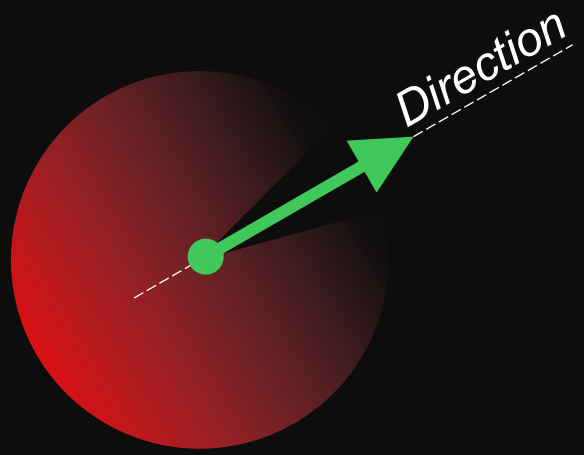
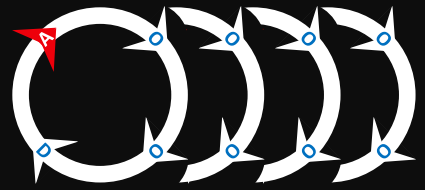
Ways R&D Managers Get the Engineers to Learn Faster, Thus More

Keep it Simple – No Premium for Complication



Full Feedback Cycles



The Team's Work Flow

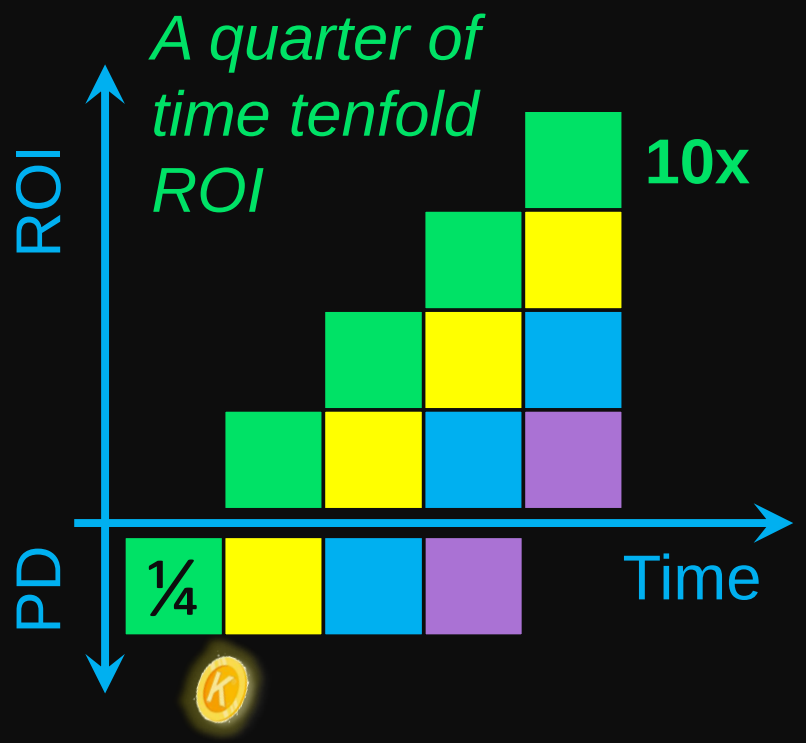
THE SCIENCE OF IMPROVING INNOVATION

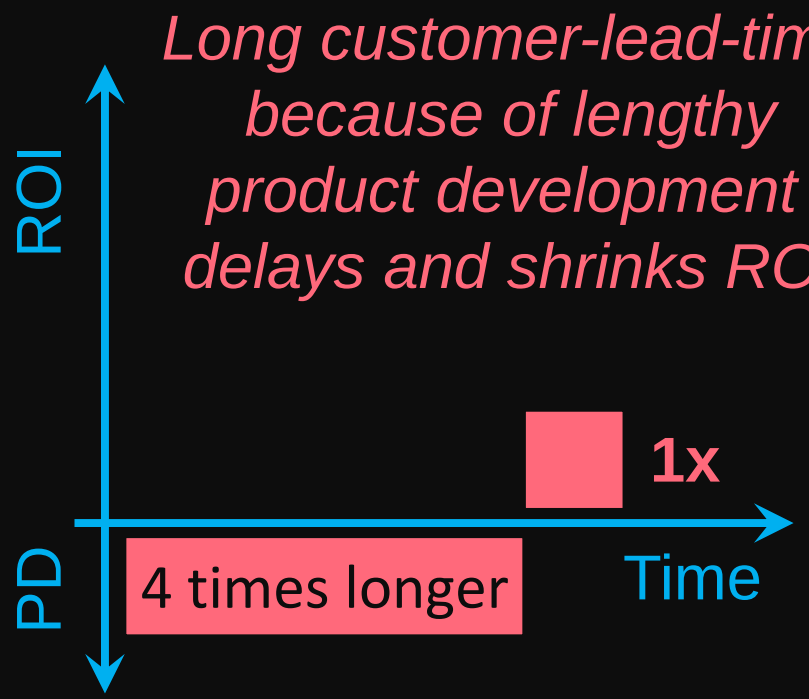
AUFERO

Winnings at Stake

AUFERO

