Continuous Modernization

Maintaining Long-Lived Federal Systems

Excella | Dane Weber

excella.com | @excellaco

571.289.0000 | dane.weber@excella.com

Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

Federal IT systems include critical enterprise functions:

- System of record
- Implementation of mandate
- Accomplishing the mission

Long-Lived Federal IT Systems

Federal IT systems include critical enterprise functions:

- System of record
- Implementation of mandate
- Accomplishing the mission

These frequently evolve over time, but rarely go away.

Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

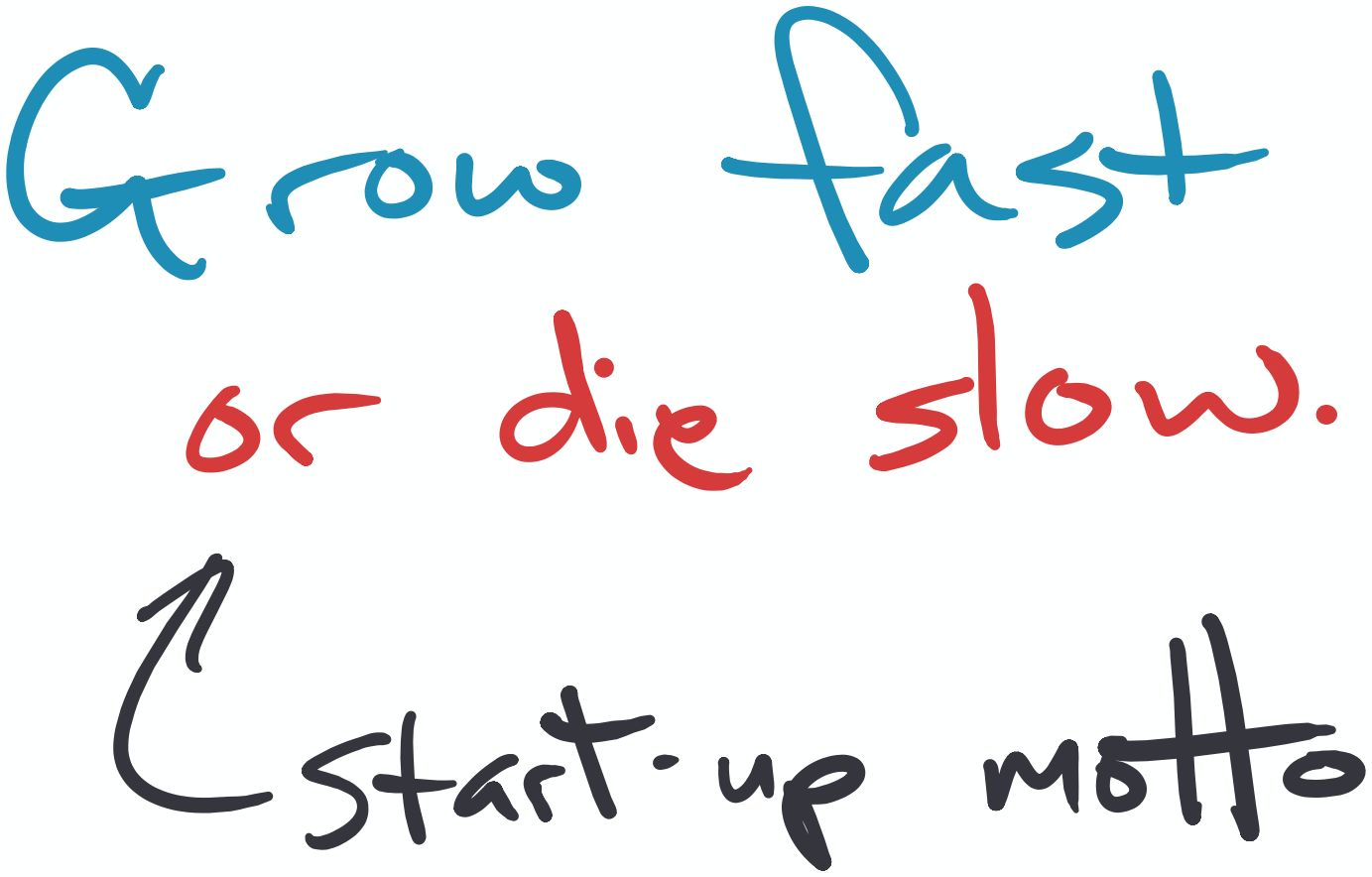
Federal systems serve uniquely long-lived missions.

Long-Lived Federal IT Systems

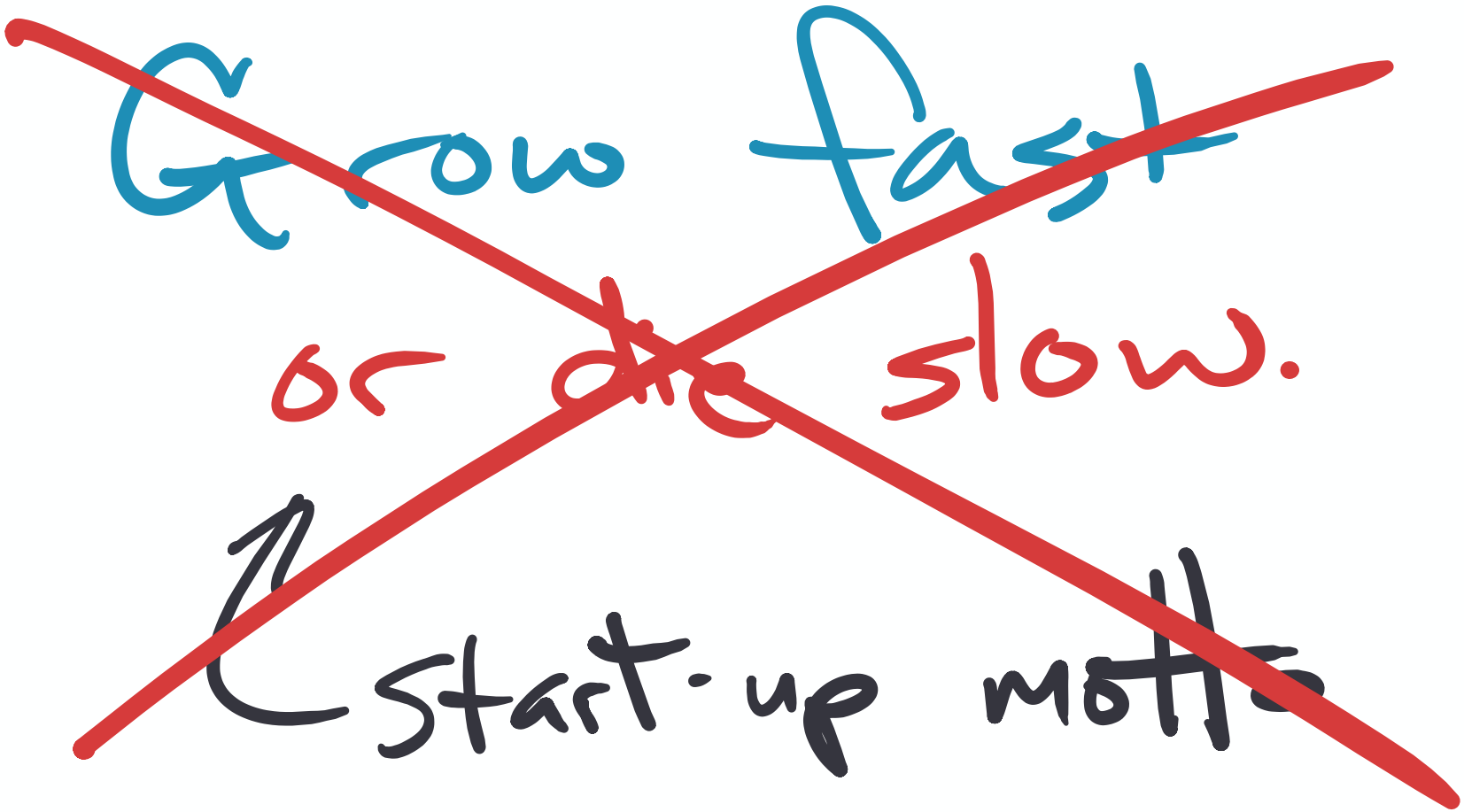
Federal systems serve uniquely long-lived missions.

Short-term thinking often leads to **risks** and **waste** that have long-term consequences.

Long-Lived Federal IT Systems



Long-Lived Federal IT Systems



Long-Lived Federal IT Systems

Long-Lived Federal IT Systems

Goals for critical enterprise functions:

Long-Lived Federal IT Systems

Goals for critical enterprise functions:

- Cost-effective

Long-Lived Federal IT Systems

Goals for critical enterprise functions:

- Cost-effective
- Dependable

Long-Lived Federal IT Systems

Goals for critical enterprise functions:

- Cost-effective
- Dependable
- Extensible

Long-Lived Federal IT Systems

Reality

~~Goals~~ for critical enterprise functions:

- Cost-effective
- Dependable
- Extensible

Long-Lived Federal IT Systems

Reality

~~Goals~~ for critical enterprise functions:

- ~~Cost-effective~~ Expensive
- Dependable
- Extensible

Long-Lived Federal IT Systems

Reality

~~Goals~~ for critical enterprise functions:

- ~~Cost-effective~~ Expensive
- ~~Dependable~~ Unreliable
- Extensible

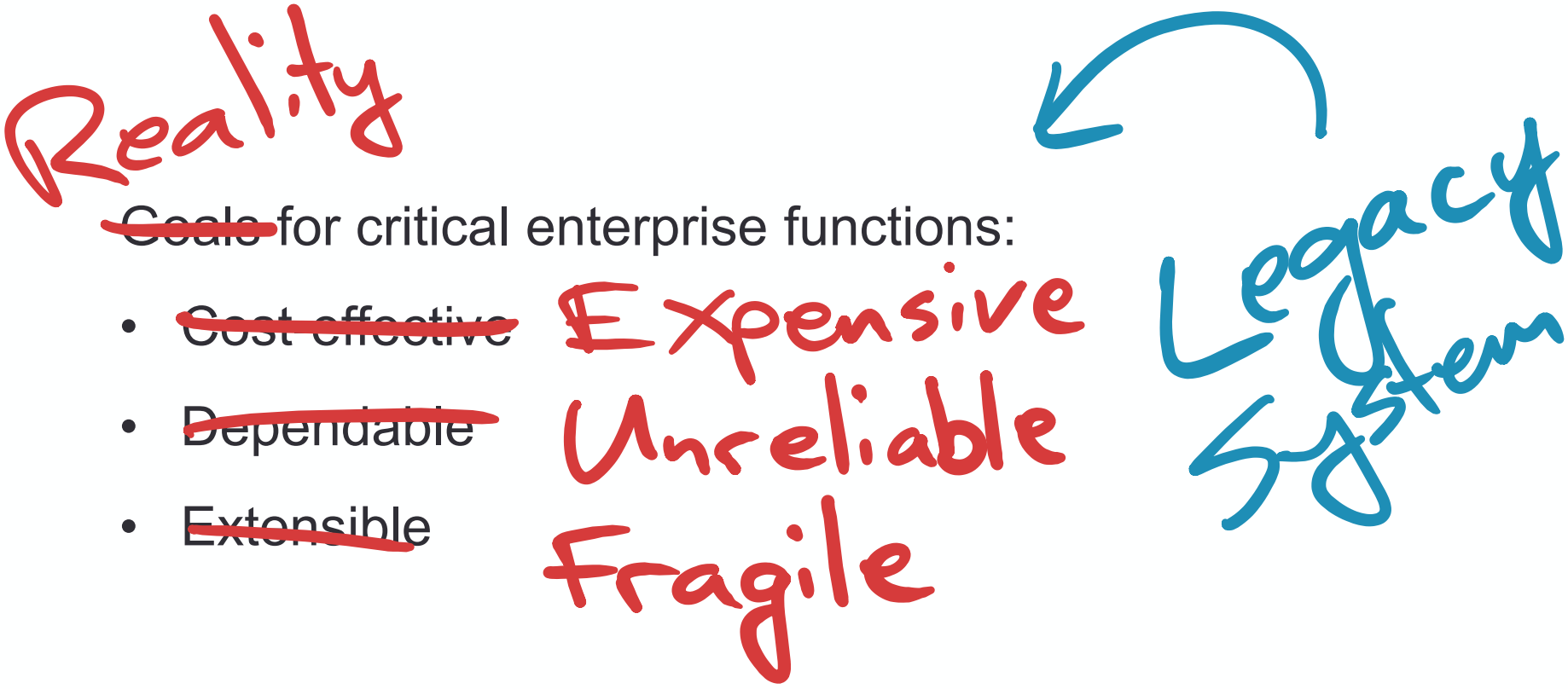
Long-Lived Federal IT Systems

Reality

~~Goals~~ for critical enterprise functions:

- ~~Cost-effective~~ Expensive
- ~~Dependable~~ Unreliable
- ~~Extensible~~ Fragile

Long-Lived Federal IT Systems



Long-Lived Federal IT Systems

What makes legacy systems so expensive, unreliable, and fragile?

Long-Lived Federal IT Systems

What makes legacy systems so expensive, unreliable, and fragile?

Technical Debt

Technical Debt

Technical Debt

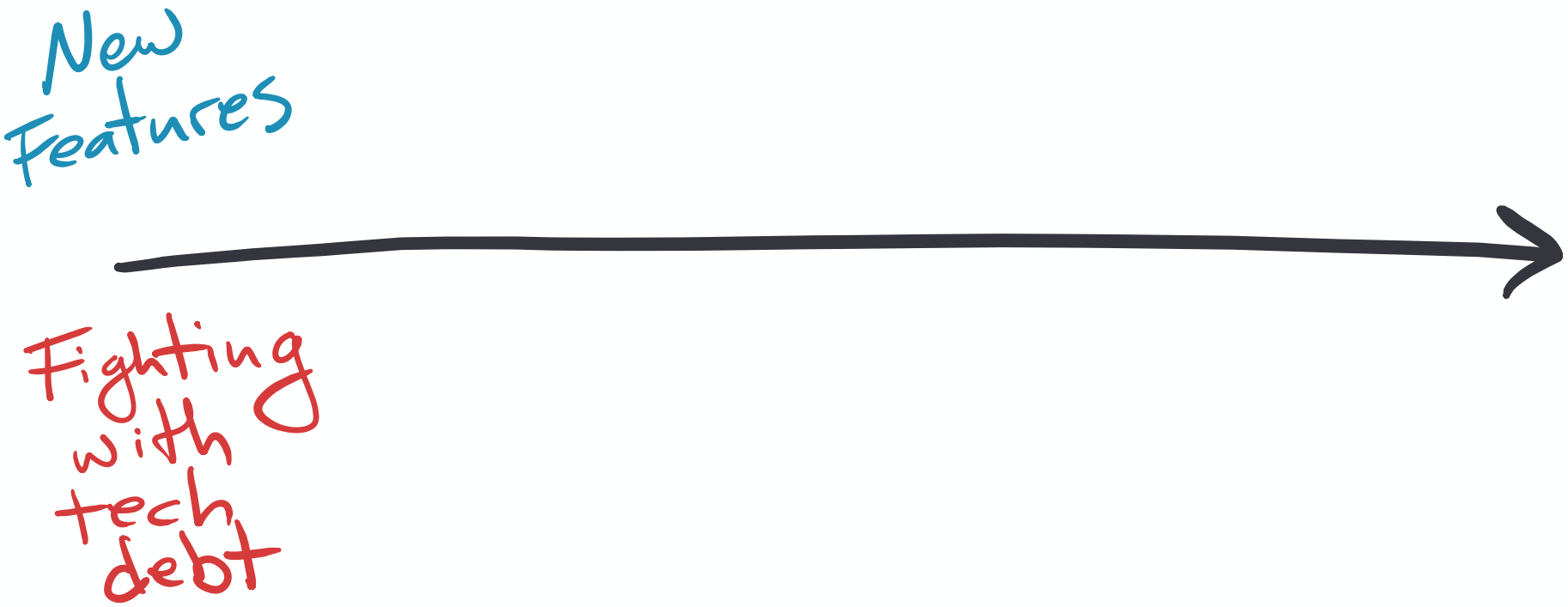
Technical debt is a metaphor for the shortcuts, onerous designs, and outdated technology that make modifying the system difficult and risky.

Technical Debt

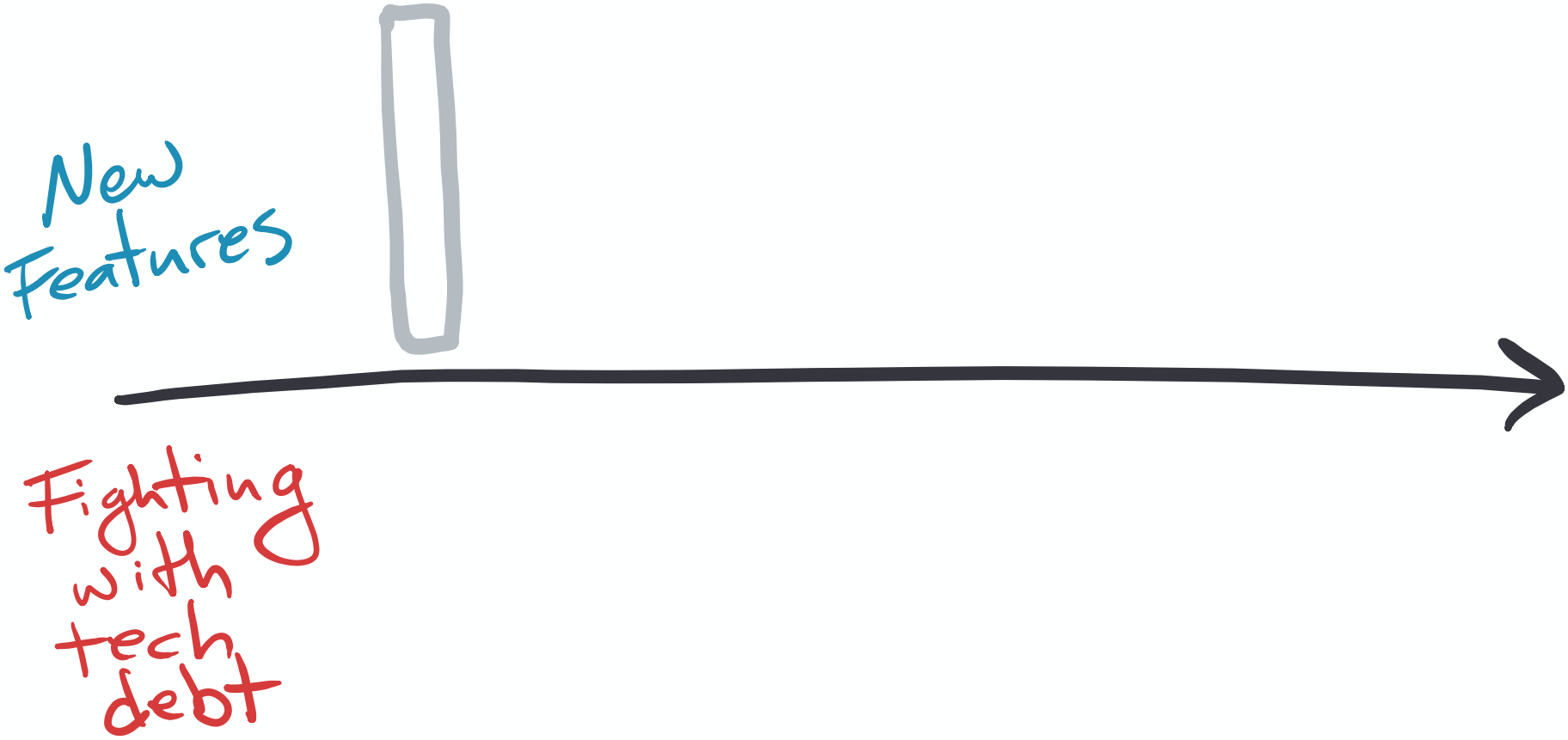
Technical debt is a metaphor for the shortcuts, onerous designs, and outdated technology that make modifying the system difficult and risky.

Like *financial debt* it compounds and accumulates over time and has ongoing costs until it is paid off.

