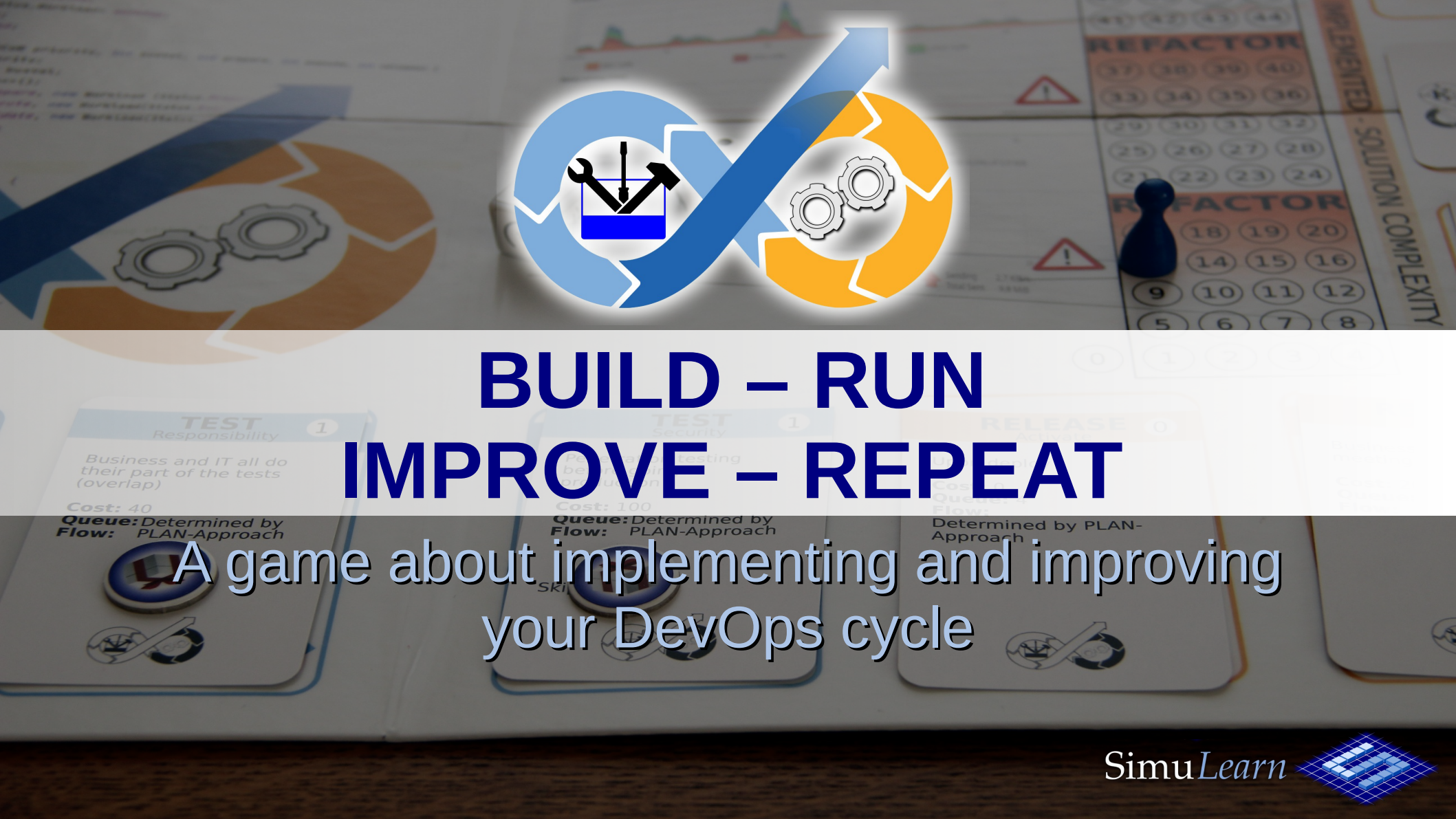


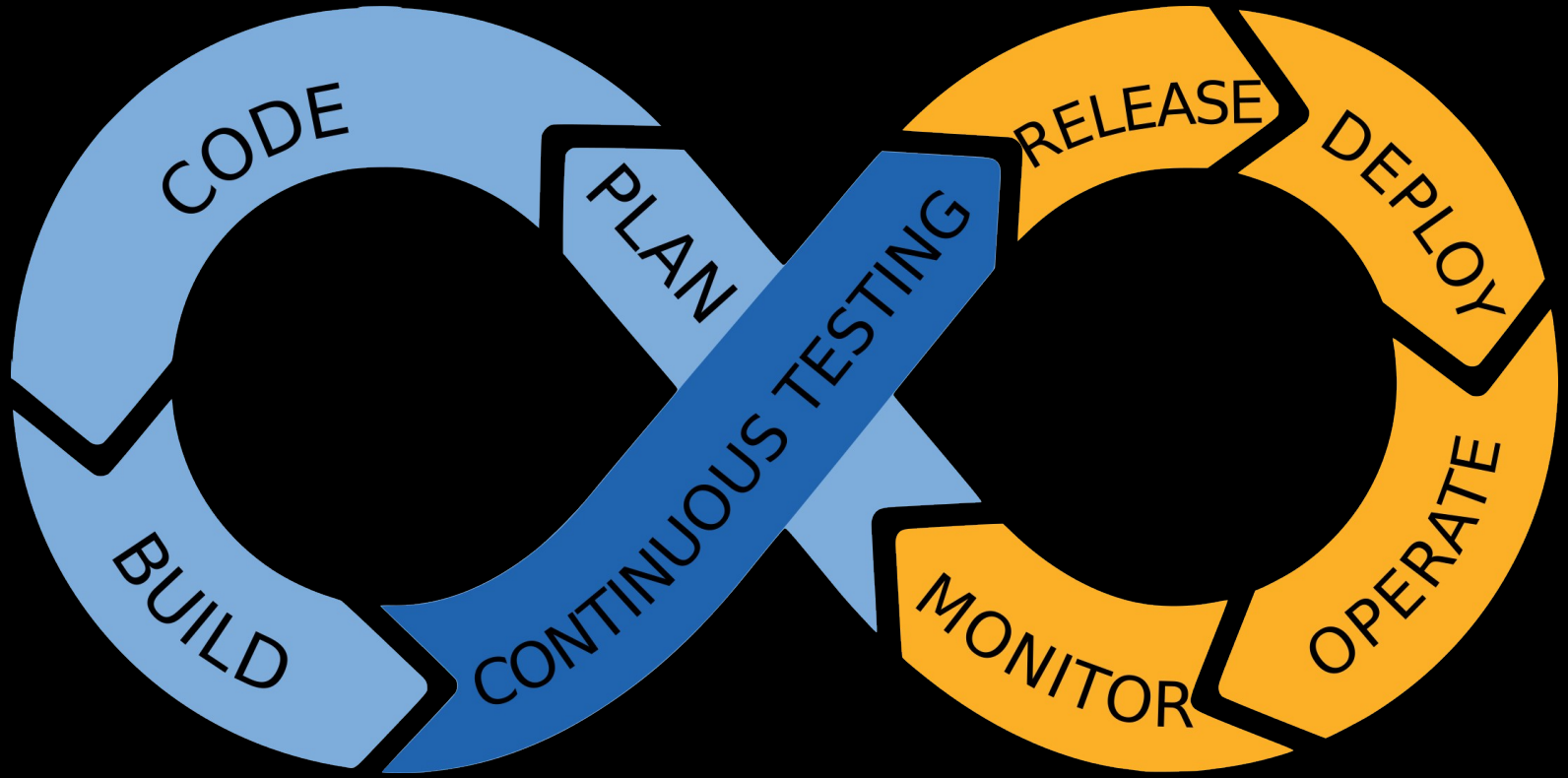
BUILD – RUN
IMPROVE – REPEAT

A game about implementing and improving your DevOps cycle

SimuLearn

Goal of this game

Understand the DevOps cycle

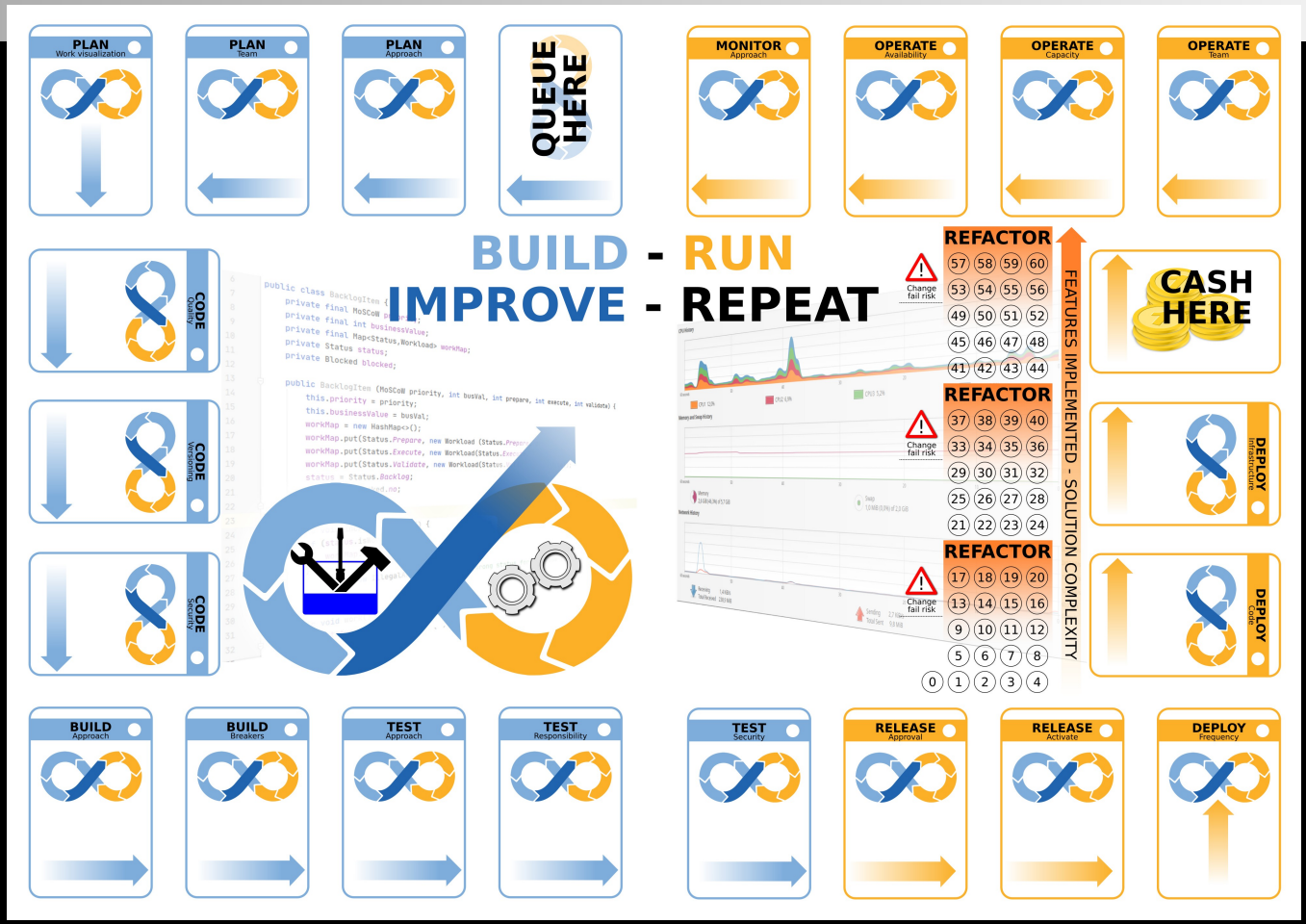


Invest wisely

- Improve your way of working
- The right investments first
- Keep money to cover losses
- **Don’t go bankrupt!**

Elements of the game

The board

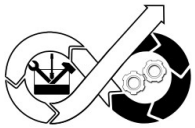

PLAN
Work visualization
PLAN
Team
PLAN
Approach
QUEUE HERE
MONITOR
Approach
OPERATE
Availability
OPERATE
Capacity
OPERATE
Team
CODE
Quality
CODE
Versioning
CODE
Testing
BUILD - RUN
IMPROVE - REPEAT
REFACTOR
REFACTOR
REFACTOR
Change fail risk
FEATURES IMPLEMENTED - SOLUTION COMPLEXITY
CASH HERE
DEPLOY
Infrastructure
DEPLOY
Code
BUILD
Approach
BUILD
Breakers
TEST
Approach
TEST
Responsibility
TEST
Security
RELEASE
Approval
RELEASE
Activate
DEPLOY
Frequency

The cards

PLAN 0
Approach

Project approach with big specification up-front (“waterfall’”)

Cost: 0
Queue: 4
Flow:
All queued features need to move together from 1 activity to the other. Only queue new items when the project is delivered.



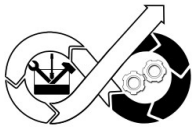
- Different activities/ aspects per stage
- 3 performance levels to invest
- Level 0 = starting point

The cards

PLAN 0
Approach

Project approach with big specification up-front (“waterfall’”)

Cost: 0
Queue: 4
Flow:
All queued features need to move together from 1 activity to the other. Only queue new items when the project is delivered.



PLAN 0
Approach

Incident impact:

Incident cost: 30

Cause:
Medium high cost: a project team tends to focus more on delivering project scope than on code quality and run stability

