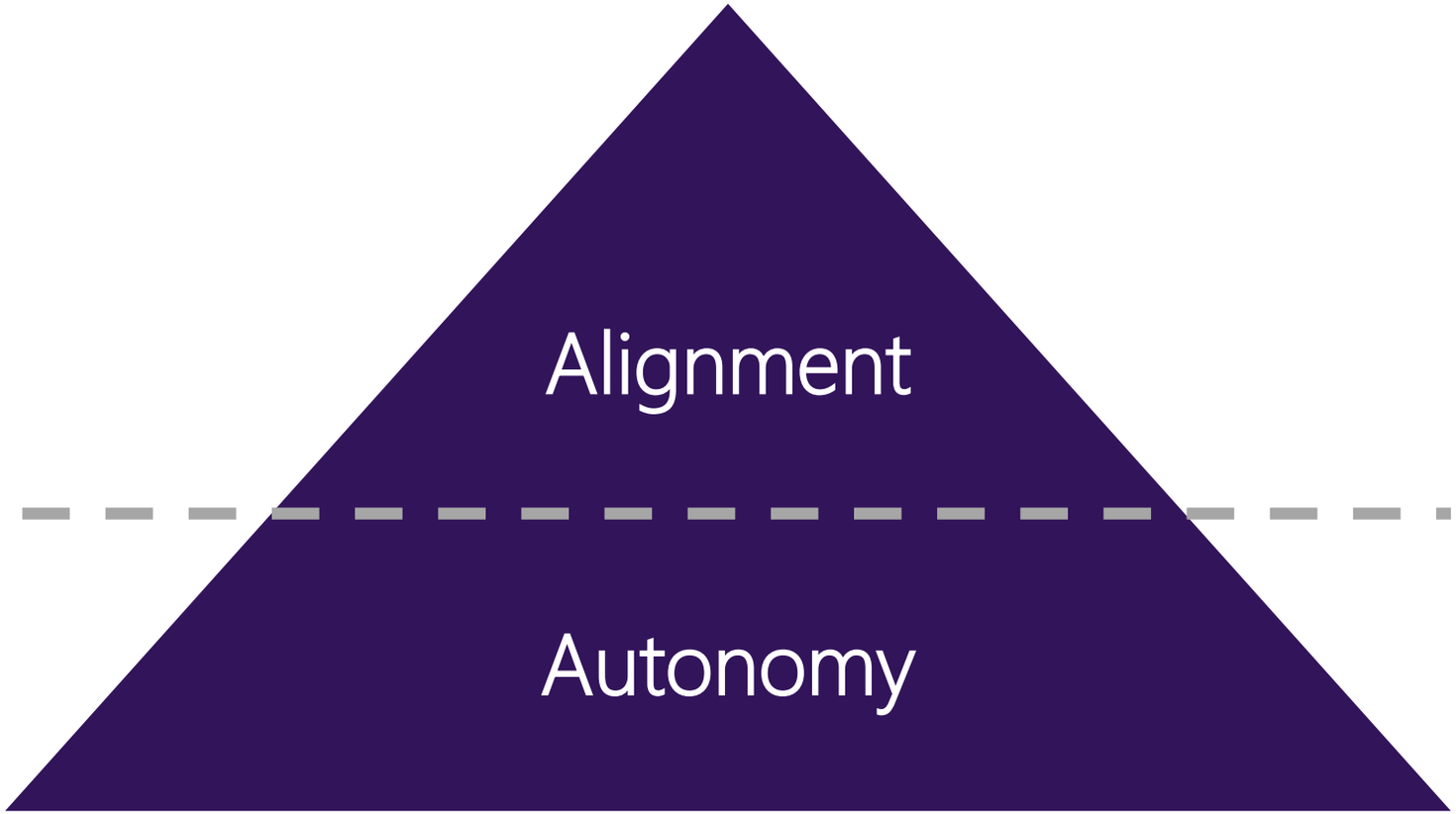
- Autonomy: Let team choose what they want to work on.
- Alignment: Ensure we’ve got the right balance across teams.