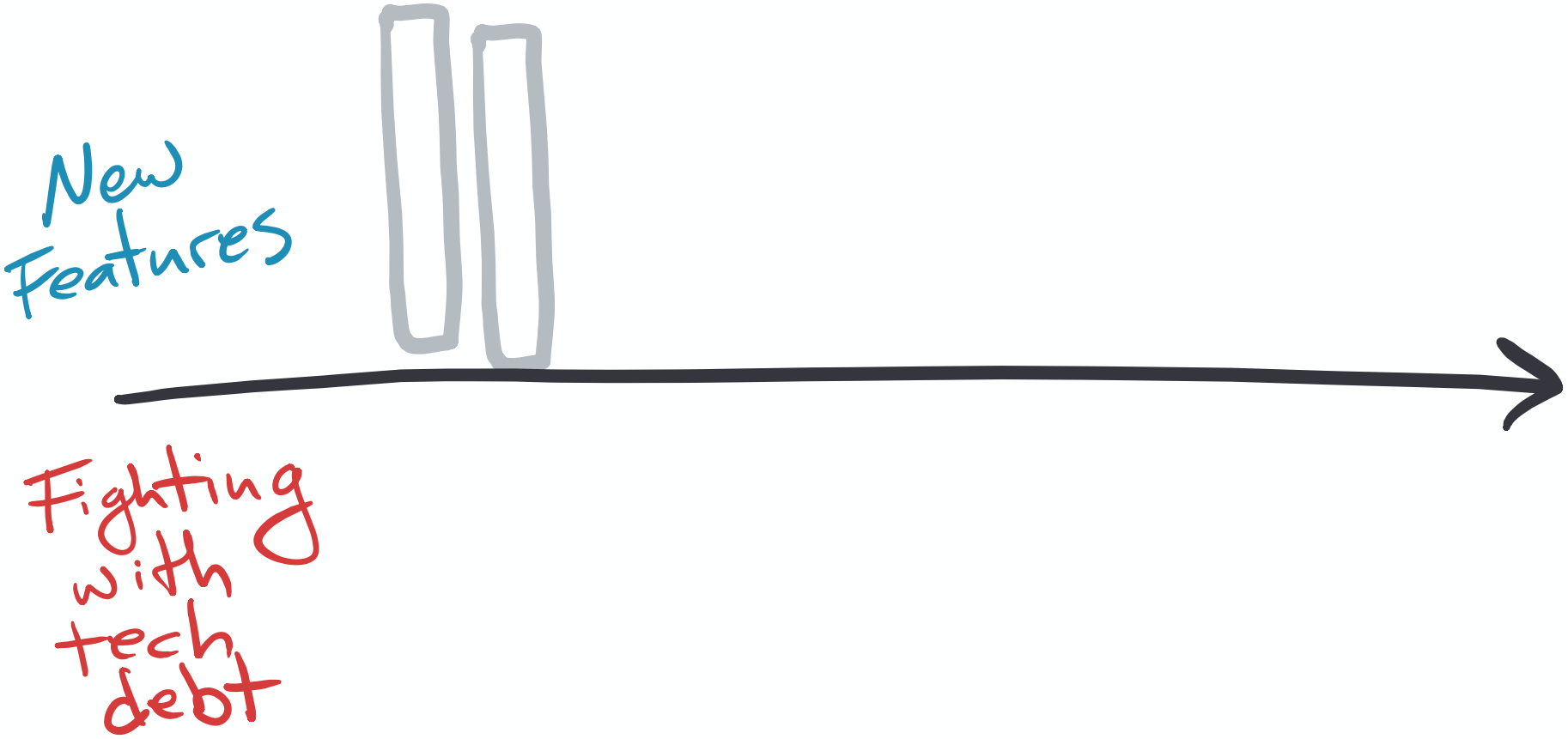
Technical Debt



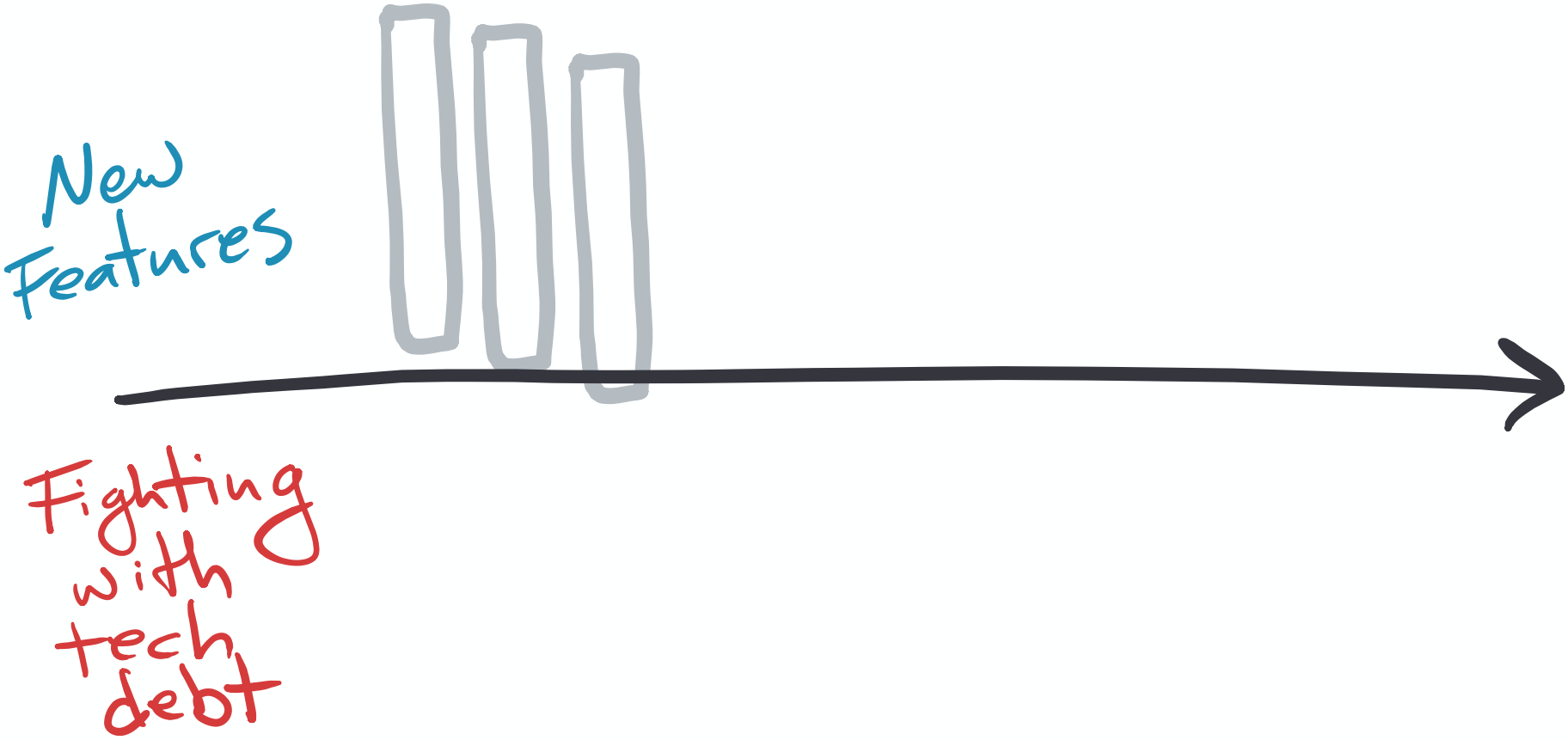
Technical Debt



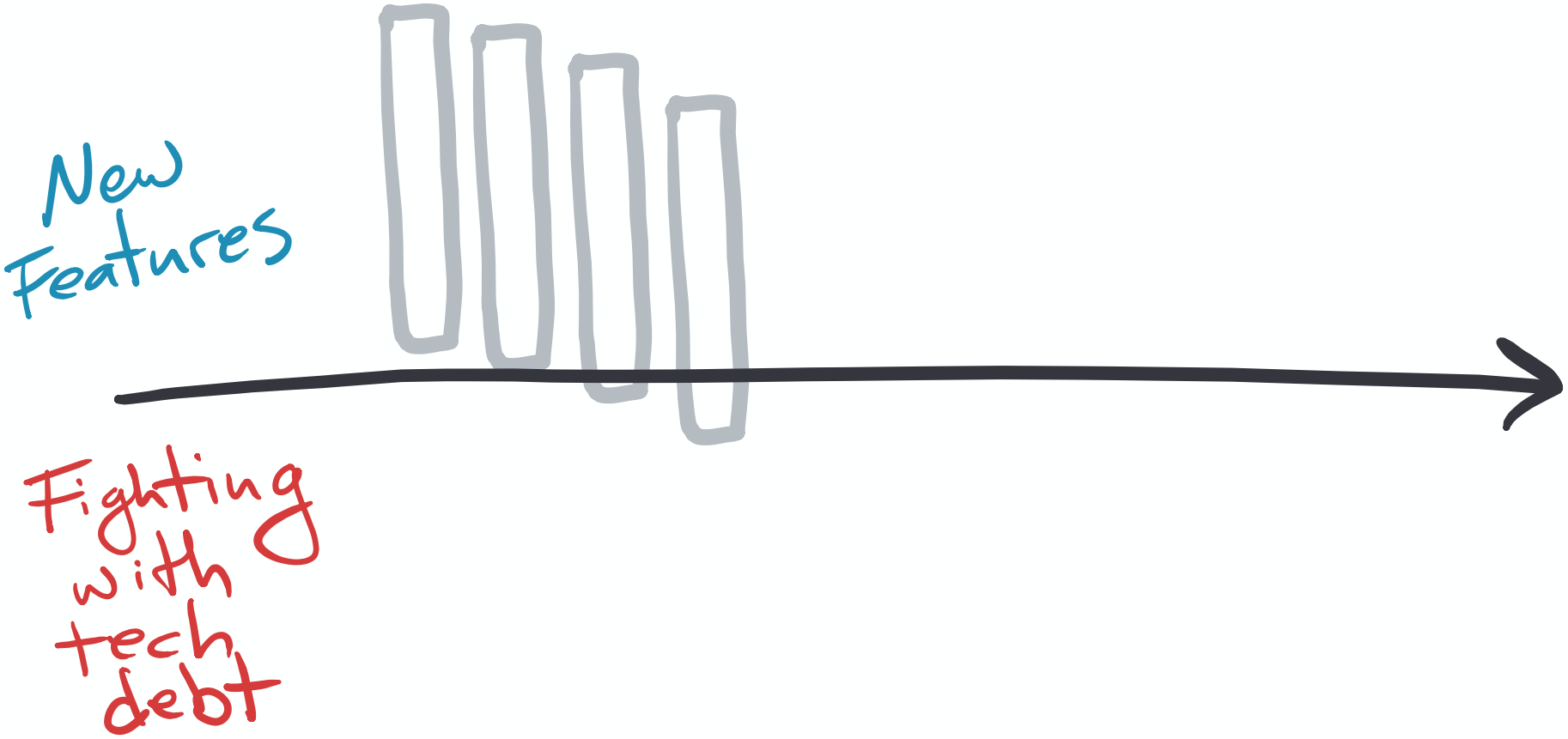
Technical Debt



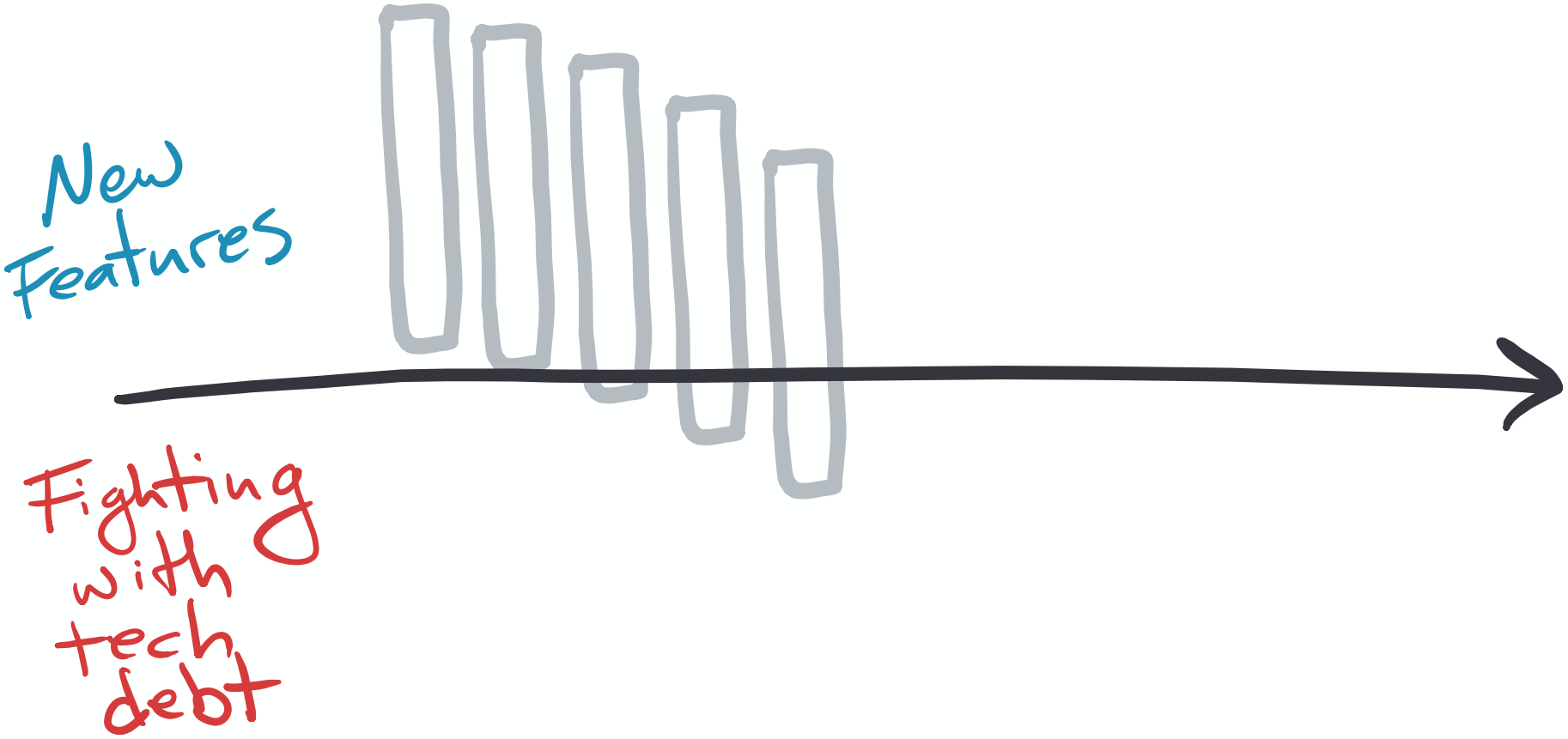
Technical Debt



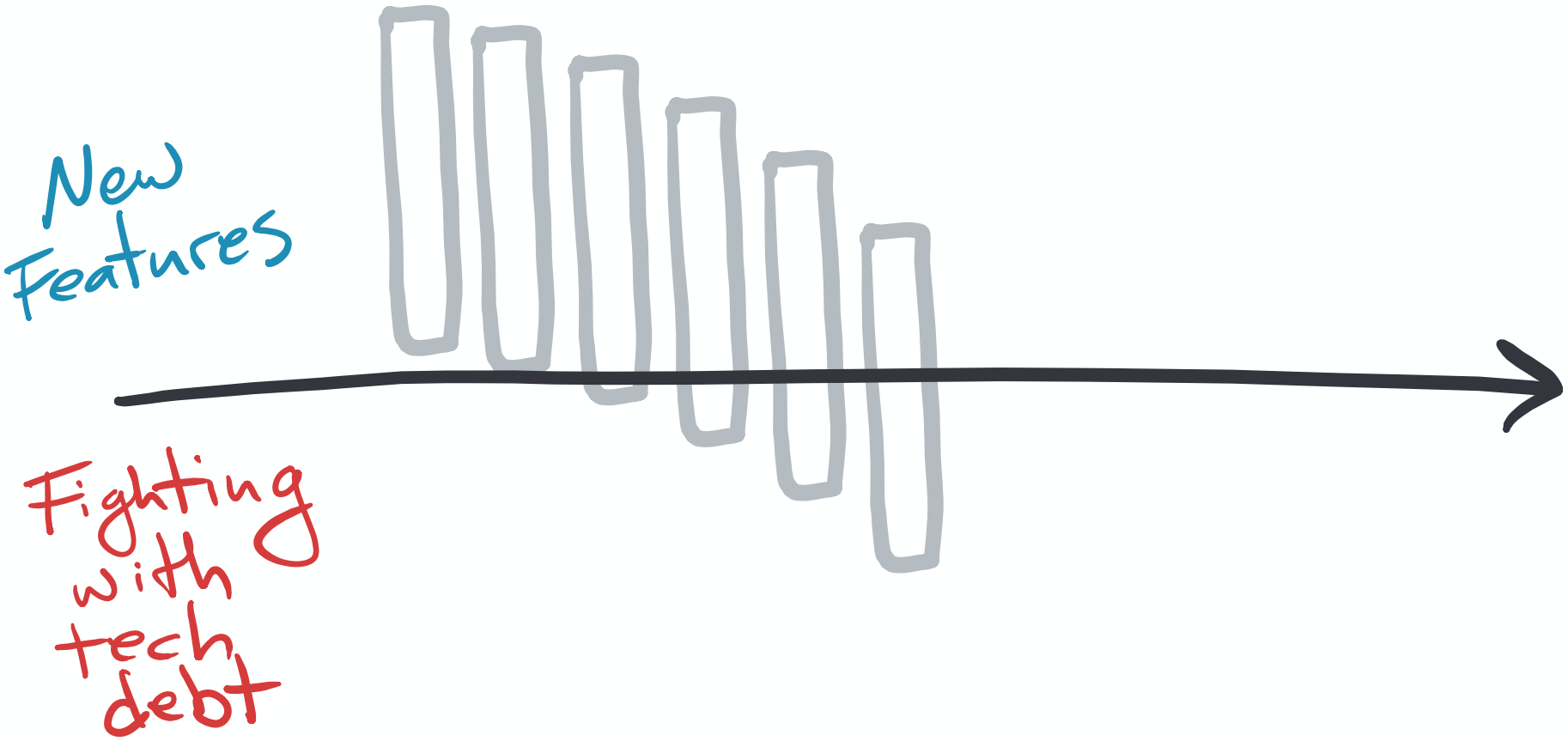
Technical Debt



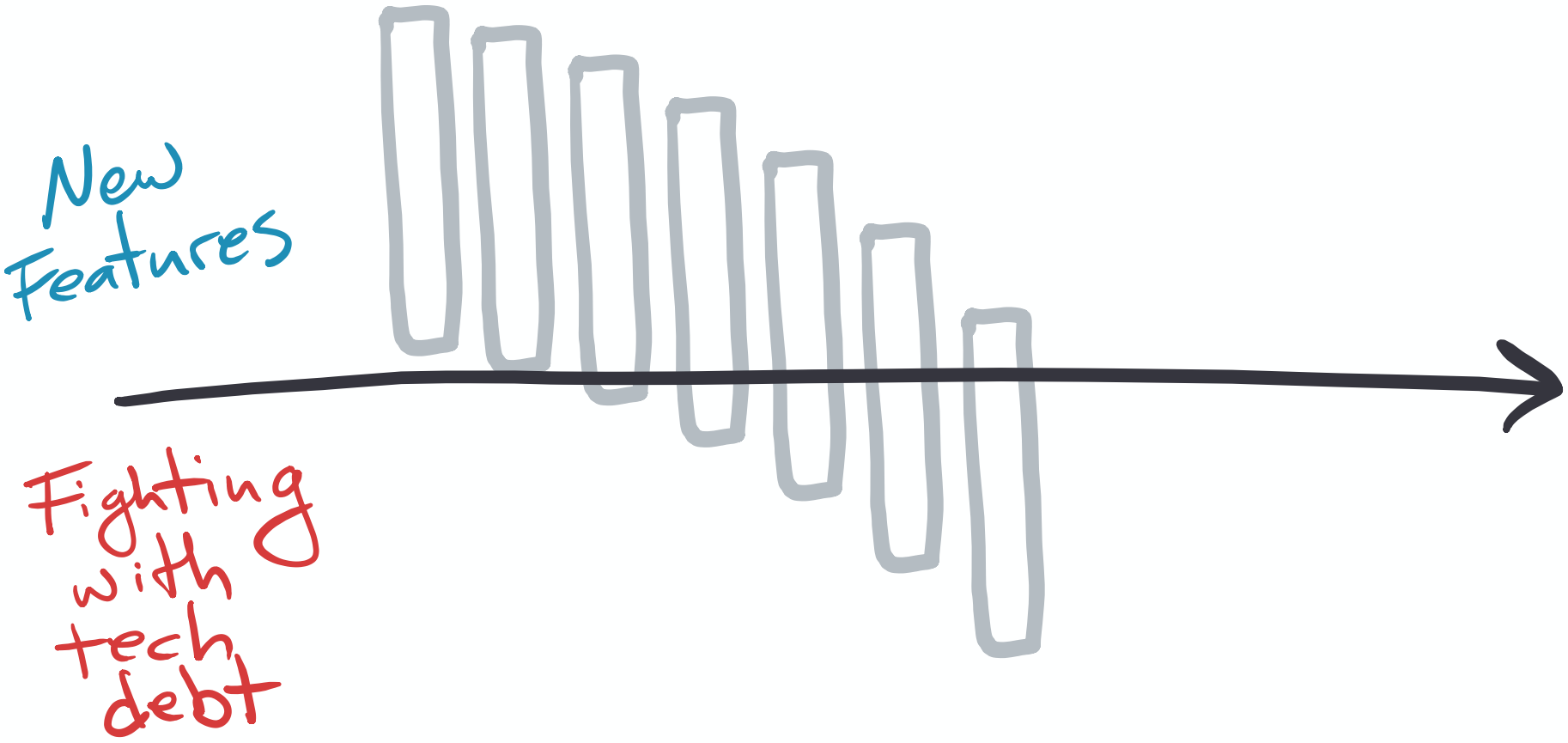
Technical Debt



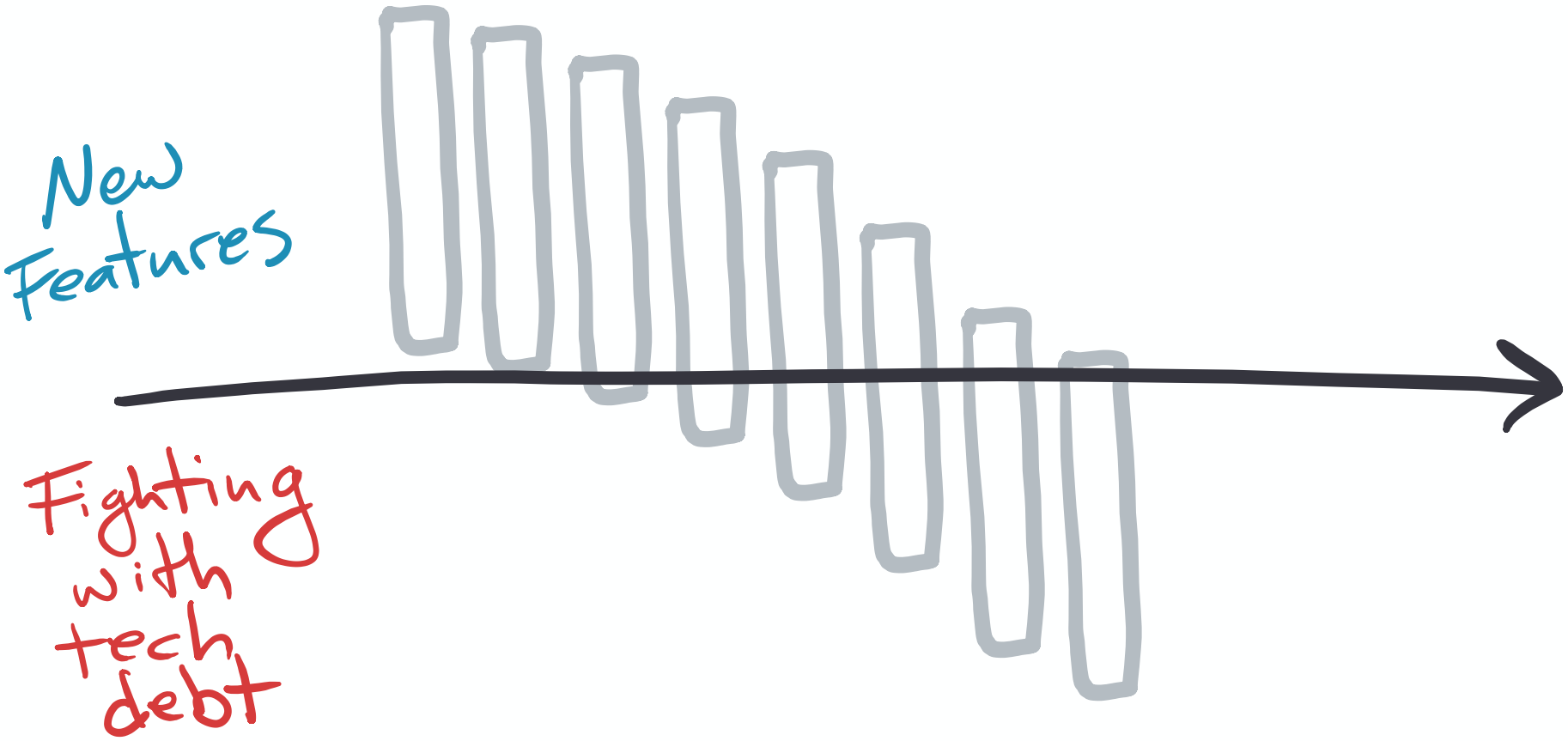
Technical Debt



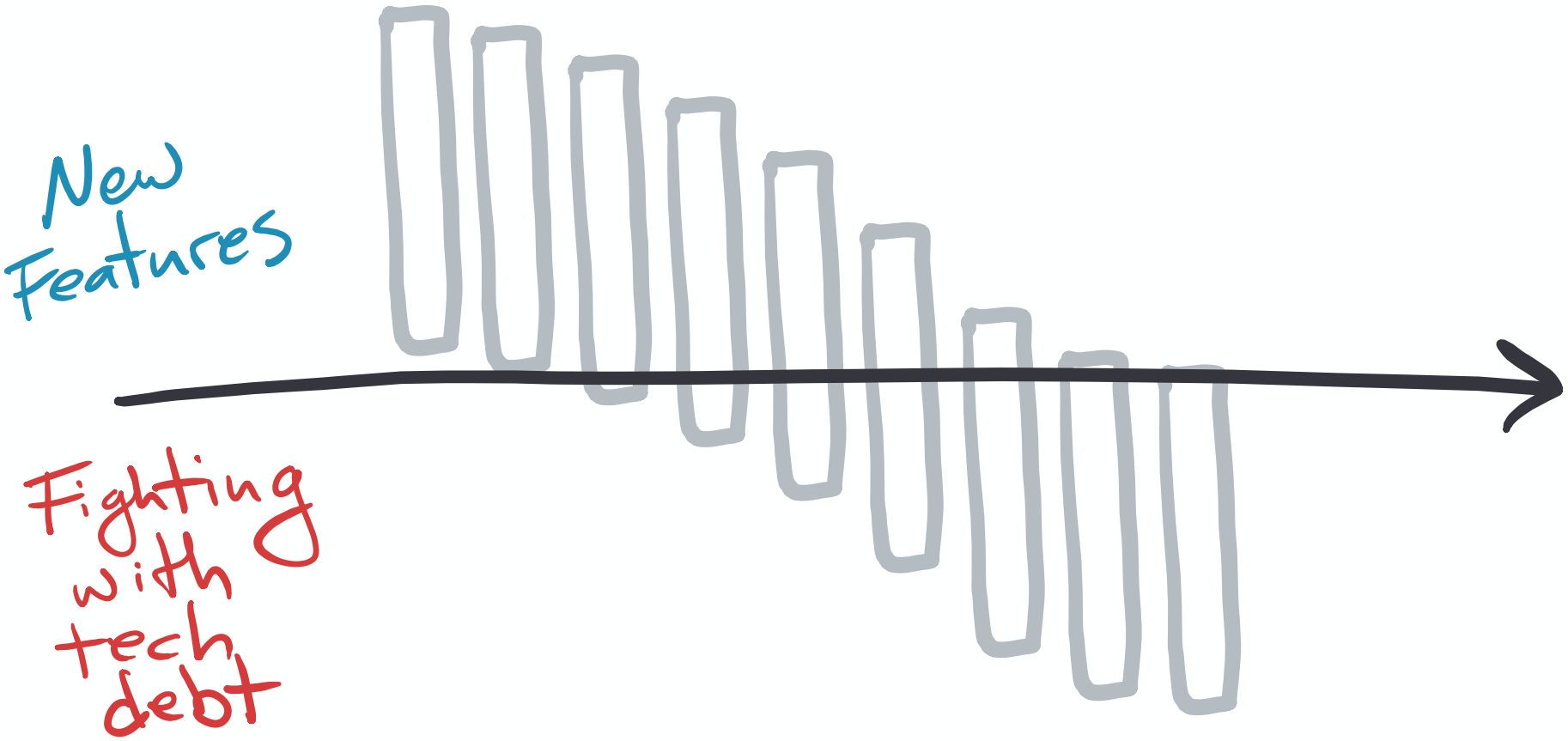
Technical Debt



Technical Debt



Technical Debt



Technical Debt

Technical debt is worse than *financial debt*.

Technical Debt

Technical debt is worse than *financial debt*.

- Not fungible; unique to the system.

Technical Debt

Technical debt is worse than *financial debt*.

- Not fungible; unique to the system.
- Cannot outrun; must fix or replace.

Technical Debt

Technical debt is worse than *financial debt*.

- Not fungible; unique to the system.
- Cannot outrun; must fix or replace.
- Does not have predictable costs.

Technical Debt

Technical debt is worse than *financial debt*.

- Not fungible; unique to the system.
- Cannot outrun; must fix or replace.
- Does not have predictable costs.

It is *compounding risk*.

Legacy Modernization

Legacy System → **Modernization** → **Modern System**

Legacy Modernization

Legacy System

The bad old system full of technical debt.

Modernization

Modern System

Legacy Modernization

Legacy System

The bad old system full of technical debt.

Modernization

Modern System

The replacement that will make all of our dreams come true.

Legacy Modernization

Legacy System

The bad old system full of technical debt.

Modernization

Modern System

The replacement that will make all of our dreams come true.

Legacy Modernization

Legacy System

The bad old system full of technical debt.

Modernization

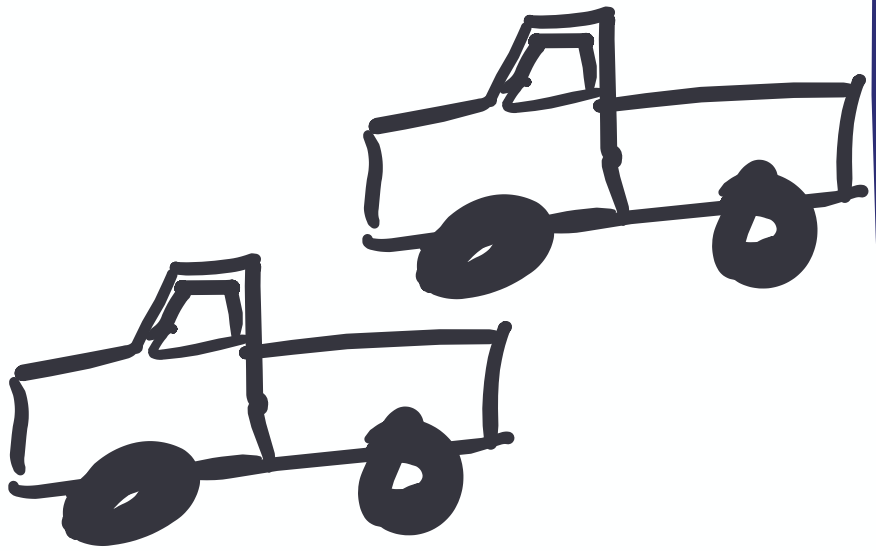
Modern System

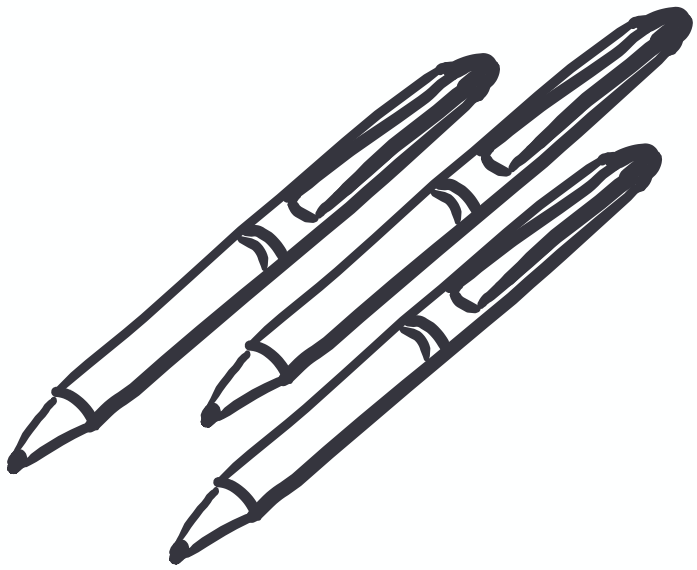
The replacement that will make all of our dreams come true.

Legacy Modernization

“Product” is a problematic metaphor

Trucks, pens, laptops, and code?