INCIDENT

The tokens

Feature

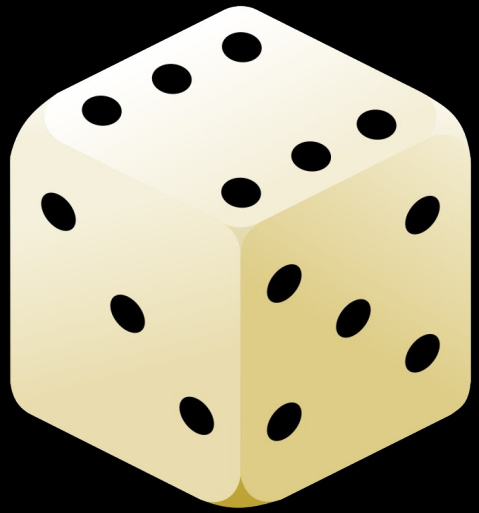
Improvement

Technical debt

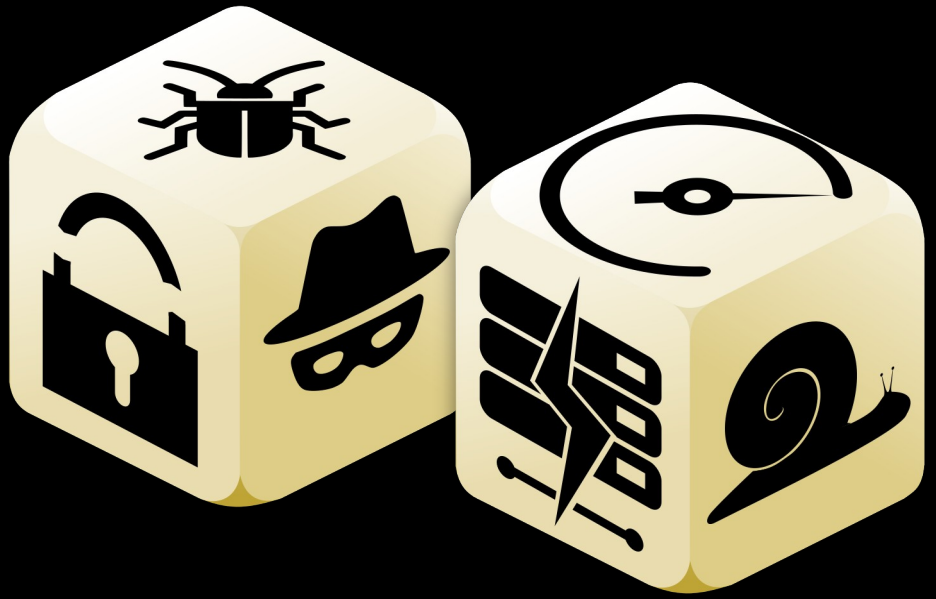
Know vulnerability

Incident

The dice

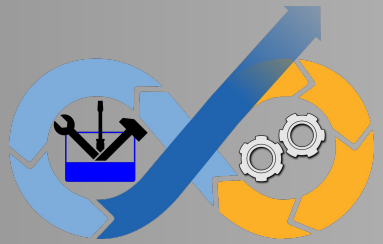


Progress of work

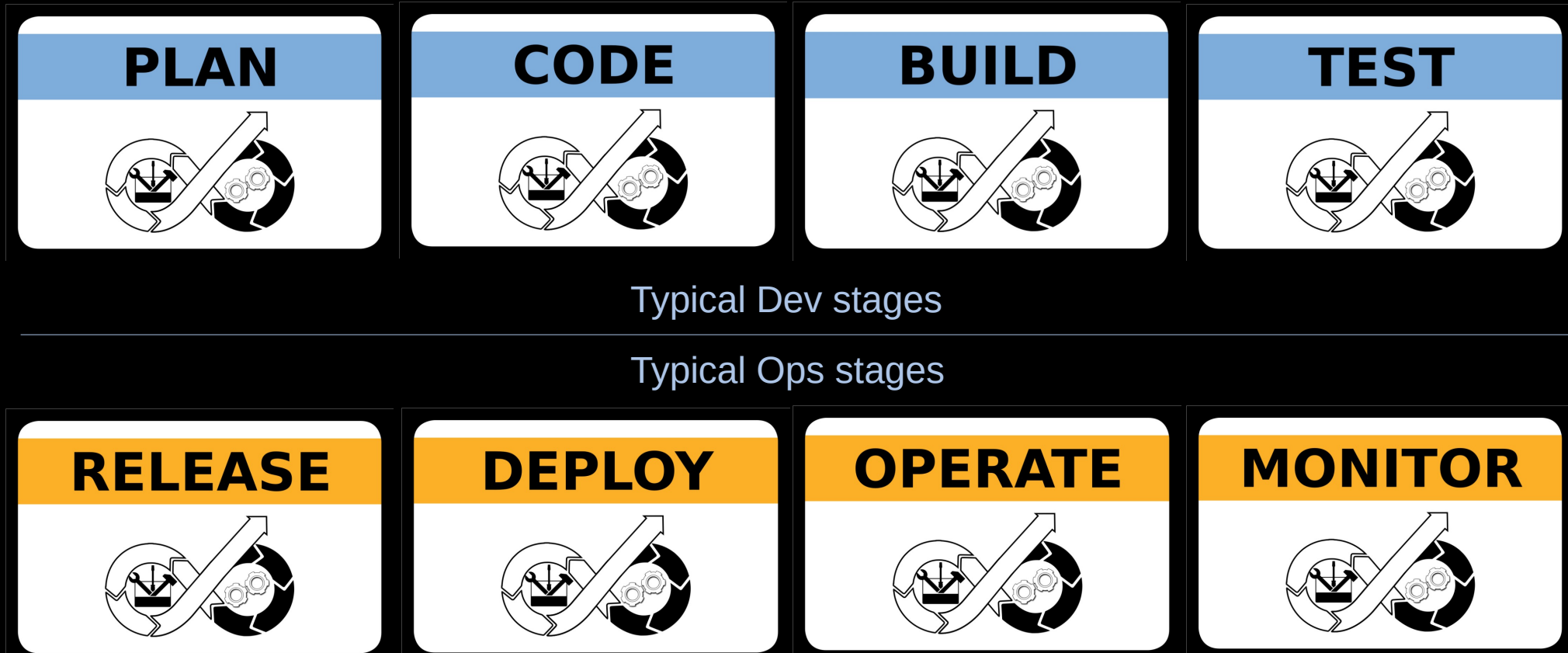


Incidents that occur

Playing the game



Divide ownerships



What is your decision strategy?

Separate responsibilities

- Everyone decides for their own domain(s)
- Everyone invests in their own domain(s)
- Everyone pays for their own losses

Shared responsibility

- Shared decision about all domains
- Global budget
  - For investments
  - For losses

Financial impact of decision strategy

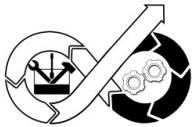
- Shared responsibility: 1000 credits for all
- Separate responsibilities: credits divided, according to:
  - DevOps stages
  - Activities

Zero-state

PLAN
Approach
0

Project approach with big specification up-front (“waterfall’”)

Cost: 0
Queue: 4
Flow:
All queued features need to move together from 1 activity to the other. Only queue new items when the project is delivered.



- All activities start with performance level 0
  - = basic or no activity
- Can potentially cause big damage
- Try to improve before starting

Variation Performance level

Start from 0

- To get to know the simulation
- Experience everything that can go wrong
- For heterogeneous groups (meetups, conferences, ...)

Your organization's situation

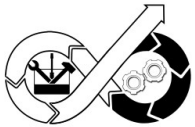
- Headstart for investments
- Better learning experience for your organization

Invest to improve

PLAN 1
Approach

Iterative project approach (agile/ ”Scrum” principles & techniques)

Cost: 100
Queue: 2
Flow:
All queued features need to move together from 1 activity to the other. Only queue new items when the project is delivered.



- Improvements come with a cost
- Spend your budget wisely!
  - Not all at once
  - The right priorities
- ***What are your initial investments?***

Invest to improve – avoid efficiency penalty

PLAN
Approach
2

Iterative product approach, driven by product backlog

Cost: 150
Queue: 2
Flow:
All
to
acti
que
buil

PLAN
Team
0

Separate build and maintenance team

Cost: 0
Queue: Determined by
Flow: PLAN-Approach

PLAN
Visualization
0

Nothing

Cost: 0
Queue: Determined by
Flow: PLAN-Approach

Inefficient organization leads to less work done than foreseen
(= less moves of tokens than die values)
Improve all PLAN aspect at the same pace!

Invest to improve – efficiency penalty: what?

- perf(A) = performance level PLAN-Approach
- perf(T) = performance level PLAN-Team
- perf(V) = performance level PLAN-Visualization
- perf(A) > perf(T) or perf(A) > perf(V)?
  - Penalty = perf (A) – low (perf (T), perf (V))

Invest to improve – efficiency penalty: example

perf(A)	perf(T)	perf(V)	penalty
0	0	0	0
1	0	0	1
2	1	0	2
1	2	2	0

Invest to improve – efficiency penalty: how?

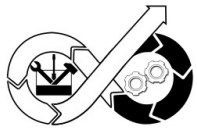
Die value	Penalty = 1	Penalty = 2
	5	4
	4	3
	3	2
	2	1
	1	1
	1	1

Flow and queue

PLAN 0
Approach

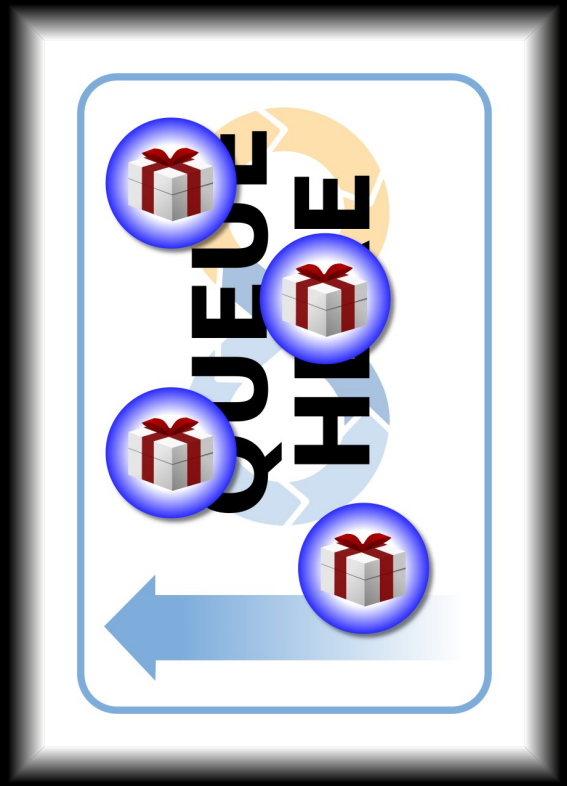
Project approach with big specification up-front (“waterfall’”)

Cost: 0
Queue: 4
Flow:
All queued features need to move together from 1 activity to the other. Only queue new items when the project is delivered.



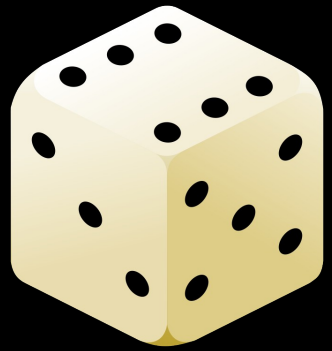
- Queue size:
  - At least how many features need to be at this activity before you can move on to the next?
- Flow:
  - How can you move the features?
  - When can you bring in new items?