http://aka.ms/selfformingteams

Blog post from Brian Harry, Corporate Vice President, describing the process

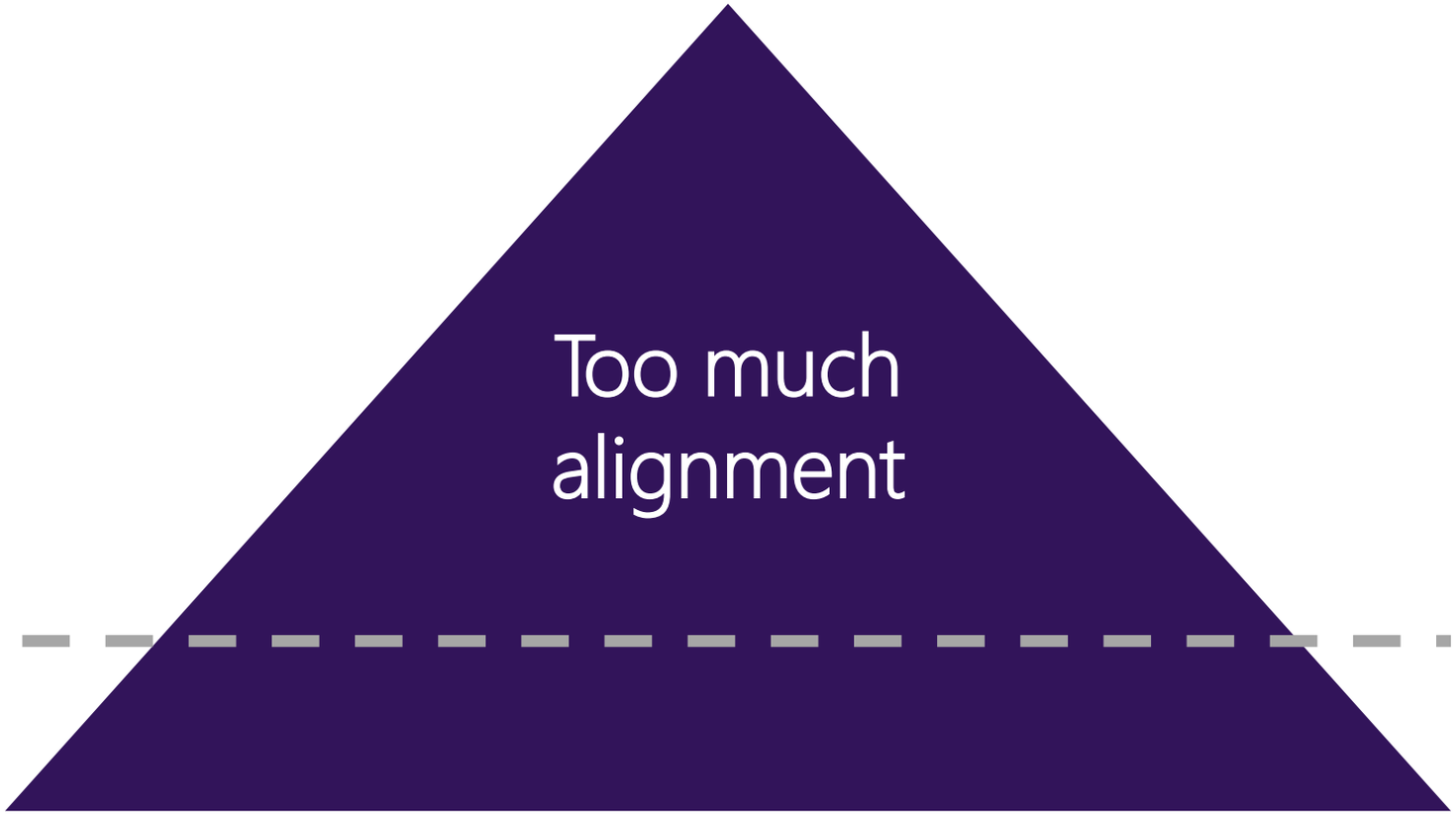
Microsoft

Aligned Autonomy



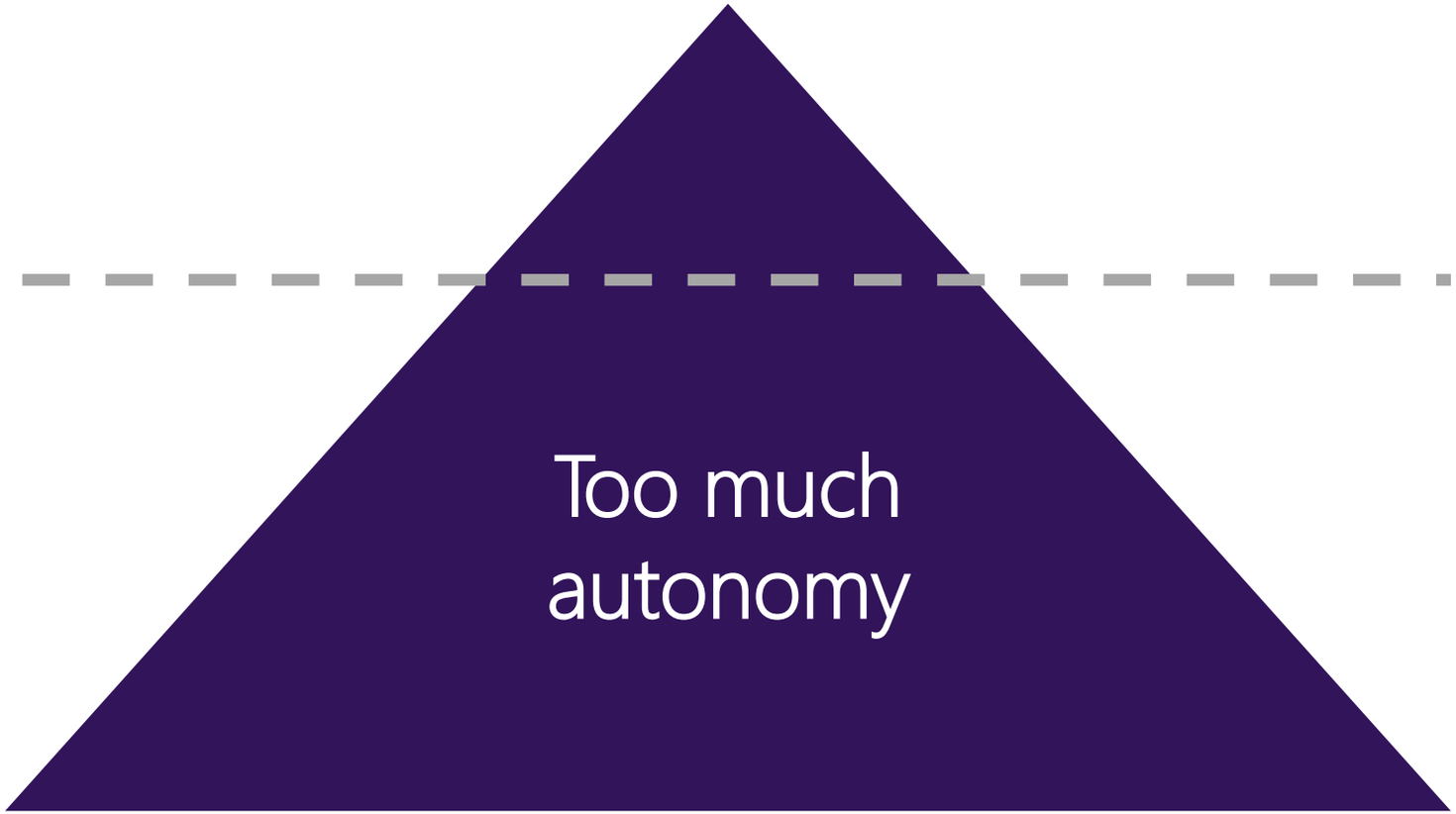
Microsoft

Aligned Autonomy



Microsoft

Aligned Autonomy

