DEALING WITH TECH DEBT

LeadDev New York

Oct 2025

Meri Williams, ChromeRose

@Geek_Manager

P&G

GDS

MOO

vodafone

monzo

M&S DIGITAL

Lead Dev

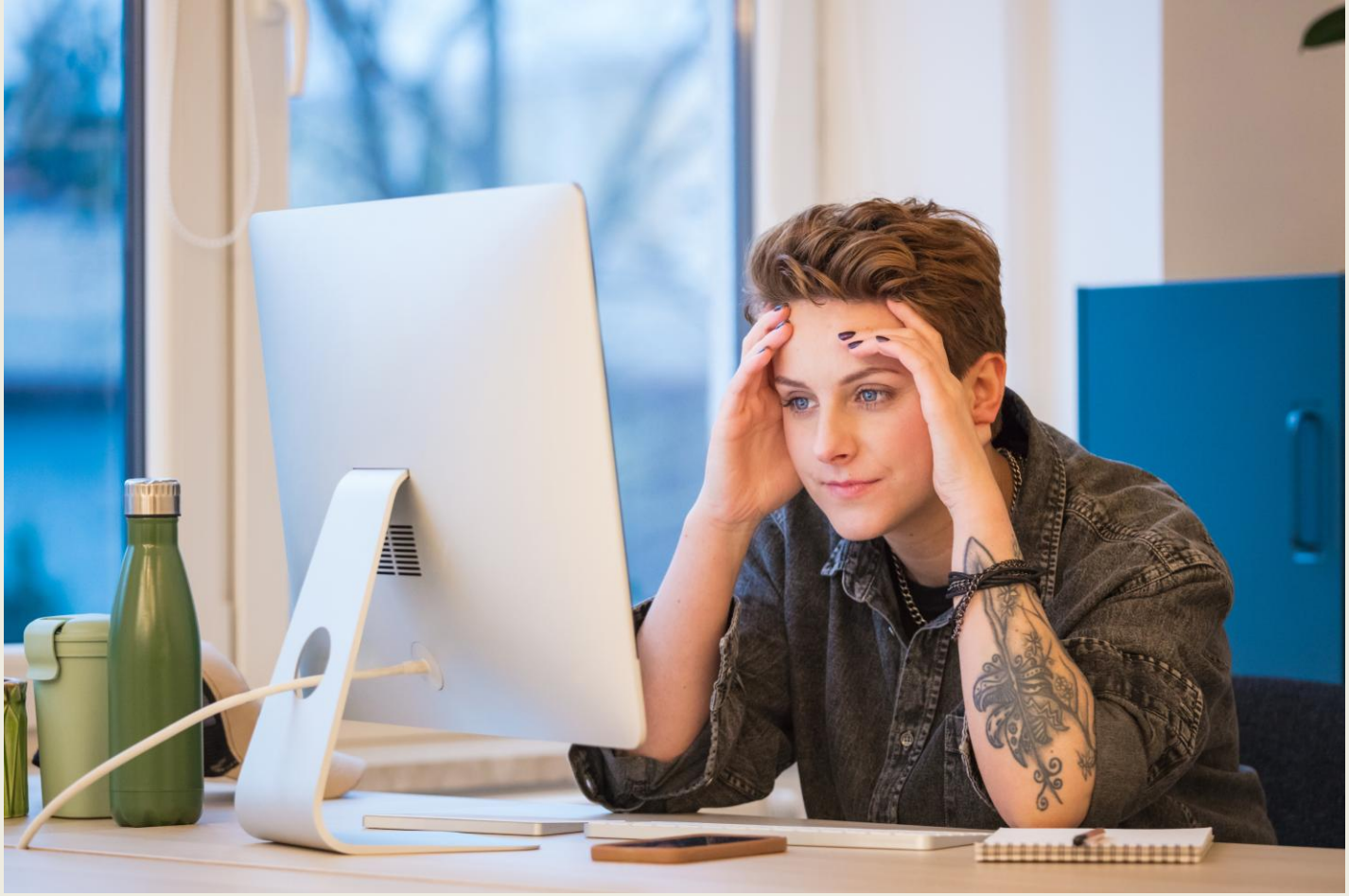
Meri Williams, ChromeRose

@Geek_Manager

WHAT IS TECH DEBT?