Start here



- Queue your feature tokens
- Move them to the first activity according to:
  - Queue size
  - Flow

Implementing features



- Each participant
- Roles the regular die
- Moves feature tokens according to:
  - Value of die
  - Queue size
  - Flow

Implementing features
4 features queued

PLAN Visualization 0

Nothing

Cost: 0
Queue: Determined by
Flow: PLAN Approach

PLAN Team 0

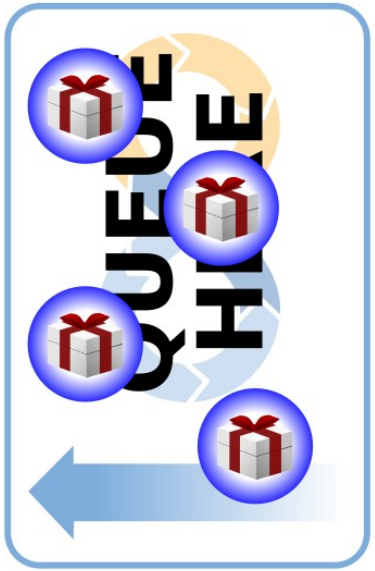
Separate build and maintenance team

Cost: 0
Queue: Determined by
Flow: PLAN Approach

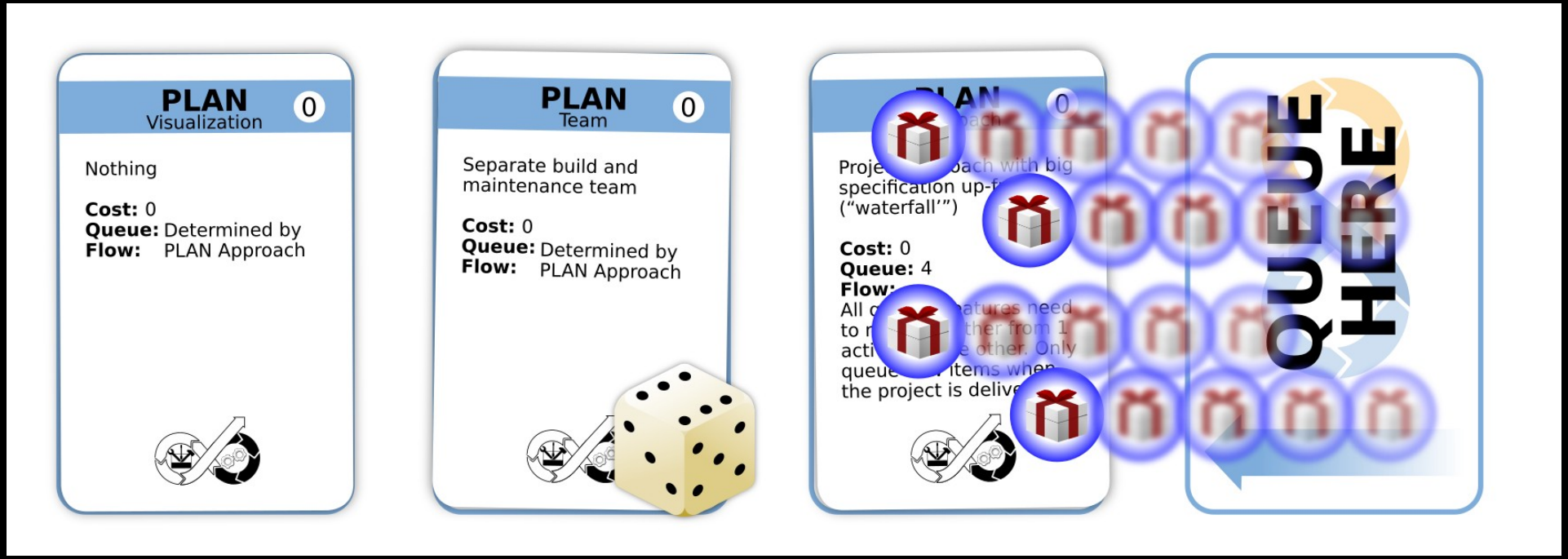
PLAN Approach 0

Project approach with big specification up-front (“waterfall’”)

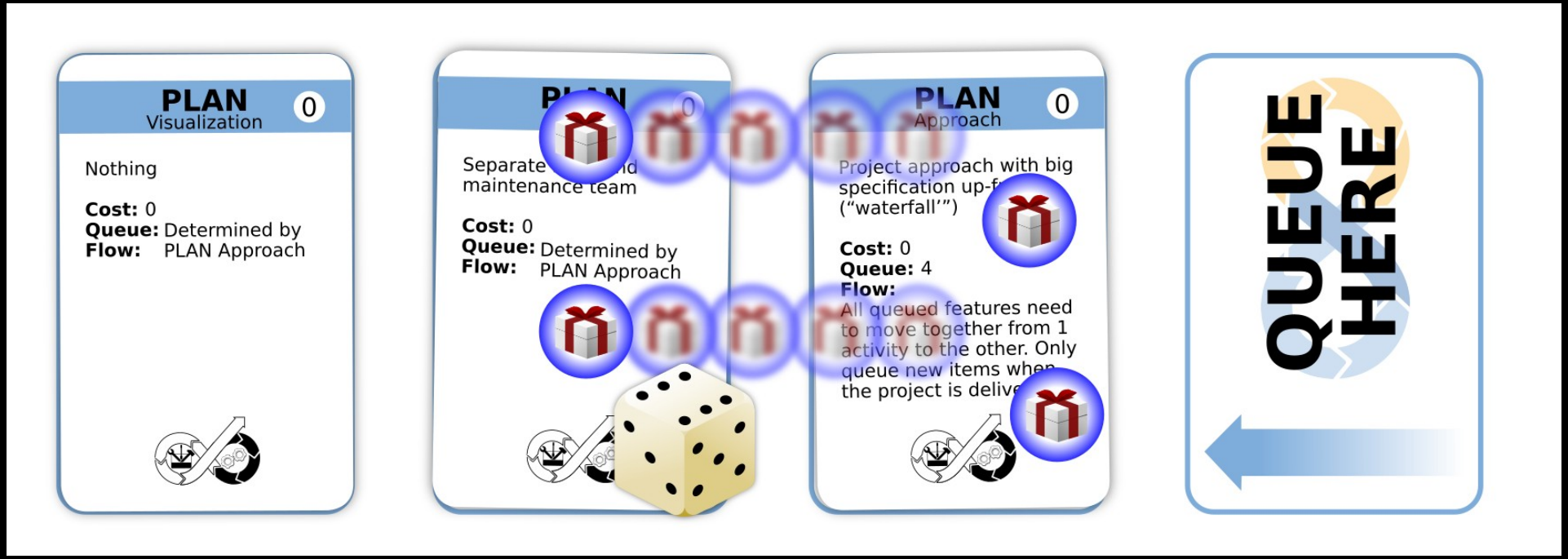
Cost: 0
Queue: 4
Flow:
All queued features need to move together from 1 activity to the other. Only queue new items when the project is delivered.