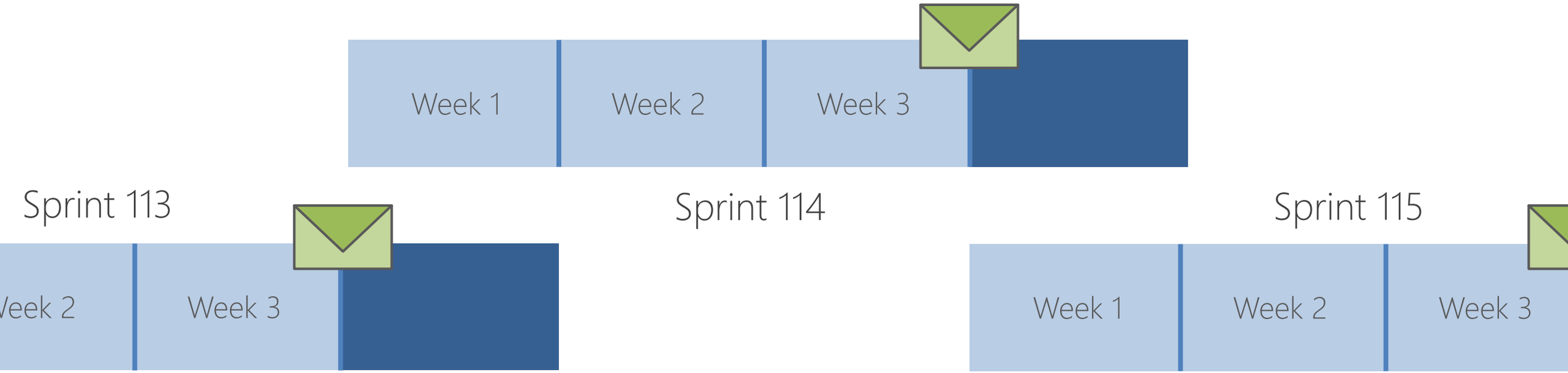


Microsoft

How do teams stay connected?

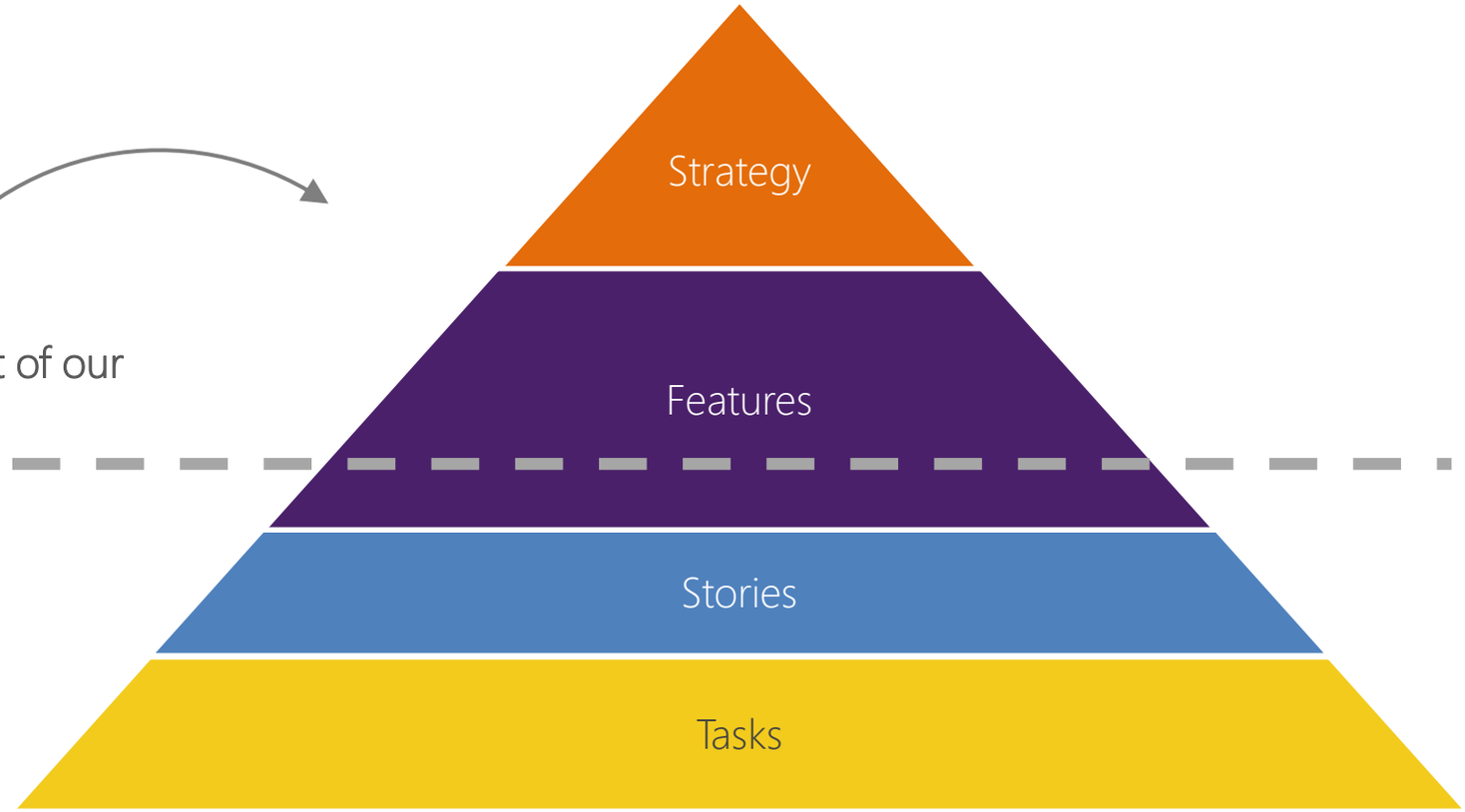
Sprint Mails

At the end of a sprint, all teams send a "sprint mail" ✉... communicating what they've accomplished in the sprint, and what they're planning to accomplish in the next sprint.