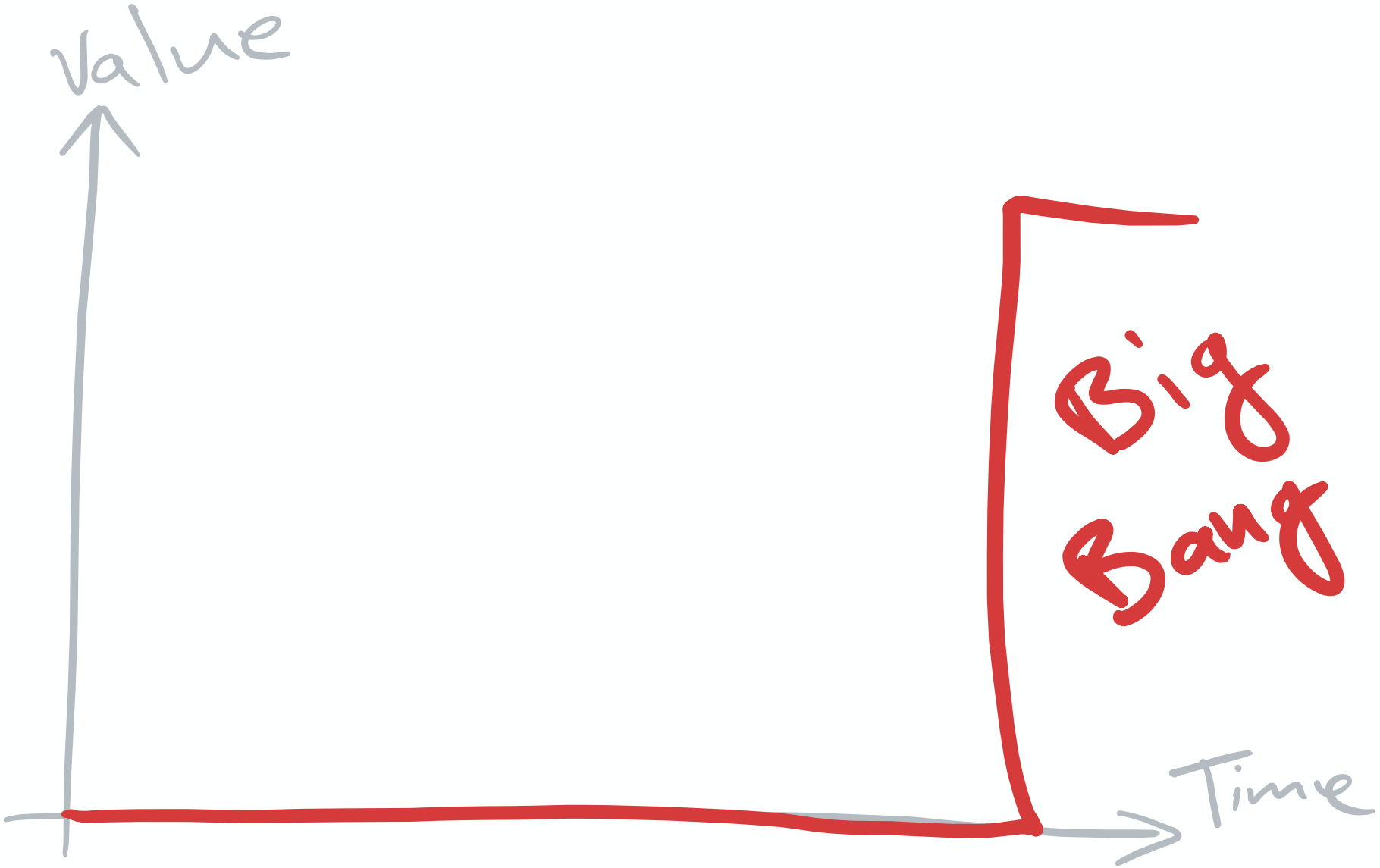


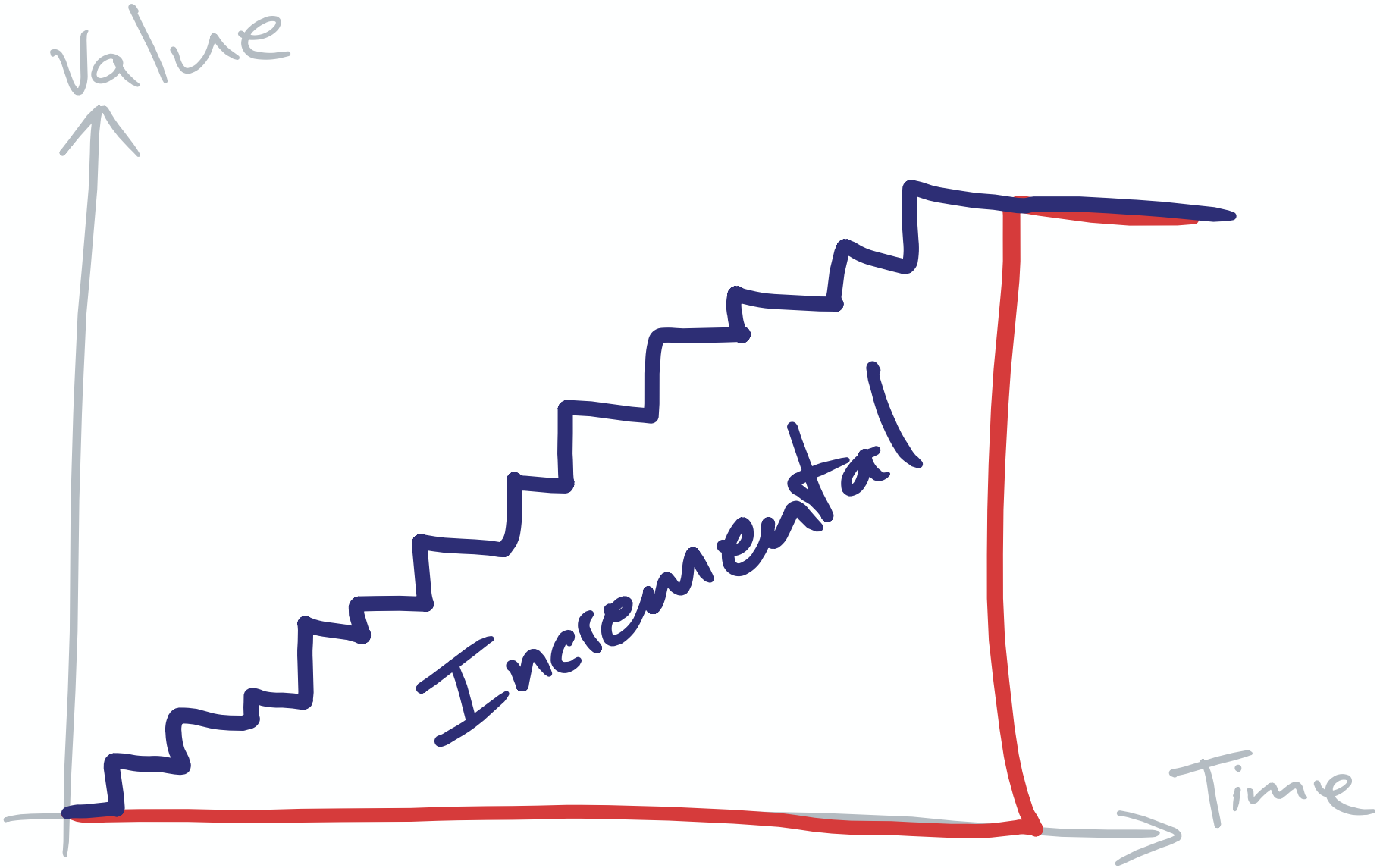


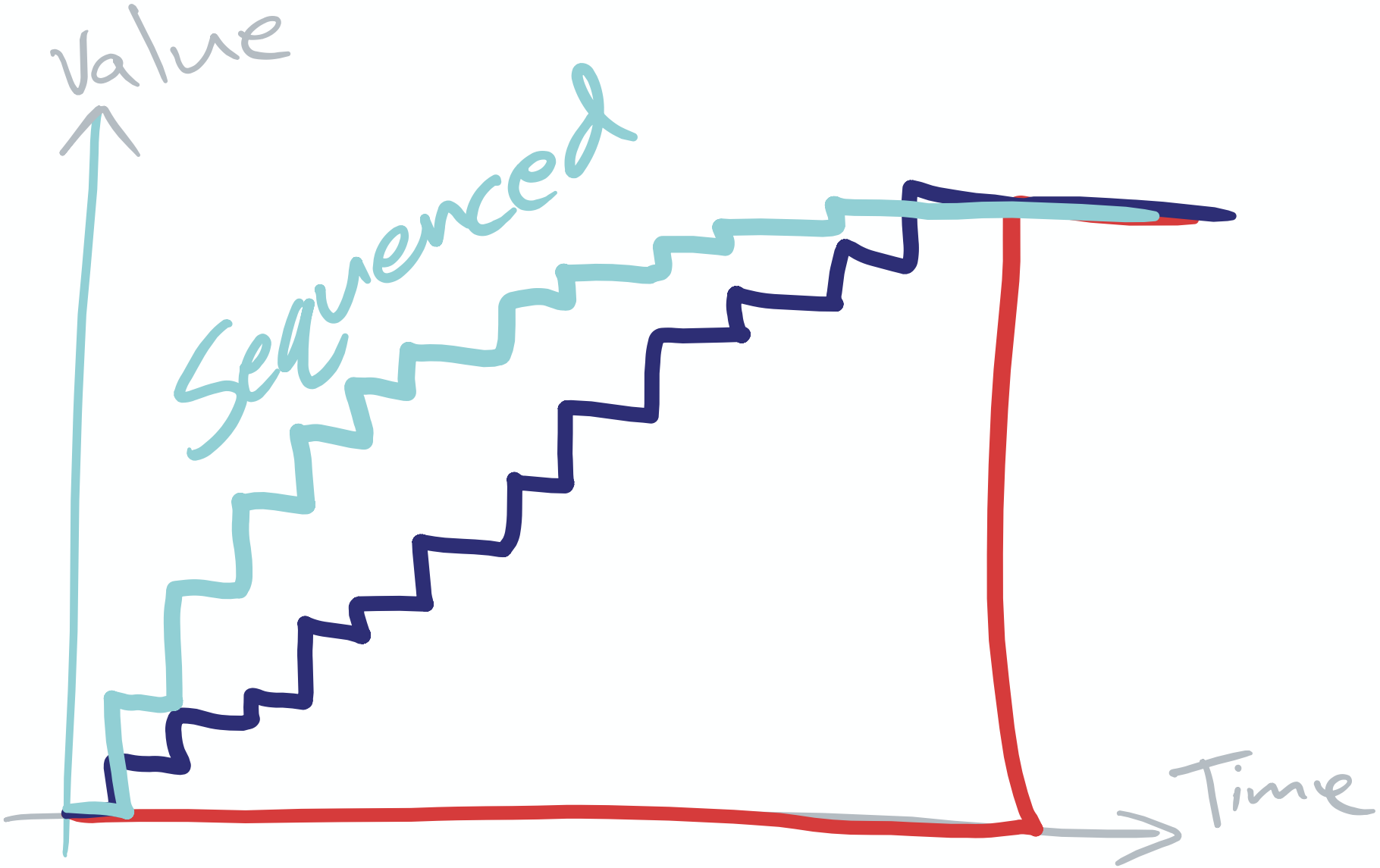
```
switch ($saves)
( {$PSItem.SideIndicator
-eq "<="} {
Copy-Item -Path
$PSItem.InputObject
```