Reducing Customer-lead-time to a Quarter (¼) can Tenfold (10) ROI





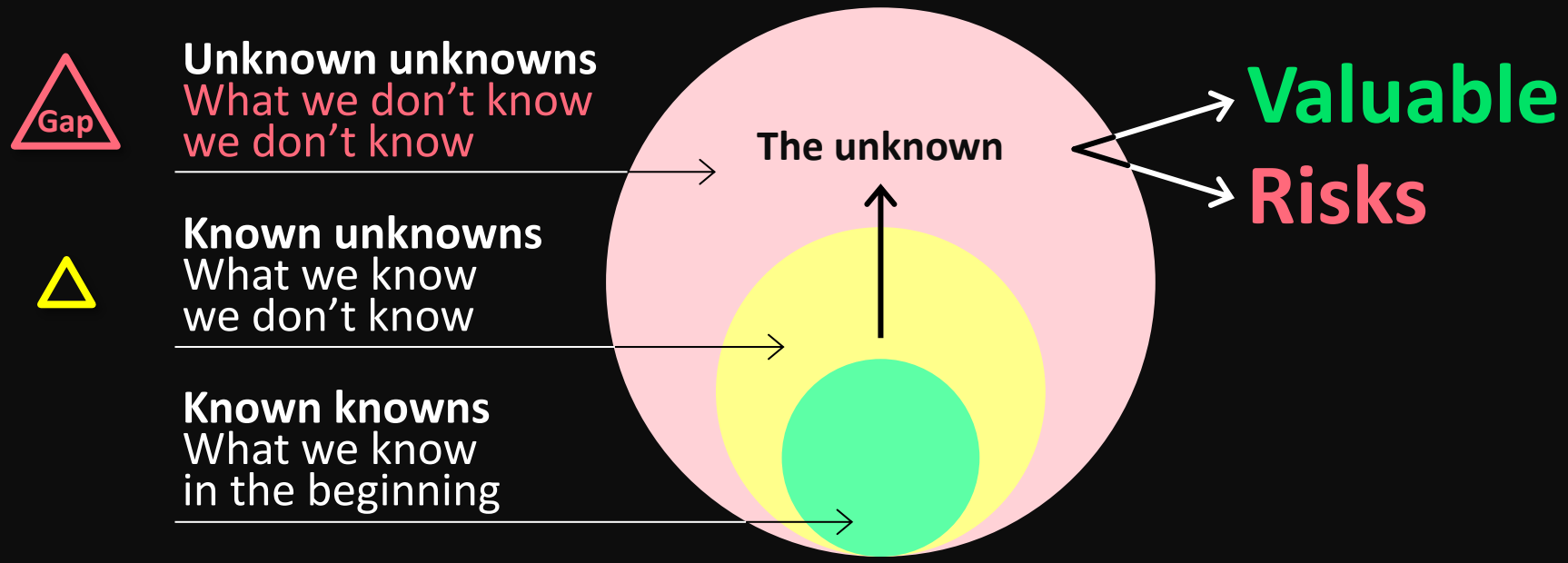
Benefits of shorter time to market cumulates. Set an aspirational target to reach development in a quarter of time and you will conquer.

Test First

AUFERO

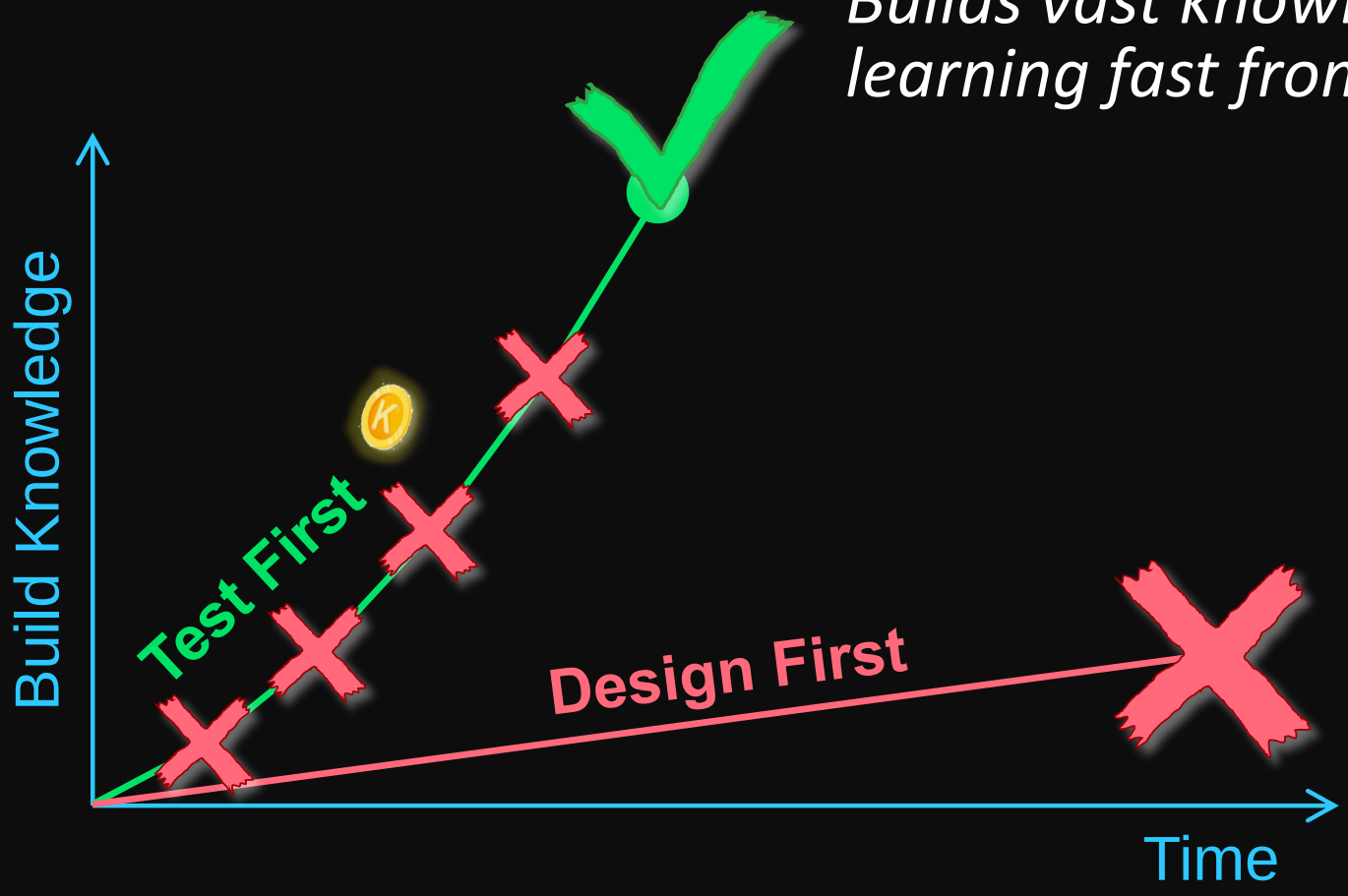
Development has a **KNOWLEDGE** Issue

Unknown Unknowns a Major Problem



"Unknowns can only be understood in retrospect of tests" [Dave Snowden]