ANYTHING THAT MAKES CODE…

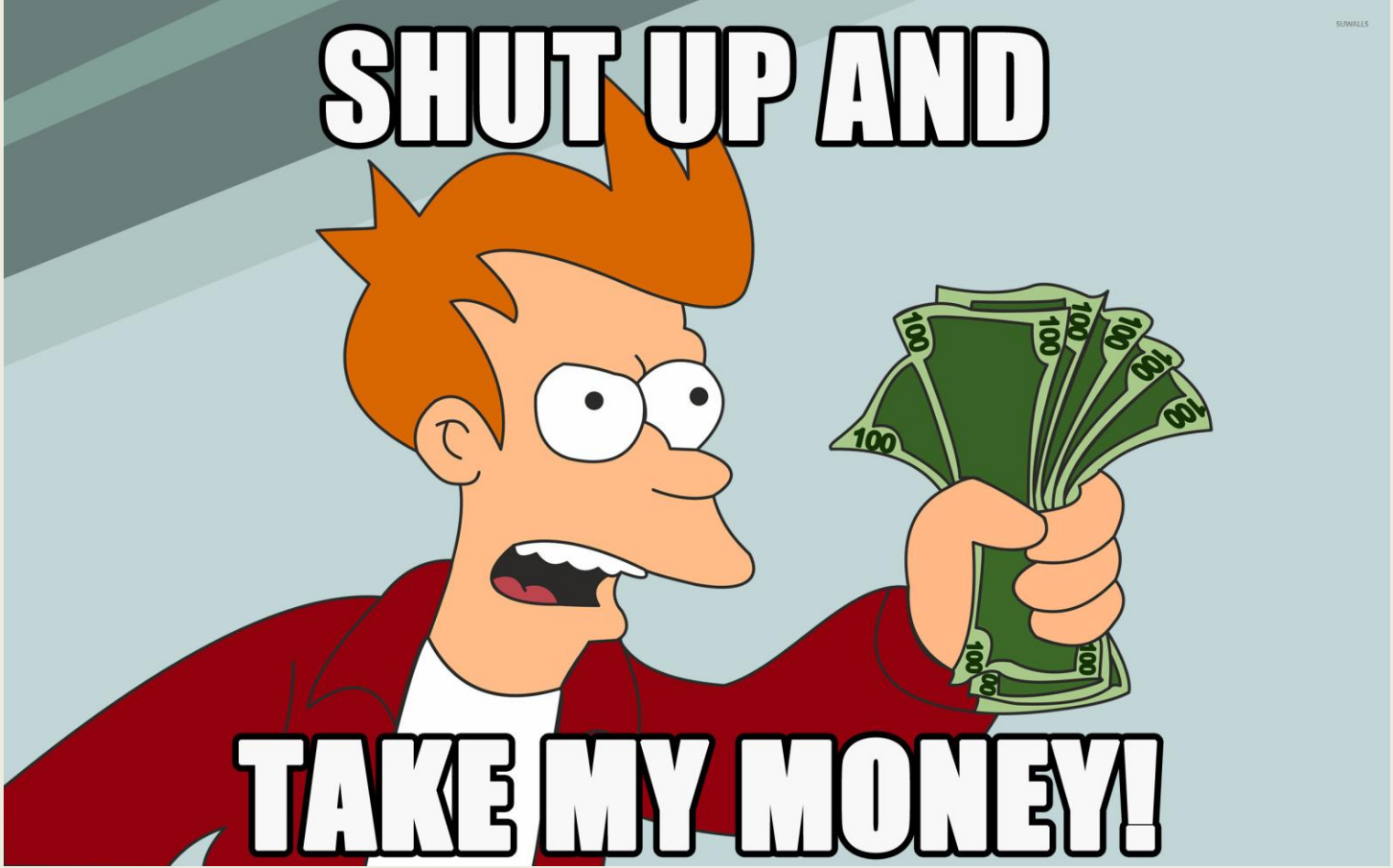
1) DIFFICULT TO UNDERSTAND
2) DIFFICULT TO SCALE
3) DIFFICULT TO CHANGE



WHY DOES TECH DEBT HAPPEN?

SAME REASON FINANCIAL DEBT HAPPENS.

WE NEED OR WANT SOMETHING NOW BUT CAN'T AFFORD TO PAY FOR IT.


SHUT UP AND
TAKE MY MONEY!

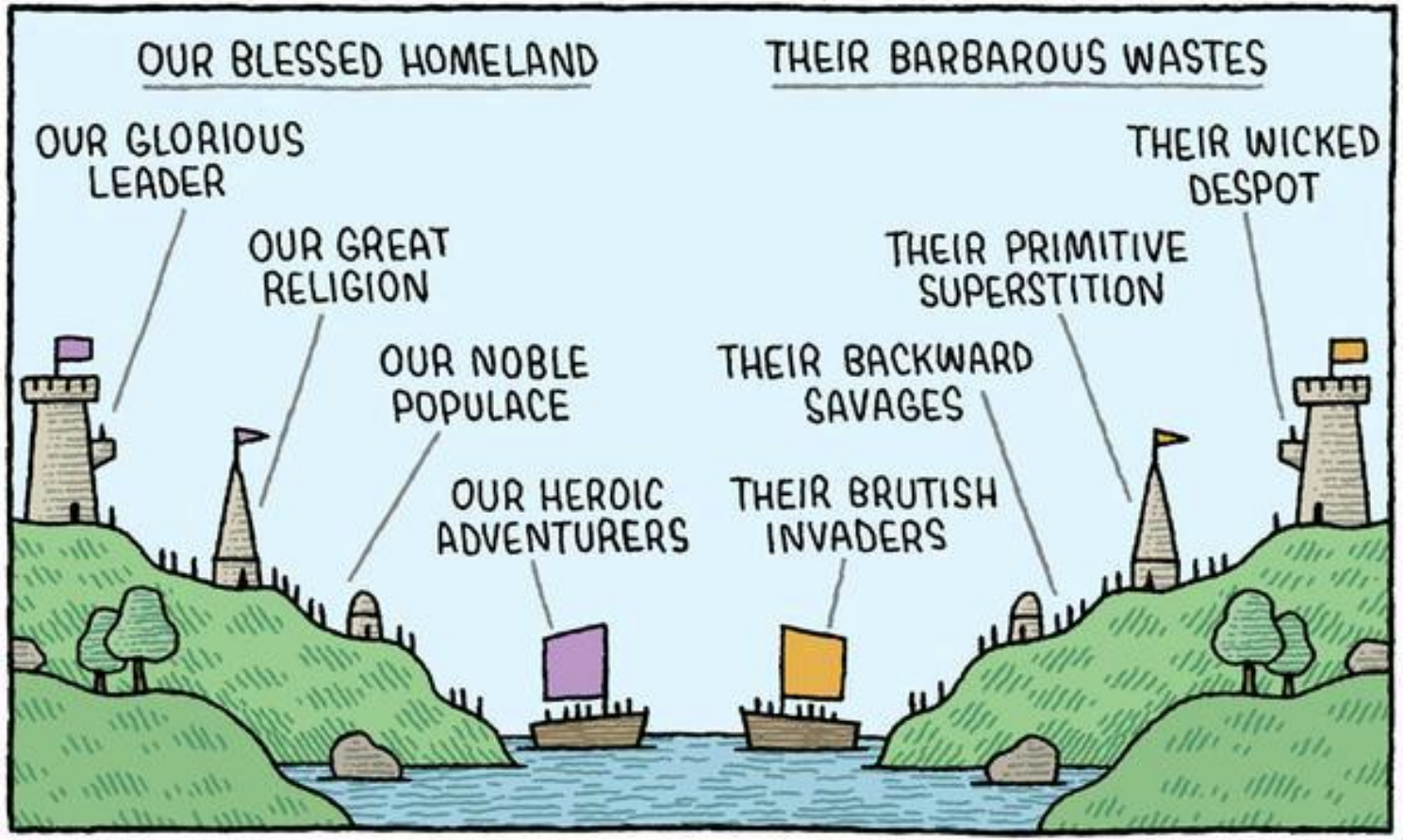
IT'S NOT EVIL OR STUPID TO BORROW AGAINST THE FUTURE.