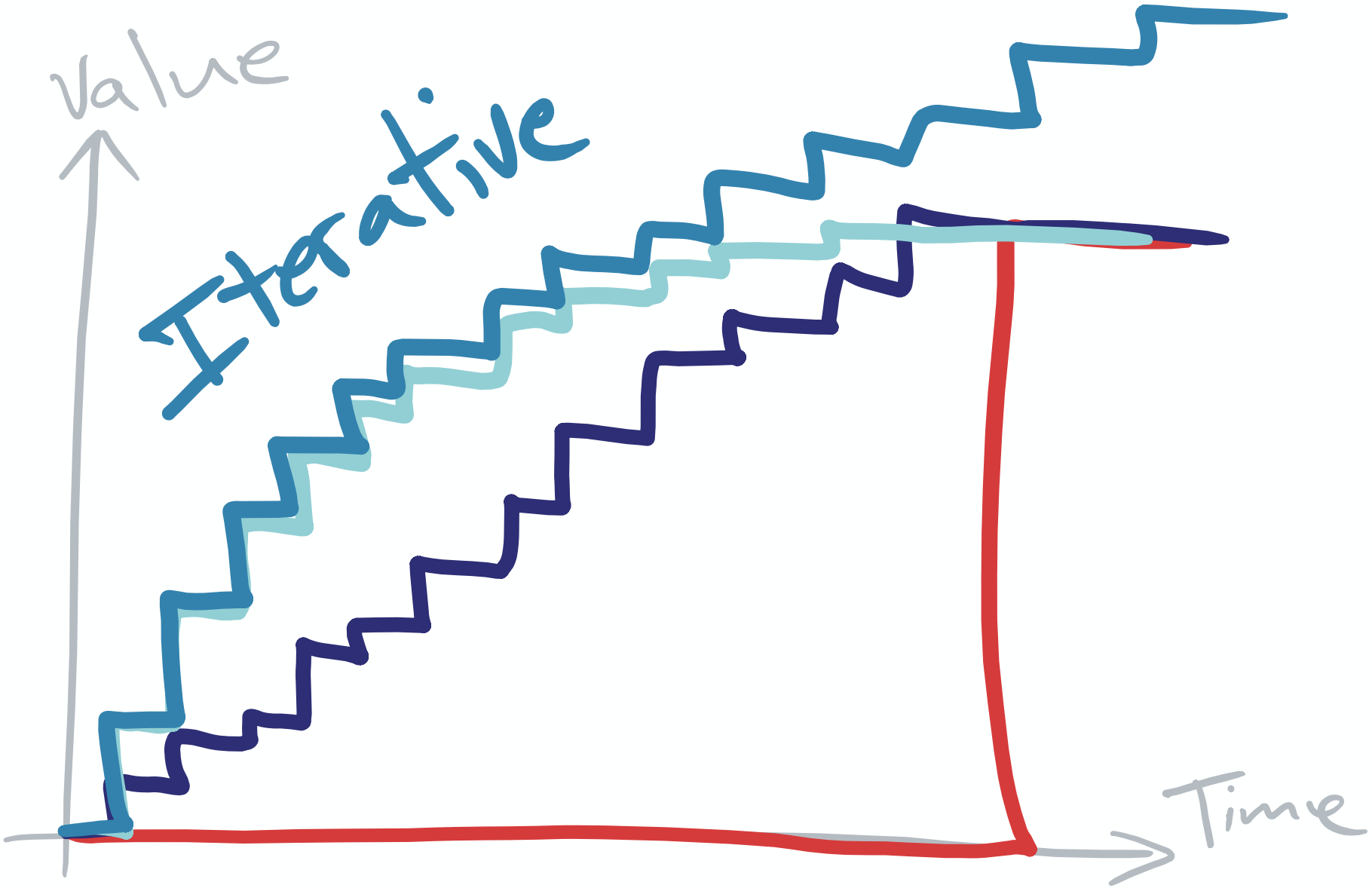
The Promise of Agile Delivery


Value
Time


Value
Big Bang
Time

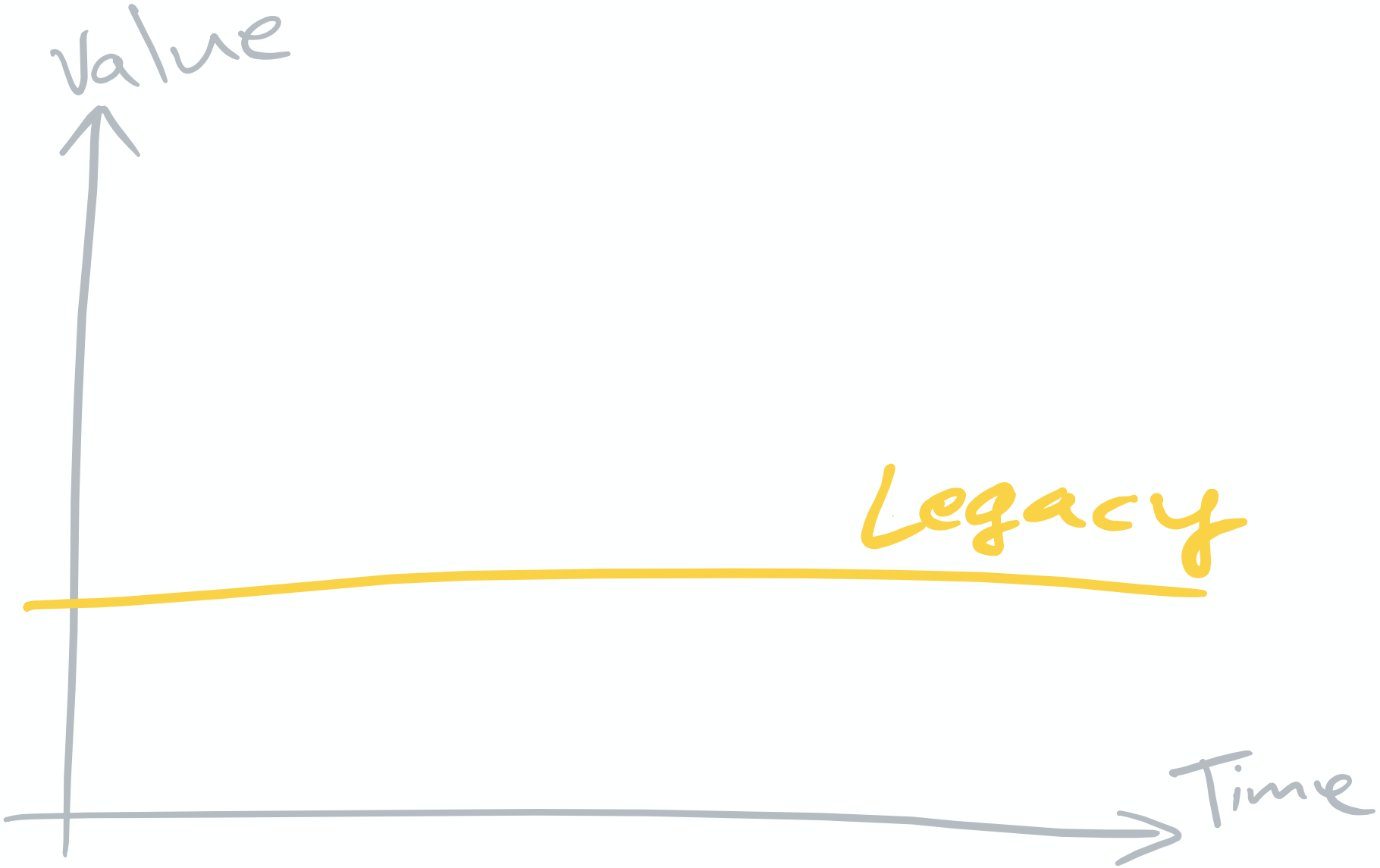

Value
Incremental
Time

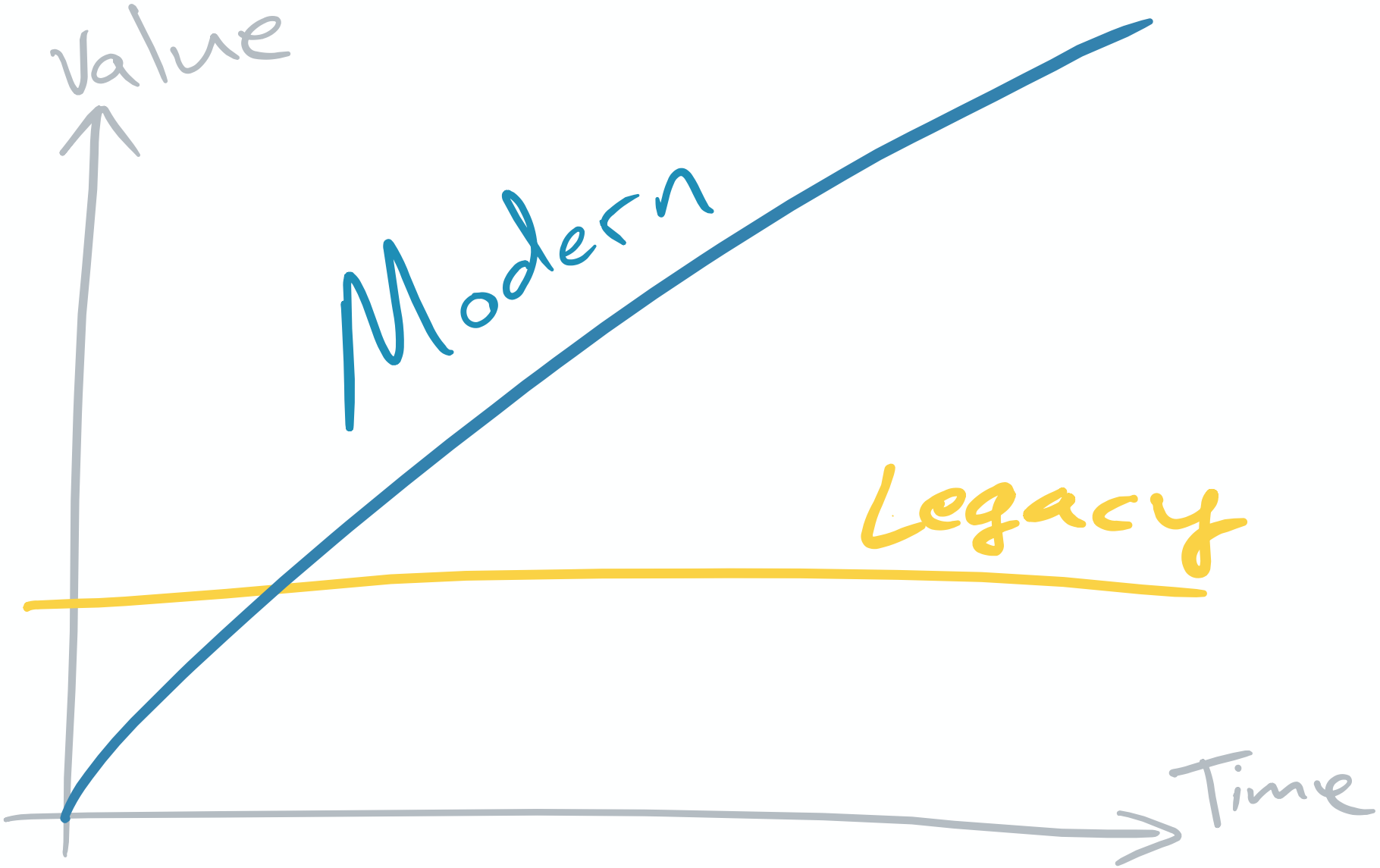

Value
Sequenced
Time

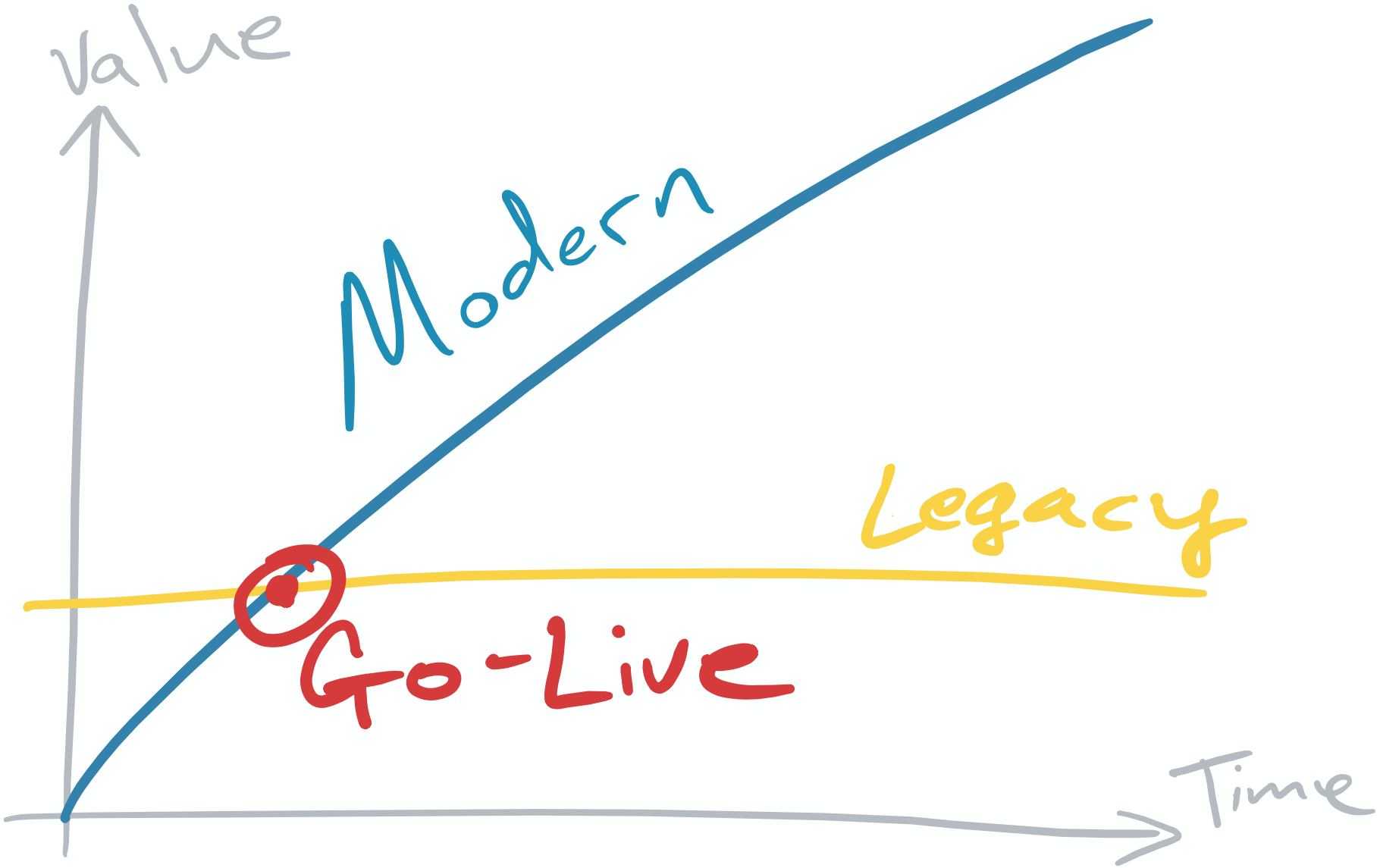

Value
Iterative
Time

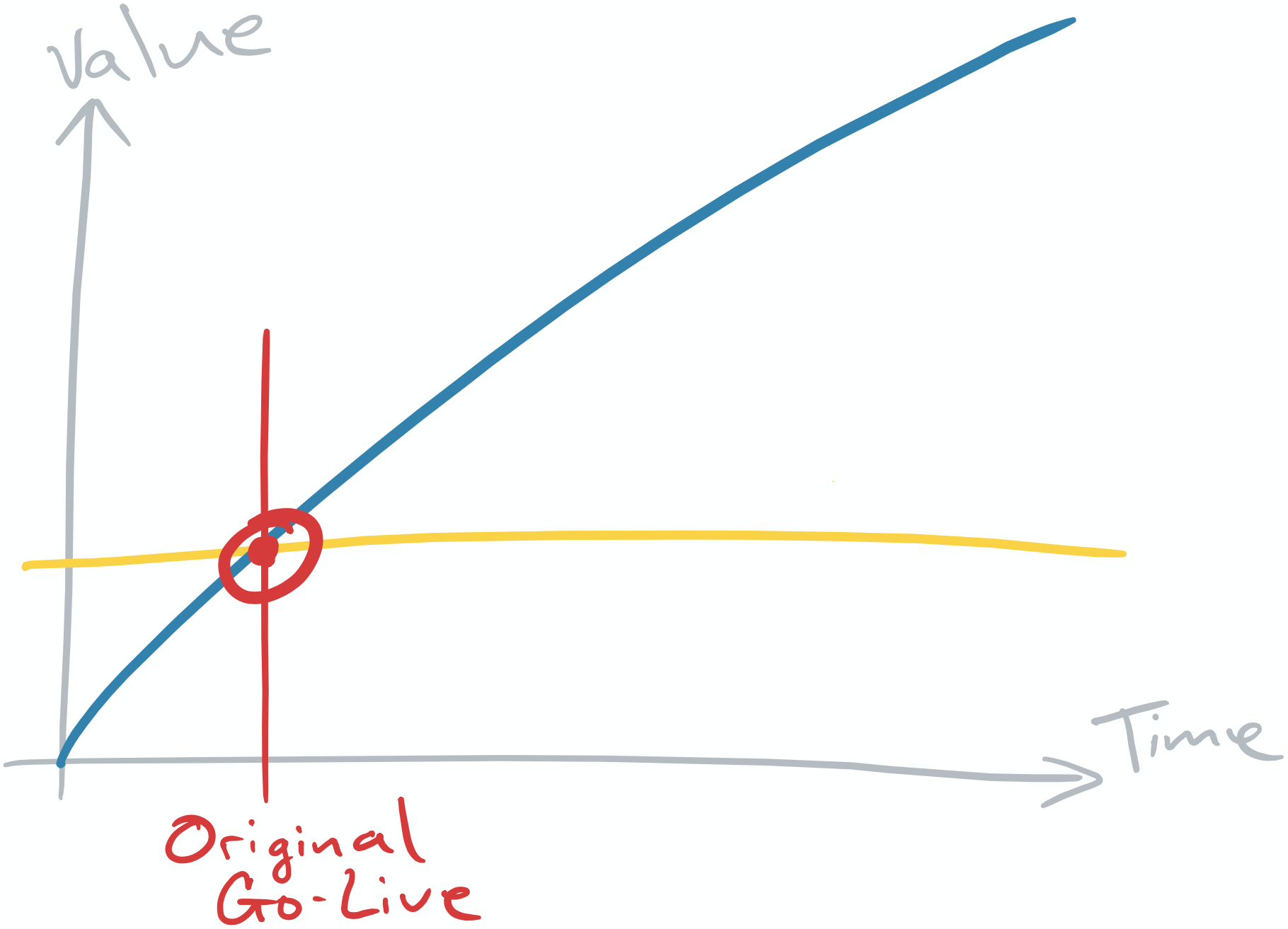
The Problem of Modernization

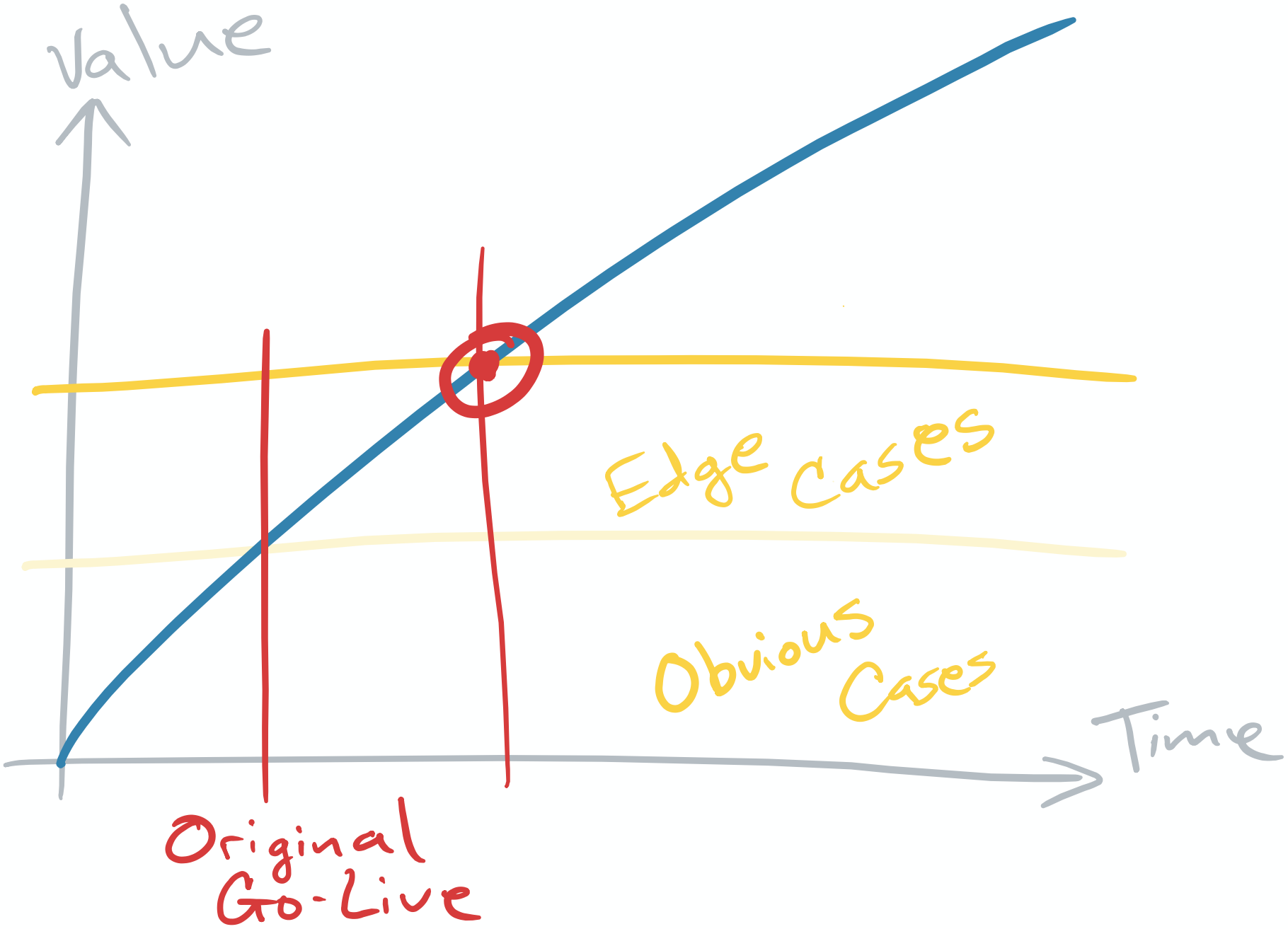

Value
Time

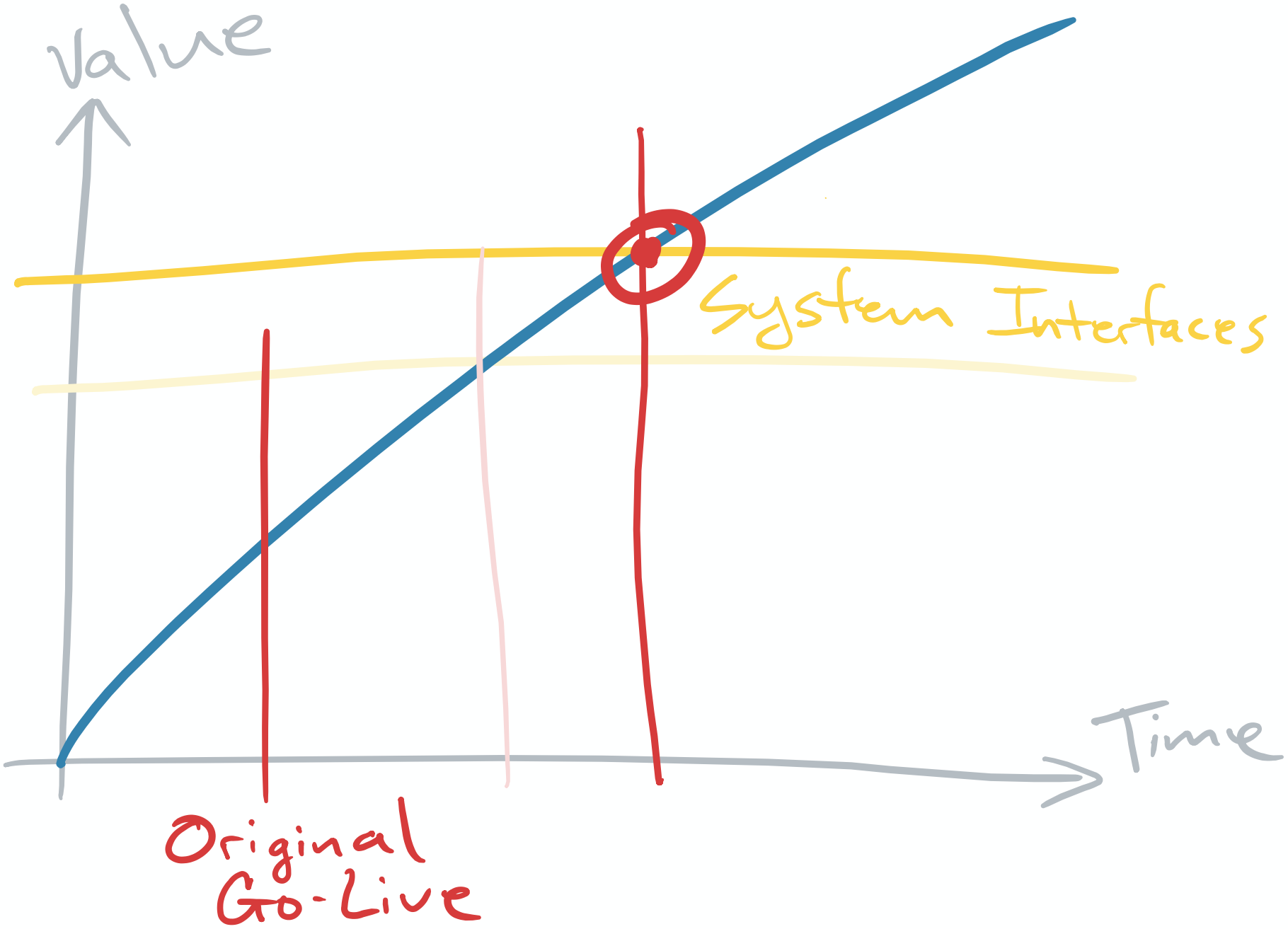

Value
Legacy
Time

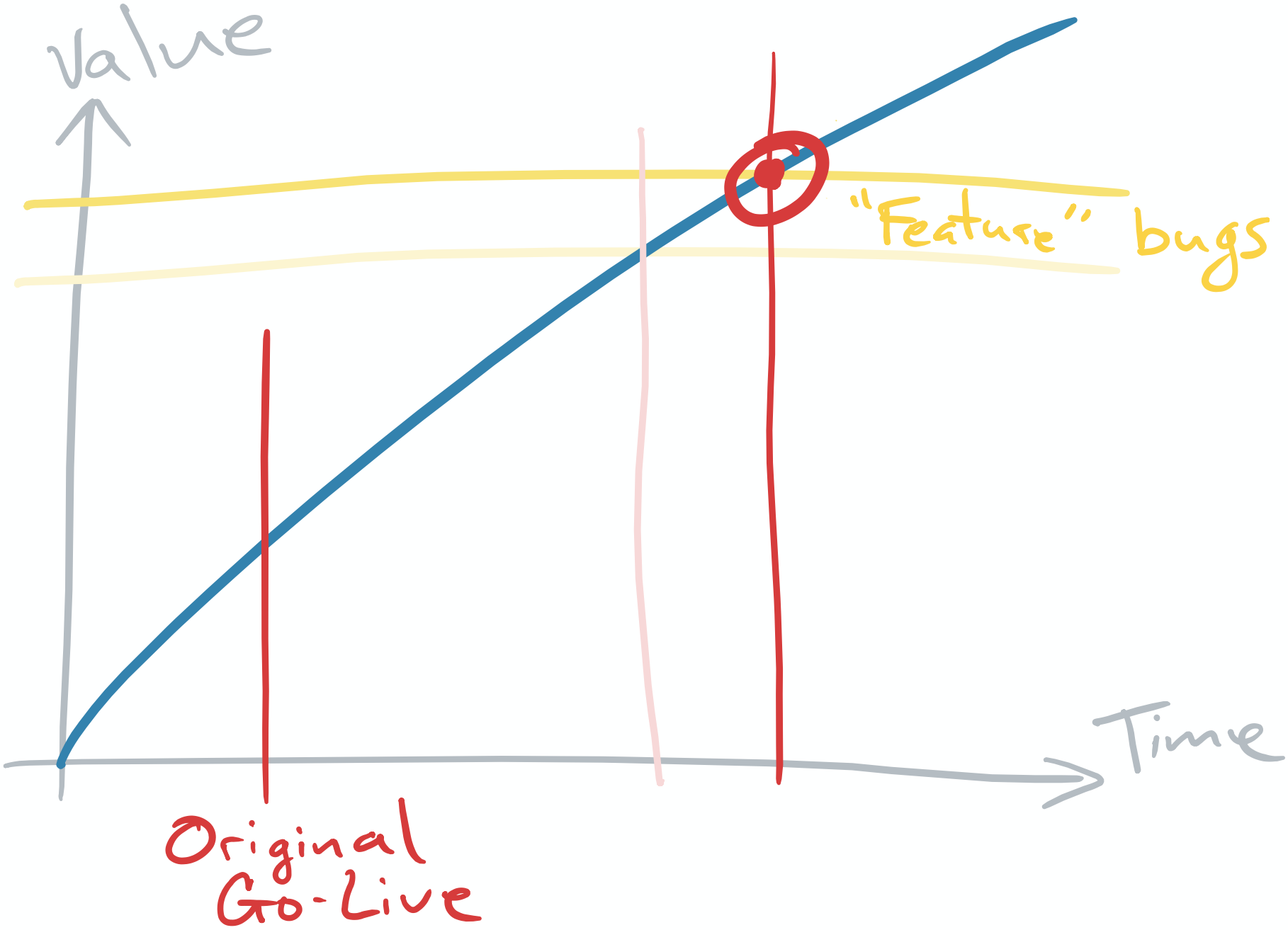

Value
Modern
Legacy
Time

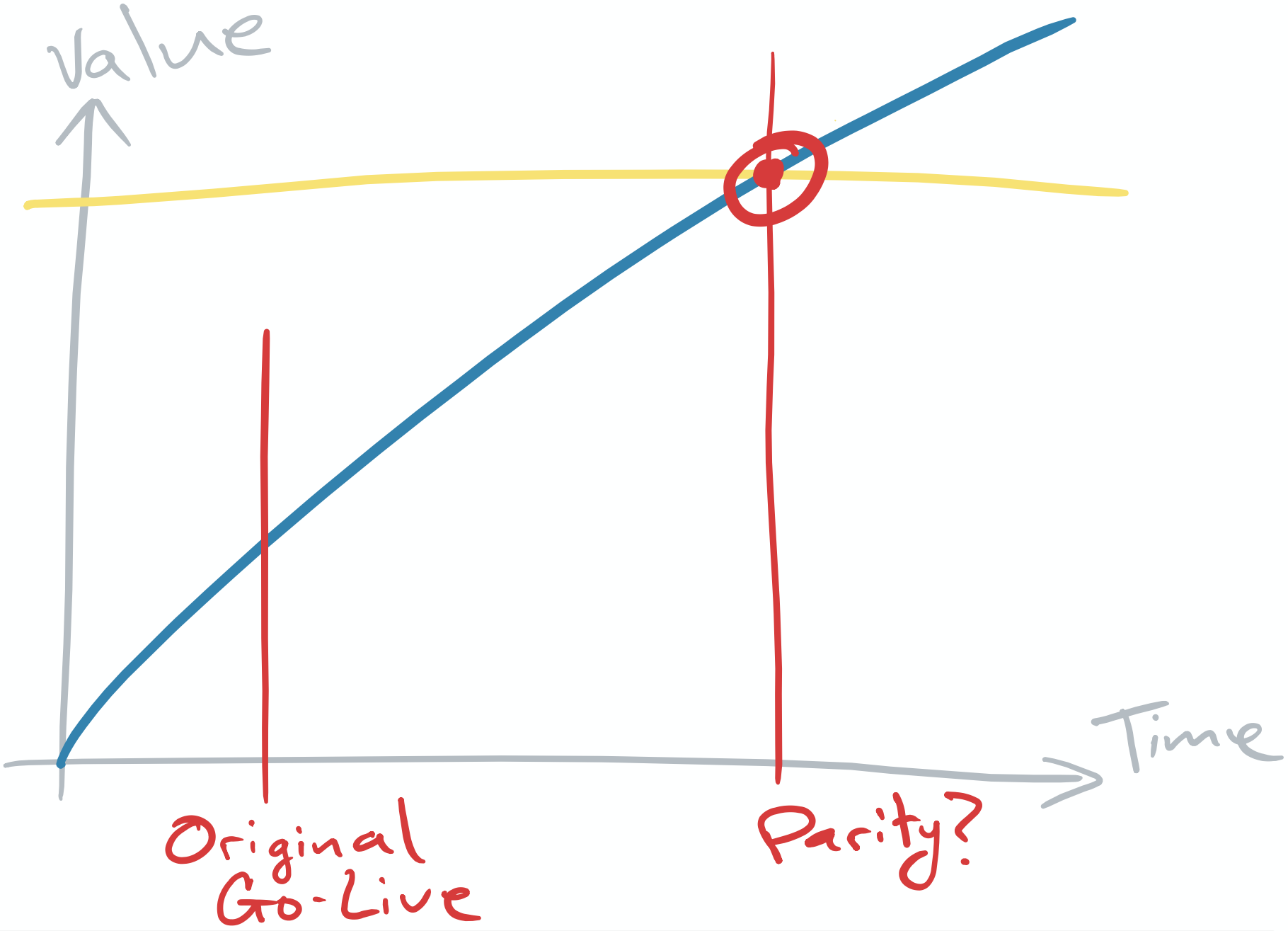

Value
Modern
Legacy
Go-Live
Time

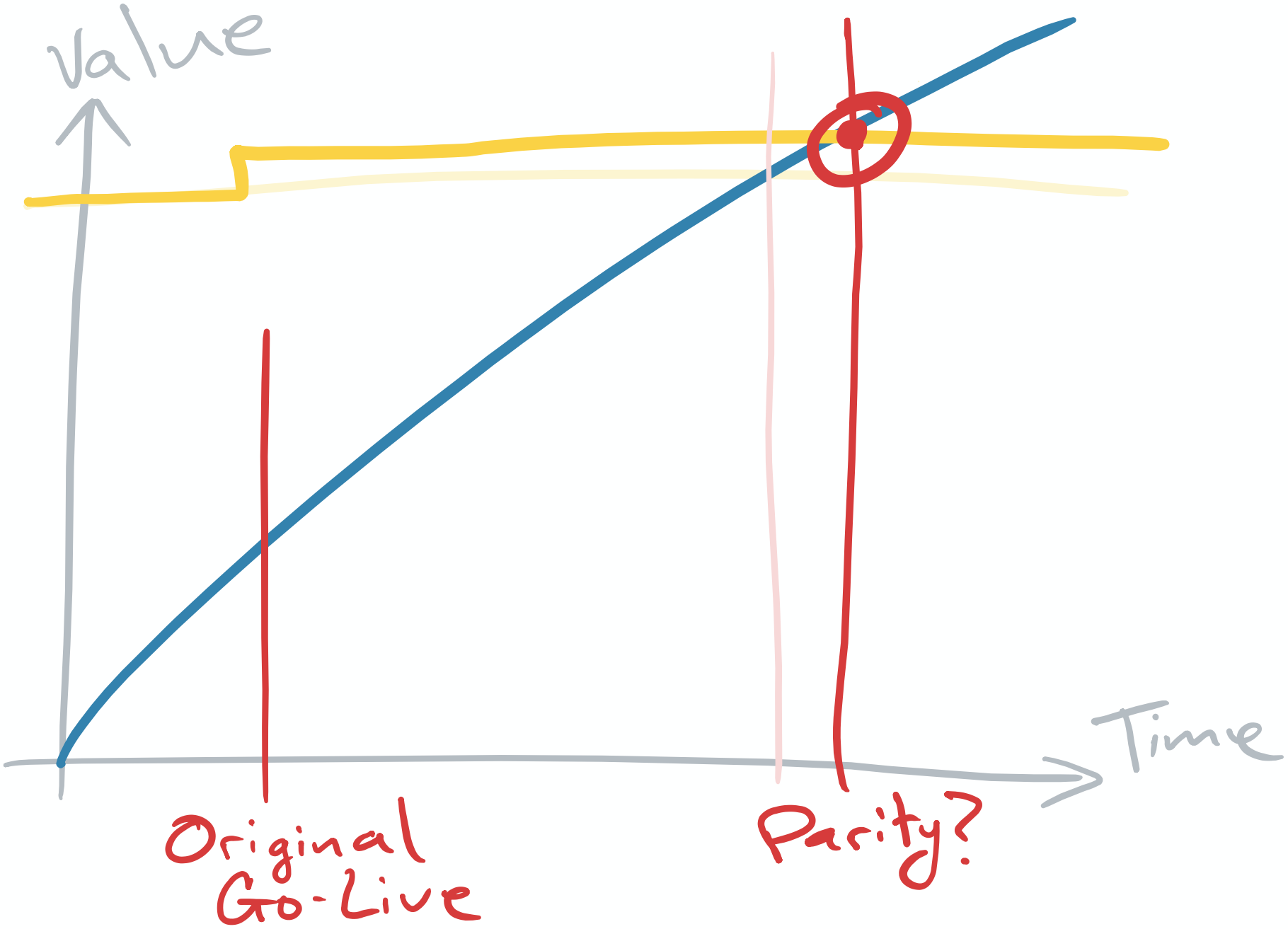

Value
Time
Original
Go-Live

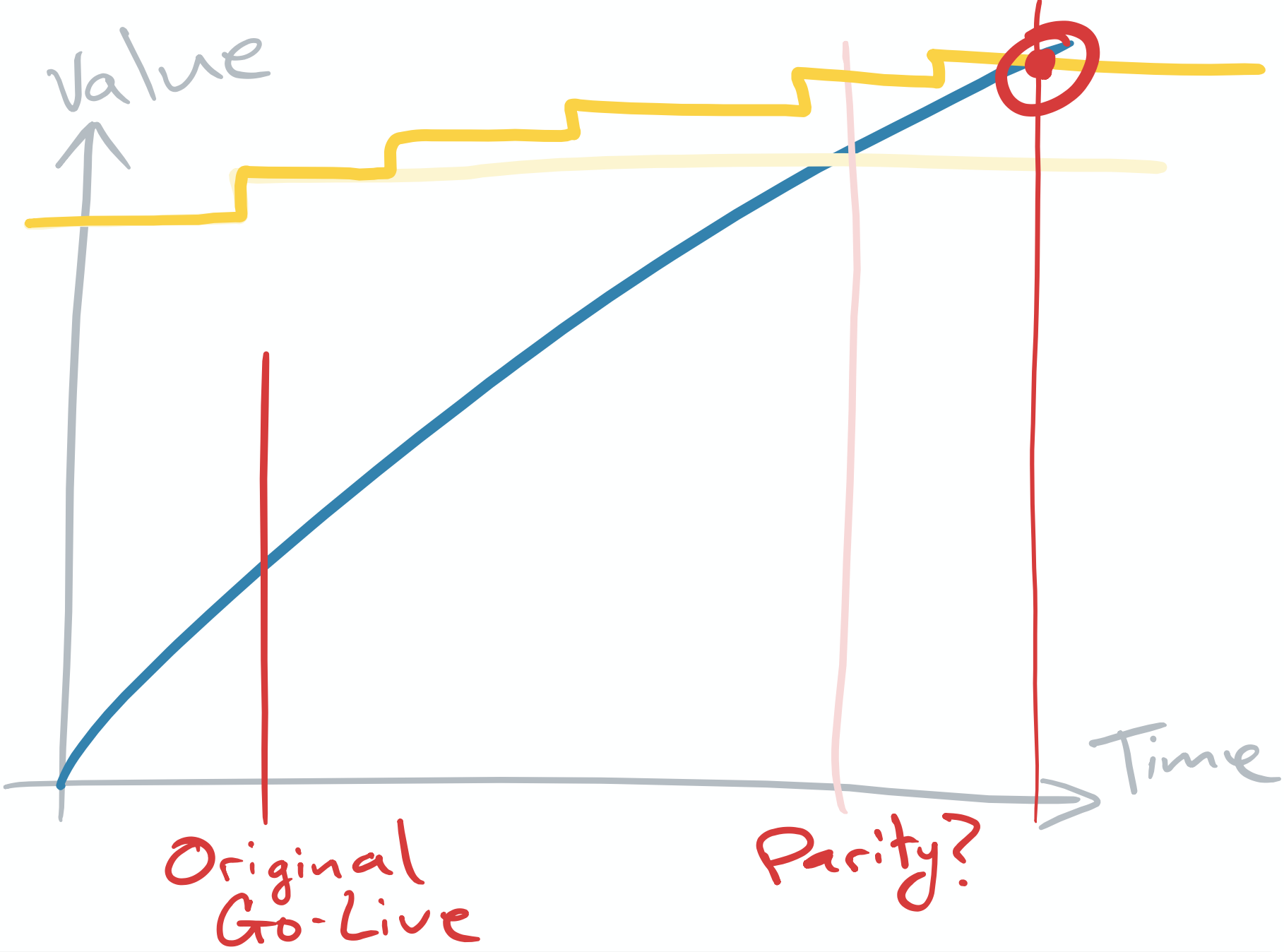

Value
Edge cases
Obvious Cases
Time
Original Go-Live

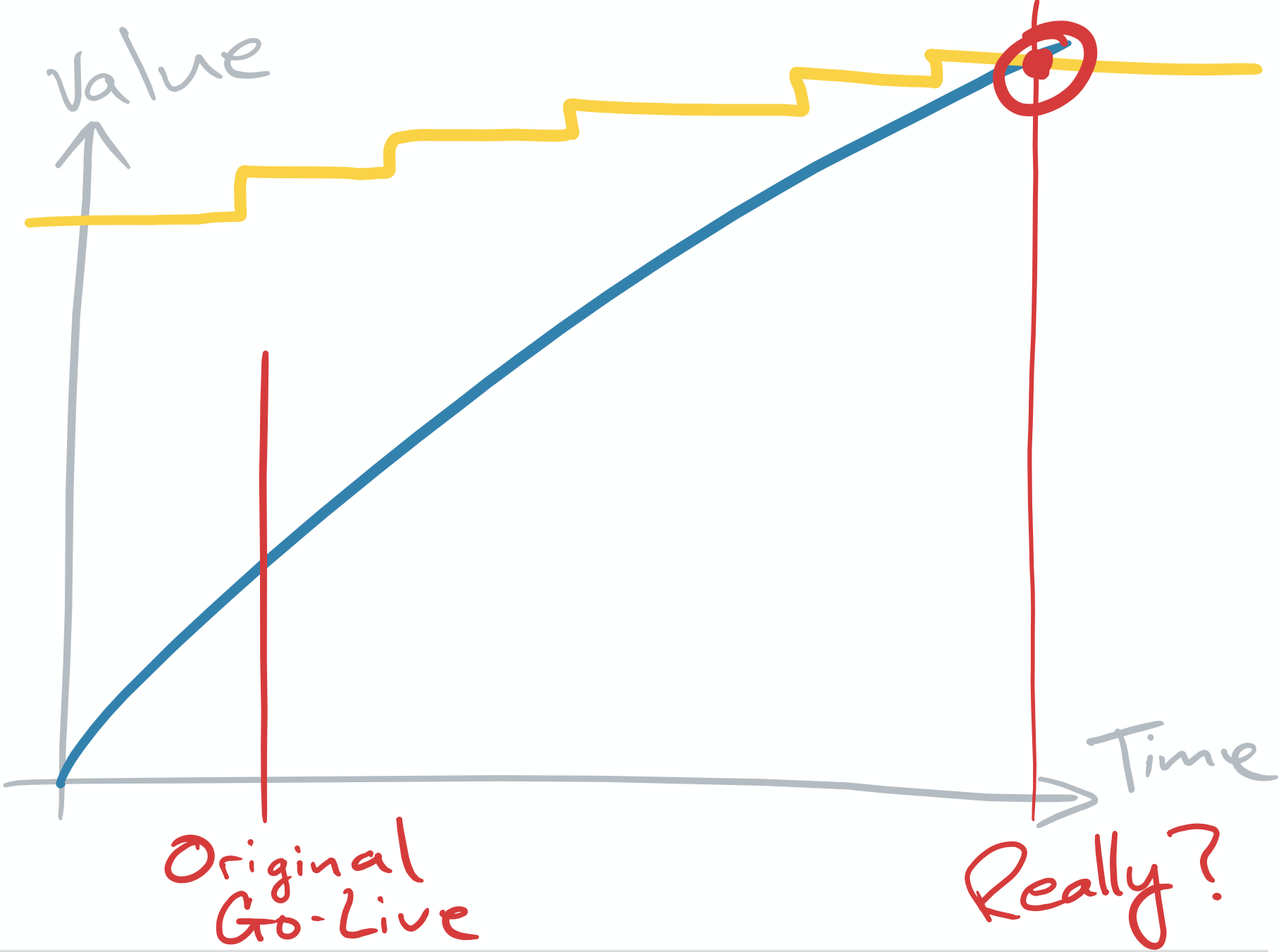

Value
System Interfaces
Time
Original
Go-Live

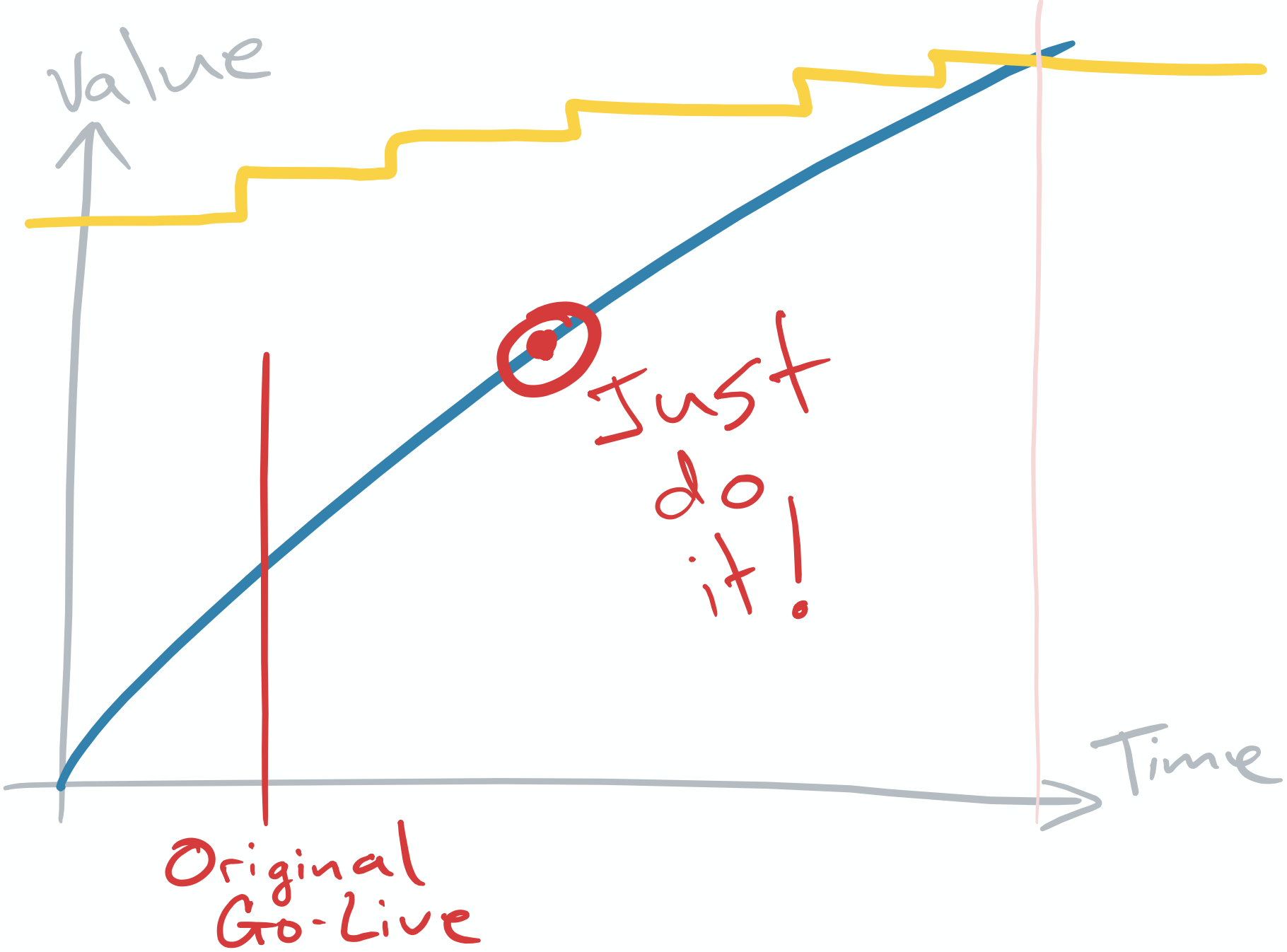

Value
"Feature" bugs
Time
Original
Go-Live

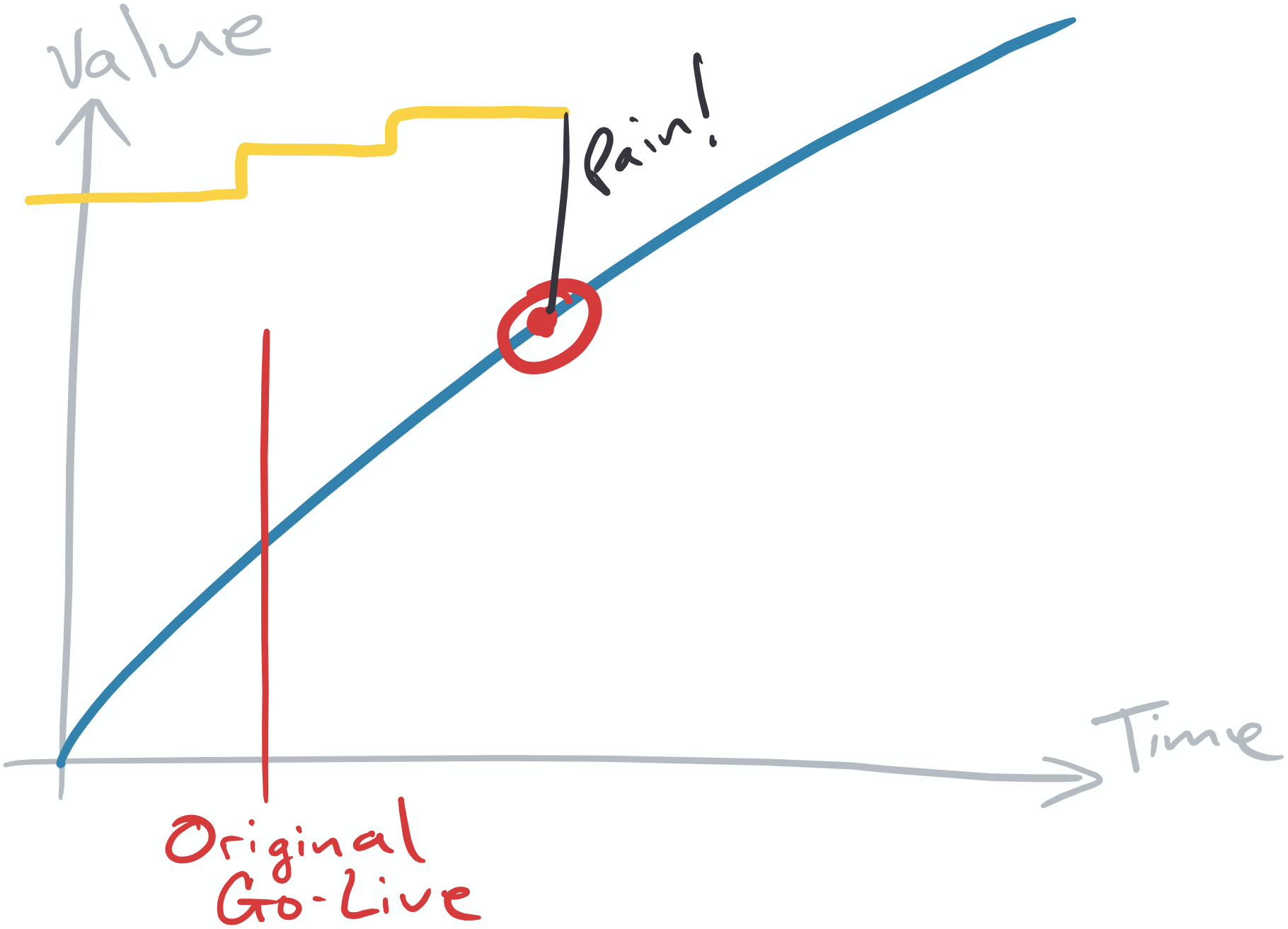

Value
Time
Original Go-Live
Parity?