Test First to Build Knowledge and Create Value



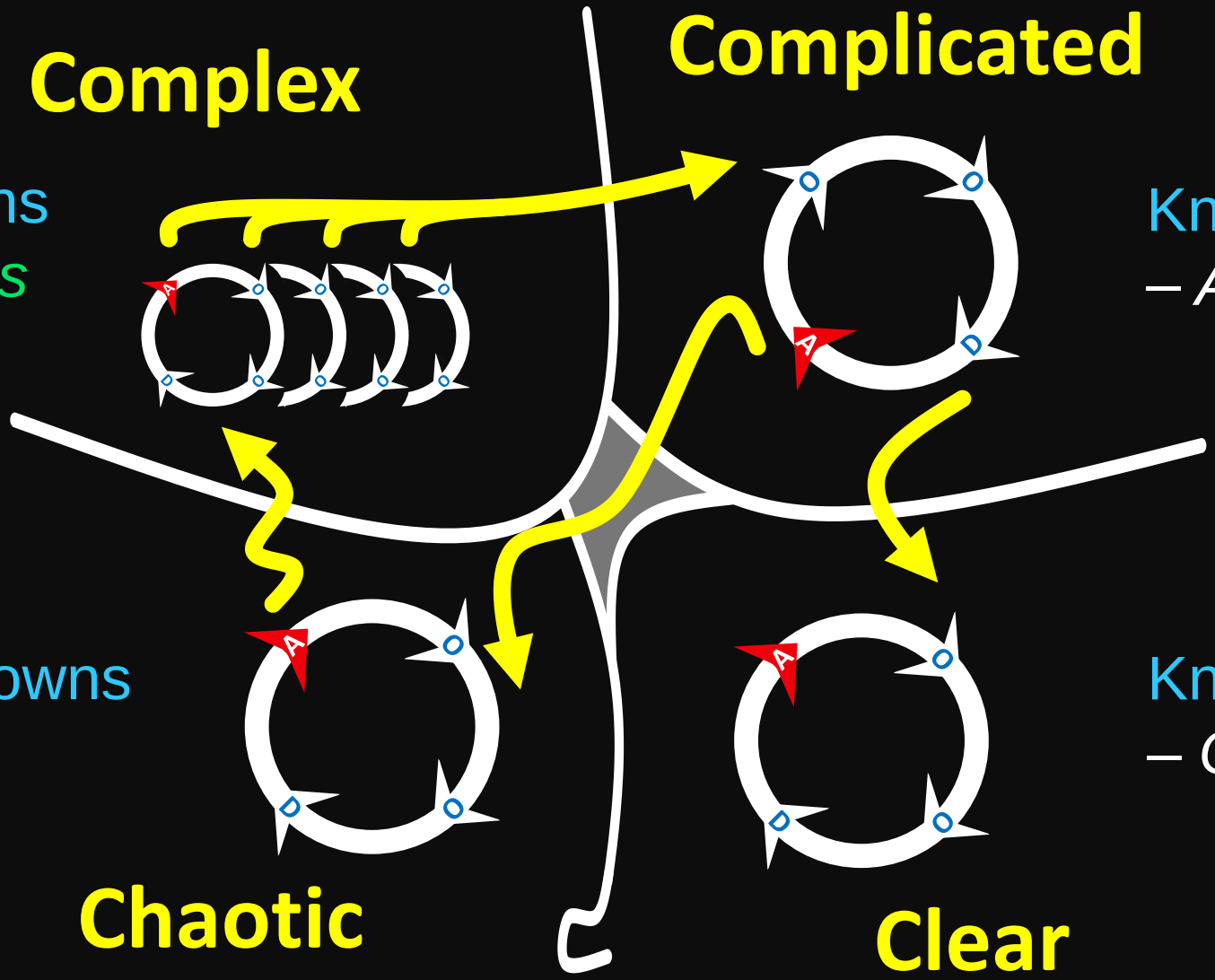
Builds vast knowledge bottom-up by learning fast from what doesn't work

Perfects the design of the super system first.

Harmed by late test failures

CYNEFIN Problem-solving Applying OODA-loop

What kind of problem am I trying to solve?



Unknown Unknowns
– *Run multiple trials with small errors*

Known Unknowns
– *Analyze*

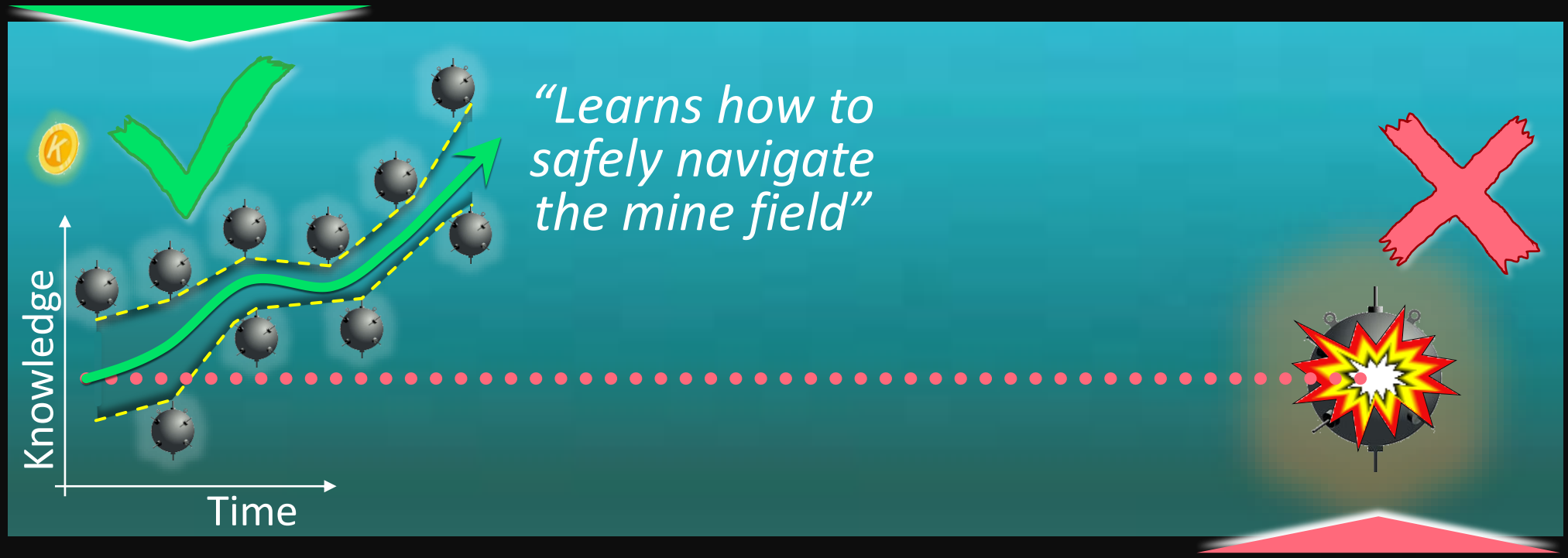
Unknowable Unknowns
– *Act, act, act*

Known Knowns
– *Categorize*

Source: Dave Snowden (CYNEFIN Framework) combined with John Boyd's OODA-loop

Learning Edge Cases = Building Knowledge ≠ Fail Fast

Learn and build knowledge from what doesn't work – outer limits. Mitigating risk in the current and superior starting point in the next development.