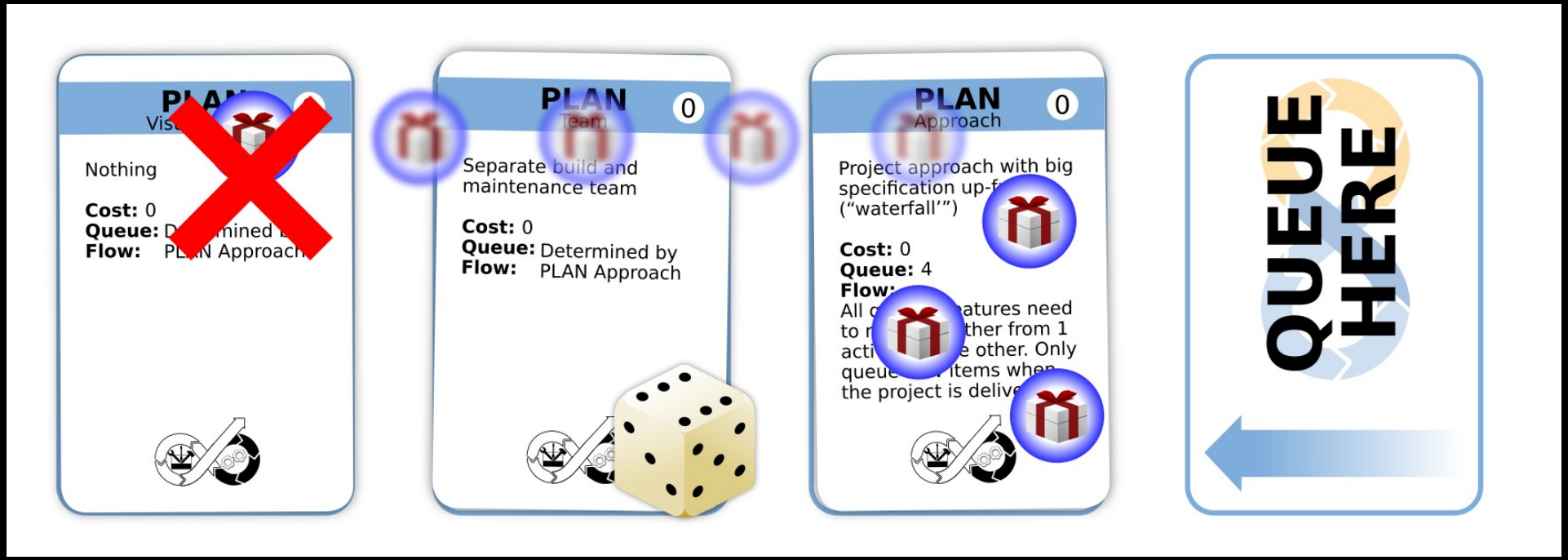
Implementing features
roll 6, move 4



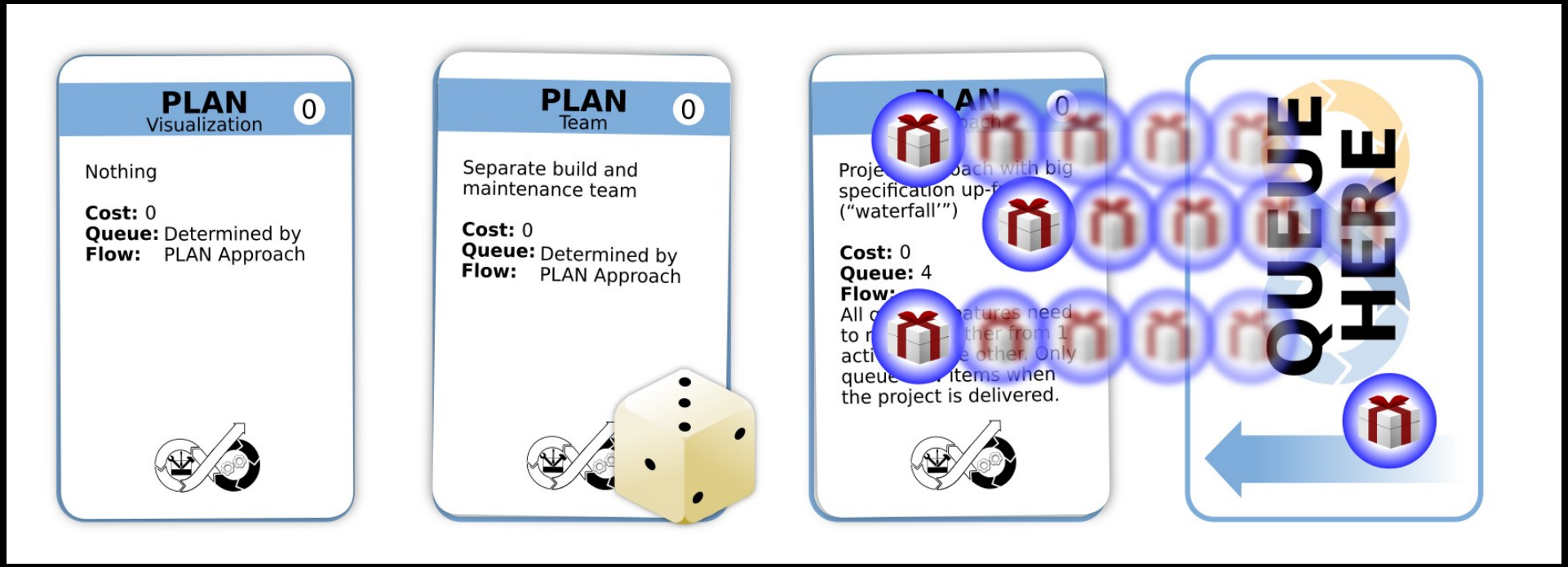
Implementing features roll 6, move 2 more