HOWEVER, IT BECOMES A PROBLEM WHEN INTEREST PAYMENTS GET TOO LARGE …

OR WE PAY THE MORTGAGE ON A CREDIT CARD!

IT'S ALL JUST A MATTER OF PERSPECTIVE

DON'T ASSUME YOUR PREDECESSORS WERE STUPID!

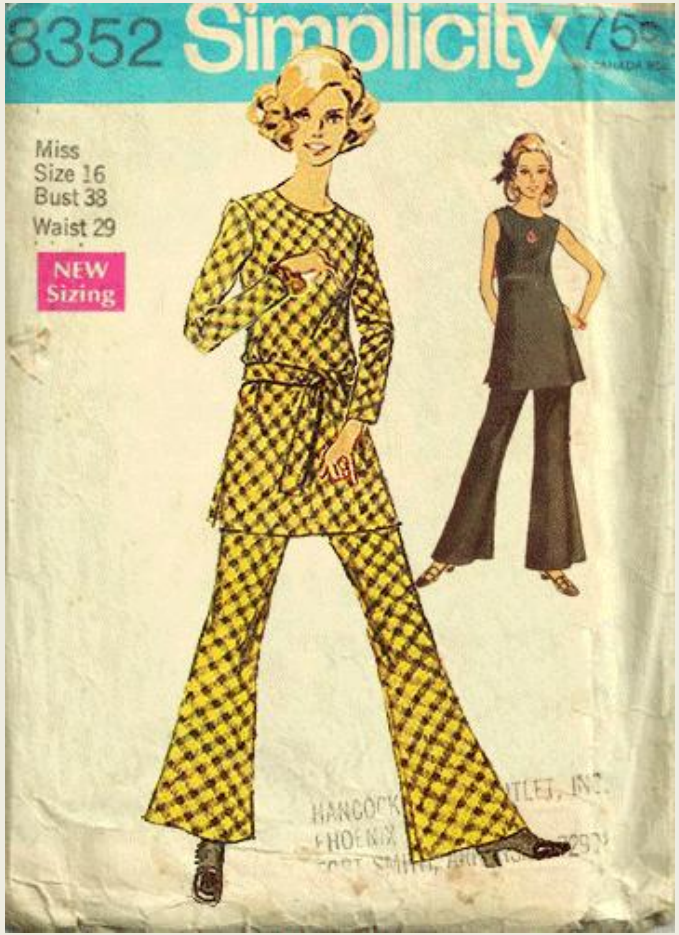


https://twitter.com/tomgauld/status/571994690289061888

(AS AN ASIDE, THIS IS WHY ARCHITECTURAL DECISION RECORDS ARE MAGICAL.

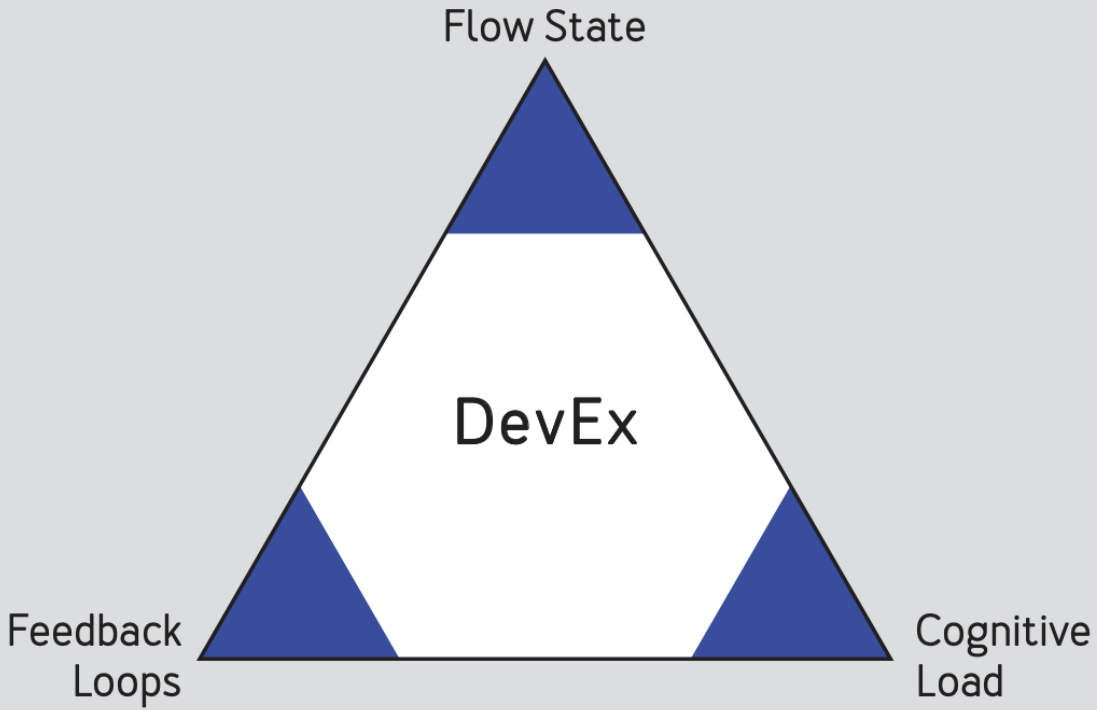
THEY GIVE YOU THE WHY AND THE CONTEXT TO CHOICES, SO AS CONTEXT CHANGES YOU CAN RE-EVALUATE ... BUT ALSO UNDERSTAND)


WE DON’T ALL HAVE THE BENEFIT
OF STARTING GREENFIELD



(IT'S NOT LEGACY
IT'S VINTAGE)

FIGURE 1: THREE CORE DIMENSIONS OF DEVELOPER EXPERIENCE



https://dl.acm.org/doi/pdf/10.1145/3595878

THREE THINGS MATTER FOR DEVELOPER EXPERIENCE:

- FEEDBACK LOOPS
- COGNITIVE LOAD
- FLOW STATE

THE WORST KIND OF TECH DEBT AFFECTS ALL THREE.