Microsoft

Aligned Autonomy

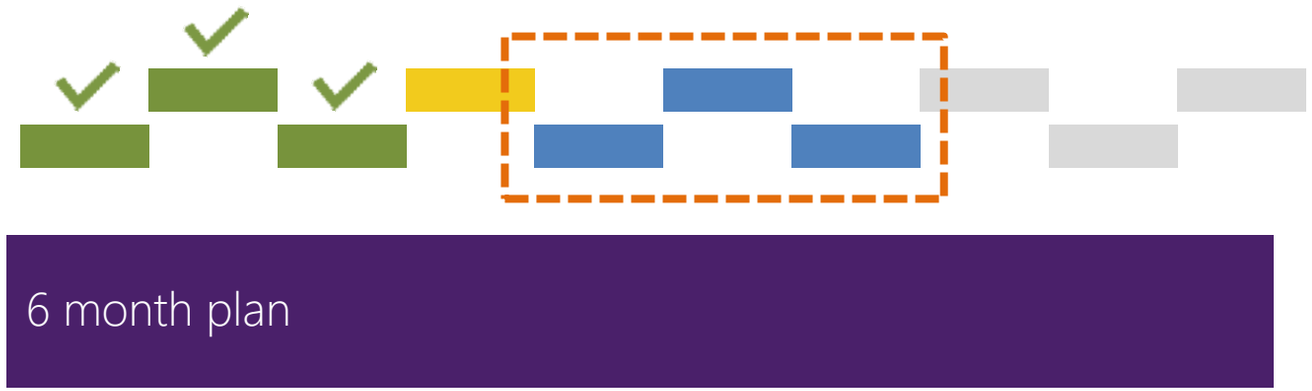


Alignment
The big picture in light of our business goals

Autonomy
The detail about what we'll deliver to achieve our business goals

Microsoft

Quarterly Feature Team Chats



Each team comes in and reviews with leadership three things:

1. What is the plan for the next 3-sprints?
2. Is the team healthy?
3. Any risks or issues to highlight?

Microsoft

Bug Cap

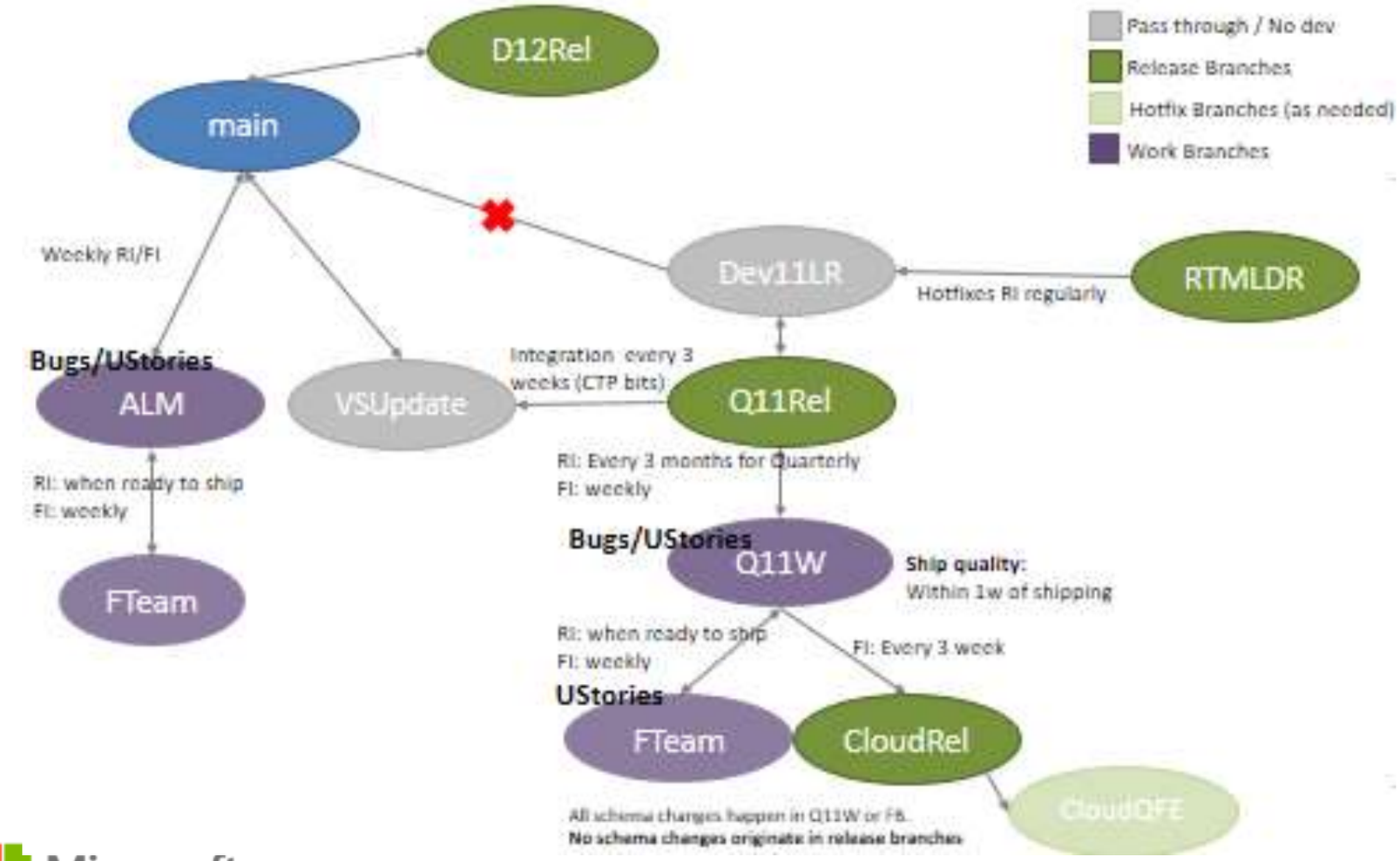
We all follow a simple rule we call the “Bug Cap”:

engineers on your team x 5 = ?