Implementing features what you can't do



Implementing features roll 3



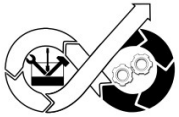
Fast forward

CODE Quality 0

Nothing

Cost: 0
Queue: 0
Flow:
Feature can immediately go to the next activity

NO ACTION

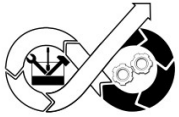


CODE Quality 3

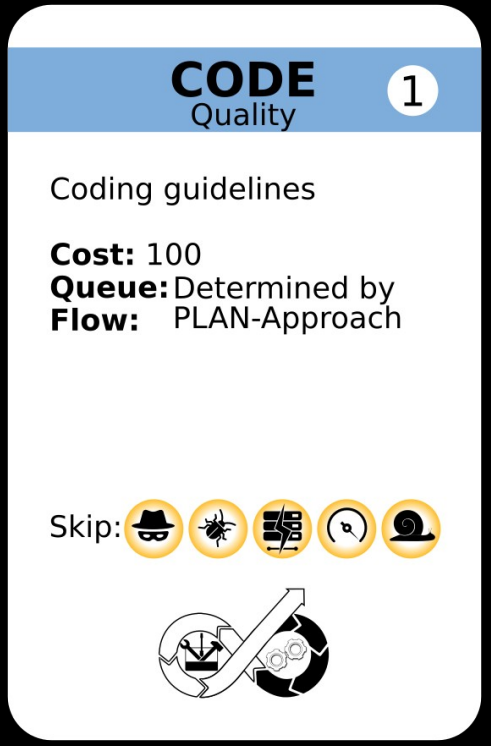
Automatic code scans

Cost: 200
Queue: 0
Flow:
Feature can immediately go to the next activity

AUTOMATED

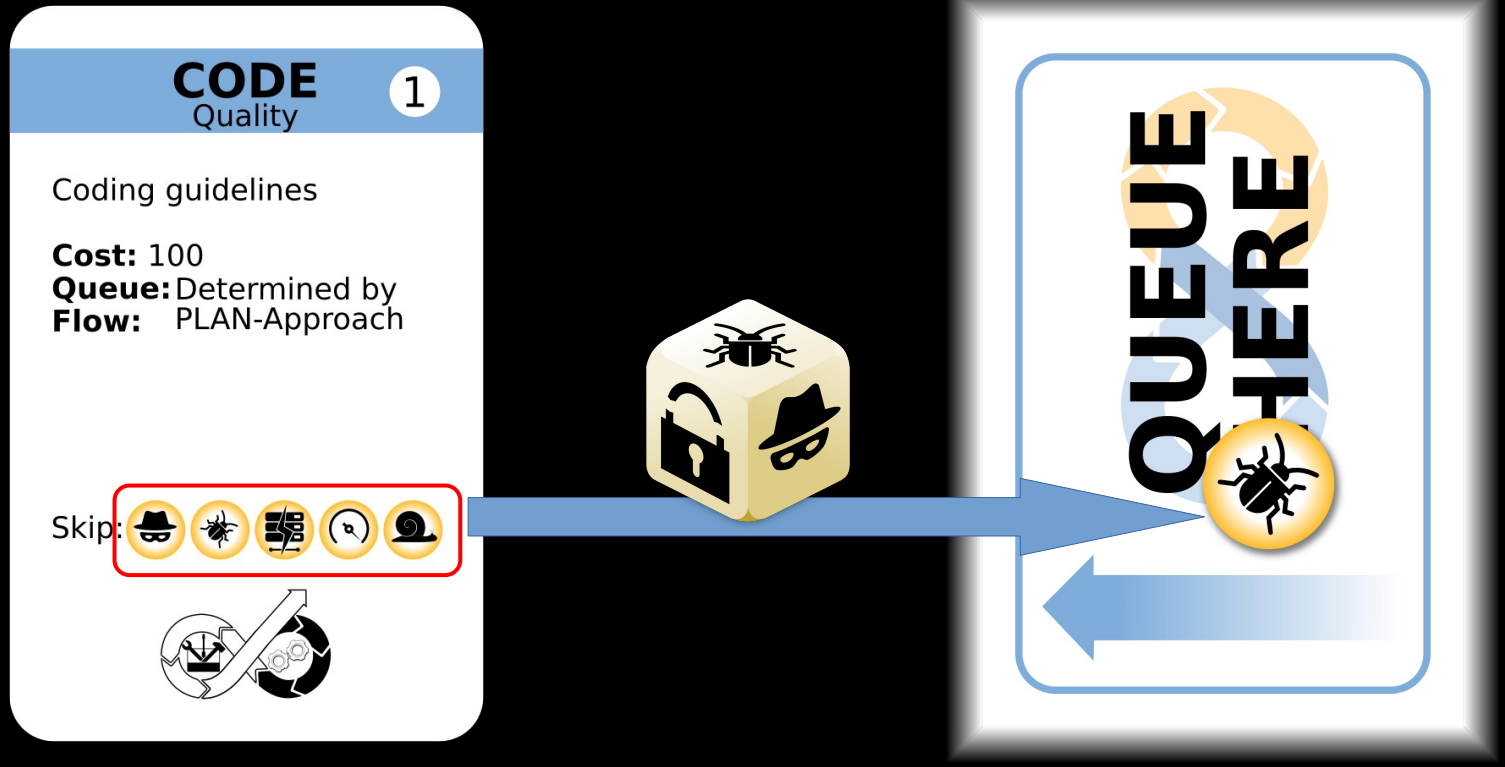


Cutting corners

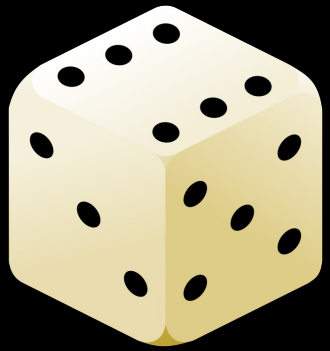


- Speed up delivery
- Bypass quality gates
- Create technical debt

Create technical debt

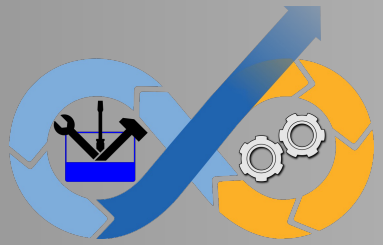


After each round

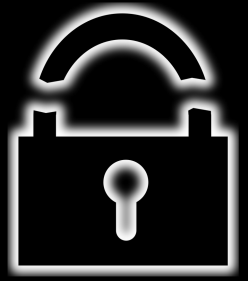


- Role both dice
  - You've got an even number?
  - You are impacted by the incident on the other die
- The even value = severity
  - 2 = low priority → 10% of incident cost
  - 4 = medium priority → 50% of incident cost
  - 6 = high priority → 100% of incident cost

What can possibly go wrong?



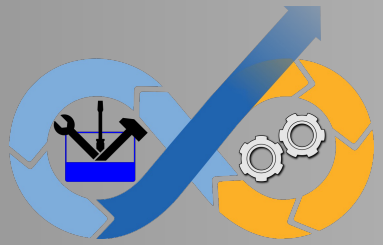
What can possibly go wrong?



Reported vulnerability
Fix ASAP

Bug
Fix ASAP
Count losses

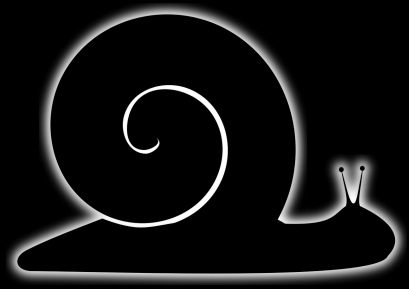
Security breach
Fix ASAP
Count losses



What can possibly go wrong?

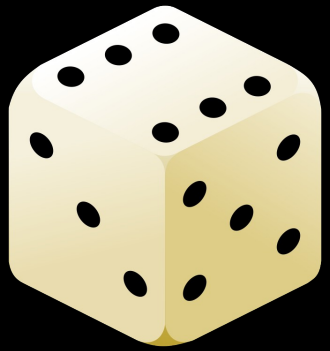
System outage

Unexpected load



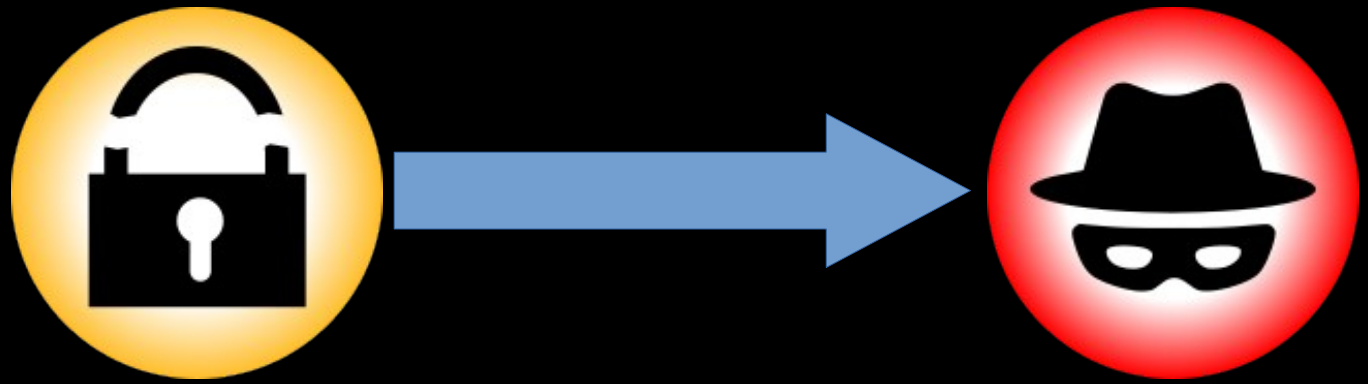
Performance issue

Solve the security vulnerability



- Take CVE token
- Skip Plan stage
- Use regular die to move fix through all stages
  - Ignore queue size
- No financial impact

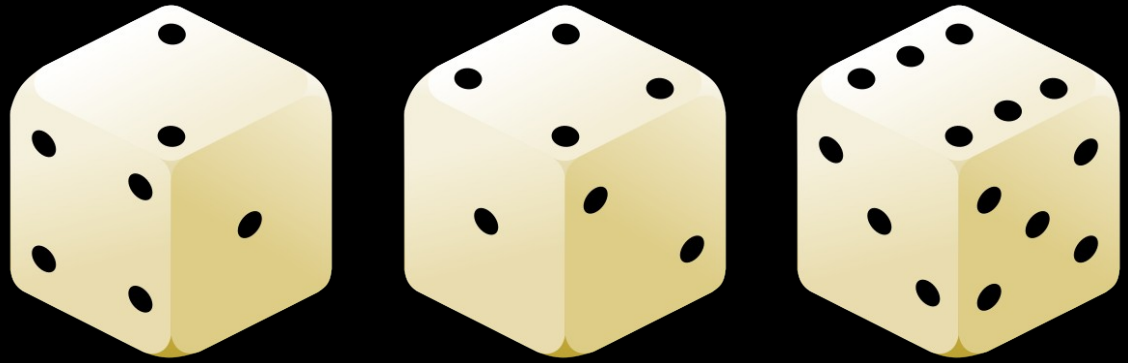
Unsolved security vulnerability



- If not solved before a new vulnerability is thrown, this becomes a security breach!
- → Replace with security breach token
- Count your losses