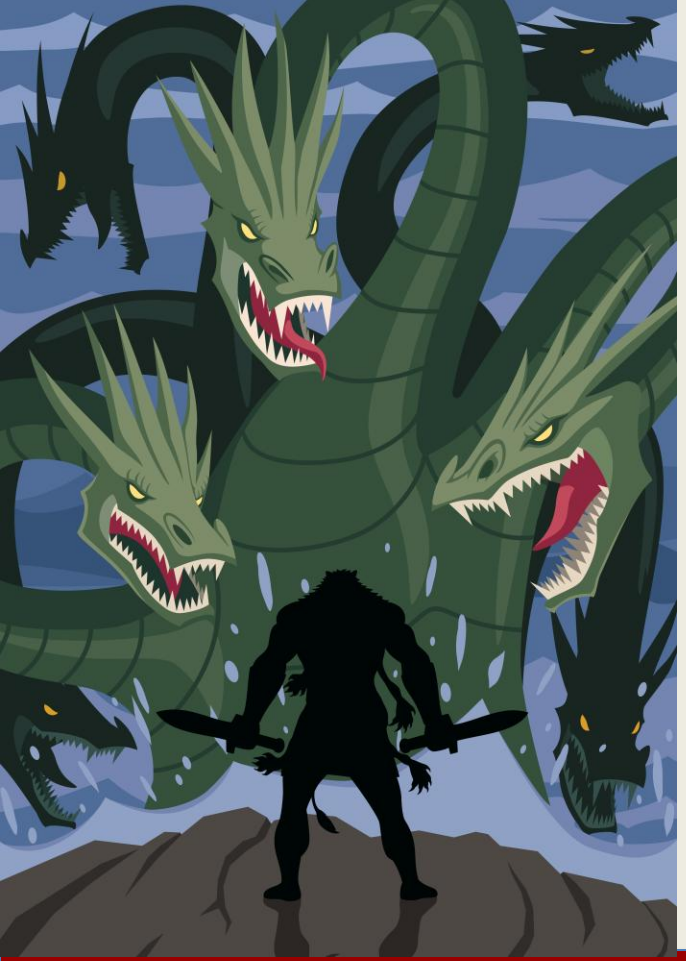
IT'S UNTESTED ← FEEDBACK LOOPS
IT'S COMPLEX OR COMPLICATED ← COGNITIVE LOAD
IT'S UNDOCUMENTED ← FLOW STATE

AT PLEO, WE HAD A CASE LIKE THIS.

THE VILLAIN OF THE STORY WAS DEIMOS, THE MONOLITH THAT IN MANY WAYS HAD MADE THE COMPANY SUCCEED IN THE EARLY DAYS (AFTER ALL, RUNNING CODE IS DELIVERING VALUE!)

BUT DEIMOS WAS

- DIFFICULT TO UNDERSTAND (”SPAGHETTI CODE”, POORLY STRUCTURED & DOCUMENTED)
- DIFFICULT TO SCALE (REACHING LIMITS)
- DIFFICULT TO CHANGE (CAUSING INCIDENTS)



WHEN I TOOK OVER AS CTO, THERE HAD ALREADY BEEN 2 OR 3 FAILED ATTEMPTS TO KILL DEIMOS.

WE REALLY REALLY NEEDED TO GET IT RIGHT THIS TIME.

WHAT TO DO?

A HUGE REWRITE WAS APPEALING.

BUT THEY ALWAYS TAKE TWICE AS LONG AS YOU WANT, AND CAN KILL COMPANIES.

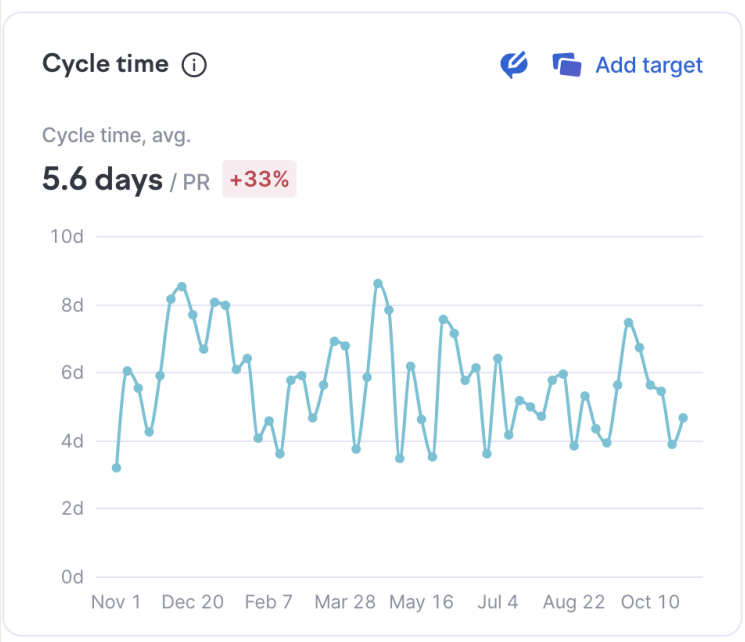
DEALING WITH TECH DEBT:

1) VISUALIZE THE PROBLEM
2) BENCHMARK VS INDUSTRY
3) EVANGELIZE & EXPLAIN
4) SECURE SOME AIR COVER
5) DELIVER, DELIVER, DELIVER!