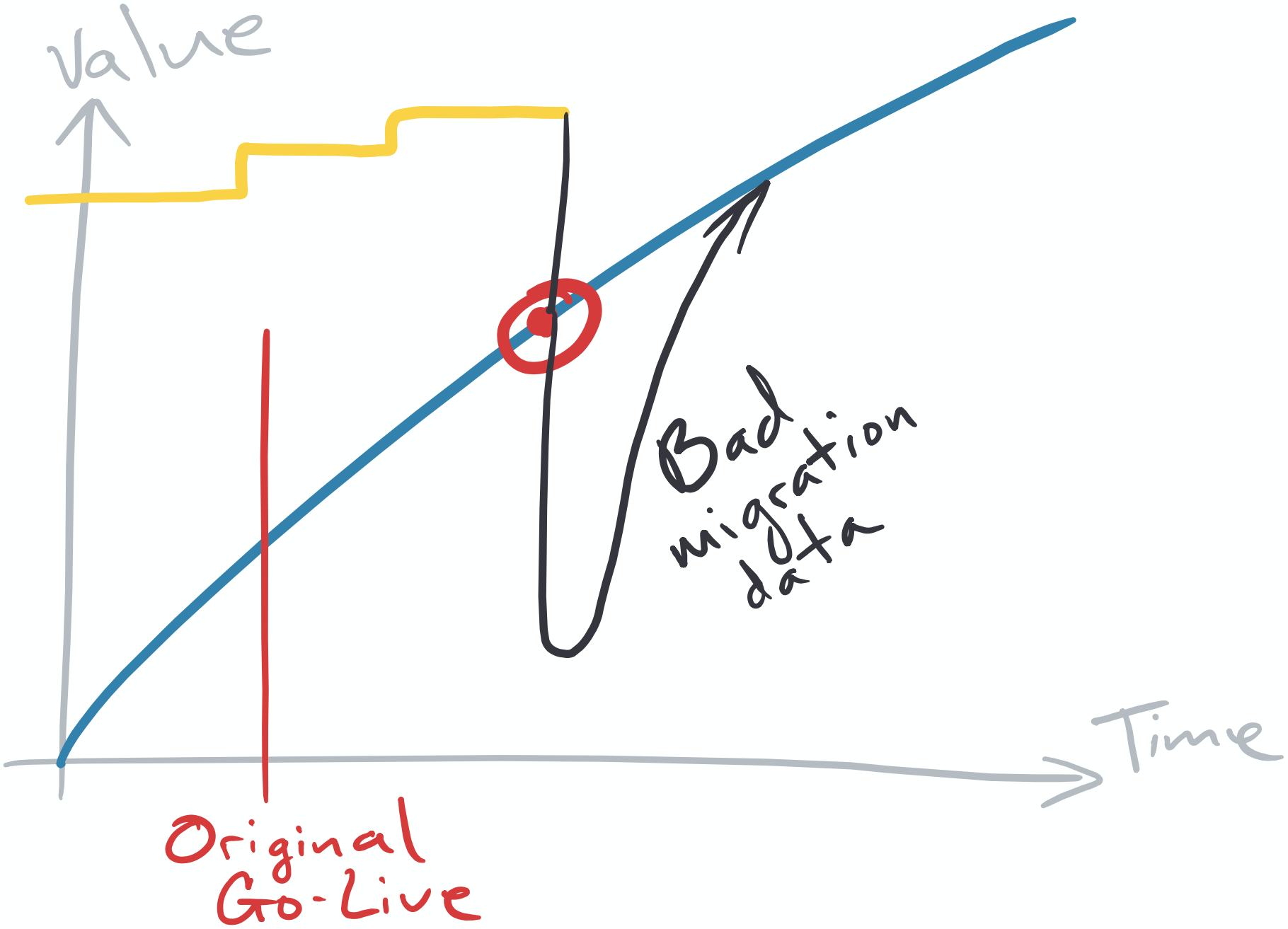

Value
Time
Original Go-Live
Parity?


Value
Time
Original Go-Live
Parity?

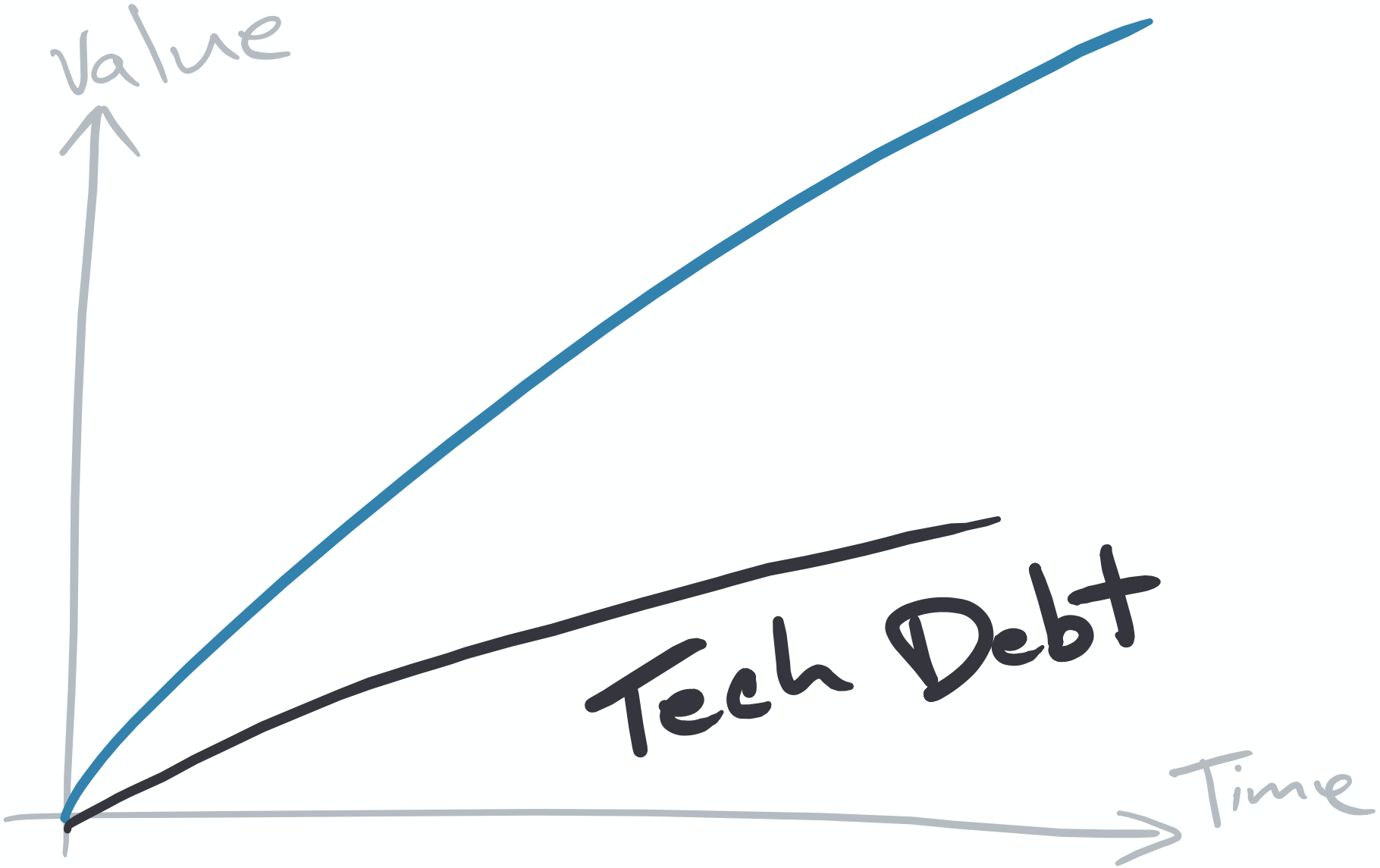

Value
Time
Original Go-Live
Really?

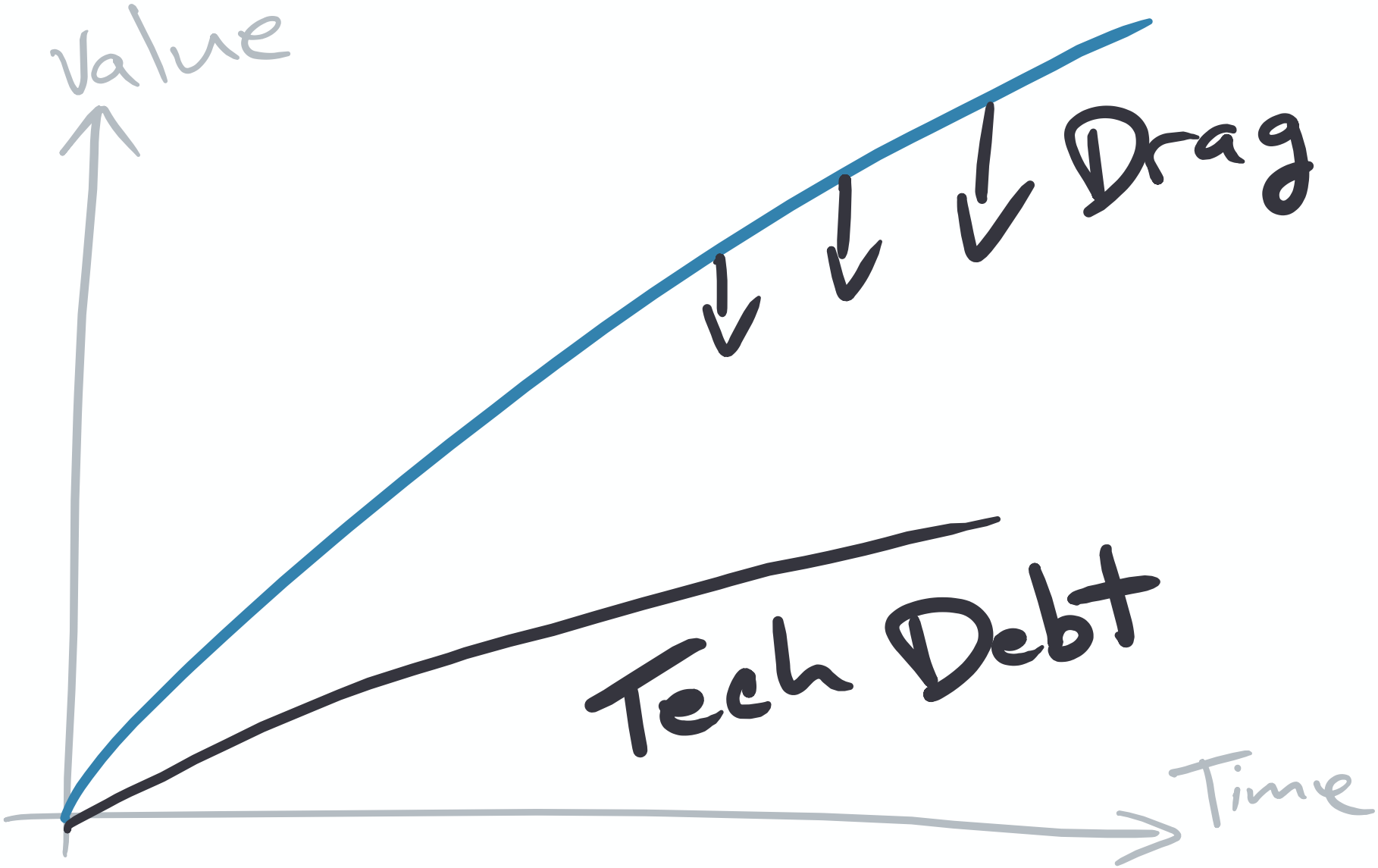

Value
Just do it!
Time
Original Go-Live

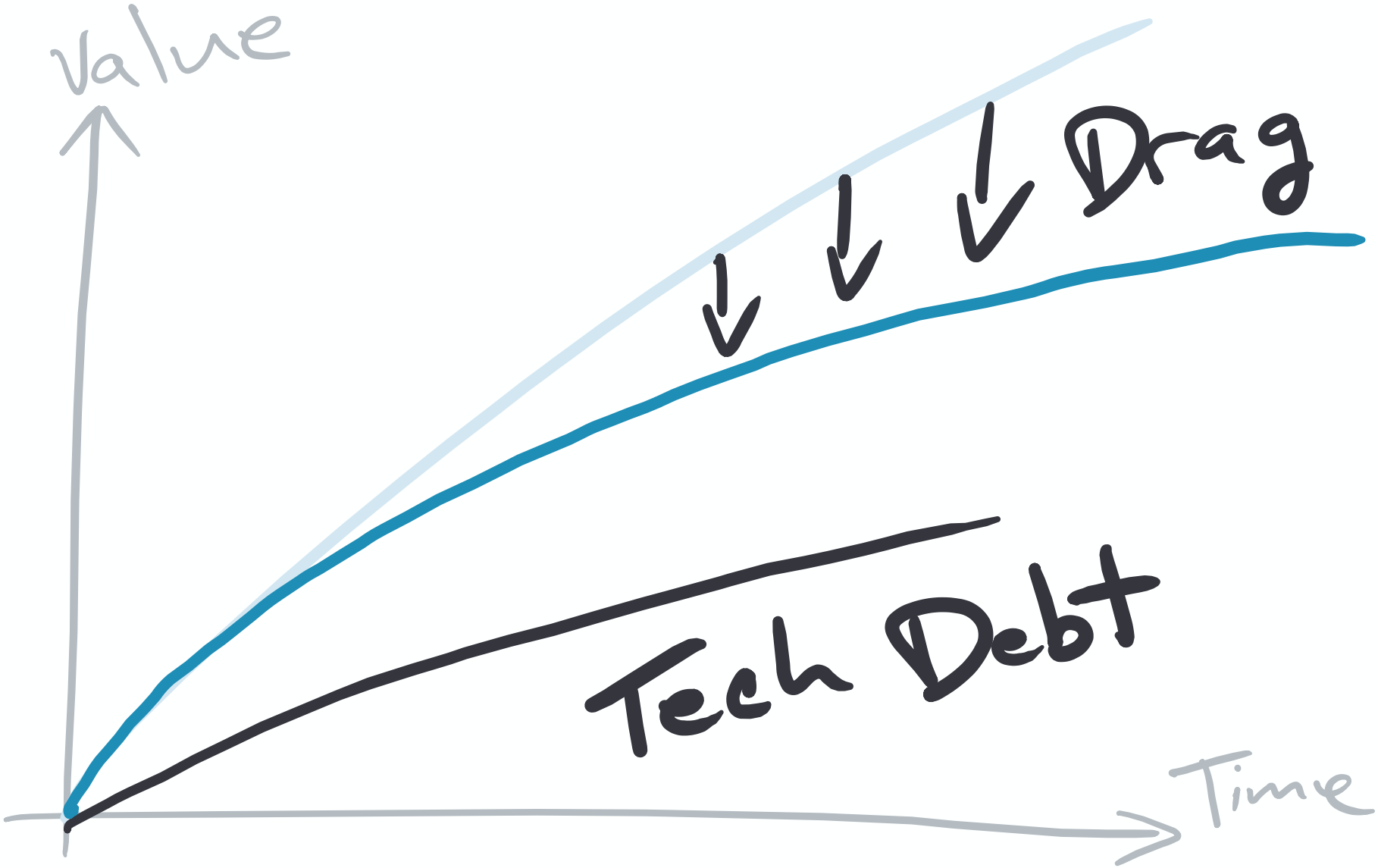

Value
Pain!
Time
Original
Go-Live

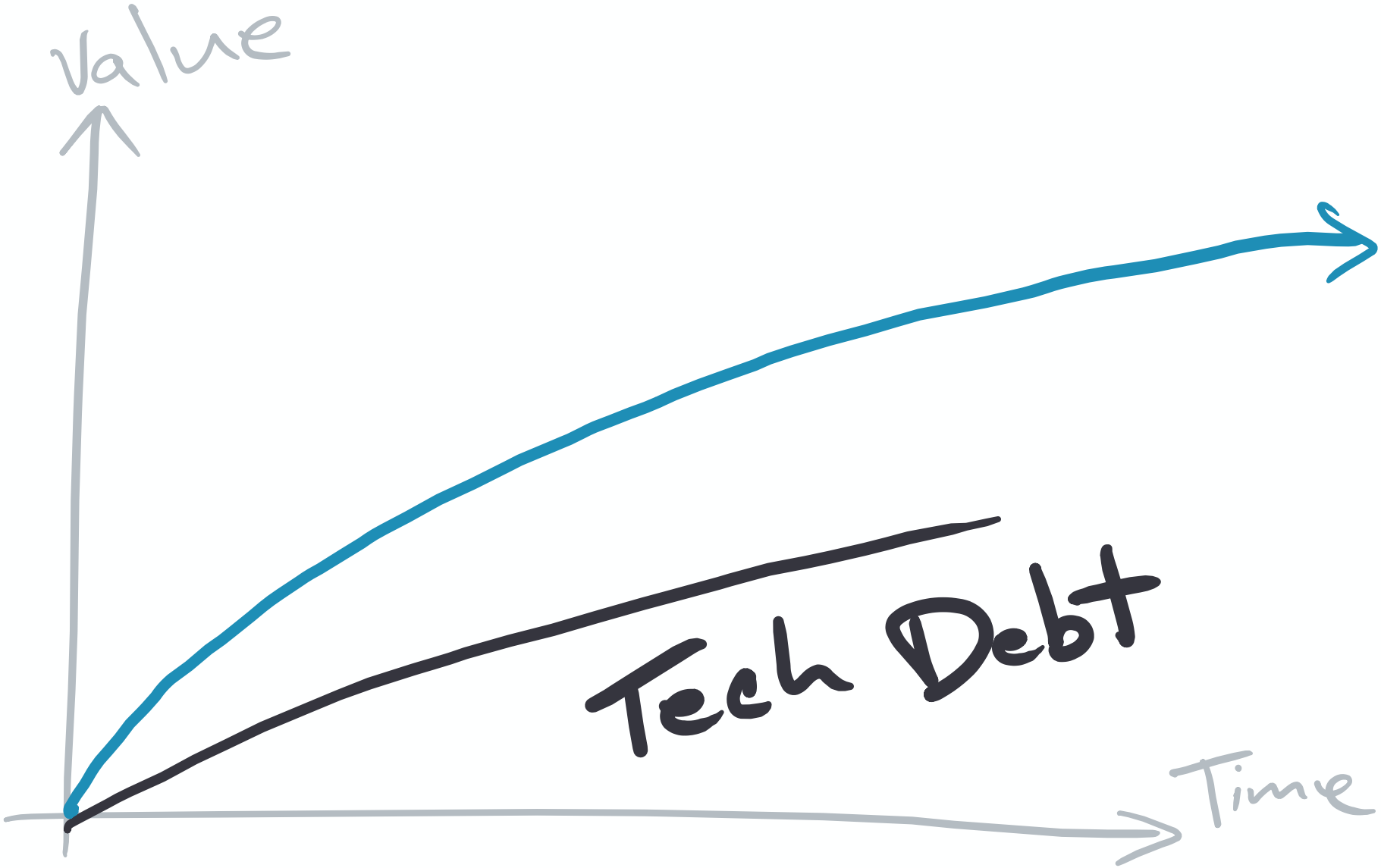

Value
Bad migration data
Time
Original Go-Live

Dirty Secret: The modernization was “rushed”


Value
Tech Debt
Time


Value
Drag
Tech Debt
Time


Value
Drag
Tech Debt
Time

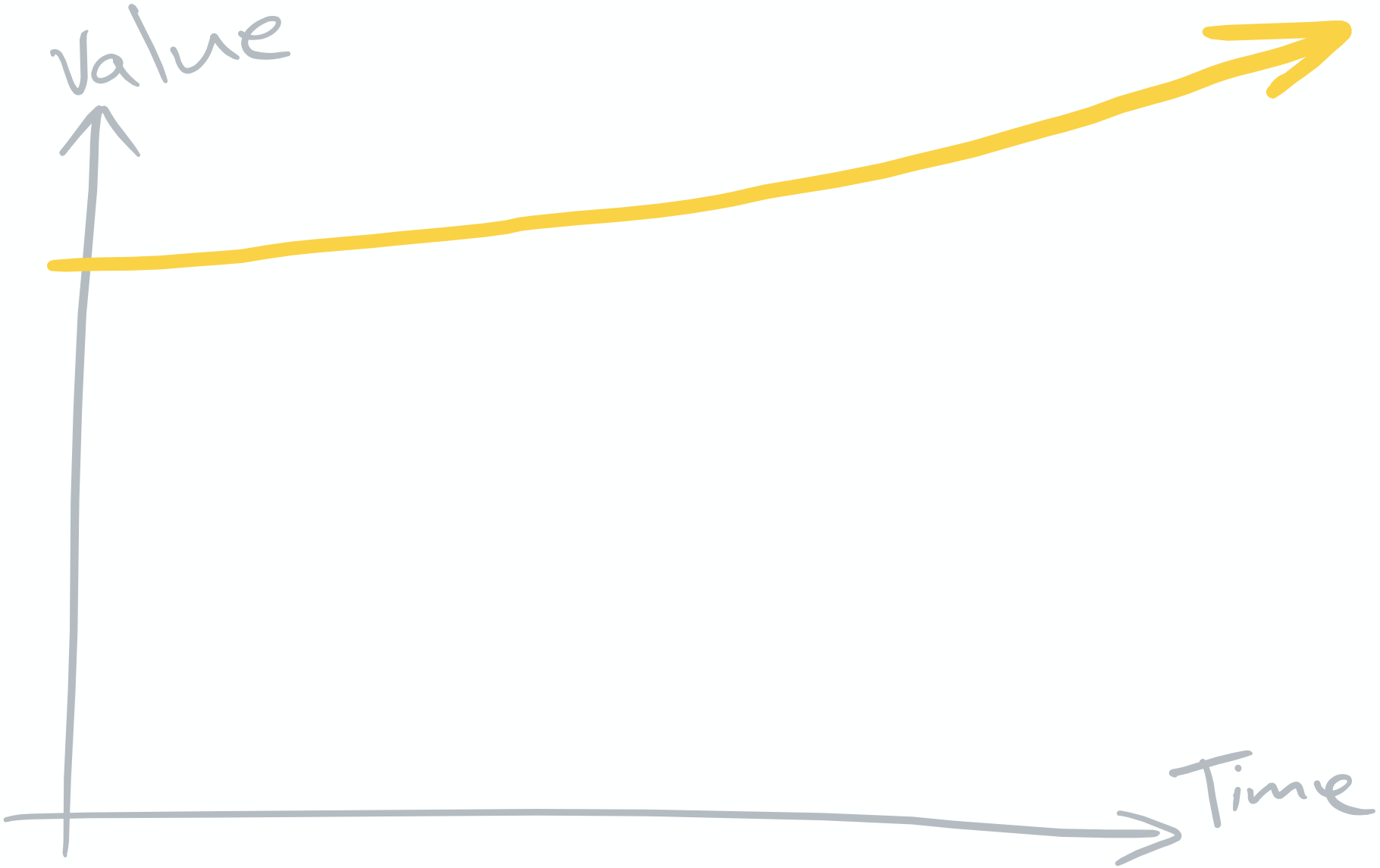

Value
Tech Debt
Time

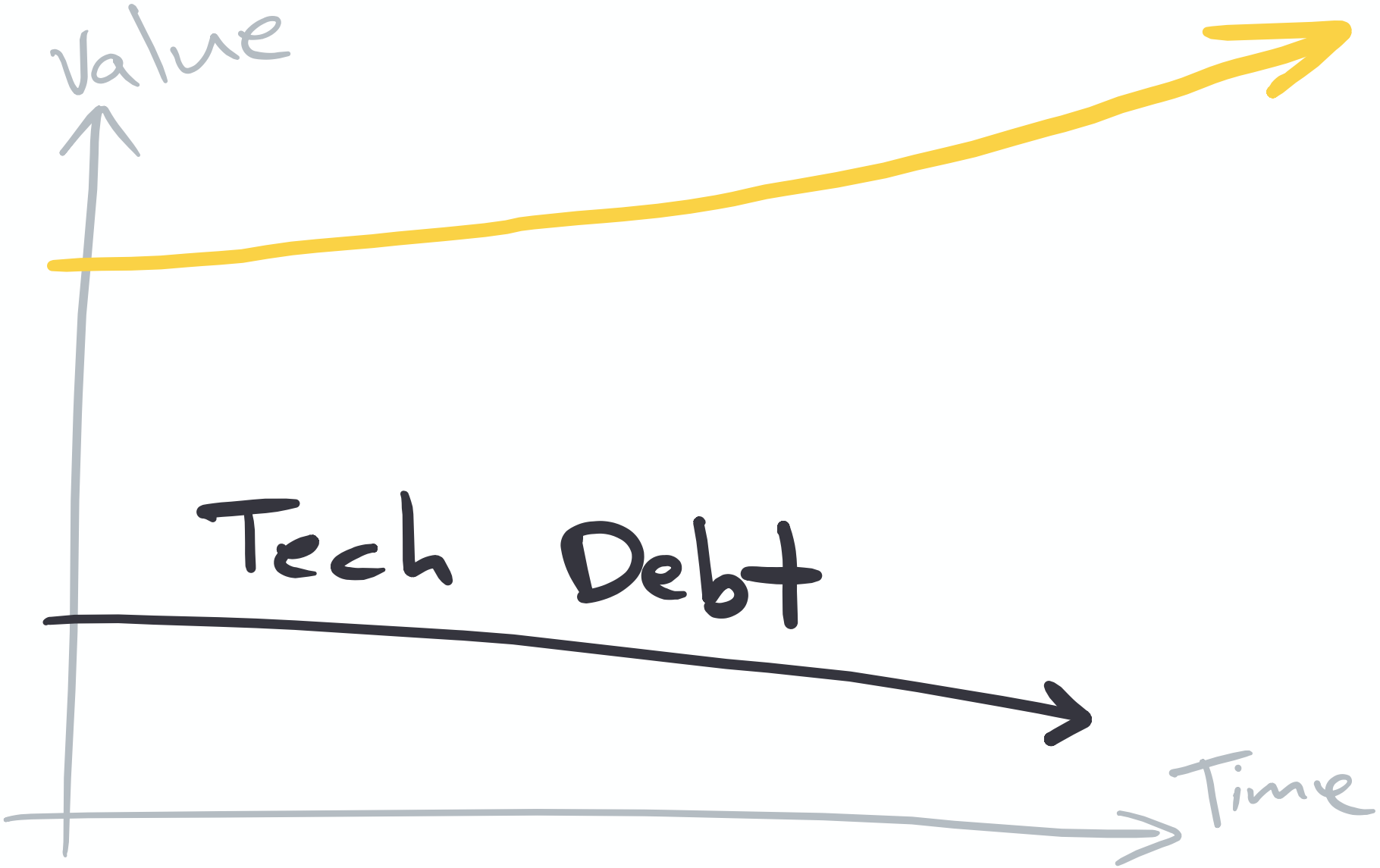
Every Legacy System was once “Modern”

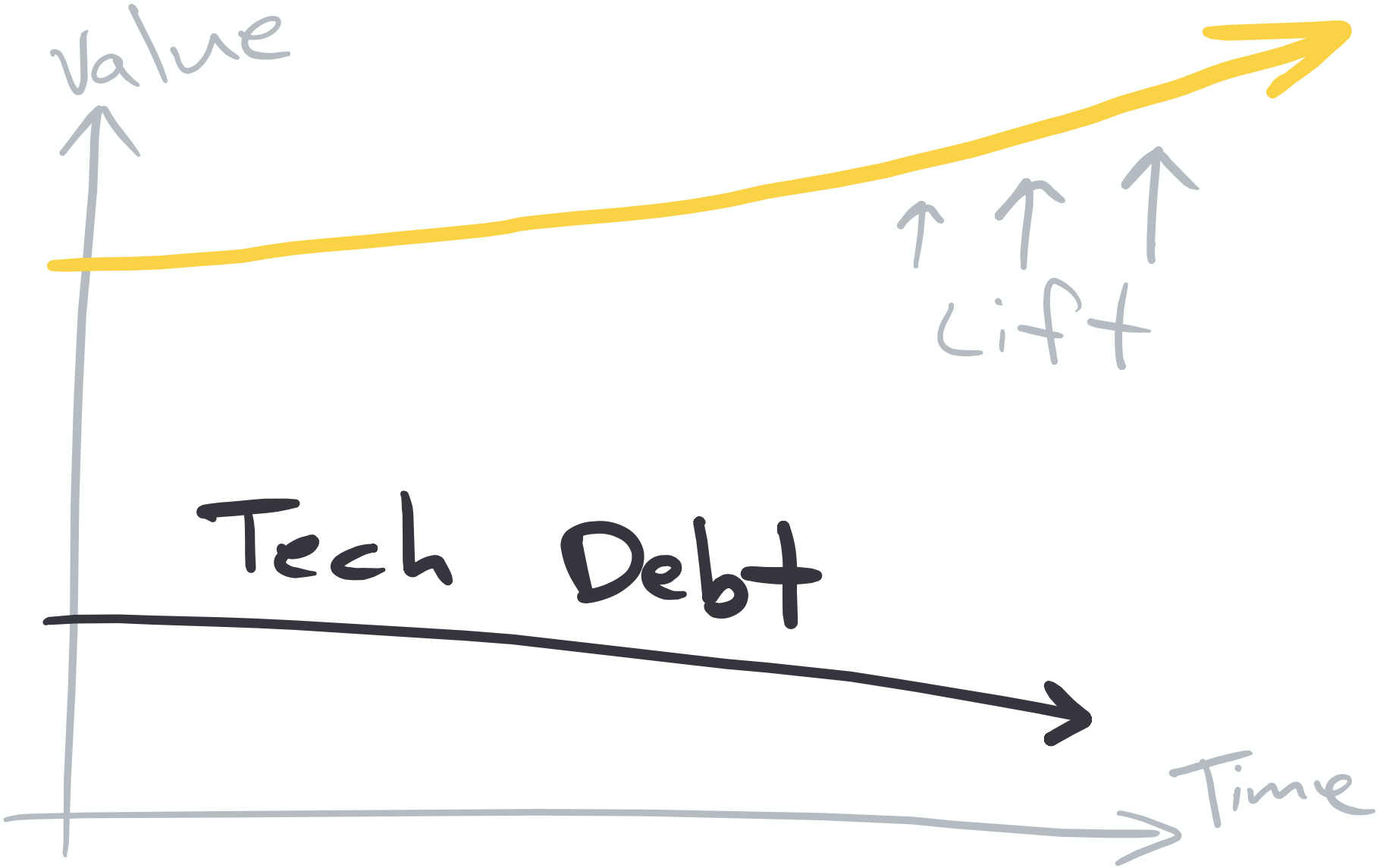
Congratulations!