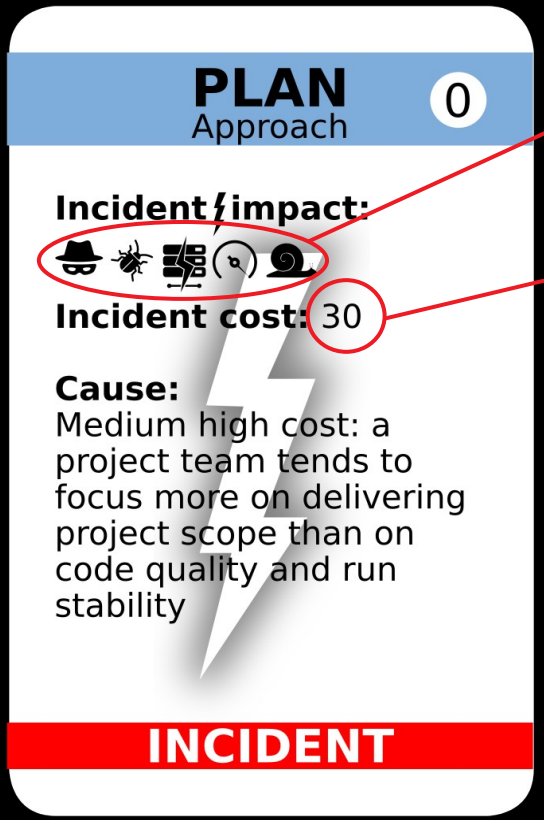
Dealing with incidents

Incident type



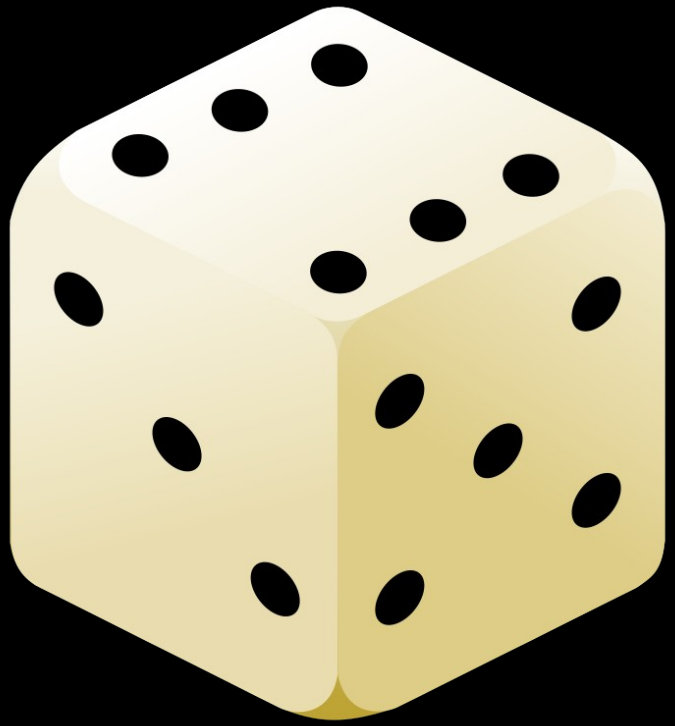
Incident severity

Flip all cards
Calculate financial loss



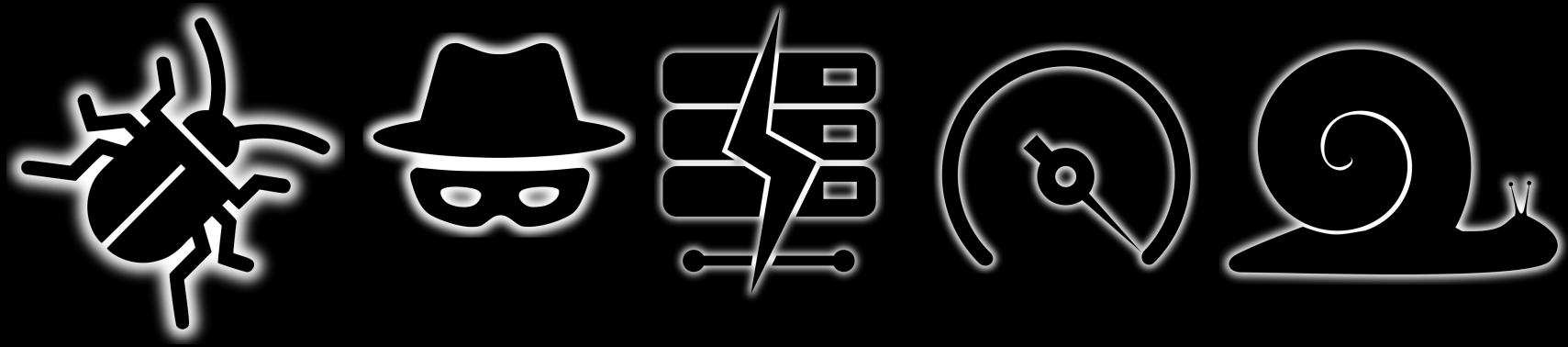
- Check the impact for each activity
- Sum the incident costs
- Apply severity multiplier
- Alternatively:
  - Only sum costs for activities you're responsible for

Severity multiplier



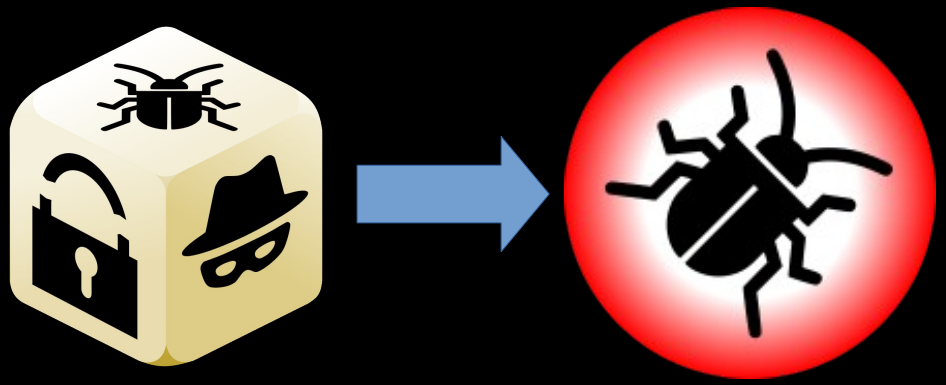
- 2 = low priority
  10% of incident cost
- 4 = medium priority
  50% of incident cost
- 6 = high priority
  100% of incident cost

Why extra cost?



- These incidents cause financial losses
- The lower your performance level, the higher the cost
  - Late detection & slow fixing = longer exposure

Fix the incident



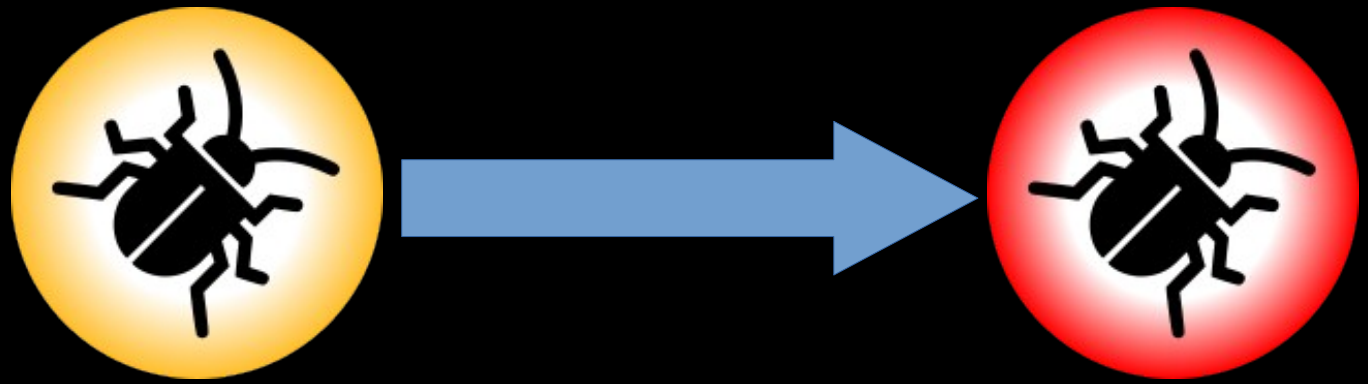
- Take corresponding red token
- Incidents skip Plan stage
- Use normal die to move fix through all stages
  - Ignore queue size

Accept incident risk

- Low prio or cost incident: pay loss
- Put token on board
- Don’t fix → accept risk

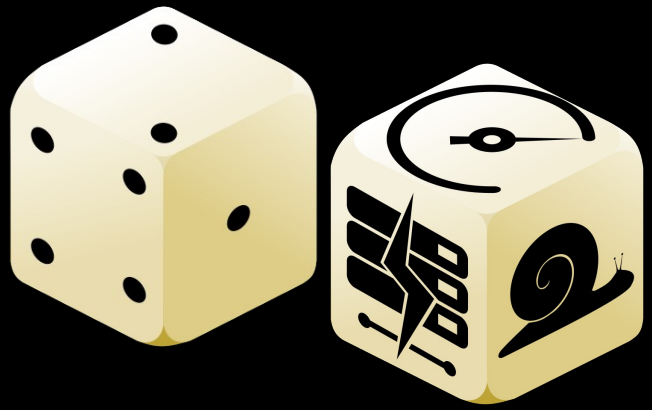
If the incident is not fixed when you roll the same incident type, you pay twice and need to solve 2 incidents!

Technical debt becomes incident



- Technical debt not solved when incident of same type occurs
- Technical debt becomes incident
- + add extra incident
- Double financial loss

Failed change?