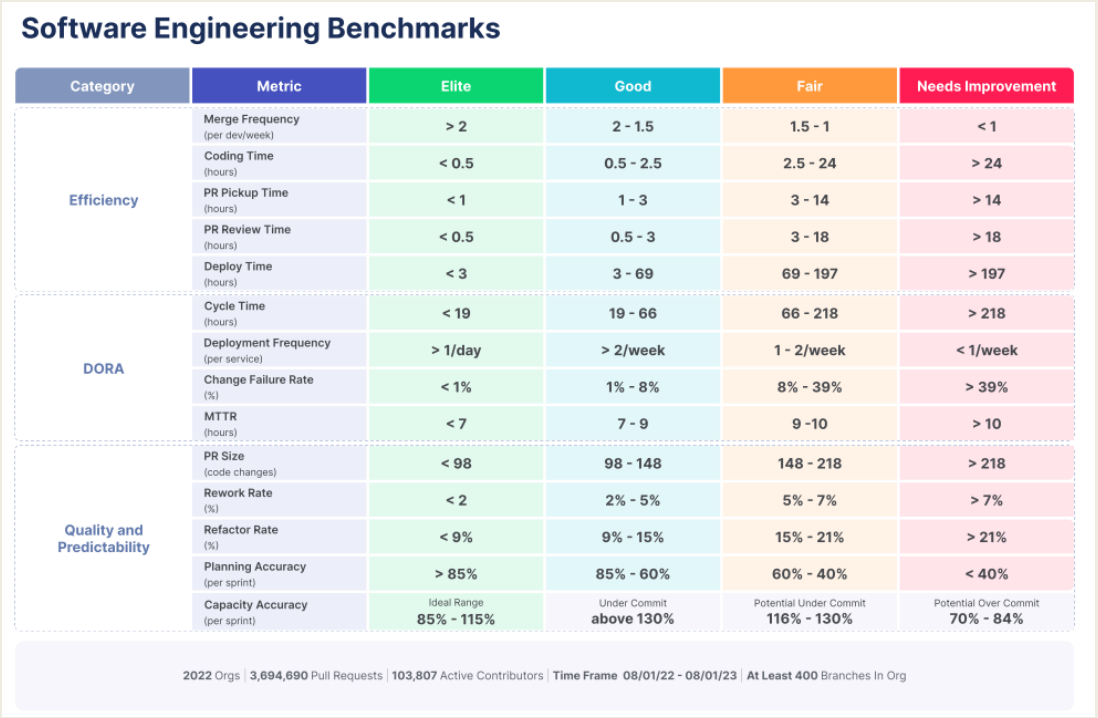
1) VISUALIZE THE PROBLEM

WE USED DORA METRICS

OUR CYCLE TIME WAS 5.6 DAYS!



2) BENCHMARK VS INDUSTRY



Software Engineering Benchmarks

Category	Metric	Elite	Good	Fair	Needs Improvement
Efficiency	Merge Frequency (per dev/week)	> 2	2 - 1.5	1.5 - 1	< 1
	Coding Time (hours)	< 0.5	0.5 - 2.5	2.5 - 24	> 24
	PR Pickup Time (hours)	< 1	1 - 3	3 - 14	> 14
	PR Review Time (hours)	< 0.5	0.5 - 3	3 - 18	> 18
	Deploy Time (hours)	< 3	3 - 69	69 - 197	> 197
DORA	Cycle Time (hours)	< 19	19 - 66	66 - 218	> 218
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	< 1%	1% - 8%	8% - 39%	> 39%
	MTTR (hours)	< 7	7 - 9	9 -10	> 10
Quality and Predictability	PR Size (code changes)	< 98	98 - 148	148 - 218	> 218
	Rework Rate (%)	< 2	2% - 5%	5% - 7%	> 7%
	Refactor Rate (%)	< 9%	9% - 15%	15% - 21%	> 21%
	Planning Accuracy (per sprint)	> 85%	85% - 60%	60% - 40%	< 40%
	Capacity Accuracy (per sprint)	Ideal Range 85% - 115%	Under Commit above 130%	Potential Under Commit 116% - 130%	Potential Over Commit 70% - 84%

2022 Orgs | 3,694,690 Pull Requests | 103,807 Active Contributors | Time Frame 08/01/22 - 08/01/23 | At Least 400 Branches In Org

THIS HELPED OUR EXEC AND BOARD TO UNDERSTAND THE PROBLEM CONCEPTUALLY

3) EVANGELIZE & EXPLAIN

WE TALKED WITH THE BROADER COMPANY ABOUT DEIMOS AND THE PROBLEMS IT WAS CAUSING.

PRIMARILY IN INCIDENTS AND STOPPING US DELIVERING NEW CUSTOMER FEATURES.

4) SECURE SOME AIR COVER

I ASKED BOARD & EXEC TO ALLOW US 50% OF OUR TIME FIXING DEIMOS WHILST STILL DELIVERING NEW CUSTOMER FEATURES.

(I ALSO TOLD THEM TO FIRE ME IN A YEAR IF I WAS STILL COMPLAINING ABOUT DEIMOS)

5) DELIVER, DELIVER, DELIVER

WE SET OUT AN API-FIRST STRATEGY AND DID A FULL PROGRAM OF TRAINING (WITH SKILLERWHALE) TO UPSKILL OUR ENGINEERS.

THEN WE PULLED PIECES OUT OF DEIMOS INTO PROPER APIS.