We just built another legacy system.

Instead: Maintain and Renovate


Value
Time

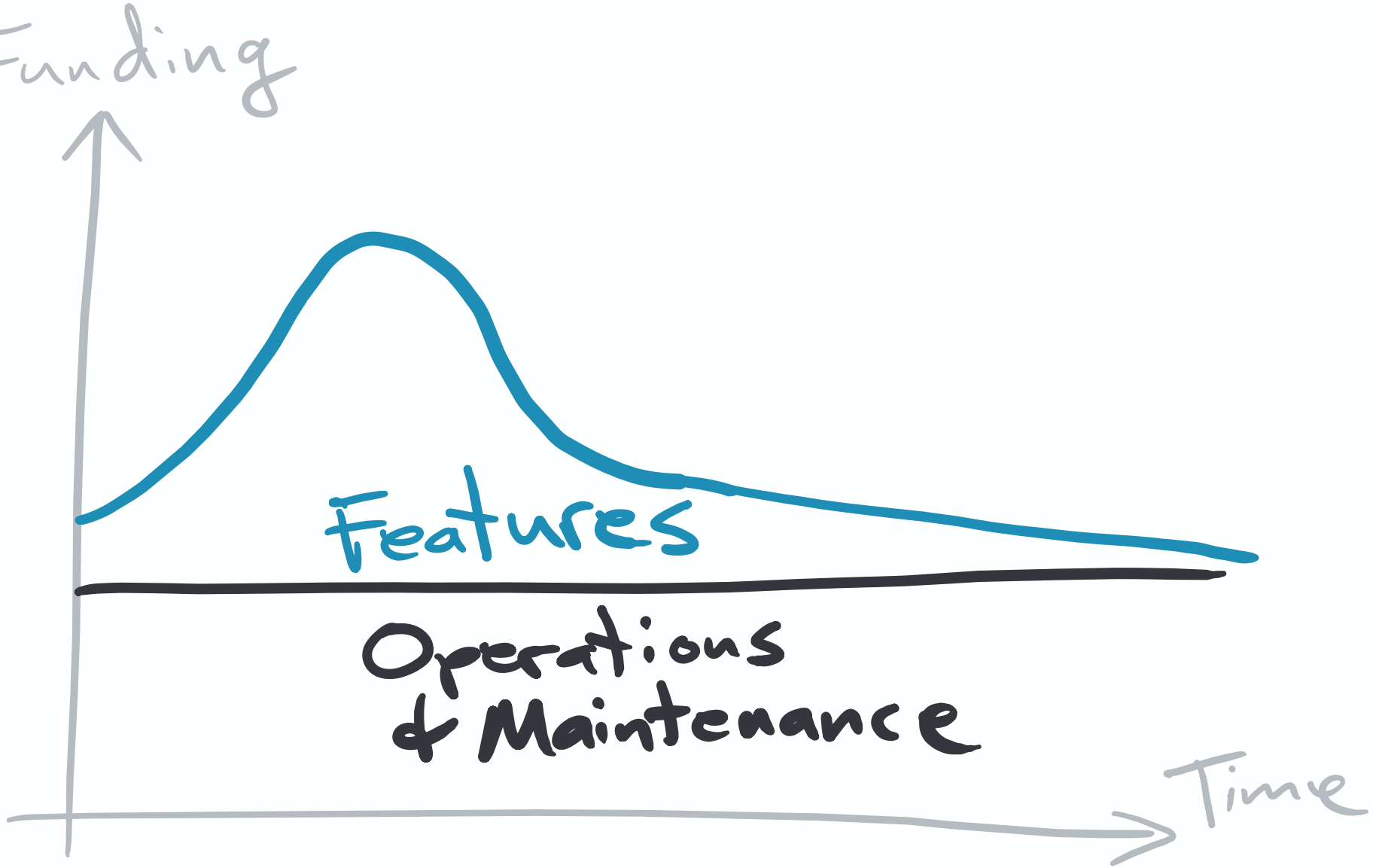

Value
Tech Debt
Time

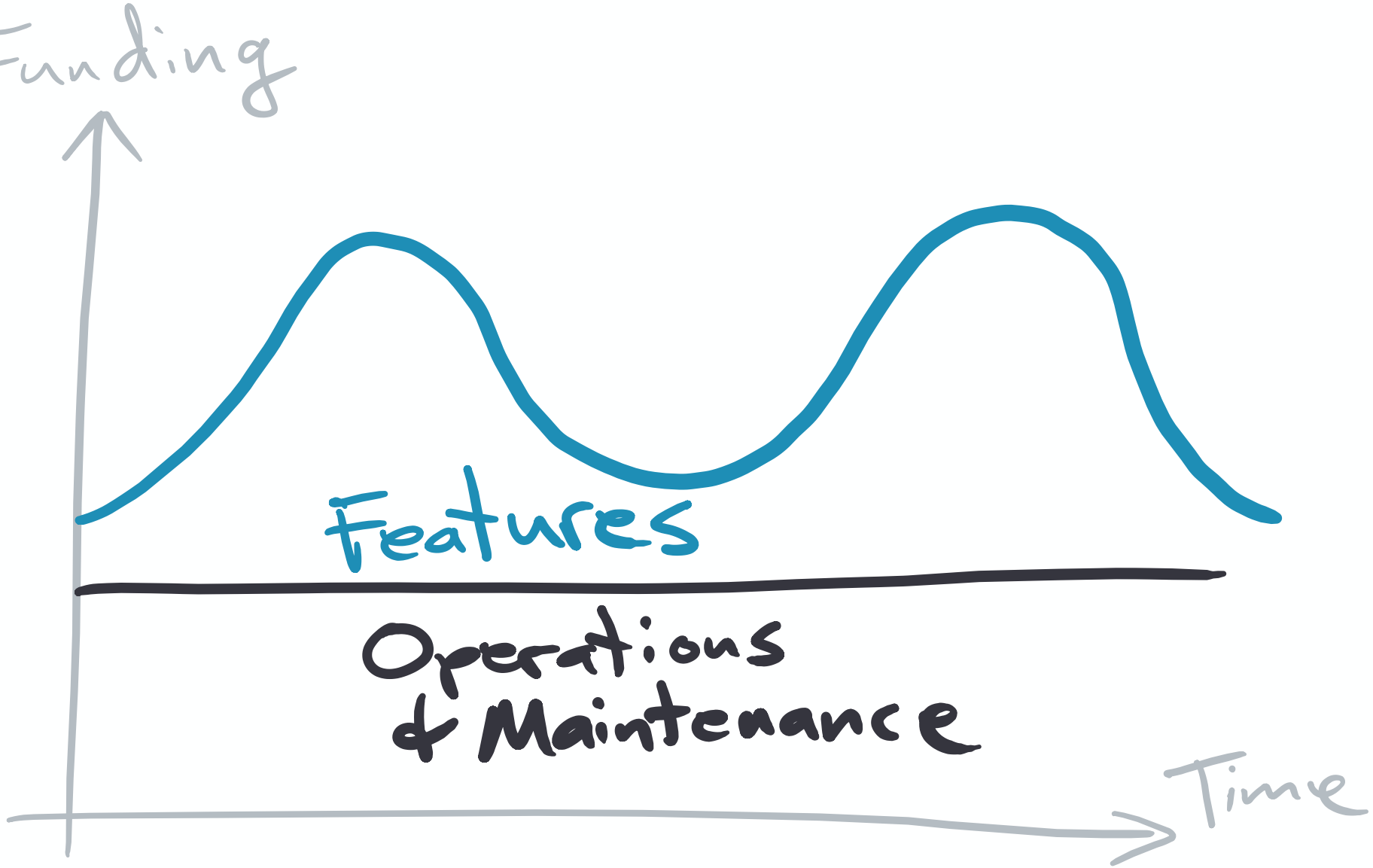

Value
Lift
Tech Debt
Time


Funding
Time

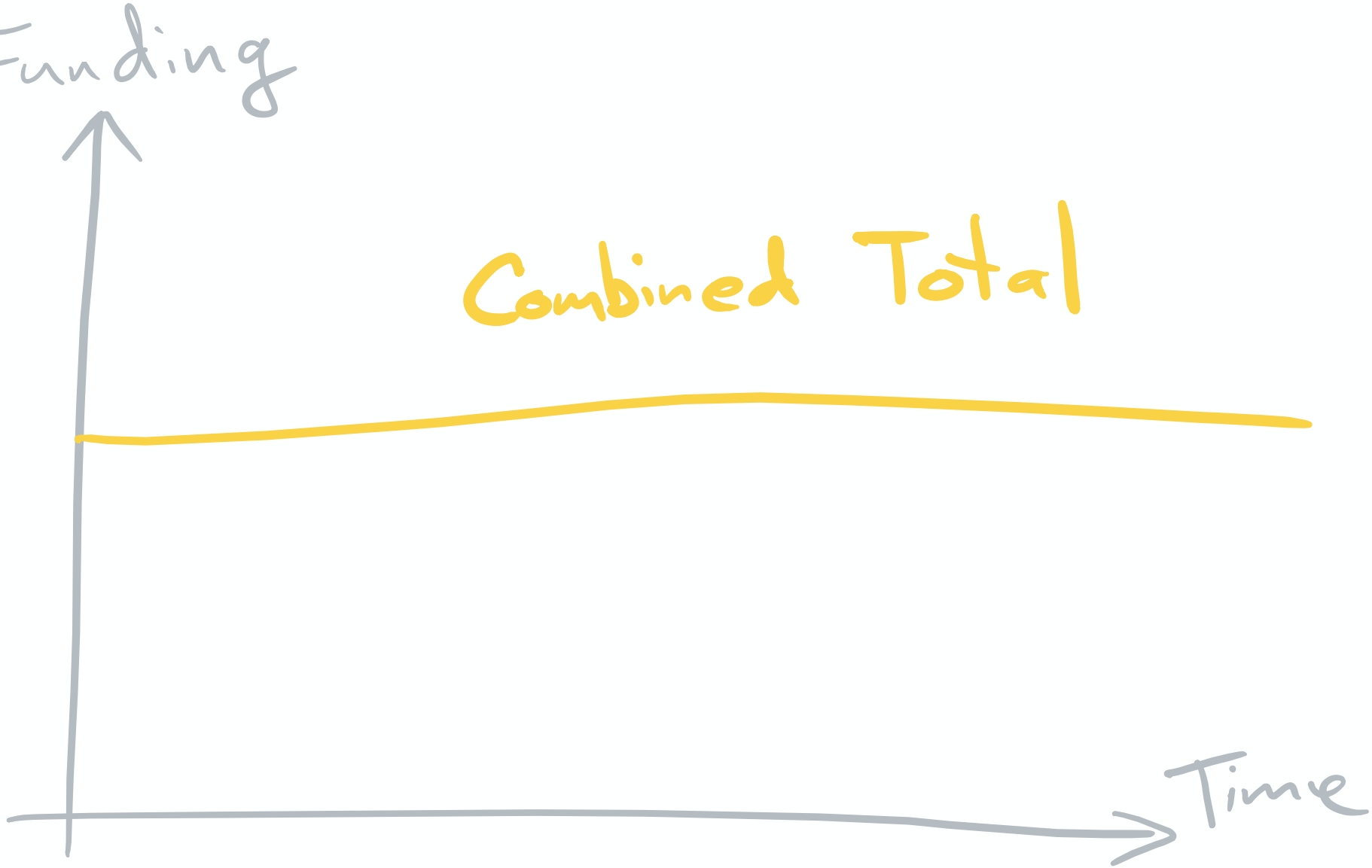

Funding
Time

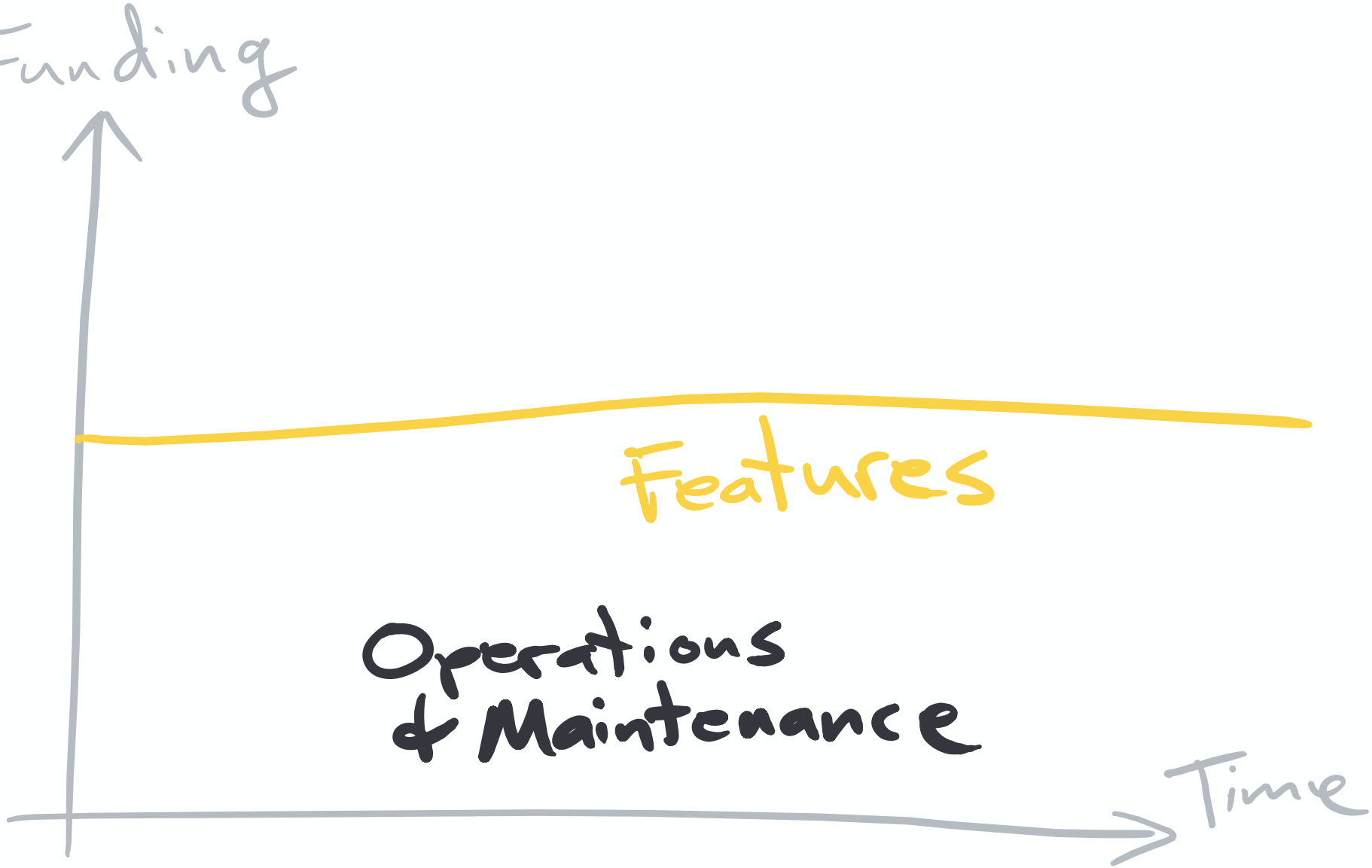

Funding
Operations
& Maintenance
Time

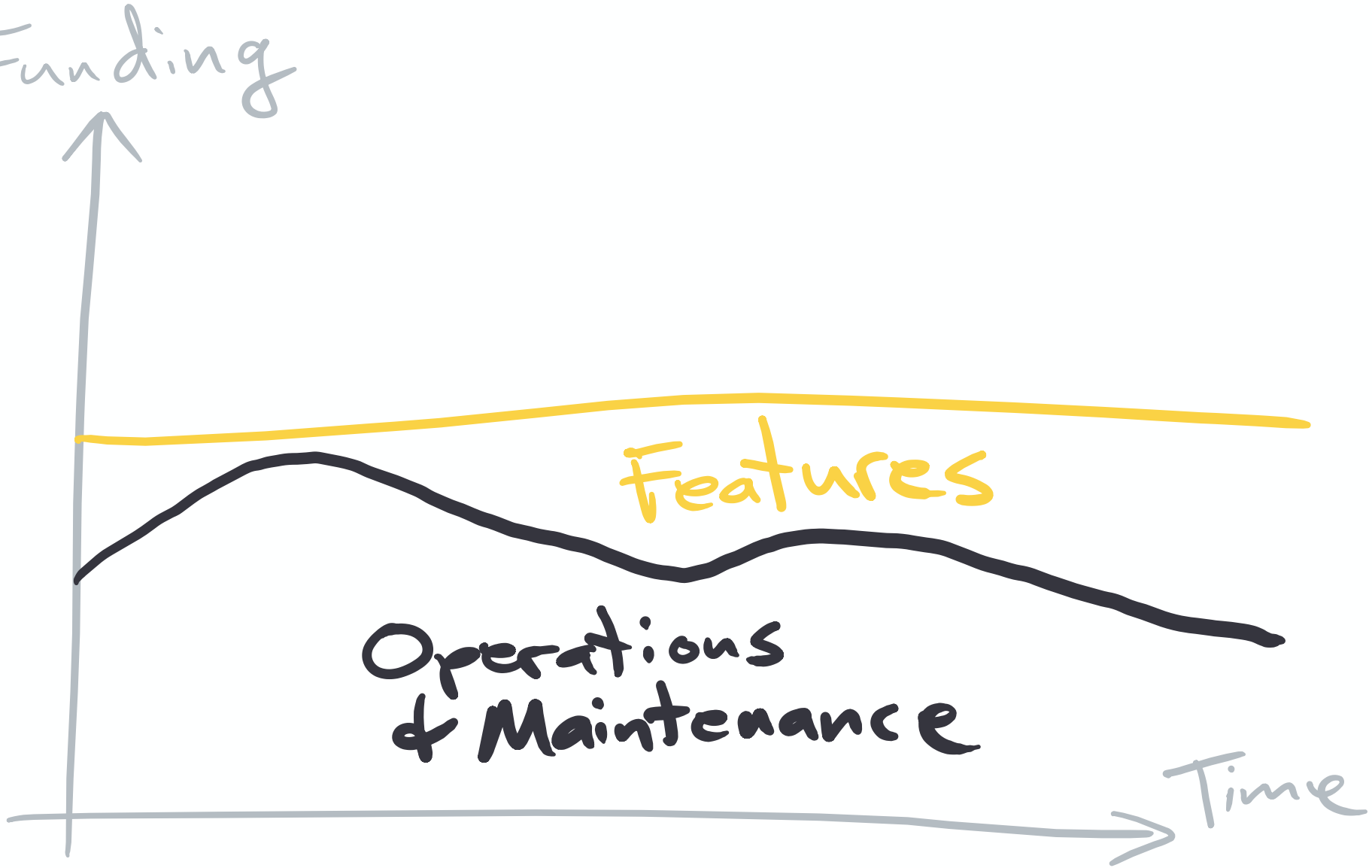

Funding
Features
Operations
& Maintenance
Time


Funding
Features
Operations
& Maintenance
Time


Funding
Time


Funding
Combined Total
Time


Funding
Features
Operations
& Maintenance
Time


Funding
Features
Operations
& Maintenance
Time

Another Metaphor: Your Home

Does it really need to be demolished?

The “Home” Metaphor

The “Home” Metaphor

- Long-term investment
- In daily use
- Needs maintenance
- Renovate as needed
- Make it safe to renovate with girders, braces, shoring, & jacks

The “Home” Metaphor

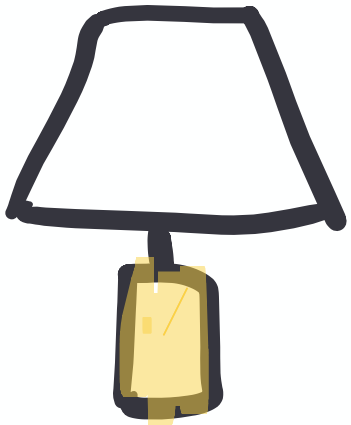
It is one thing to build a new house, but to build a new house in the same exact spot?

The “Home” Metaphor

The “Home” Metaphor

Live On-Site

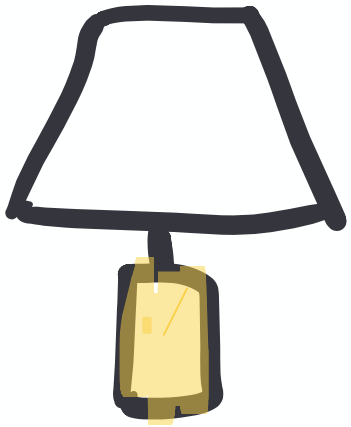
This is critical to the enterprise. Moving out for a year is a costly and risky option.



The “Home” Metaphor

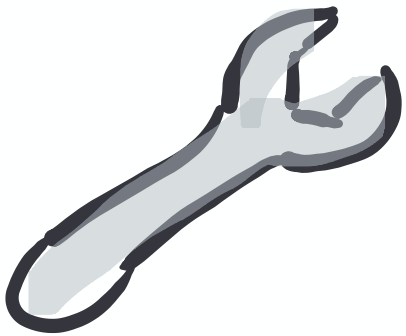
Live On-Site

This is critical to the enterprise. Moving out for a year is a costly and risky option.



Maintain

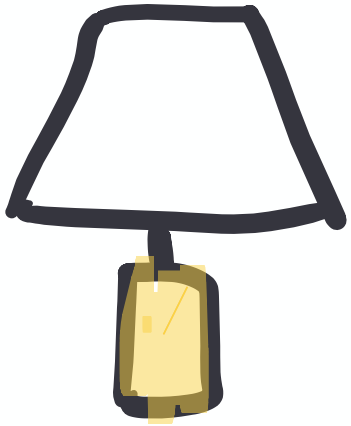
Prevention over cure. Don’t save money now at the risk of catastrophe later.



The “Home” Metaphor

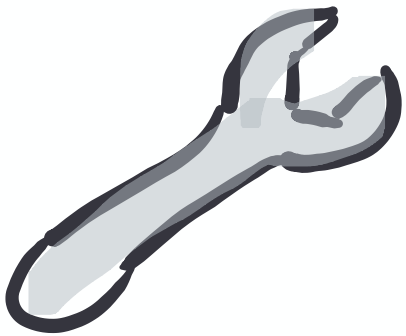
Live On-Site

This is critical to the enterprise. Moving out for a year is a costly and risky option.



Maintain

Prevention over cure. Don’t save money now at the risk of catastrophe later.



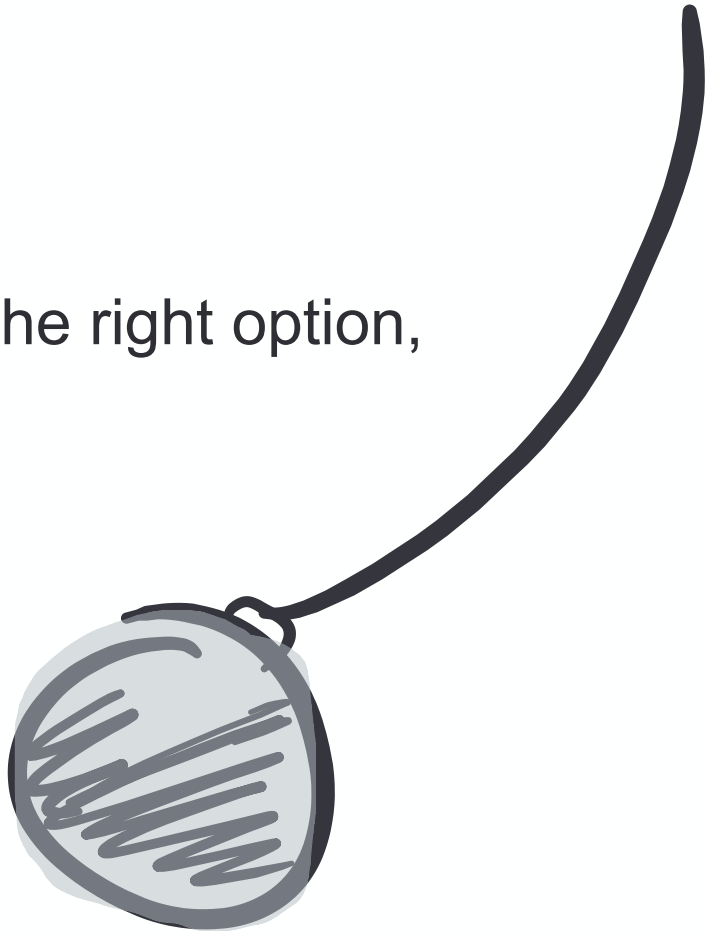
Renovate

Drastic changes may include remodeling, additions, and major retrofitting.

The “Home” Metaphor

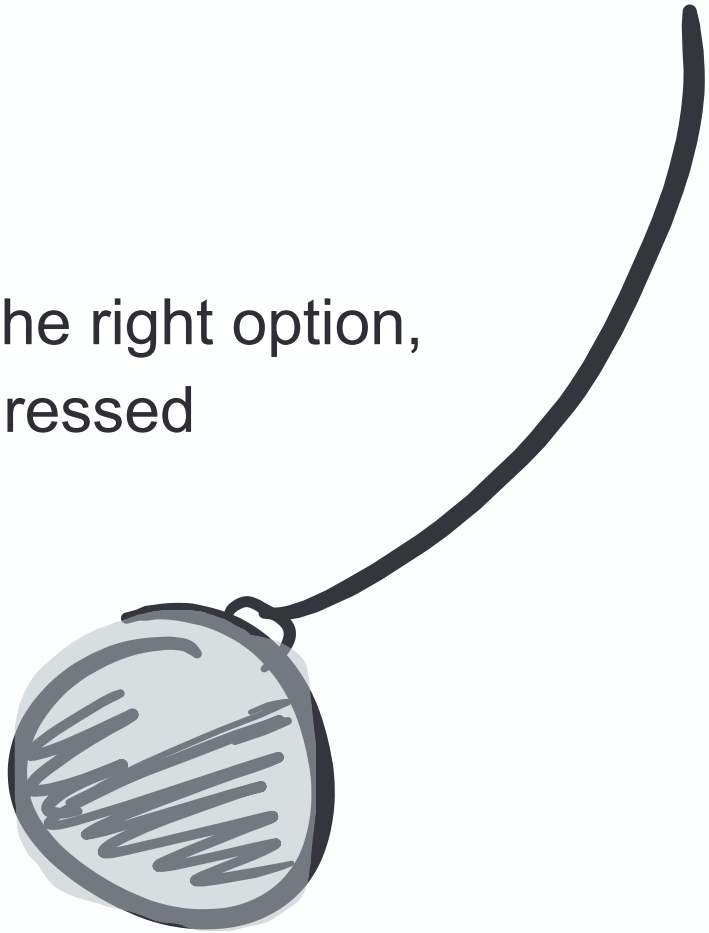
The “Home” Metaphor

Demolition and rebuilding may be the right option,



The “Home” Metaphor

Demolition and rebuilding may be the right option, although usually because of unaddressed deterioration.



How to maintain and renovate

Maintain & Renovate

Maintain & Renovate

Treat long-lived Federal IT systems like homes.

Maintain & Renovate

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Maintain & Renovate

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Pay down technical debt.

Maintain & Renovate

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Pay down technical debt.

Steadily invest in modernizing the system.

Maintain & Renovate

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Pay down technical debt.

Steadily invest in modernizing the system.

Reduce Technical Debt

Technical Debt is a huge topic.

Reduce Technical Debt

Technical Debt is a huge topic.

- Lack of process or understanding
- Tightly-coupled components
- Lack of a test suite
- Lack of documentation
- Lack of collaboration
- Parallel development
- Delayed refactoring
- Lack of alignment to standards
- Lack of knowledge
- Lack of ownership
- Poor technological leadership

Reduce Technical Debt

Technical Debt is a huge topic.

- Lack of process or understanding
- Tightly-coupled components
- Lack of a test suite
- Lack of documentation
- Lack of collaboration
- Parallel development
- Delayed refactoring
- Lack of alignment to standards
- Lack of knowledge
- Lack of ownership
- Poor technological leadership

Reduce Technical Debt

Technical Debt is a huge topic.

- Lack of process or understanding
- Tightly-coupled components
- Lack of a test suite
- Lack of documentation
- Lack of collaboration
- Parallel development
- Delayed refactoring
- Lack of alignment to standards
- Lack of knowledge
- Lack of ownership
- Poor technological leadership

“As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.”