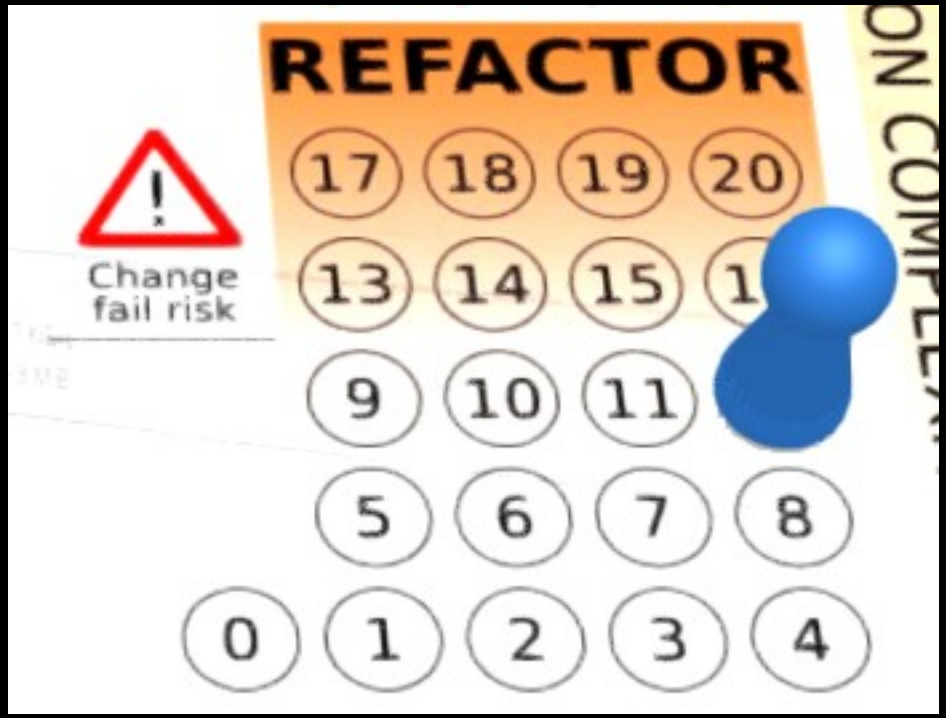
- New change in production = risk of failure
- When entire batch is delivered:
  Roll dice to see if an incident occurred after activation

Create revenue

Create revenue

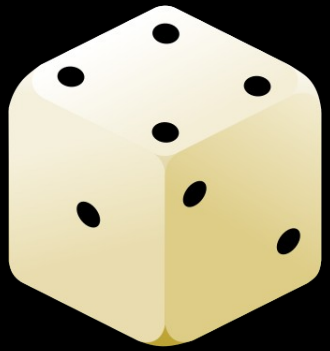
- Move features to this spot
  - According to queue size and flow
  - Earn money: 100 credits/feature
- No money for incidents, improvements, technical debt, CVE’s!
- Remove tokens

Track implemented features



- More features implemented = increased complexity
- As of 13: risk of failing changes, potential incidents
  → roll dice
- Above 20: refactoring necessary!

Invest to improve



- Put improvement token on “Queue here”
- Implement by rolling die
- Own cadence, dedicated people?

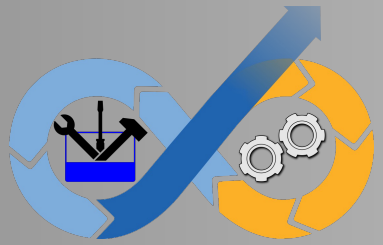
When to invest?

- Typically:
  - At the start of the game
  - After delivering features – when you get revenue
  - When a serious incident occurred
- But in general: whenever you want to and have the means to

And now…
Let the game begin!

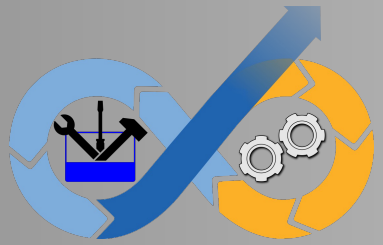

BUILD - RUN
ROVE - REPEAT
REFACTOR
REFACTOR
REFACTOR
FEATURES IMPLEMENTED - SOLUTION COMPLEXITY
CASH HERE
DEPLOY 2
Changes to infrastructure are done through self servicing
Cost: 150
Queue: Determined by
Flow: PLAN-Approach
DEPLOY 2
Code
Automated deploy to test, manual approval for UAT, production
Cost: 100
Queue: Determined by
Flow: PLAN-Approach
TEST 1
Security
Penetration testing before going to production
Cost: 100
Queue: Determined by
Flow: PLAN-Approach
RELEASE 0
Activate
Upon deploy
Cost: 0
Queue: 4
Flow:
Determined by PLAN-Approach
RELEASE 1
Approval
Business go-no go meeting
Cost: 20
Queue: Determined by
Flow: PLAN-Approach
DEPLOY 0
Frequency
Determined by PLAN-Approach

Debrief



Learnings

- First focus on the build quality
  - **Don’t be tempted to start delivering faster!**
- Slow progress in the beginning
- Will prove good foundation once you improve delivery
- Security issues can have high financial impact
  - Improve these first!

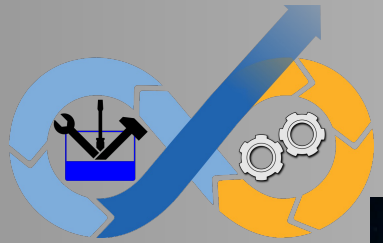


Learnings

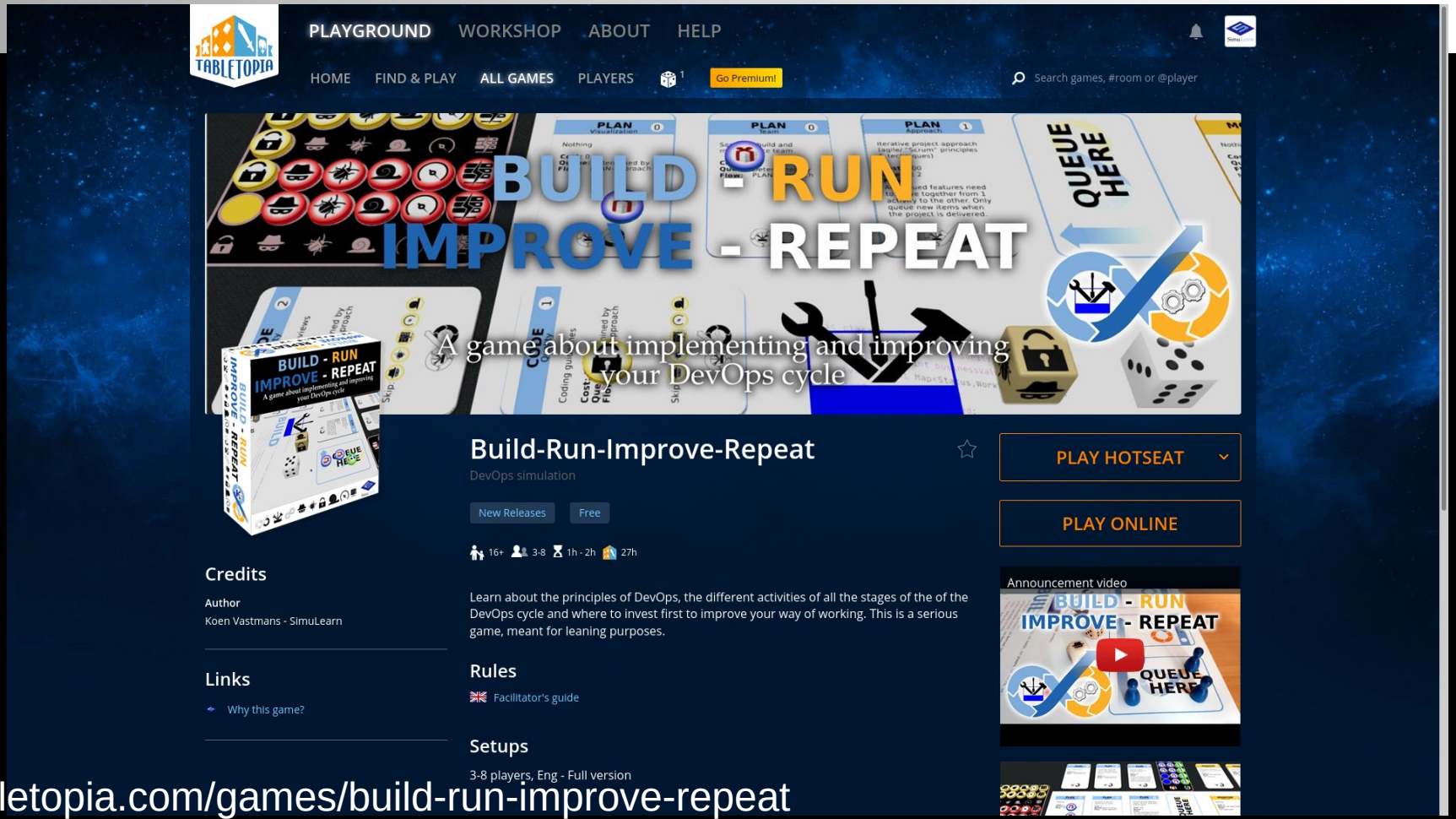
- Evolve to small batches and automation for faster revenue

  Smaller batches will get full benefit with shorter deployment intervals
- Don’t forget availability, stability and performance of your system!
- Shared responsibilities, budget and decisions are better than split responsibilities

Get started yourself



Go to Tabletopia.com



https://www.tabletopia.com/games/build-run-improve-repeat

THANK YOU!

SimuLearn