5) DELIVER, DELIVER, DELIVER

Component	Team	Internal status	External s	
Export API ↑ Move out Core Entity Operatio...	Accounting Experience	Taking Traffic	Public	V1
Tags API ↑ Move out Core Entity Operatio...	Accounting Experience	Taking Traffic	Public	V1
User Provisioning (SCIM) API ↑ Move out Core E	Manage Entities	Deployed to Produ...	Development	V1
Accounting Entries (READ) API ↑ Move out Core	Expense Core	Taking Traffic	Development	V1
Accounting Entries (WRITE) API ↑ Move out Cor	Expense Core	Development	Not planned	Later
Receipts API ↑ Move out Core Entity Operatio...	Expense Core	Taking Traffic	Should not be exposed	Never
Cards API ↑ Move out Core Entity Operatio...	Card Experience	Deployed to Produ...	Not planned	TBD
Limits API ↑ Move out Core Entity Operatio...	Card Experience	Taking Traffic	Not planned	TBD
Companies API v4 ↑ Move out Core Entity Opera	Manage Entities	Taking Traffic	Should not be exposed	Never
Employees API v4 ↑ Move out Core Entity Opera	Manage Entities	Taking Traffic	Should not be exposed	Never
Attendees API (sub resource of AccEntry) ↑ Mc	Expense Core	Testing	Not planned	Never
Contacts API ↑ Move out Core Entity Operatio...	Expense Core	Testing	Not planned	TBD
Balance Entries API ↑ Move out Core Entity Oper	Team Pocket	Taking Traffic	Not planned	Never
Categories API ↑ Move out Core Entity Operatio.	Accounting Experience	Taking Traffic	Not planned	V1
Taxes API ↑ Move out Core Entity Operatio...	Accounting Experience	Taking Traffic	Public	V1
AggregatedAccountingEntriesAPI ↑ Move out Core	Expense Core	Development	Not planned	Later
Employees API v5 ↑ Move out Core Entity Opera	Manage Entities	Taking Traffic	Planned	V1
Accounts API ↑ Move out Core Entity Operatio...		Development	Development	
Activity Log API ↑ Move out Core Entity Operatio...	Expense Core	Development	Not planned	Later
Vendor API ↑ Move out Core Entity Operatio...	Accounting Experience	Development	Partner validation	Later

API ALL THE THINGS!

Offering maturing | 2023 snapshot 80+% of development on the core product verticals development was held back by tech debt

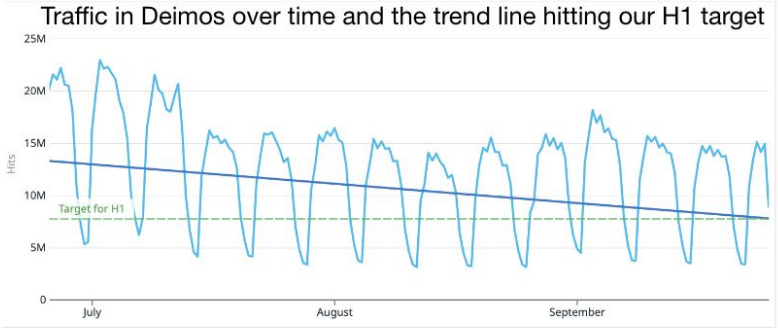
! Monolith entangled

	Easy Pay	Connect & Manage	Be in Control	Flowing Cashflows	Optimized Spending
	! Reimbursements ! Invoices ! Cards	Auto Enrichment ! Automated 'pre-bookkeeping' ! Accounting sync ! User sync ! Multi-entity Vendor management	! Budgets ! Approval workflows ! Automated controls & Blocks ! Policy management ! Audit & Reporting	Automatic funds management (top-up) Reserve Overdraft	Analytics AI spend insights Benchmarks Rewards ! Spend Category Partnerships/VAS (e.g. TravelPerk) ESG
Impact on new build	80+% Reimbursement balances Card ordering Card Limit	80+% All logic APIs needed for accounting integrations and user mgmt	80+% Card Limit Review management	20+%	20+%

PREREAD

Tech maturing | We have moved review management out of our monolith (Deimos) providing improved feature functionality AND vastly faster experience

Deimos traffic is now significantly down, reaping the benefits of our late 2023 and early 2024 investments in decommissioning (vastly reducing risk)

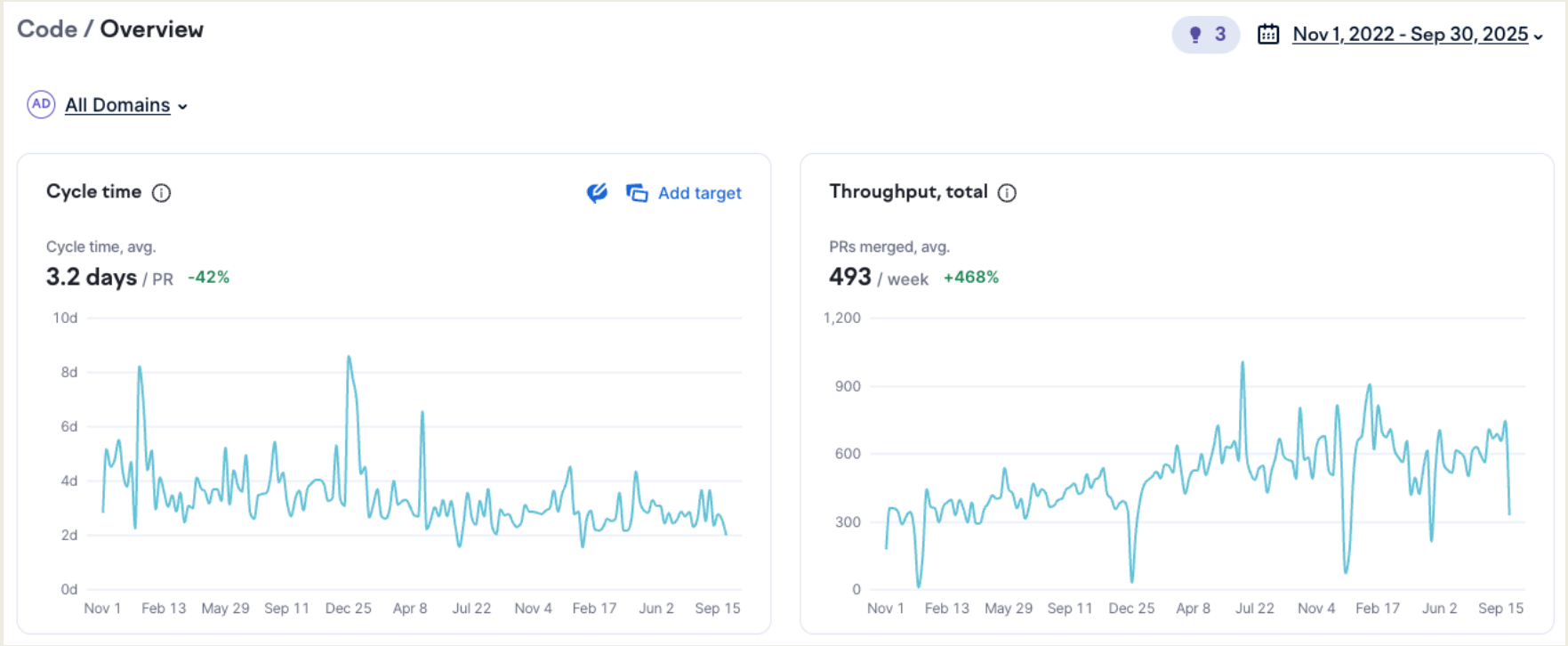


5) DELIVER, DELIVER, DELIVER

MOST IMPORTANTLY, WE DIDN'T GO HIDE IN A CORNER AND JUST DEAL WITH TECH DEBT.