Microsoft

Traditional branch structure

The OLD way



Deep branch hierarchy

Creates merge and integration debt

Significant costs to code flow

Complex logistics Engineers must understand

Microsoft

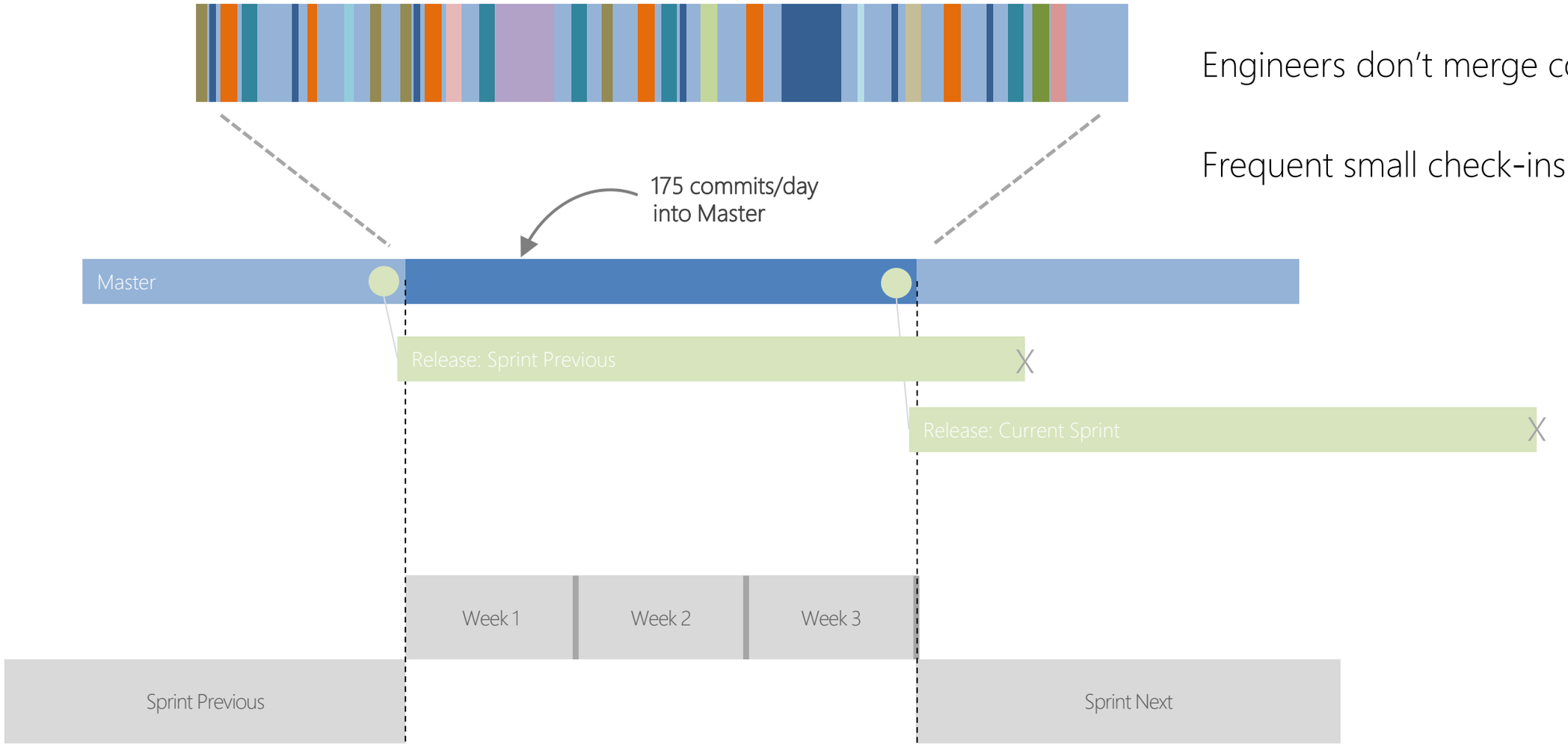
Work out of master

Source in a single git repo

All code flows to master

Engineers don’t merge code they didn’t write

Frequent small check-ins



Microsoft

Testing circa 2010 – arrival of the Cloud Services

New constraints and requirements

Faster cadence, even faster cadence, and more

Lack of customer validation through Beta, RC etc.

Micro-services deployed independently

High availability, no downtime deployments

Initial response and approach

Do the traditional waterfall dev/test model but faster

Pushed for faster automation

Test Selection techniques as a way of survival

Testing in Cloud cadence – problems

New problems emerged, old ones exacerbated

Testing became major bottleneck – we reached a breaking point

Trains didn't run on time

Lack of accountability on the Developers – no real incentive to change

High frustration among SDETs. Major retention issues.

Our model was broken

Bing, being first major cloud service at Microsoft, noticed it first

Over next few years, every team at Microsoft moved to the Cloud and changed their testing approach