Big batch testing of complete systems, backend verification, increases the risk. Team gets squeezed by late failure detection. Knowledge is discarded.

Shorter Iterations

AUFERO

The Feedback Cycle is the Heart of Learning & Development. Where the No. of Cycles Makes a Huge Difference.

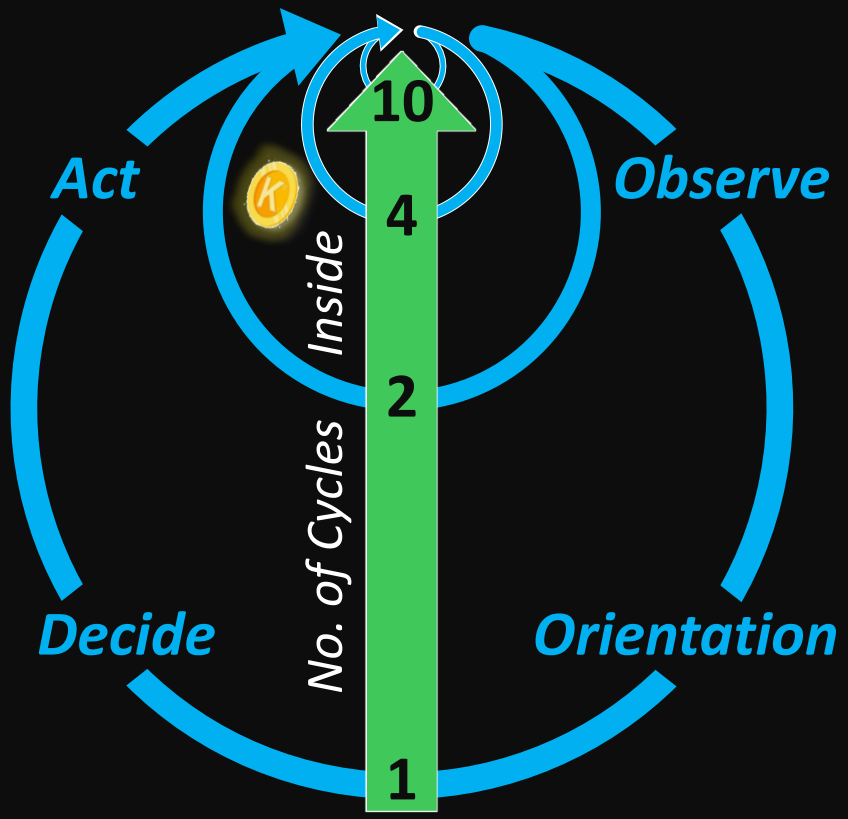


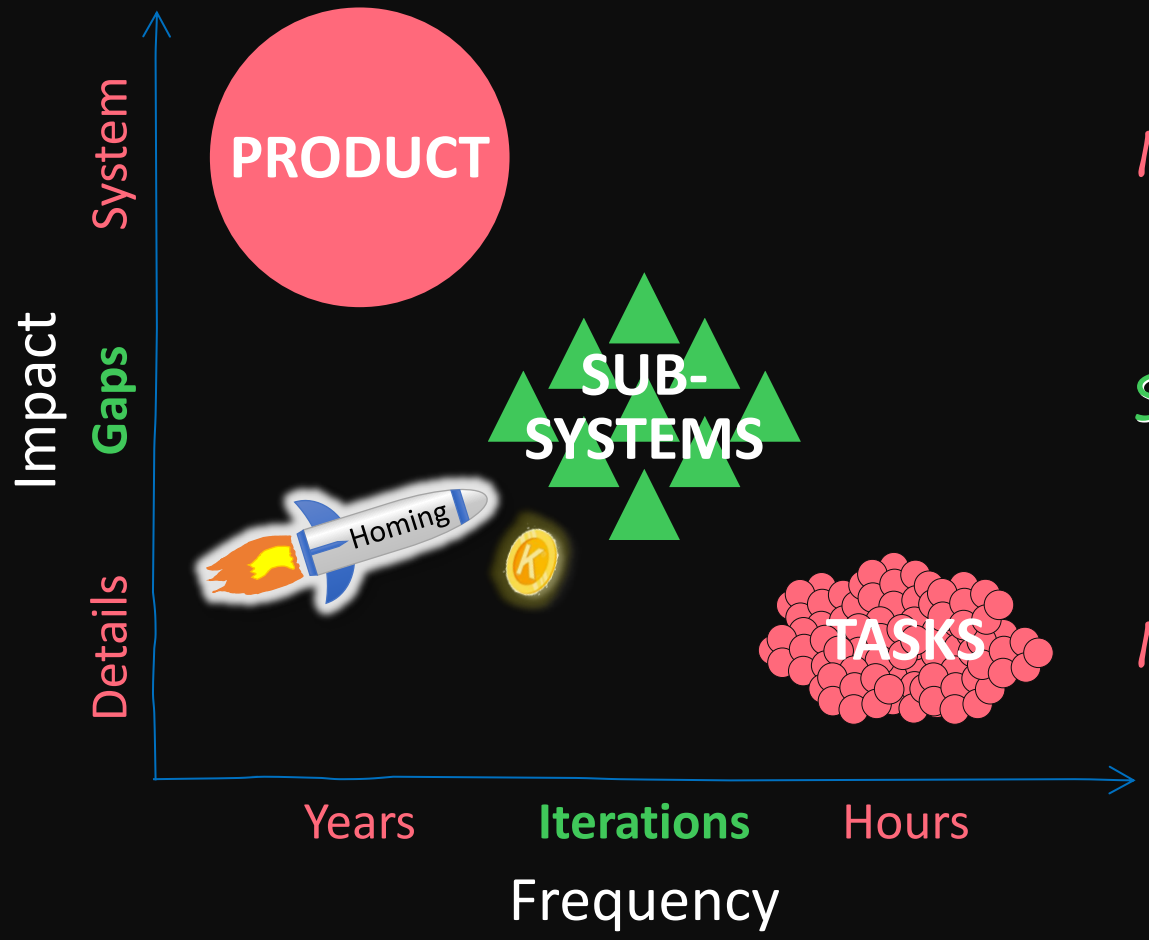
Image. John Boyd's OODA–loop, as example of efficient feedback loop.