WE DELIVERED CUSTOMER VALUE ALL ALONG THE WAY INCLUDING SOME LONG-AWAITED FEATURES DEIMOS WAS BLOCKING.

5) DELIVER, DELIVER, DELIVER



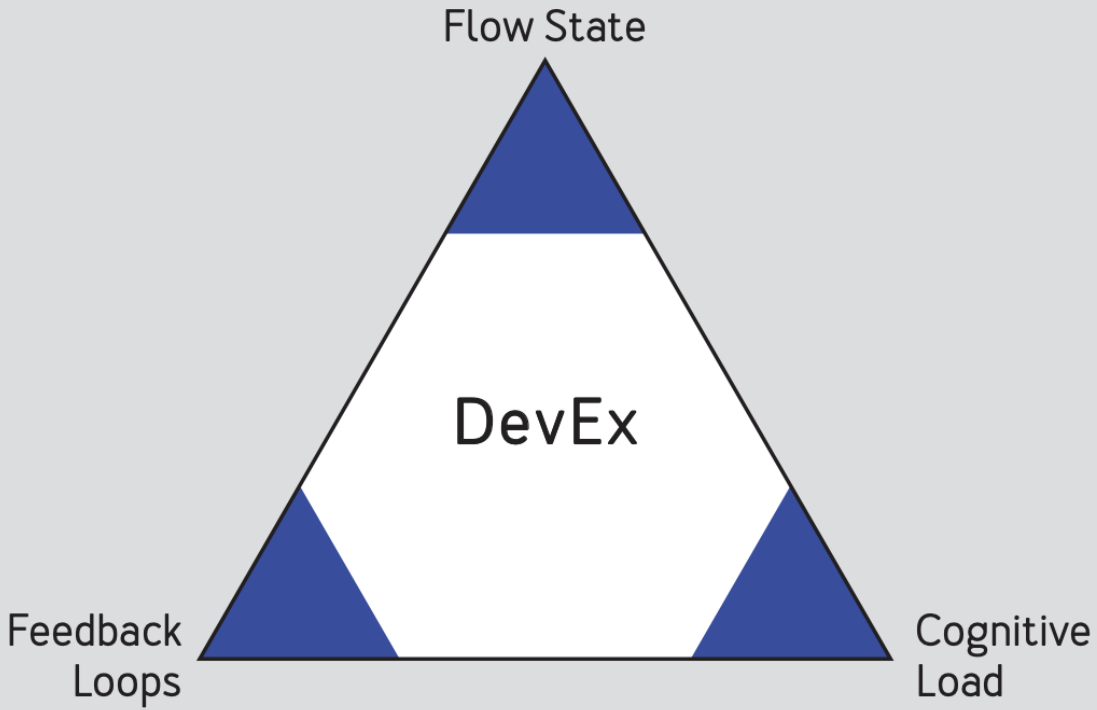
IT WOULD HAVE BEEN BETTER THOUGH IF THINGS HAD NEVER GOTTEN SO BAD THAT THIS MAJOR APPROACH WAS NEEDED.

SO HOW DO WE TAME TECH DEBT ON A DAILY BASIS?

IS TECH DEBT INEVITABLE?


I NEVER MEANT TO START A WAR,
I JUST WANTED YOU TO LET ME IN

FIGURE 1: THREE CORE DIMENSIONS OF DEVELOPER EXPERIENCE



https://dl.acm.org/doi/pdf/10.1145/3595878

THREE THINGS MATTER FOR DEVELOPER EXPERIENCE:

- FEEDBACK LOOPS
- COGNITIVE LOAD
- FLOW STATE

FEEDBACK LOOPS: ADD TESTS WHERE THEY ARE MISSING

COGNITIVE LOAD: REFACTOR COMPLEX & COMPLICATED CODE (SEE CRAP MEASURE)