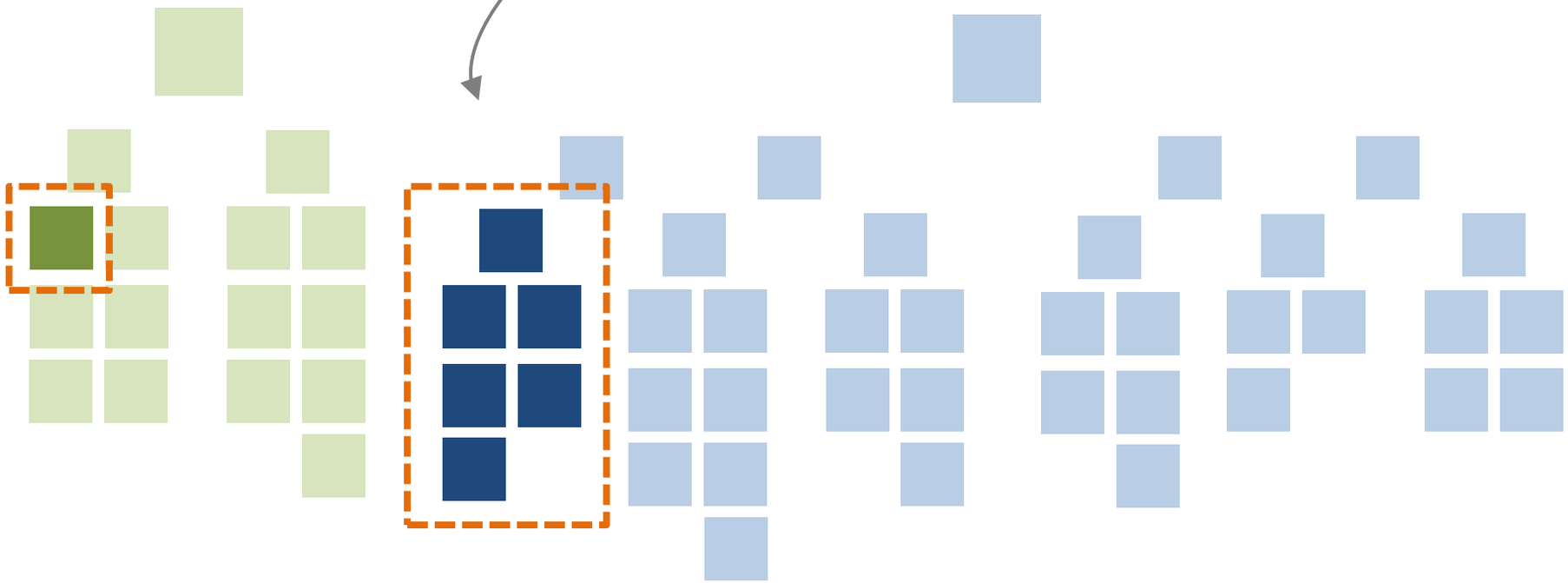
Quality ownership

Microsoft

Teams

Program Management is responsible for:
WHAT we're building, and
WHY we're building it

Engineering is responsible for
HOW we're building it, and that
we're building it with QUALITY



Microsoft

Shift-Left

Master is always shippable

Microsoft

Our problems: Sept '14

Tests took too long

Over 22 hours for nightly run and 2 days for the full run

Tests failed frequently

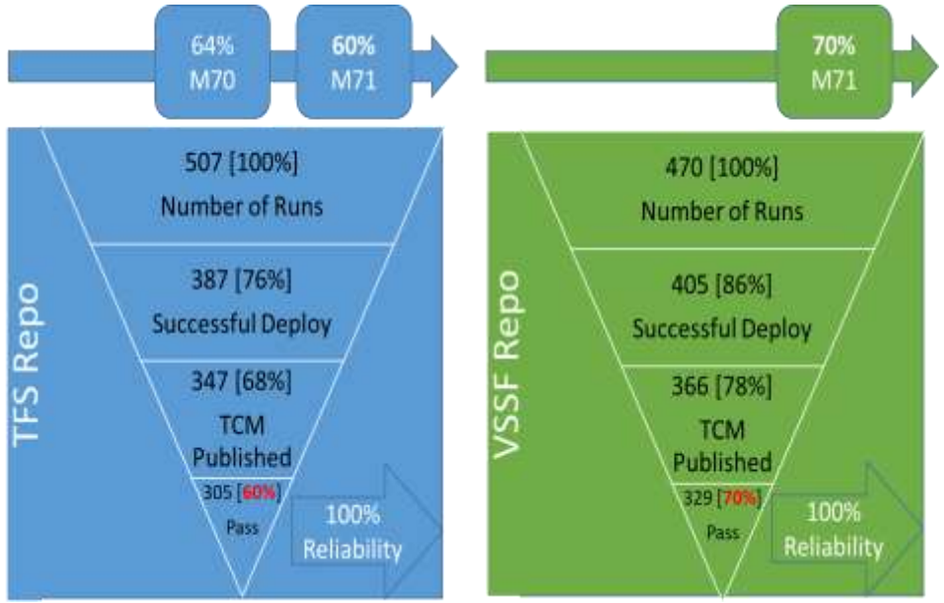
Only ~60% of P0 runs passed 100%; Each NAR suite had many failures

Quality signal unreliable in Master

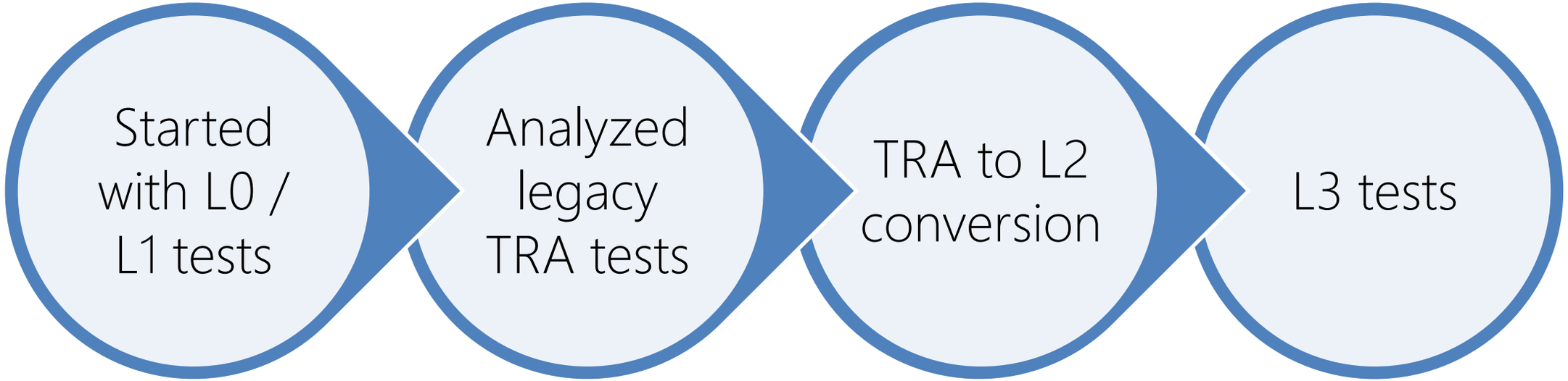
Test failure analysis was too costly

Huge cliff to reach Release quality at the end of Sprint

Took days to sift through failures before deployment could start



Shifting the test portfolio journey



- Made it easy to author and execute high quality L0/L1 tests
- Stopped creating new TRA tests as much as possible