Continuously shorten the cycle time of your feedback cycles.

Today or in ten days?
In ten or in 100 days?

Radically short feed-back cycles becomes self-reinforcing.

Make Difference, Focus on Problem-Solving (right-sized)

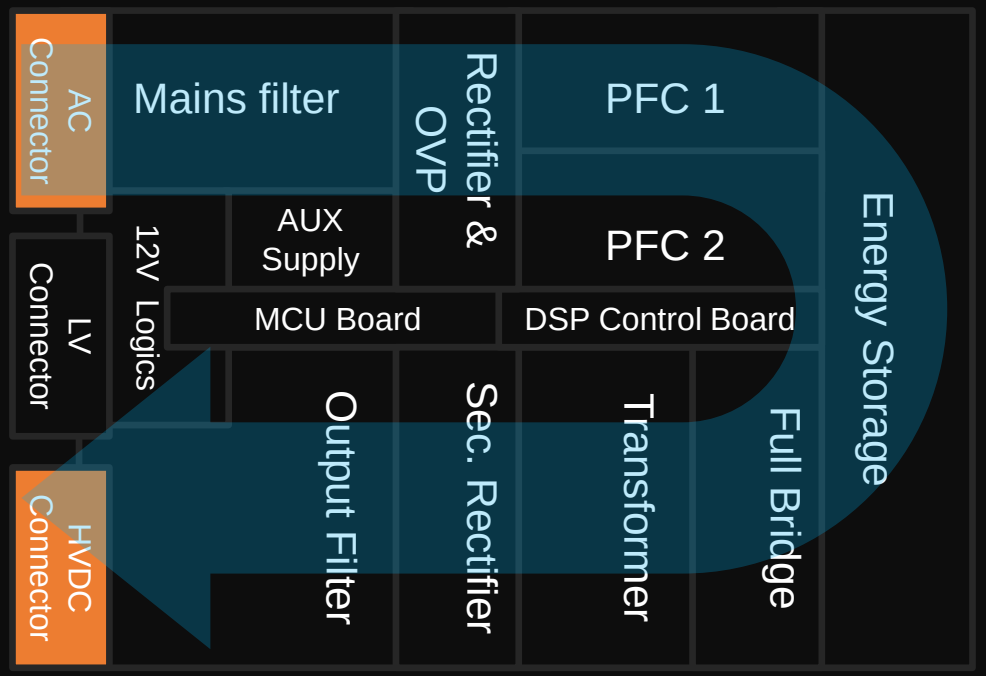


Not products - single, too large

Sub-systems - dozens, right-sized

Not tasks - countless, too small

Modularization – Run XS Missions on Sub-systems in Parallel



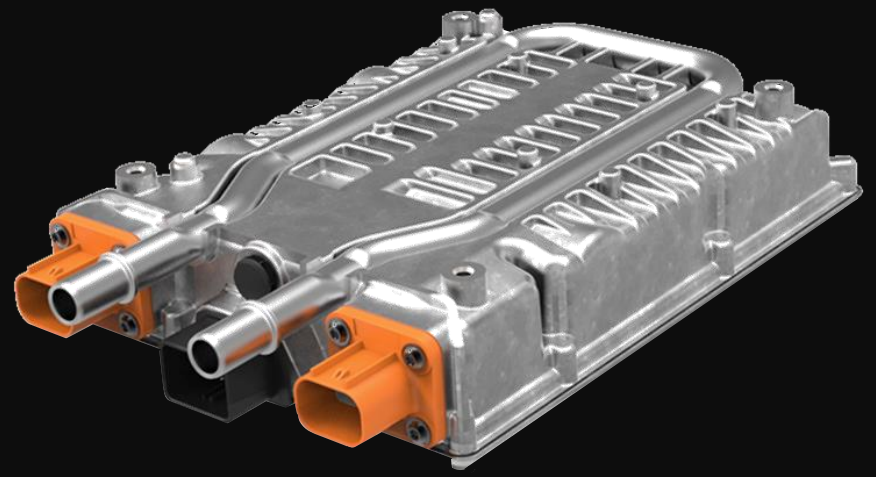
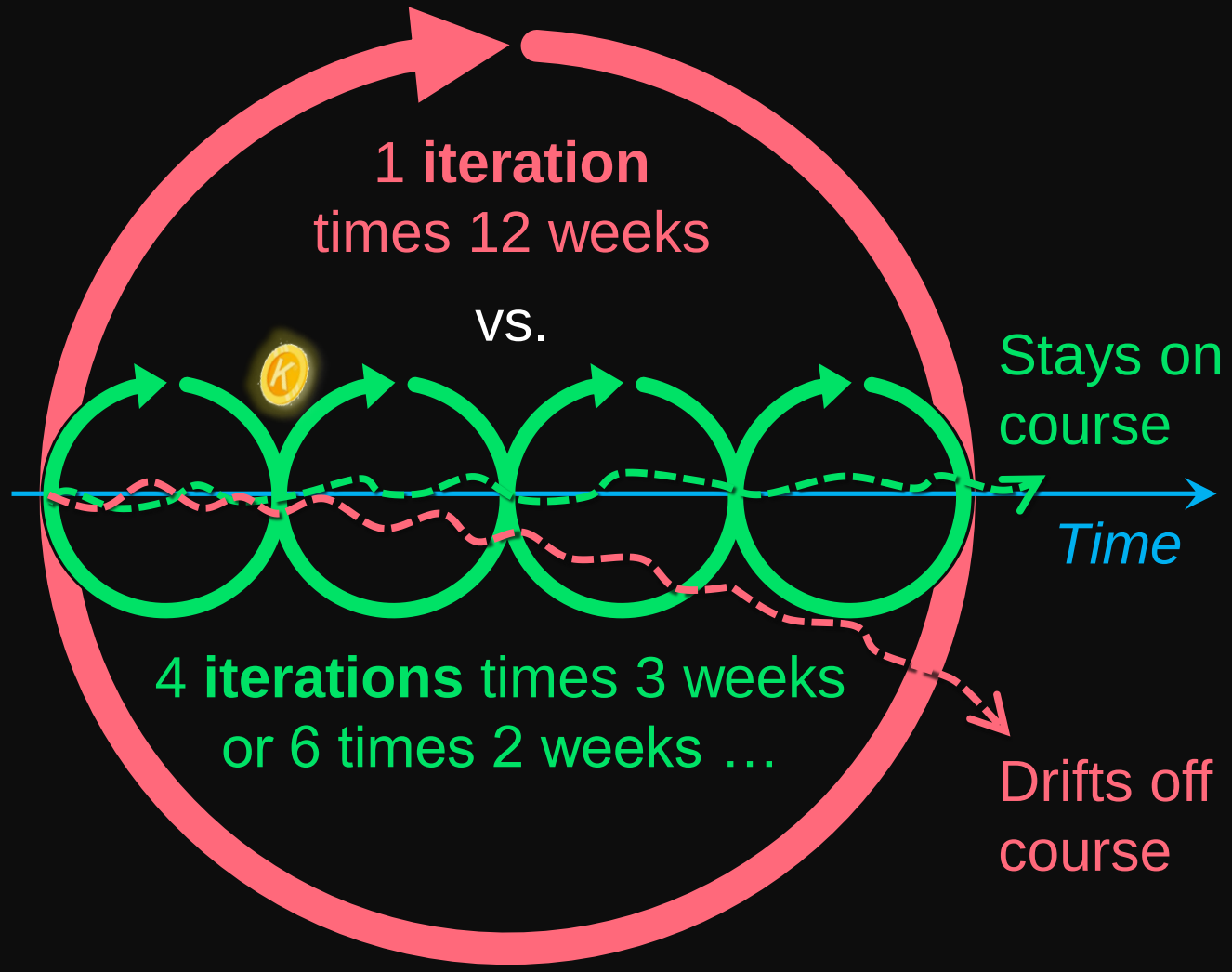