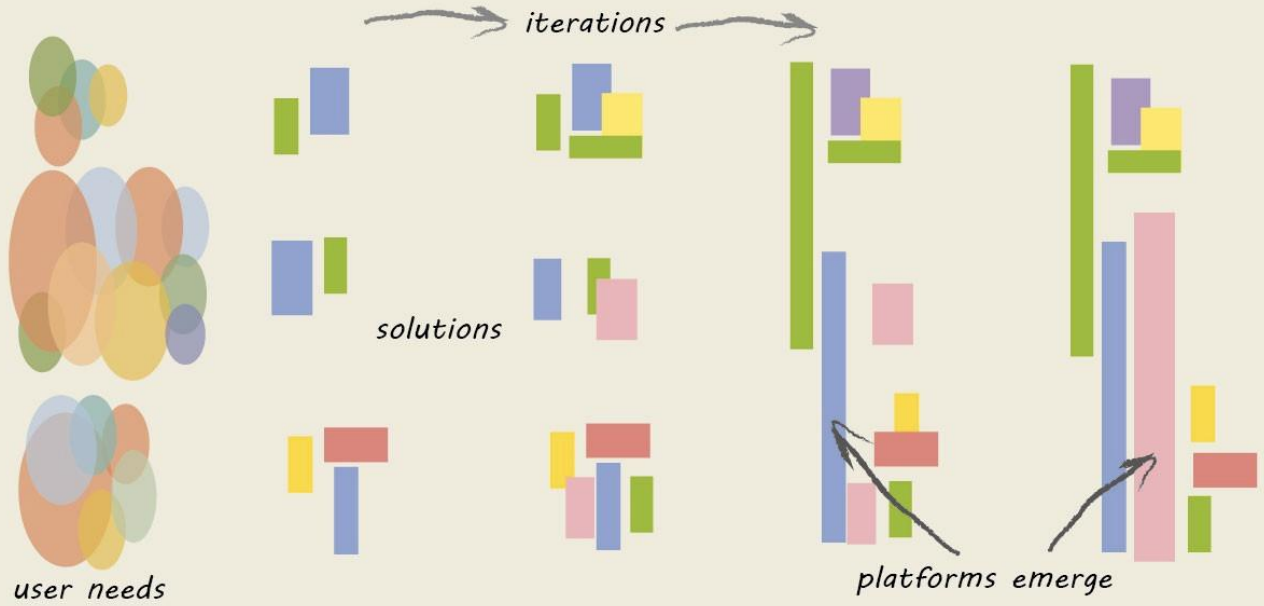
FLOW STATE: ADD DOCUMENTATION

THE GREAT NEWS?

DOING THESE THINGS HELPS YOU TO GRAPPLE WITH THE CODE AND DELIVER FASTER IN ANY CASE!

START WITH TECH DEBT IMPROVEMENT

BUILD, REFACTOR, LET PLATFORMS EMERGE



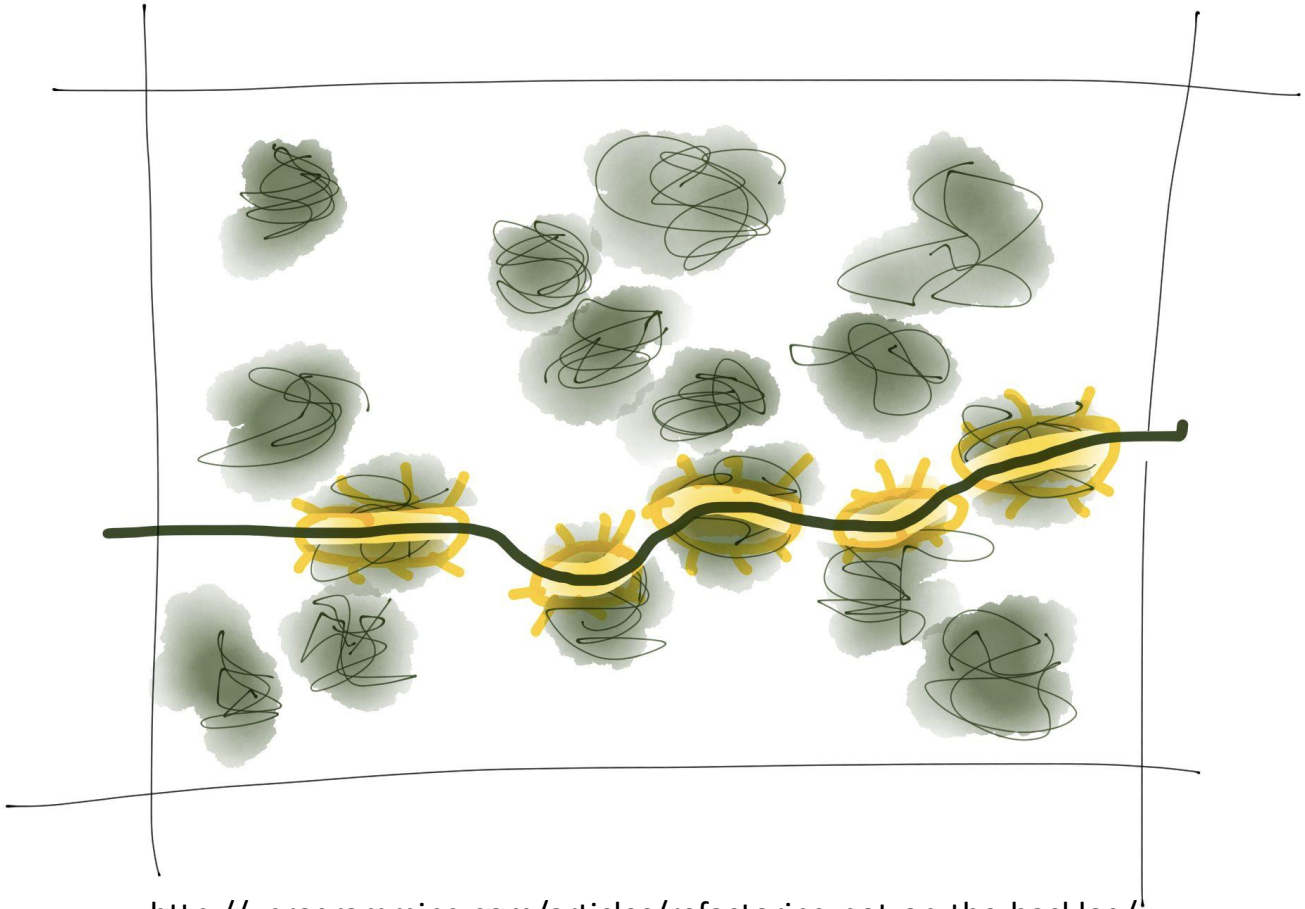
https://medium.com/@postenterprise/the-abuse-of-reuse-96b2e0af01a7

PUT AN EXPIRY DATE ON ANY (NECESSARY) DIRTY HACKS

DON’T WAIT FOR PERMISSION TO IMPROVE THINGS

ADOPT “BOYSCOUT” RULE

BUILD IMPROVEMENT IN



http://xprogramming.com/articles/refactoring-not-on-the-backlog/

REFACTOR YOUR MONOLITH

START CONSUMING APIS INTERNALLY TOO

EAT YOUR OWN DOGFOOD


I HAVEN'T EATEN IN AN HOUR
THAT'S 7 DOG HOURS

Meri Williams, ChromeRose

@Geek_Manager

DEALING WITH TECH DEBT:

1) VISUALIZE THE PROBLEM
2) BENCHMARK VS INDUSTRY
3) EVANGELIZE & EXPLAIN
4) SECURE SOME AIR COVER
5) DELIVER, DELIVER, DELIVER!