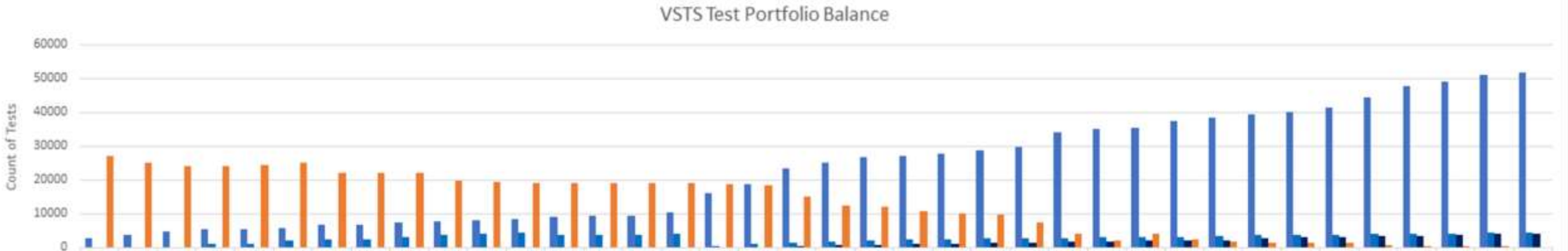
- Tests that can be deleted
- Tests that can move to L0/L1
- Tests that will move to VSSF Test SDK
- On-prem tests we expect to maintain in lights-on mode

- Test Arch v-team re-wrote L2 test framework
- Top-down push from management with org wide scorecard

- Building these now

Microsoft

Test portfolio over time



	M78	M79	M80	M81	M82	M83	M84	M85	M86	M87	M88	M89	M90	M91	M92	M93	M95	M98	M101	M102	M103	M104	M105	M106	M107	M108	M109	M110	M111	M112	M113	M114	M115	M116	M117	M118	M119	M120
L0 Tests	2723	3744	4582	5297	5381	5831	6613	6684	7232	7668	7983	8486	8908	9385	9385	10286	15968	18598	23272	24945	26594	27109	27807	28709	29787	34020	34983	35405	37404	38467	39412	40000	41452	44430	47909	49007	51113	51775
L1 Tests				888	958	2130	2241	2525	3095	3753	4046	4215	3723	3786	3786	4166	479	1153	1465	1774	2177	2289	2438	2597	2735	2813	2935	2966	3127	3528	3592	3750	3849	3928	4023	4187	4310	4353
L2 Tests																			430	590	652	980	1102	1200	1410	1693	1858	1918	2068	2190	2585	2883	3021	3228	3469	3719	3917	3969
TRA Tests	27054	24941	24135	24097	24381	25212	22198	21981	21908	19577	19529	19124	19098	19041	19041	18937	18749	18252	14983	12449	11959	10554	10070	9686	7500	4155	2199	3886	2254	1593	1519	1386	1386	755	181	136	6	0

Microsoft

What we track

		Livesite Health				Engineering Debt					
		# LSI repair WIs older than 2 sprints	% LSI repair WIs completed within 2 sprints	# of DTSs over SLA	% Perf Scenarios within SLA	# of P0 or P1 bugs older than 21 days	# of Active bugs per engineer	# of Security WIs older than 21 days	Accessibility Bugs	# of Stale Feedback Tickets	# of Active Test Reliability Bugs
Agile	Modern WIT	0	100	0		0	1	0	2	2	0
	WIT X	0	100	0	0	0	7	0	0	10	2
	WIT PI	0	100	0		0	4	0	2	4	0
	WIT IQ	0	100	1	20	0	4	1	0	3	1
	Engineering Productivity	0	N/A	0		0	4	0	0	0	1
Code Sharing	Package Experience	0	N/A	0	33	0	5	1	1	1	0
	Package Platform	3	50	0		1	3	2	0	1	0
	Component Governance	1	50	0		0	6	0	0	0	0
Continuous Integration	Build	0	100	2	0	0	3	1	0	30	0
	Talos	1	90	2		0	3	1	0	15	0
	CIX	0	N/A	2		0	3	0	1	15	0
enSPIRe	Keros	0	N/A	0		0	8	0	1	0	0
	Subscriptions	1	50	28	40	0	7	5	0	1	0
	Enterprise Requirements	0	N/A	0		1	3	5	0	0	0
	Service Insights	0	100	0		0	2	1	0	0	0
Insights	Compass	0	N/A	0	50	0	3	0	3	9	0
	Atlas	6	82	1		0	7	0	0	2	0

Live Site Health/Debt

Time to Detect, Time To Mitigate

Incident prevention items

Aging live site problems

Customer support metrics (SLA, MPI, top drivers)

Engineering Health/Debt

Bug cap per engineer

Aging bugs in important categories

Pass rate & coverage

Velocity

Time to build

Time to self test

Time to deploy

Time to learn (Telemetry pipe)

Feature Flags

Controlling exposure in the cloud

Microsoft

What do feature flags give us?