— Meir Manny Lehman, 1980

Reduce Technical Debt

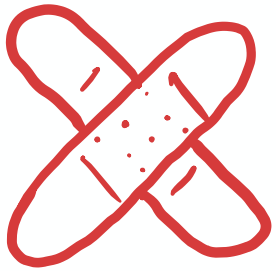
Technical Debt is a huge topic.

- Lack of process or understanding
- Tightly-coupled components
- Lack of a test suite
- Lack of documentation
- Lack of collaboration
- Parallel development
- Delayed refactoring
- Lack of alignment to standards
- Lack of knowledge
- Lack of ownership
- Poor technological leadership

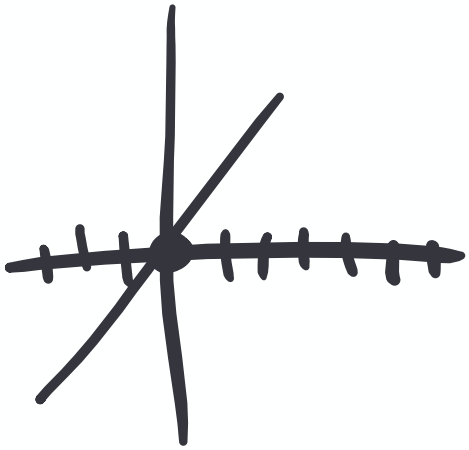
"As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it."

— Meir Manny Lehman, 1980

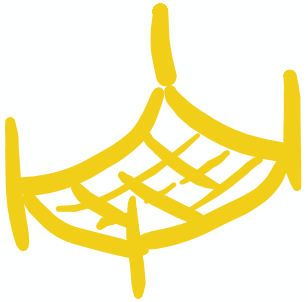
Reduce Technical Debt



Stop the bleeding

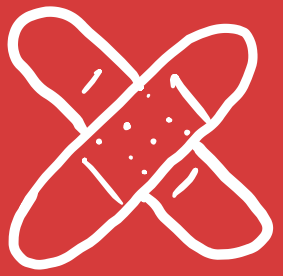


Create time and space

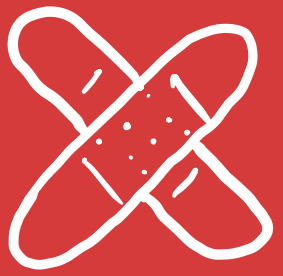


Ensure safety

Stop the Bleeding

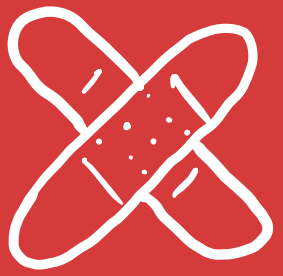


Stop the Bleeding



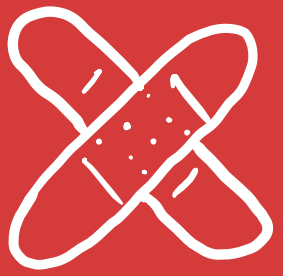
- Every piece of work should result in equal or reduced technical debt.

Stop the Bleeding



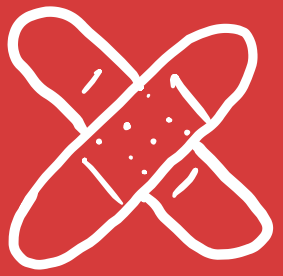
- Every piece of work should result in equal or reduced technical debt.
- Boy Scout Rule: leave it better than you found it.

Stop the Bleeding



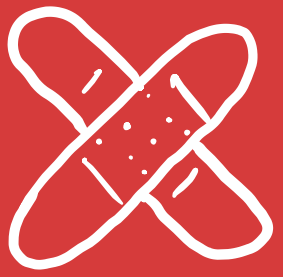
- Every piece of work should result in equal or reduced technical debt.
- Boy Scout Rule: leave it better than you found it.
- Refactor as you go.

Stop the Bleeding

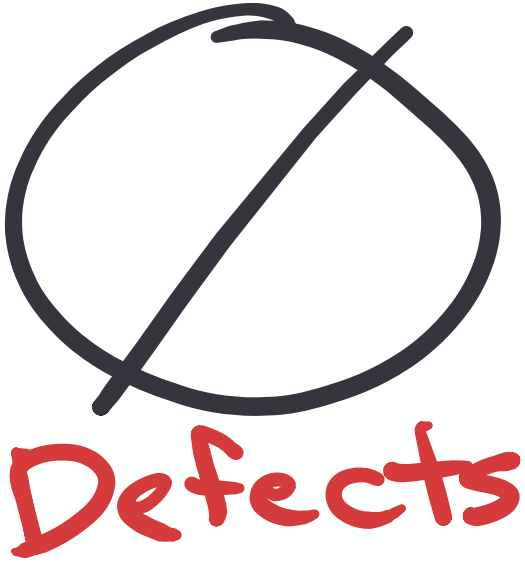


- Every piece of work should result in equal or reduced technical debt.
- Boy Scout Rule: leave it better than you found it.
- Refactor as you go.
- Fix defects ASAP.

Stop the Bleeding



Zero Defects:
fix all known defects before adding functionality.



Create Time and Space

Create Time and Space

- Software development teams usually feel pressure to deliver new functionality rapidly, both explicit and implicit.

Create Time and Space

- Software development teams usually feel pressure to deliver new functionality rapidly, both explicit and implicit.
- Paying down technical debt, including major re-writes, should be supported, rewarded, and celebrated.

Create Time and Space

- Software development teams usually feel pressure to deliver new functionality rapidly, both explicit and implicit.
- Paying down technical debt, including major re-writes, should be supported, rewarded, and celebrated.
- Educate stakeholders about the effort: these systems are long-term investments.

Create Time and Space

Create Time and Space

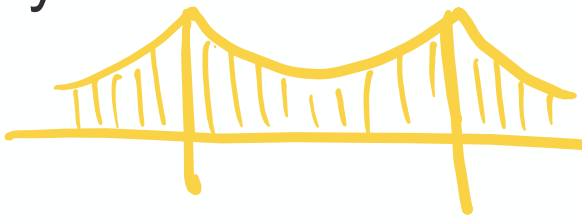
- Adding test coverage, applying updates, and rewriting portions of the code should be celebrated as much as adding new functionality.

Create Time and Space

- Adding test coverage, applying updates, and rewriting portions of the code should be celebrated as much as adding new functionality.
- Metrics that emphasize adding functionality, such as the usual velocity charts, are dangerous.

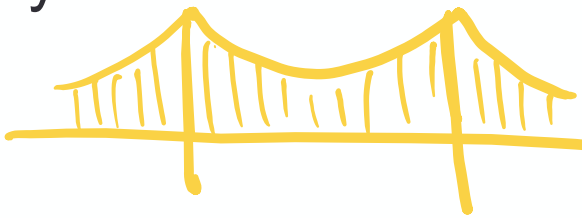
Create Time and Space

- Adding test coverage, applying updates, and rewriting portions of the code should be celebrated as much as adding new functionality.
- Metrics that emphasize adding functionality, such as the usual velocity charts, are dangerous.
- Imagine if bridge infrastructure was only measured by number of miles added!



Create Time and Space

- Imagine if bridge infrastructure was only measured by number of miles added!



Create Time and Space

- Imagine if bridge infrastructure was only measured by number of miles added!

Create Time and Space

- Imagine if bridge infrastructure was only measured by number of miles added!
- If you target a metric, you will likely improve it, although at some cost elsewhere.

Create Time and Space

- Imagine if bridge infrastructure was only measured by number of miles added!
- If you target a metric, you will likely improve it, although at some cost elsewhere.
- I want to drive on bridges where the key metric is on-time inspection and maintenance.

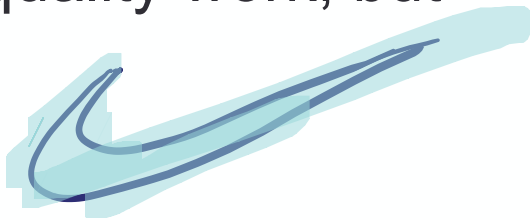
Create Time and Space

How can we support software caretakers as they maintain the system and do high-quality work, but produce less new functionality?

Create Time and Space

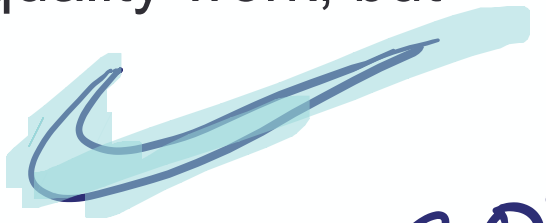
How can we support software caretakers as they maintain the system and do high-quality work, but produce less new functionality?

- Just do it.



Create Time and Space

How can we support software caretakers as they maintain the system and do high-quality work, but produce less new functionality?

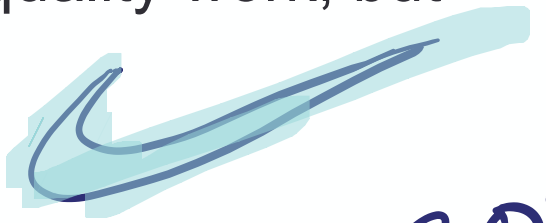


- Just do it.
- Set aside time & capacity for maintenance.

20%

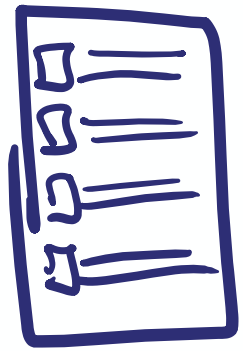
Create Time and Space

How can we support software caretakers as they maintain the system and do high-quality work, but produce less new functionality?



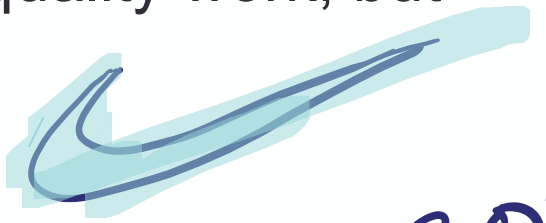
- Just do it.
- Set aside time & capacity for maintenance.

- Track maintenance and refactors the same as PBIs/User Stories/etc.



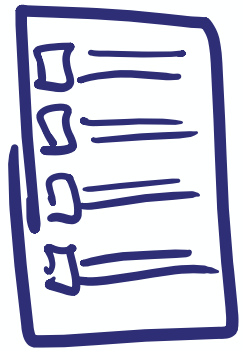
Create Time and Space

How can we support software caretakers as they maintain the system and do high-quality work, but produce less new functionality?

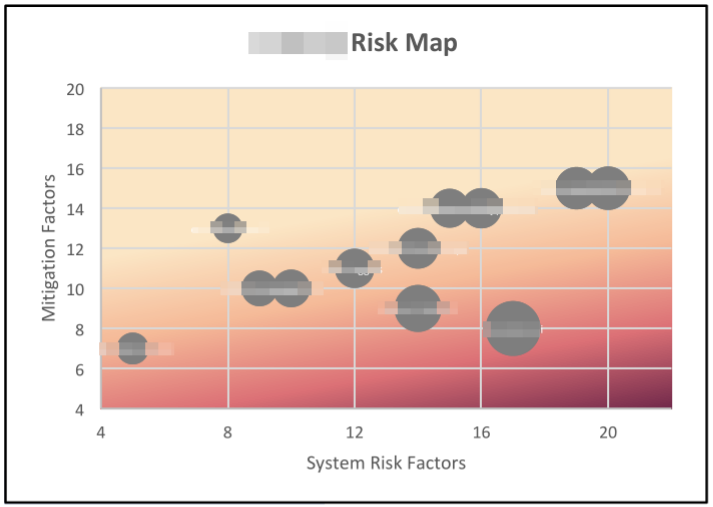


- Just do it.
- Set aside time & capacity for maintenance.

- Track maintenance and refactors the same as PBIs/User Stories/etc.
- Inspect the code and create metrics around maintenance to be done.



Create Time and Space



Maximum Risk Score:	
10.6	
1 week ago	10.6
5 weeks ago	10.6
Report Date:	10/14/18

Microservice	Risk Score	Mitigation	Testing	Monitoring	Logging	Alerting
	10.6	25%	2	2	1	3
	7.8	31%	3	3	2	1
	6.7	69%	4	4	4	3
	6.3	69%	4	4	4	3
	5.8	50%	3	3	3	3
	5.7	63%	4	3	4	3
	5.5	44%	3	3	2	3
	5.4	63%	4	3	4	3
	5.0	38%	2	3	2	3
	4.5	38%	1	3	3	3
	3.6	19%	2	2	2	1
	3.1	56%	4	3	3	3

Ensure Safety

Ensure Safety

- Psychological safety: re-writes can be risky.

Ensure Safety

- Psychological safety: re-writes can be risky.
- Risks can be reduced in many ways.
  Learn about them and use what makes sense.

Ensure Safety

- Psychological safety: re-writes can be risky.
- Risks can be reduced in many ways.
  Learn about them and use what makes sense.
- Testing, especially automated testing, is huge.

Ensure Safety

- Psychological safety: re-writes can be risky.
- Risks can be reduced in many ways.
  Learn about them and use what makes sense.
- Testing, especially automated testing, is huge.
- Blue/green deployments & shadowing (running new code in parallel with the old) can be even more effective.

Ensure Safety

Automated Tests!

Automated tests pay off over time and make future changes safer.

Ensure Safety

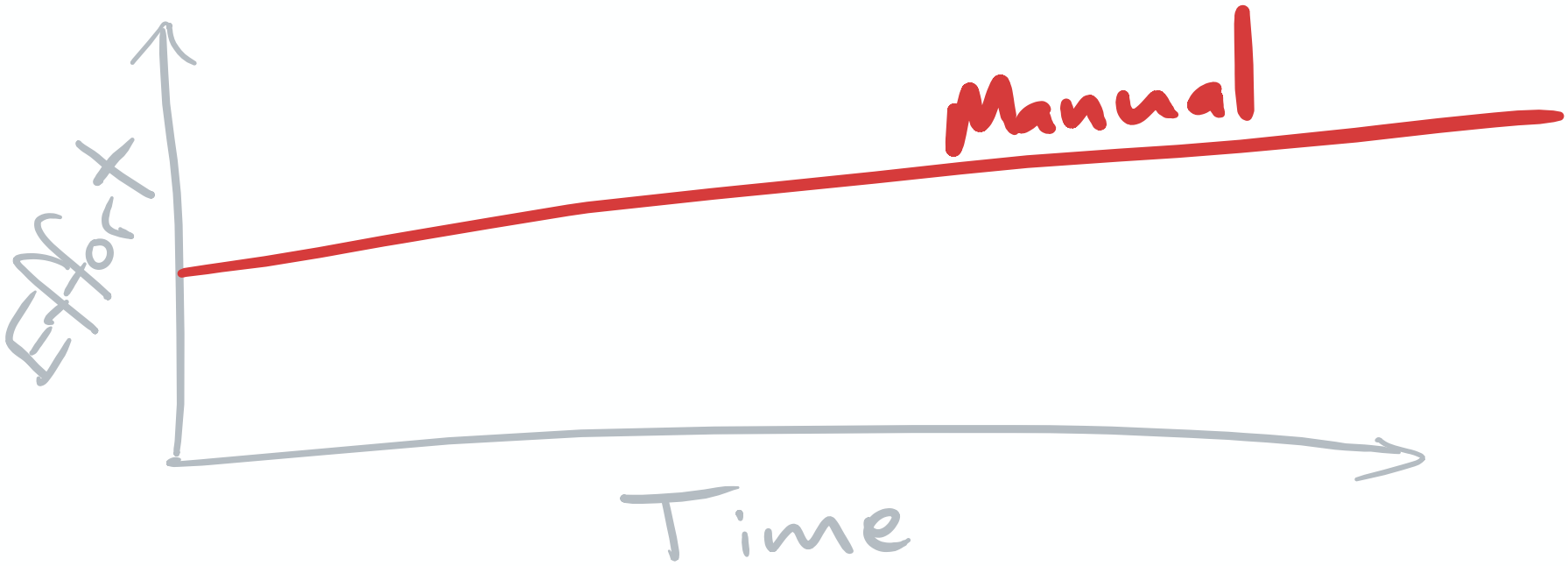
Automated Tests!

Automated tests pay off over time and make future changes safer.

Ensure Safety

Automated Tests!

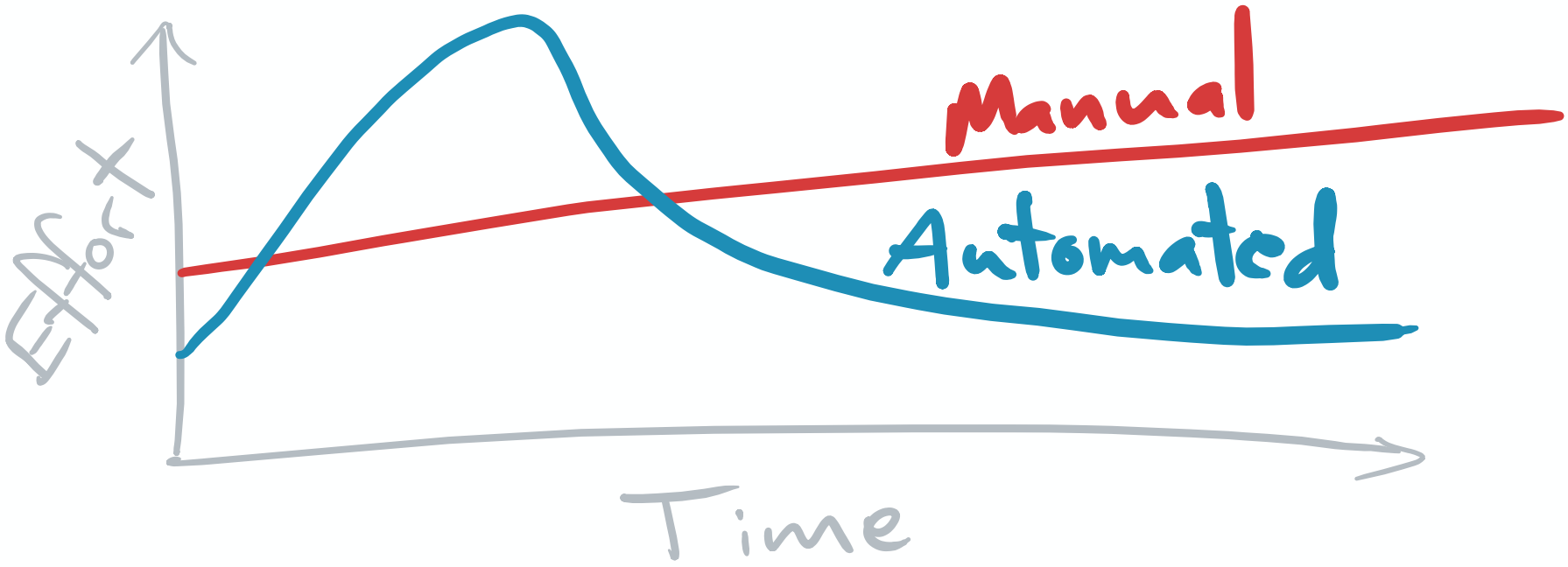
Automated tests pay off over time and make future changes safer.



Ensure Safety

Automated Tests!

Automated tests pay off over time and make future changes safer.



Ensure Safety

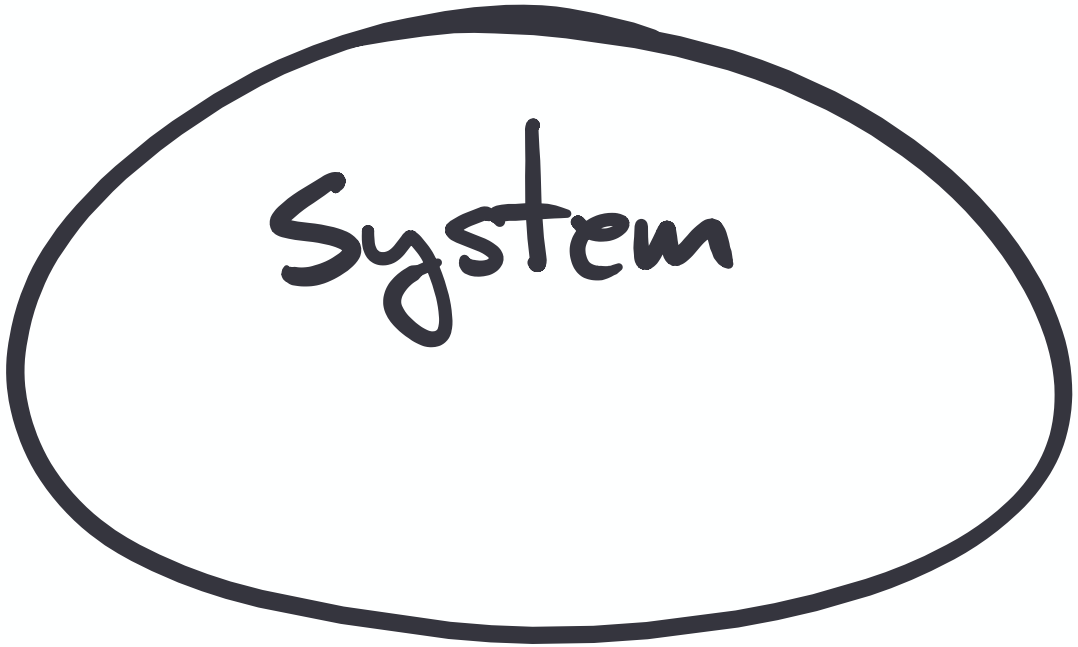
Automated Tests!

Test system behavior, interfaces, and new code.

Ensure Safety

Automated Tests!

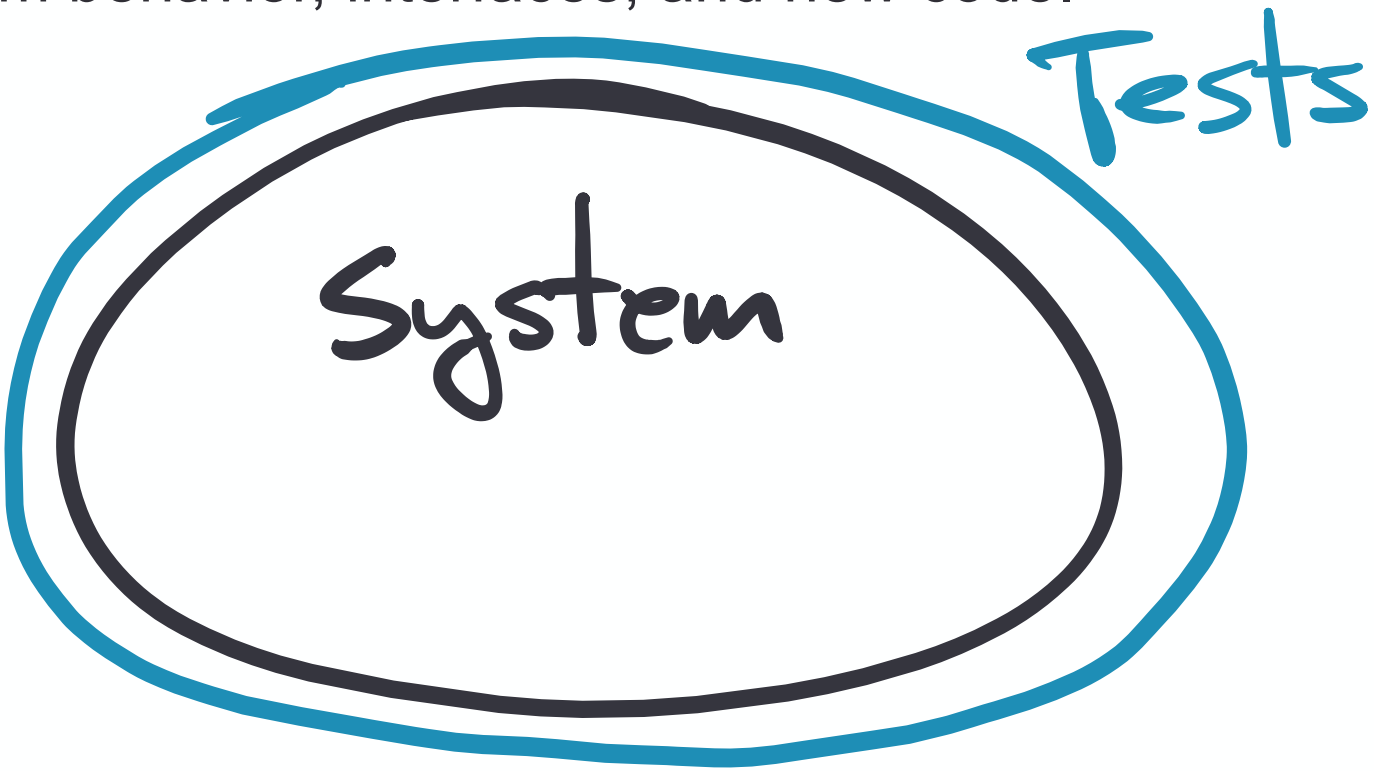
Test system behavior, interfaces, and new code.



Ensure Safety

Automated Tests!

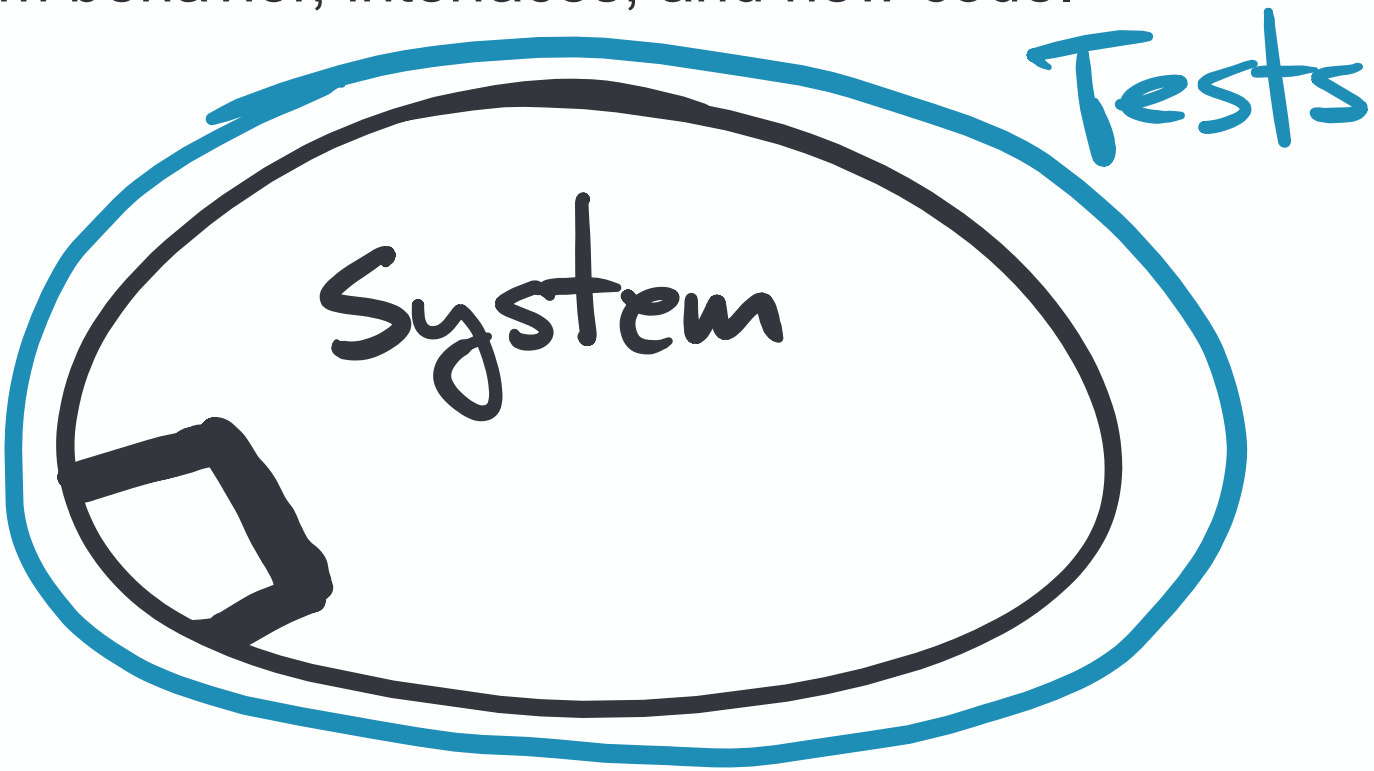
Test system behavior, interfaces, and new code.



Ensure Safety

Automated Tests!