Image. Electrical board design structure. Flow of power (blue arrow – AC input to DC output) connecting input and output of the different subsystems / -modules / -features.

Stays the Course Better with Small Iterations



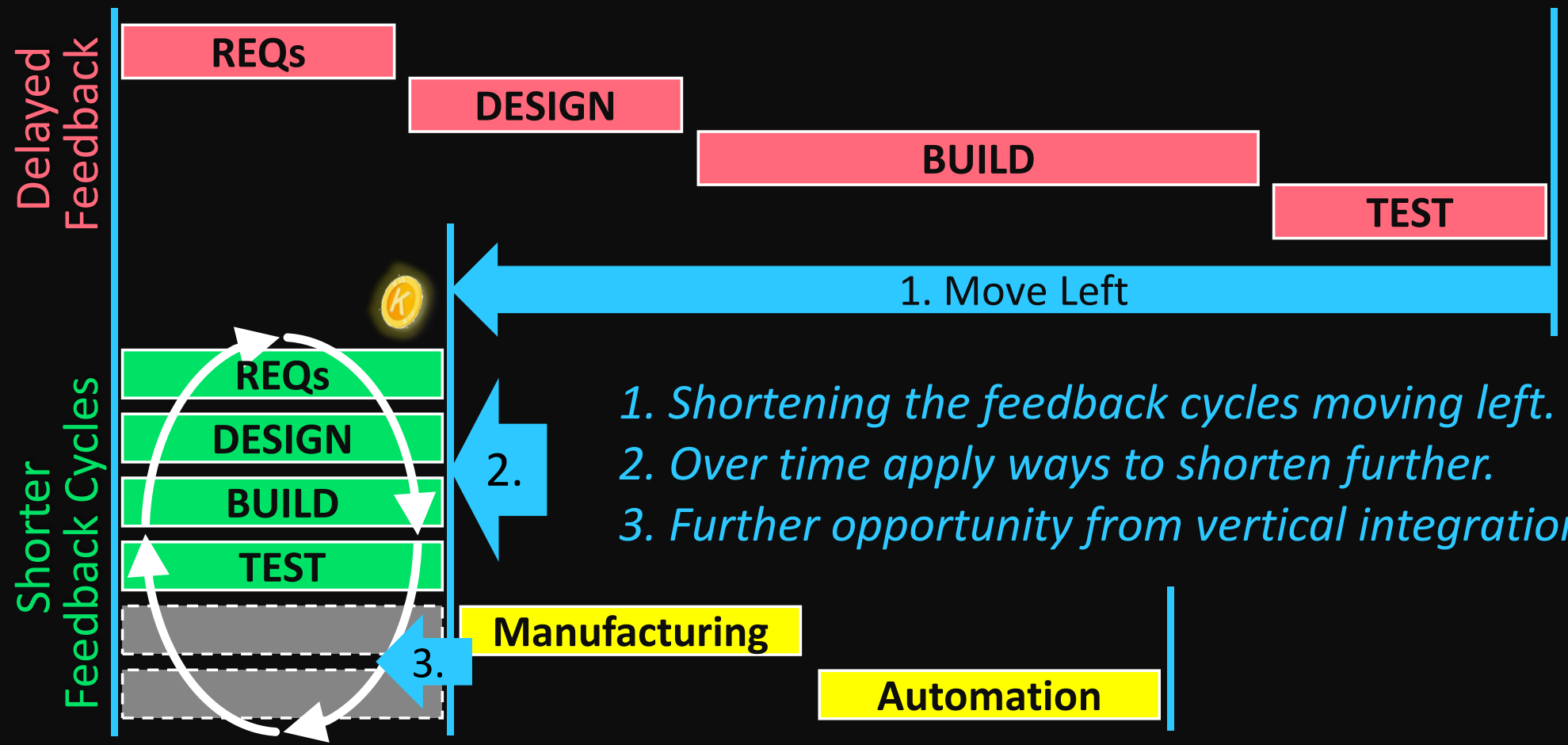
Plans need to:

1) Stay flexible *with frequent ways out,*

and counterintuitive:

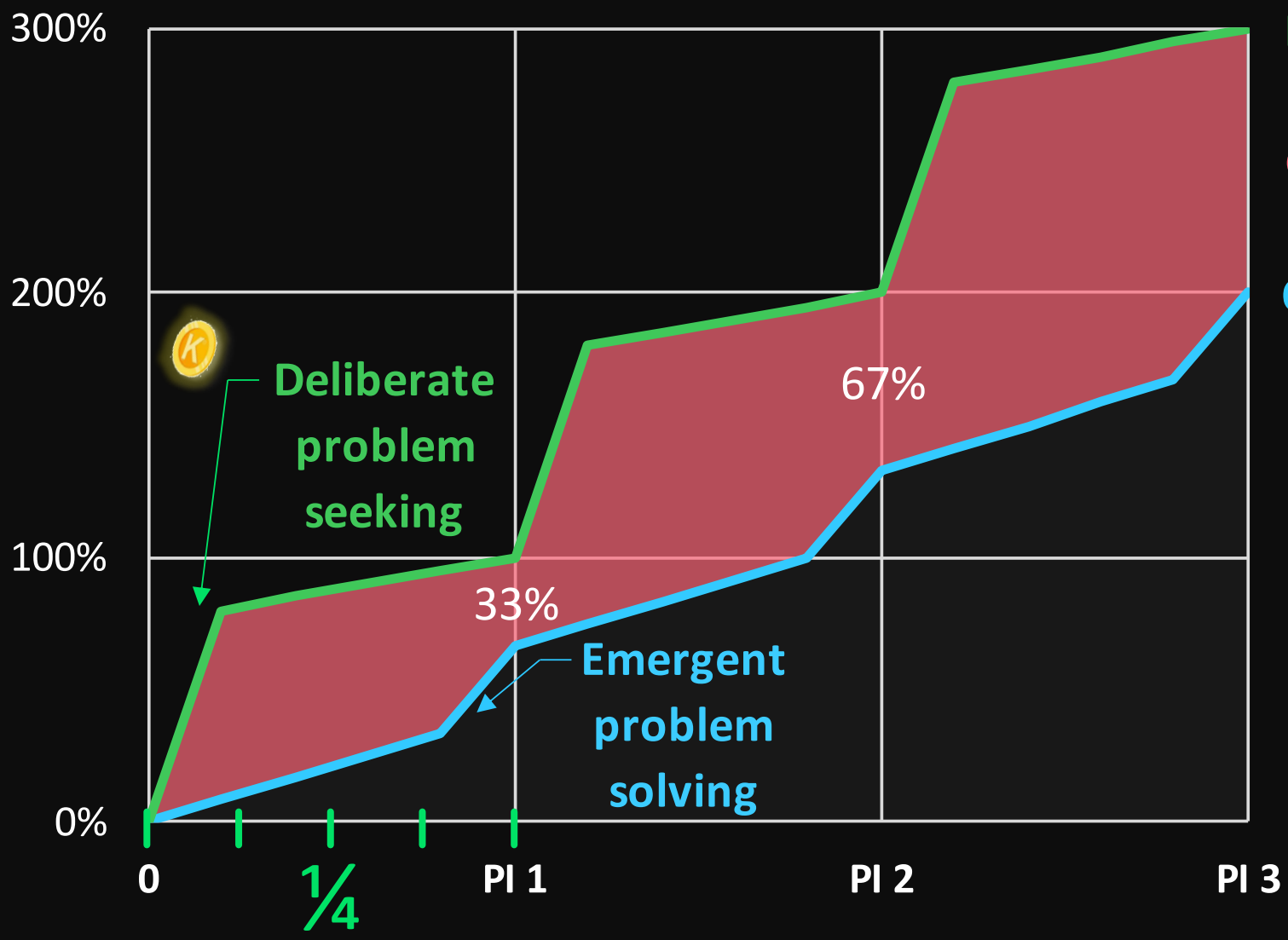
2) Be very short term*, in order to properly capture the long term.*

Shorter Feedback Cycles Moving Left



1. Shortening the feedback cycles moving left.
2. Over time apply ways to shorten further.
3. Further opportunity from vertical integration.

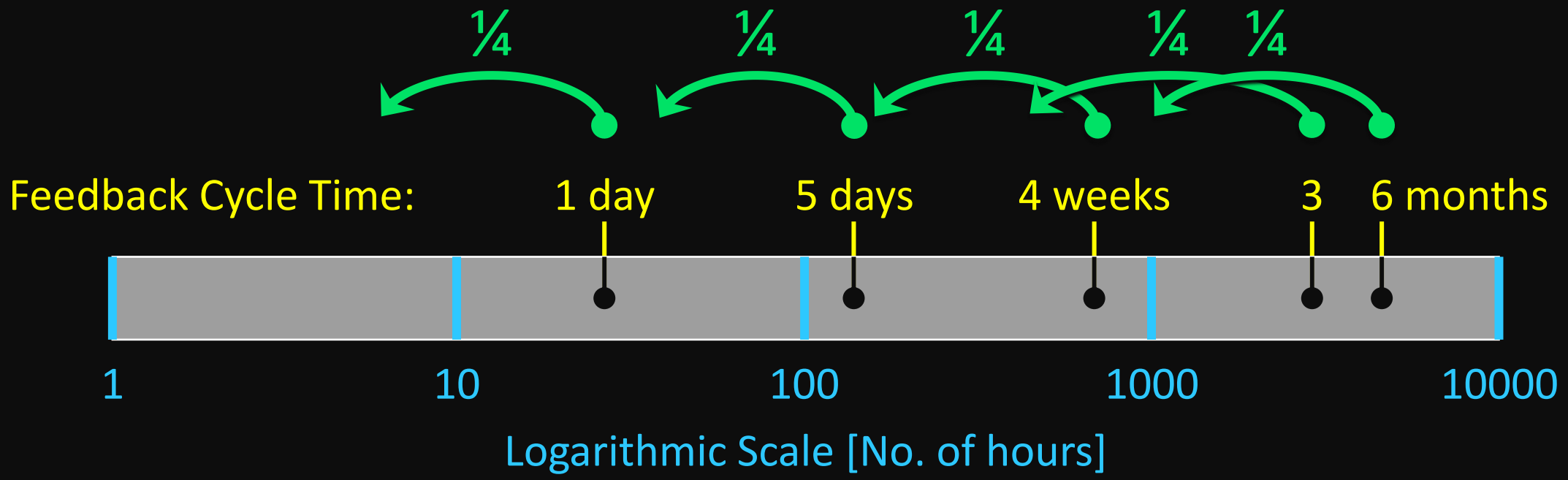
Trapped by Detailed Plans – Cumulative Remaining Backlog



Contrary to what many hope, development deals with complexity, uncertainty and variability.