Decouple deployment and exposure

Flags provide runtime control down to individual user

Change without redeployment

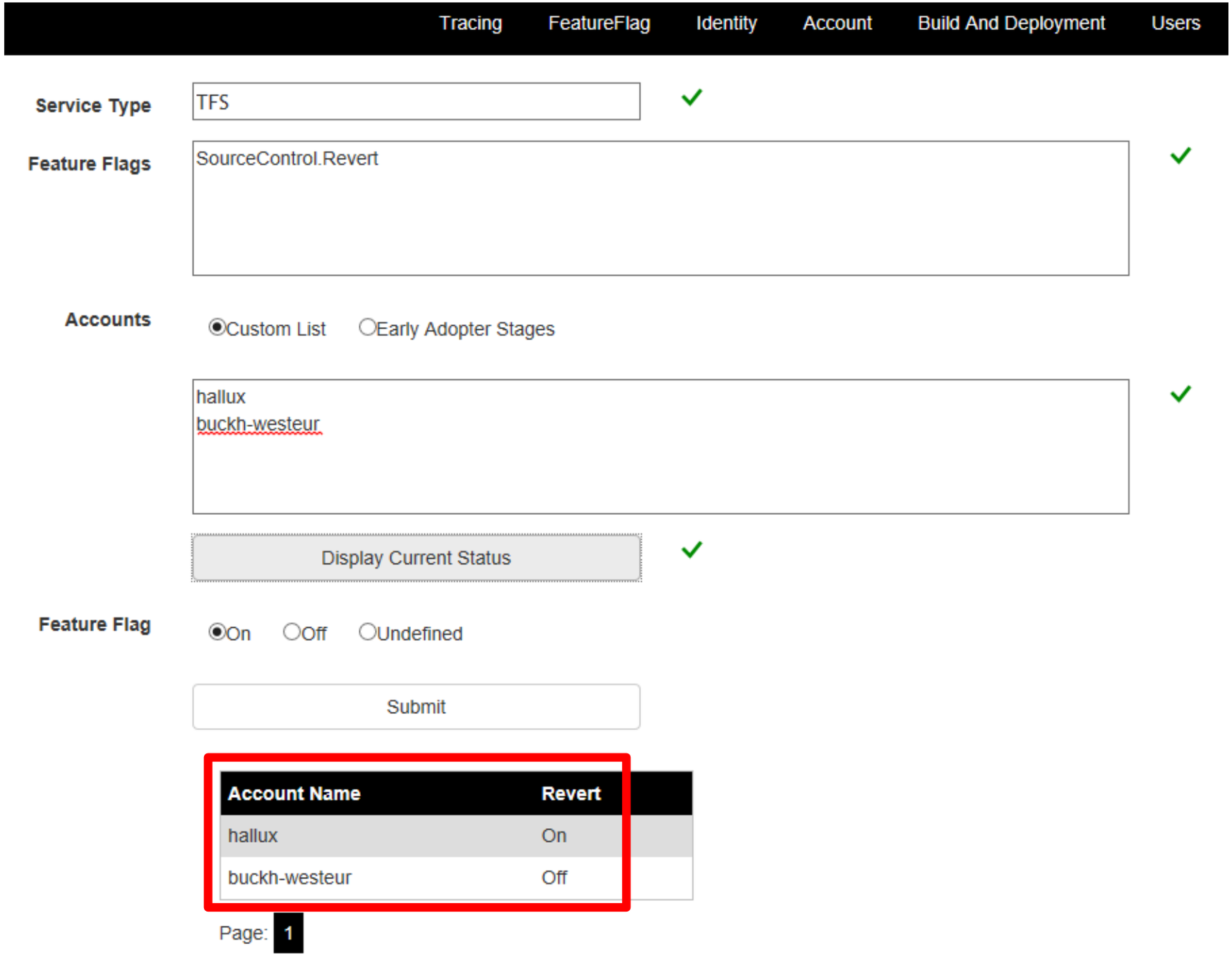
Controlled via PowerShell or web UI

Support early feedback, experimentation

Quick off switch

Control

- PowerShell
  - Get-FeatureFlag
  - Set-FeatureFlag
- Web UI



Microsoft

Deployment Practices

Microsoft

Early Principles

- The same tools we use to deploy to production we use in dev and test environments
- The quality signals we look at to green light deployments are tracked constantly every day
- Deployments take zero down time
- Deployments happen during working hours

Are We Ready To Deploy?

Check for blocking bugs

Check test results

Choose a good build

Release Branch Runs - Default

Environments								
Sps.SelfTest	100%	100%	100%	100%	100%	100%	100%	
Sps.SelfHost	100%	100%	100%	100%	100%	100%		
Tfs.SelfHost Set 1	100%	100%	100%	100%	100%	100%		
Tfs.SelfHost Set 2	100%	100%	100%	100%	98.69%	100%		
Tfs.SelfTest	100%	100%	100%	100%	100%	100%		
Tfs.Deploy		100%	100%	100%		100%		
TfsOnPrem.SelfHost	100%	100%	100%	100%	100%	100%		
TfsOnPrem.SelfTest	100%	100%	100%	100%	100%	100%		

Microsoft

Safe Deployment

Microsoft

What is Safe Deployment?

- Deploy changes to risk tolerant customers first, progressively roll out to larger and larger sets of customers
- Automated health checks and roll back

Deployment Rings

Ring	Purpose	Customer type	Data center
0	Surface most of the customer-impacting bugs introduced by the deployment	Internal only, high tolerance for risk and bugs	US West Central
1	Surface bugs in areas that we do not dogfood	Customers using a breadth of the product, especially areas we do not dogfood (TFVC, hosted build, etc). Should be in a US time zone.	A small data center
2	Surface scale-related issues	Public accounts. Ideally free accounts, using a diverse set of the features available.	A medium to large US data center
3	Surface scale issues common in internal accounts and international related issues	Large internal accounts European accounts	Internal data center and a European data center
4	Update the remaining scale units	Everyone else	All the rest

Wrap-Up

Microsoft

Before	After
4-6 month milestones	3-week sprints
Horizontal teams	Vertical teams
Personal offices	Team rooms
Long planning cycles	Continual Planning & Learning
PM, Dev, Test	PM & Engineering
Yearly customer engagement	Continual customer engagement
Feature branches	Everyone in master
20+ person teams	8-12 person teams
Secret roadmap	Publicly shared roadmap
Bug debt	Zero debt
100 page spec documents	Specs in PPT
Private repositories	Open source
Deep organizational hierarchy	Flattened organization hierarchy
Success is a measure of install numbers	User satisfaction determines success
Features shipped once a year	Features shipped every sprint

Resources

https://aka.ms/devops

https://youtube.com/devopsatmicrosoft

https://aka.ms/devopslab

Thank you

Microsoft