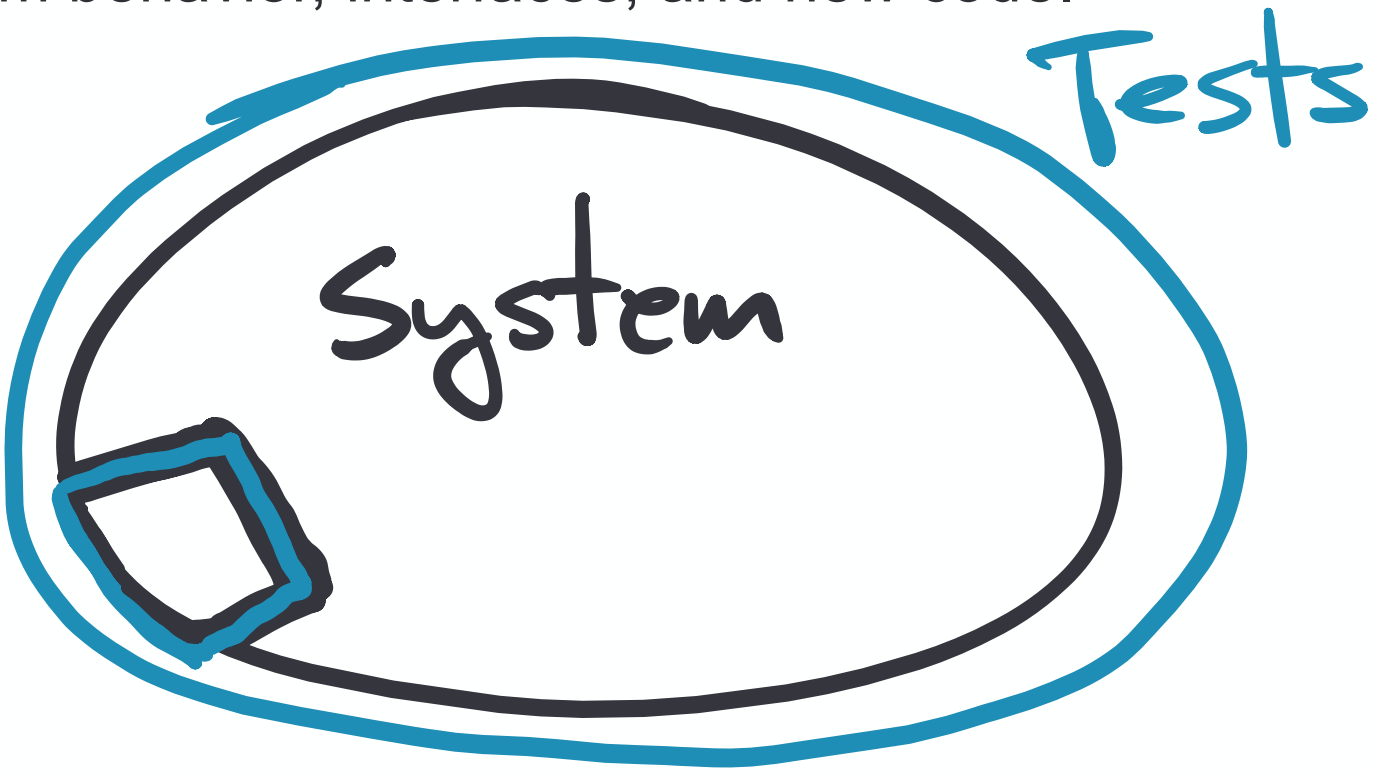
Test system behavior, interfaces, and new code.



Ensure Safety

Automated Tests!

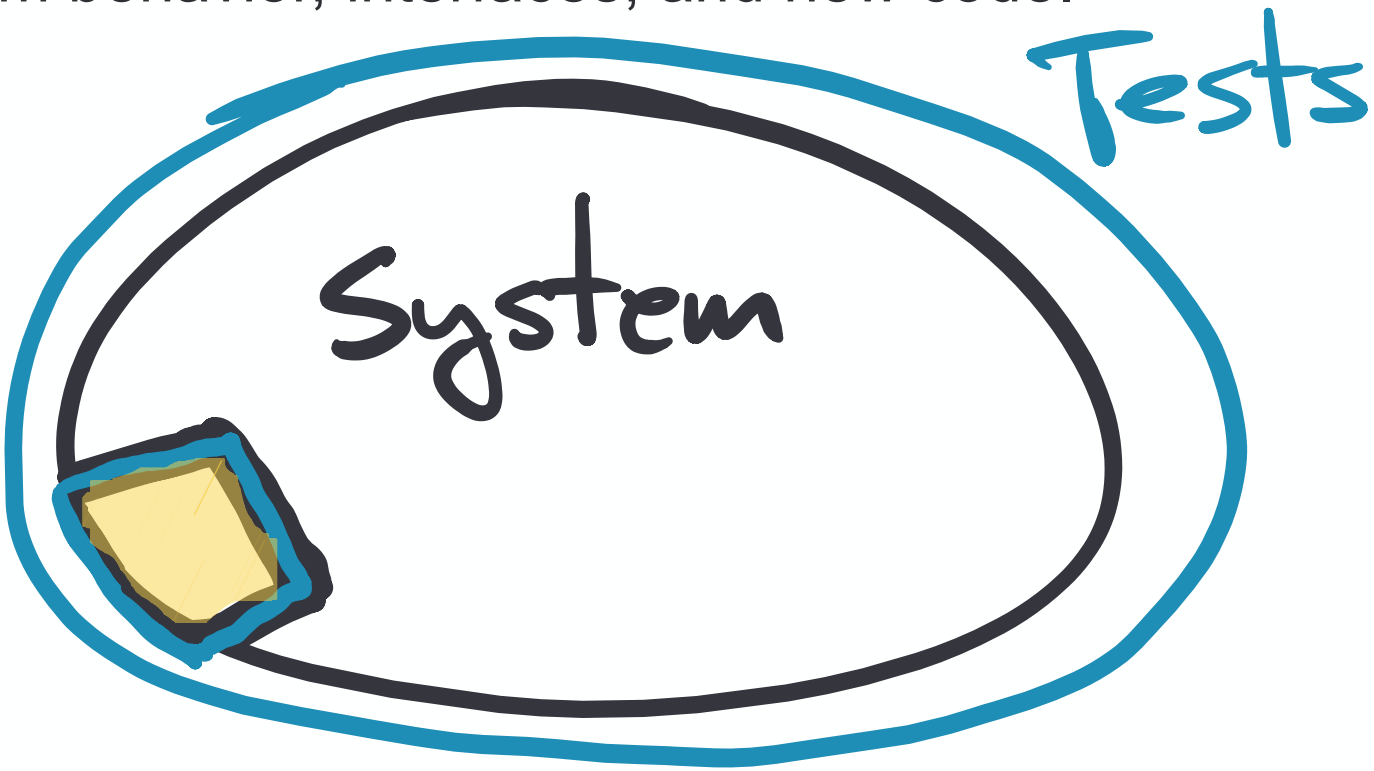
Test system behavior, interfaces, and new code.



Ensure Safety

Automated Tests!

Test system behavior, interfaces, and new code.

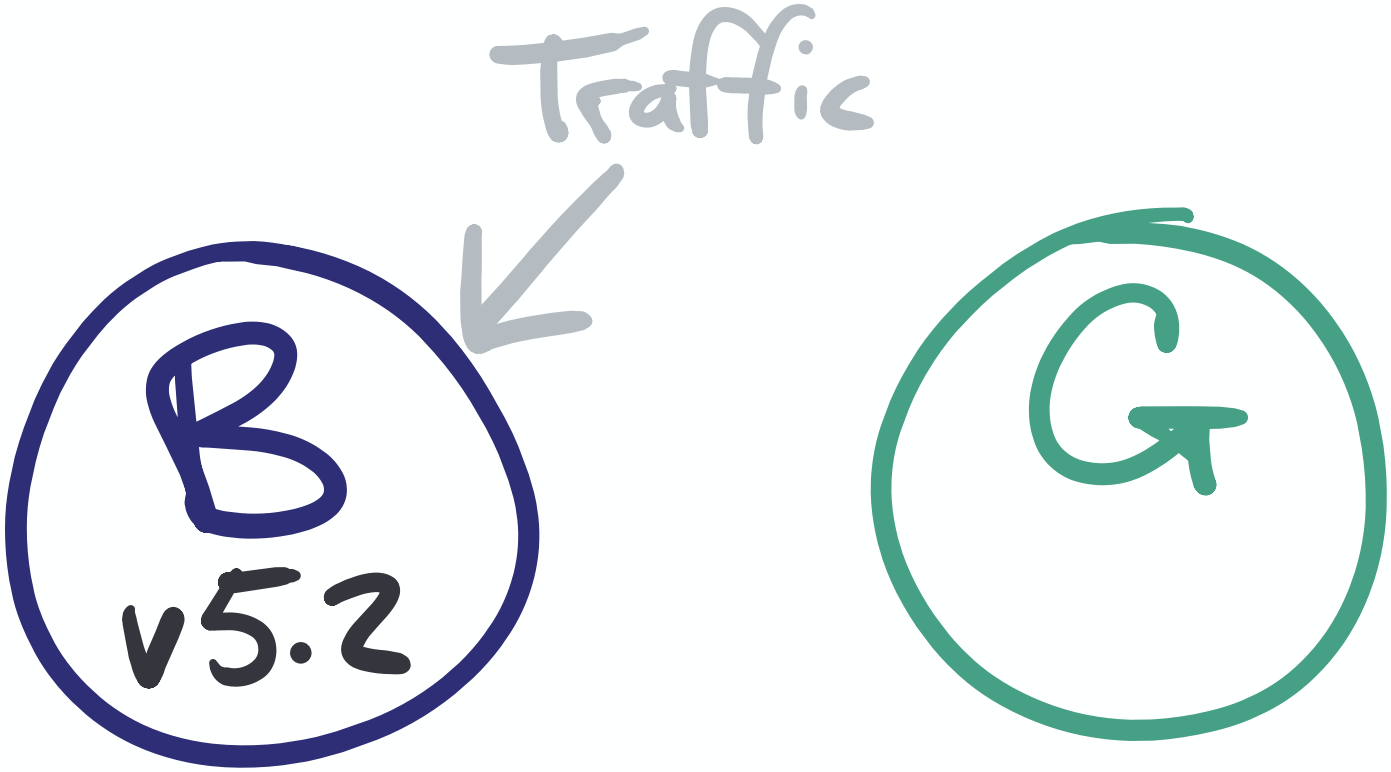


Ensure Safety

Blue/green deployments

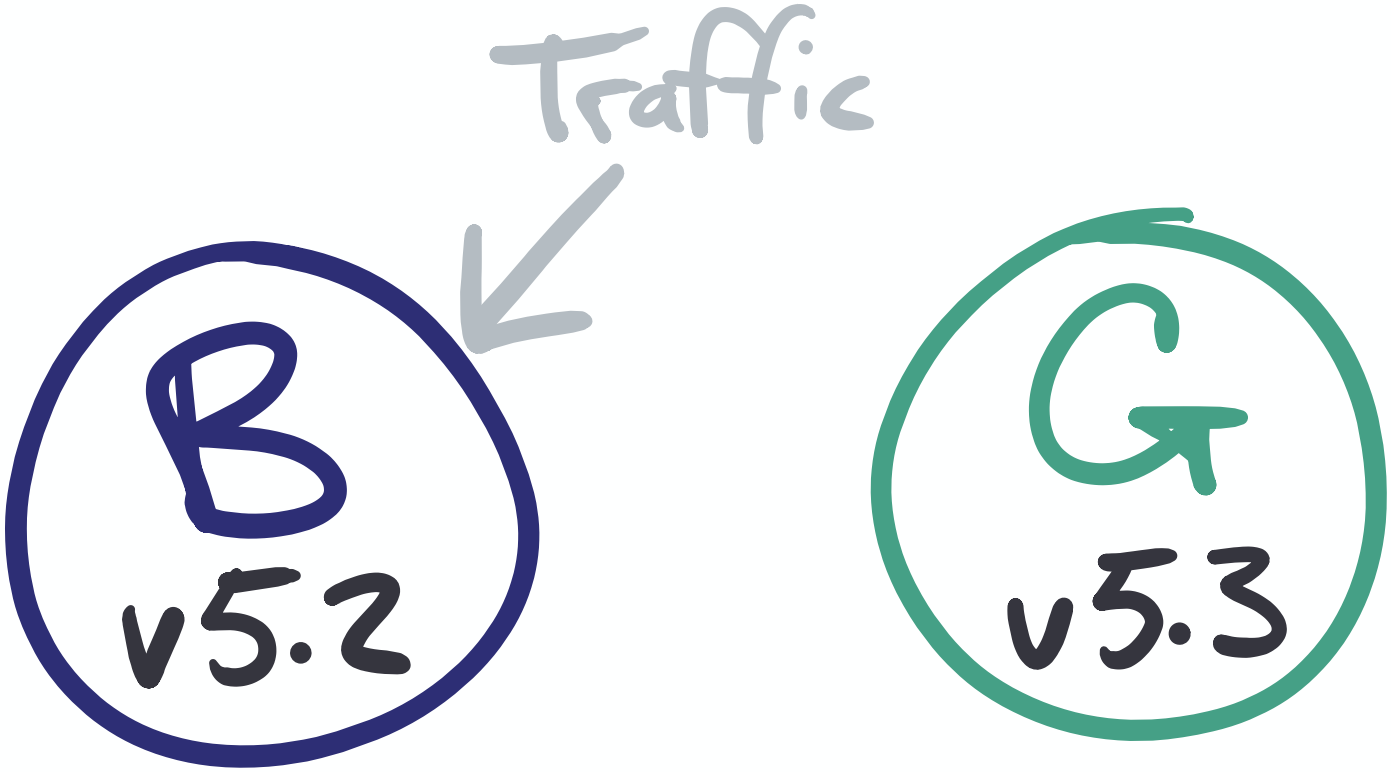
Ensure Safety

Blue/green deployments



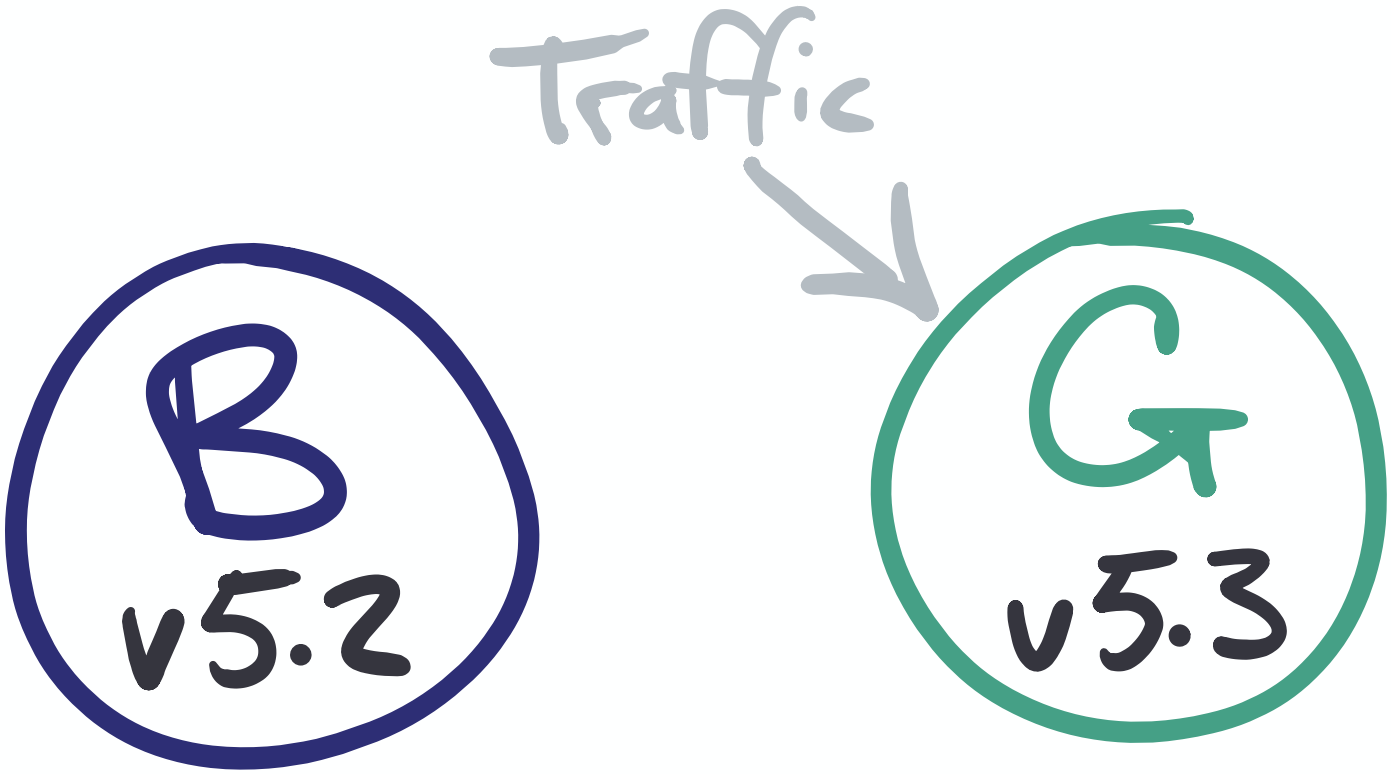
Ensure Safety

Blue/green deployments



Ensure Safety

Blue/green deployments

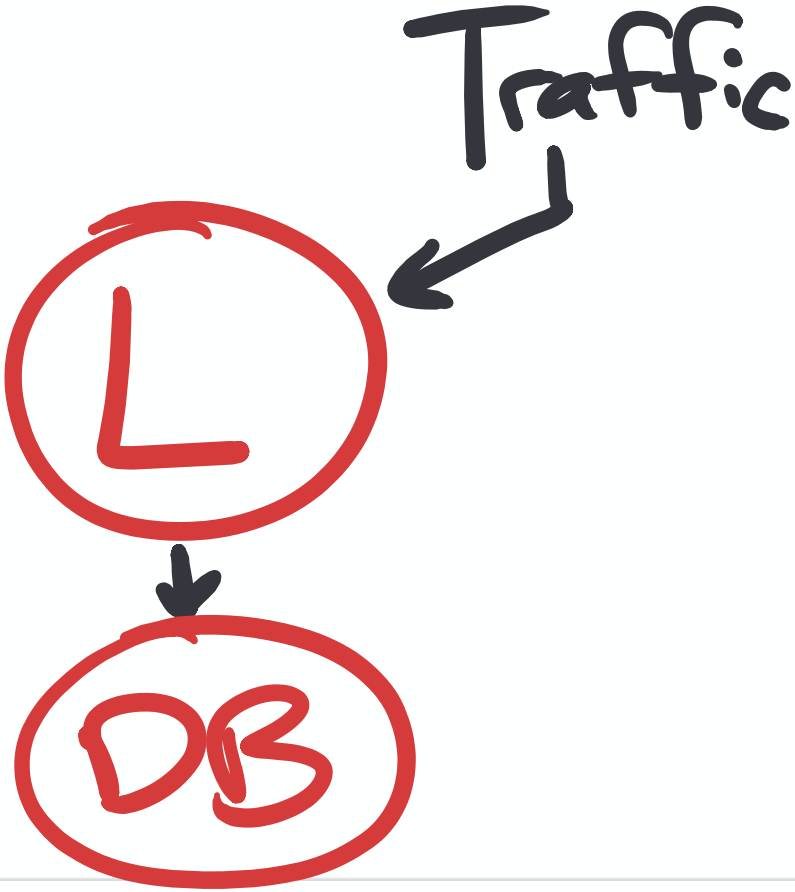


Ensure Safety

Shadowing (Transform & Eliminate)

Ensure Safety

Shadowing (Transform & Eliminate)

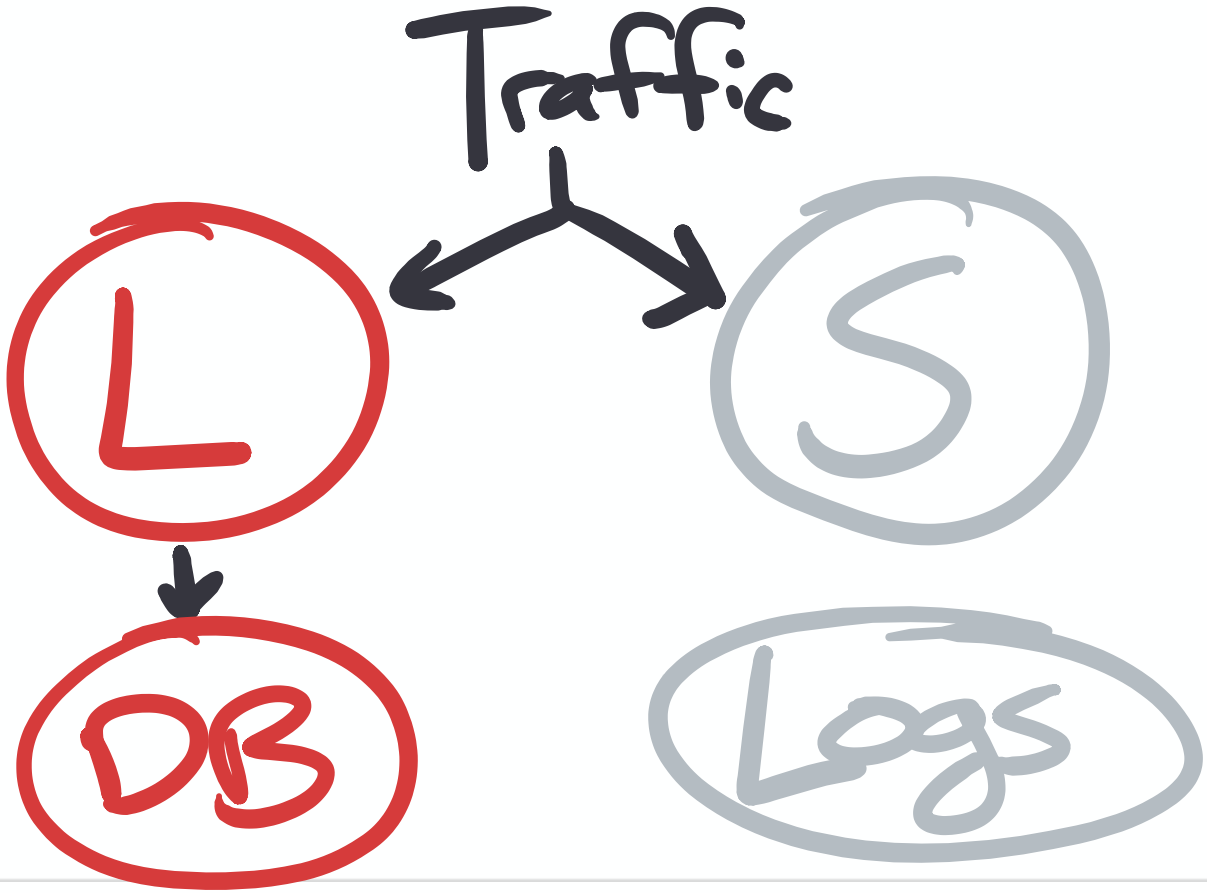


Ensure Safety

Shadowing (Transform & Eliminate)

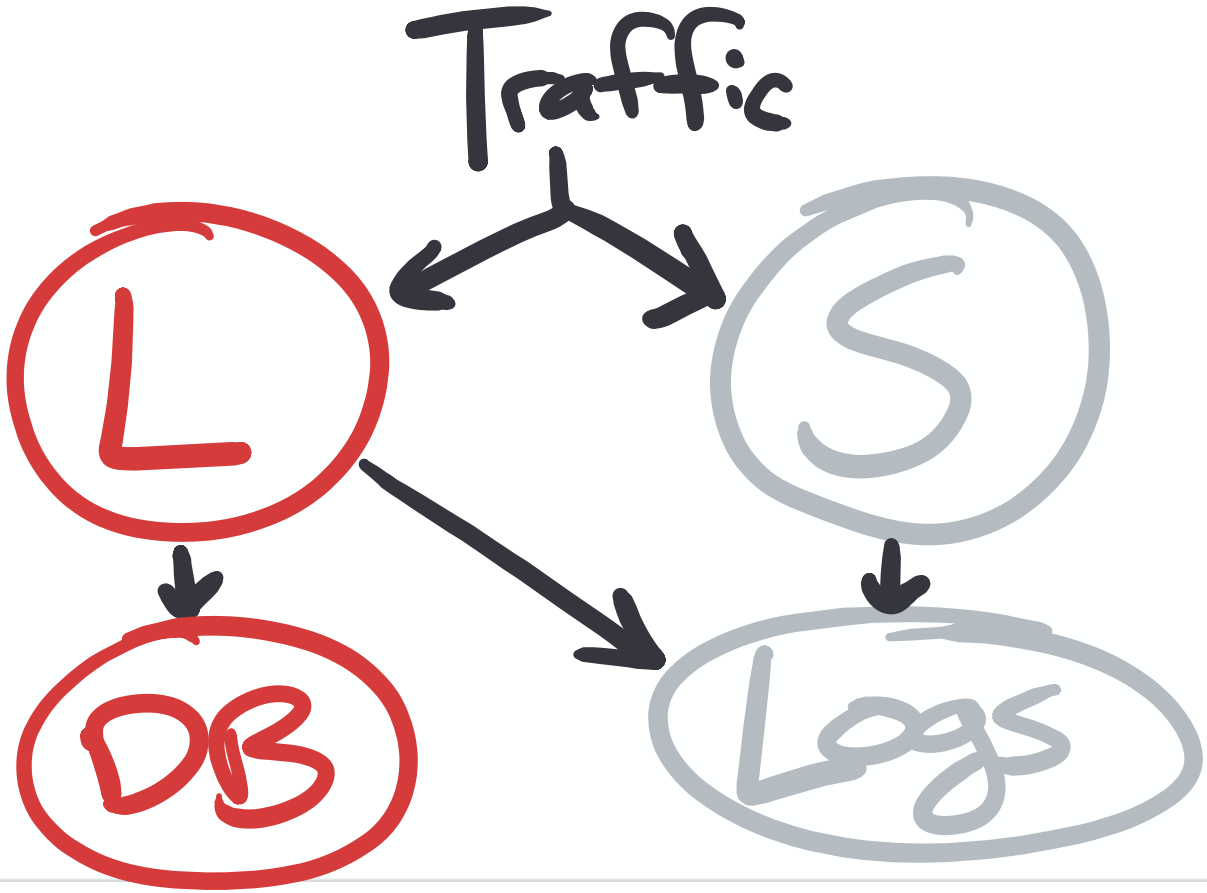
Ensure Safety

Shadowing (Transform & Eliminate)



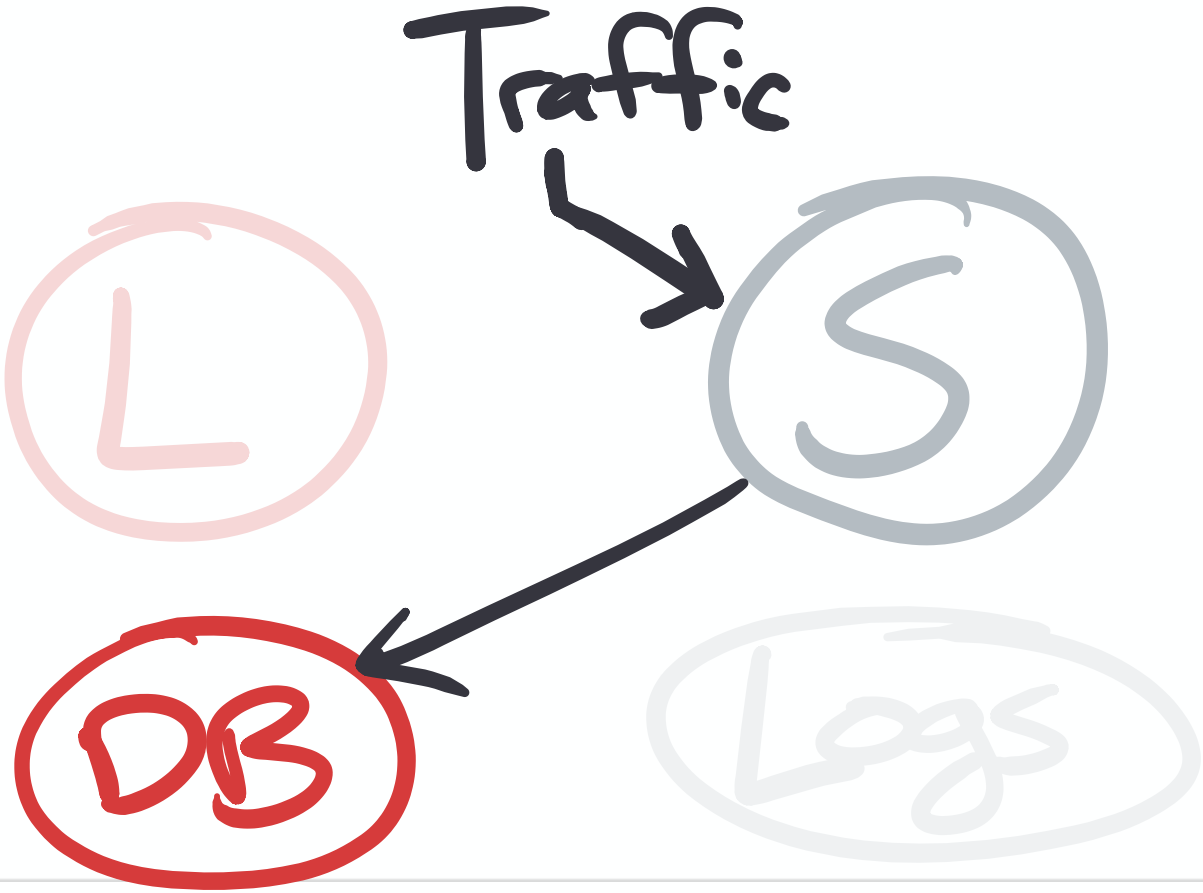
Ensure Safety

Shadowing (Transform & Eliminate)



Ensure Safety

Shadowing (Transform & Eliminate)



Ensure Safety

Further reading:

Ensure Safety

Further reading:

- The “strangler vine” pattern

Ensure Safety

Further reading:

- The “strangler vine” pattern
- Agile engineering

Ensure Safety

Further reading:

- The “strangler vine” pattern
- Agile engineering
- The world of DevOps

Continuous Modernization

Continuous Modernization

Continuous Modernization

Treat long-lived Federal IT systems like homes.

Continuous Modernization

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Continuous Modernization

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Pay down technical debt.

Continuous Modernization

Treat long-lived Federal IT systems like homes.

Avoid demolition and rebuilding.

Pay down technical debt.

Steadily invest in modernizing the system.

Never Stop Modernizing!

DO IT CONTINUOUSLY

excella.com | @excellaco