Shorten the Lead Time of Feedback Cycles – to Build and Test

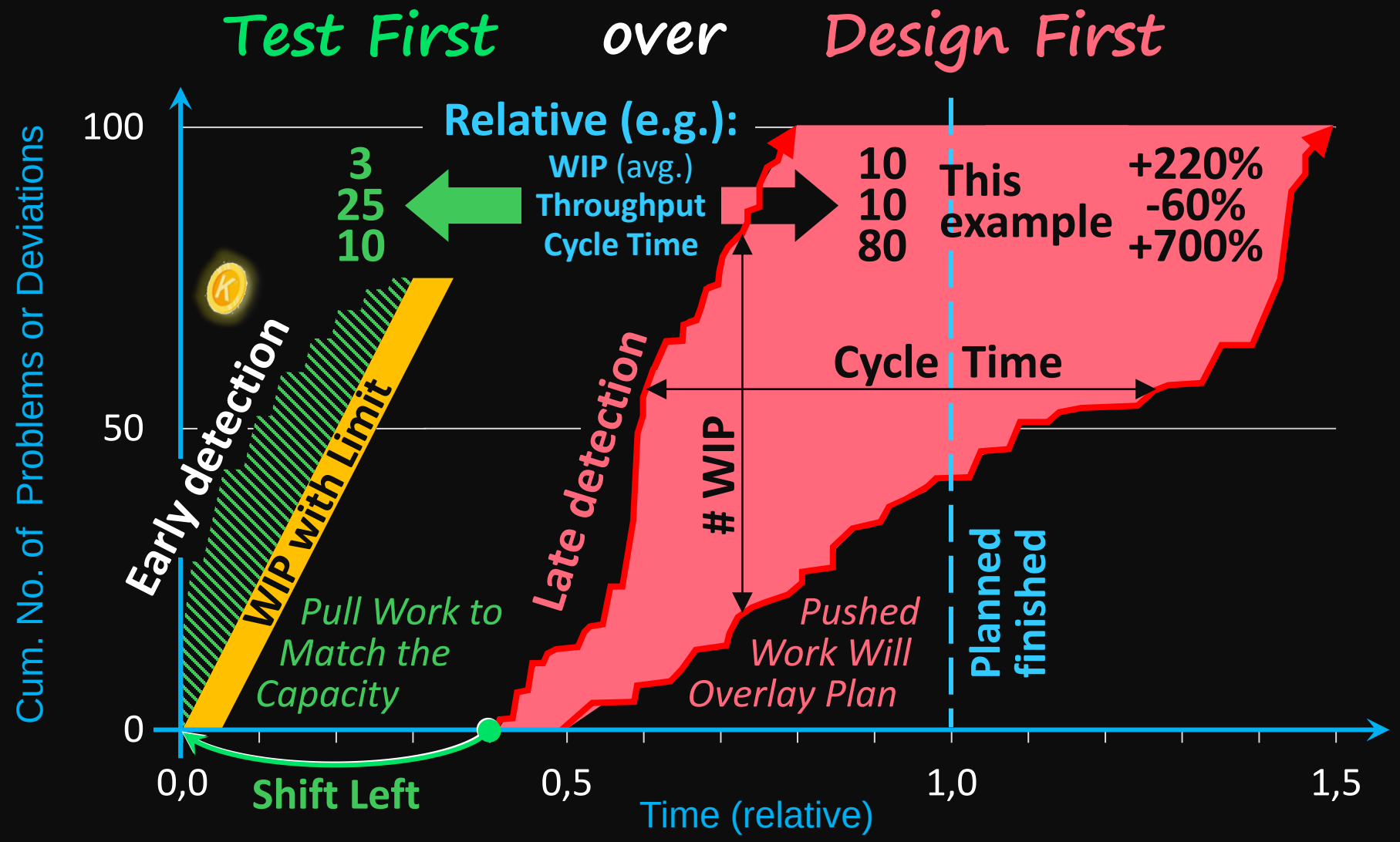
"State and support an aspirational target to acceleration the feedback cycle time."



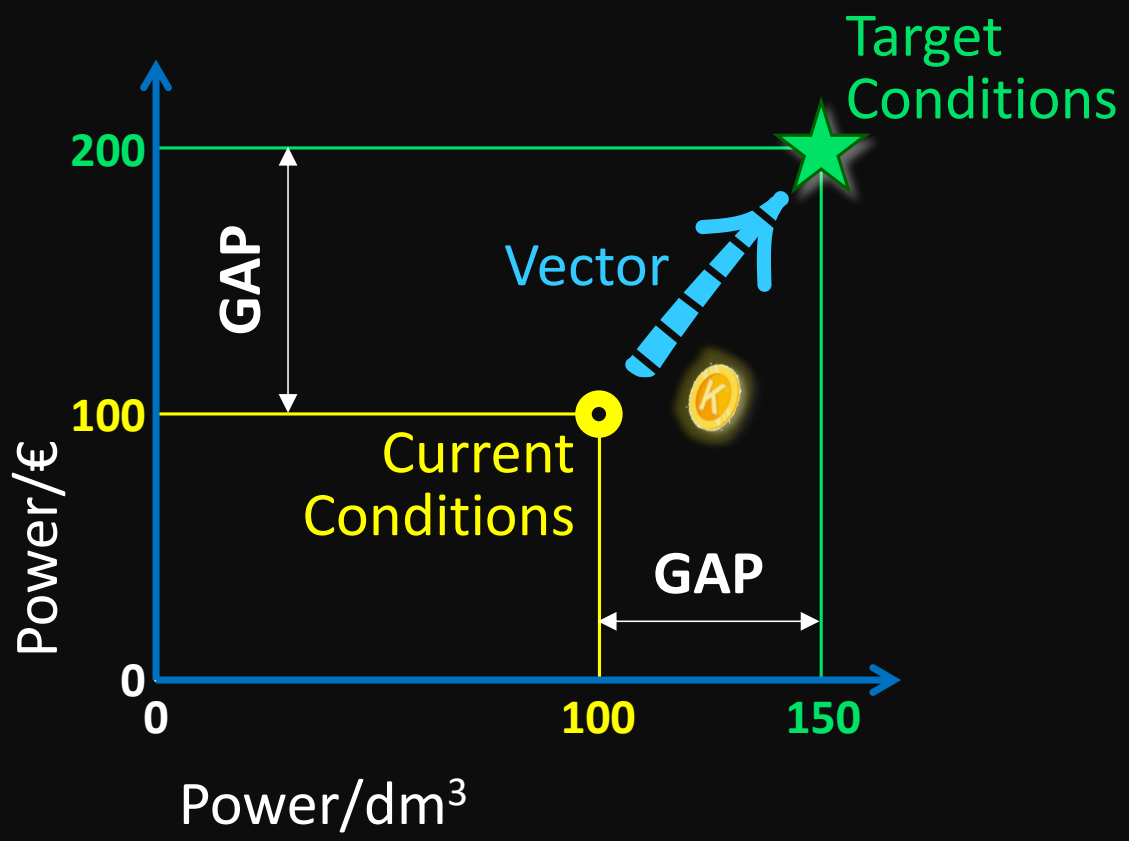
Deliberate Problem Seeking

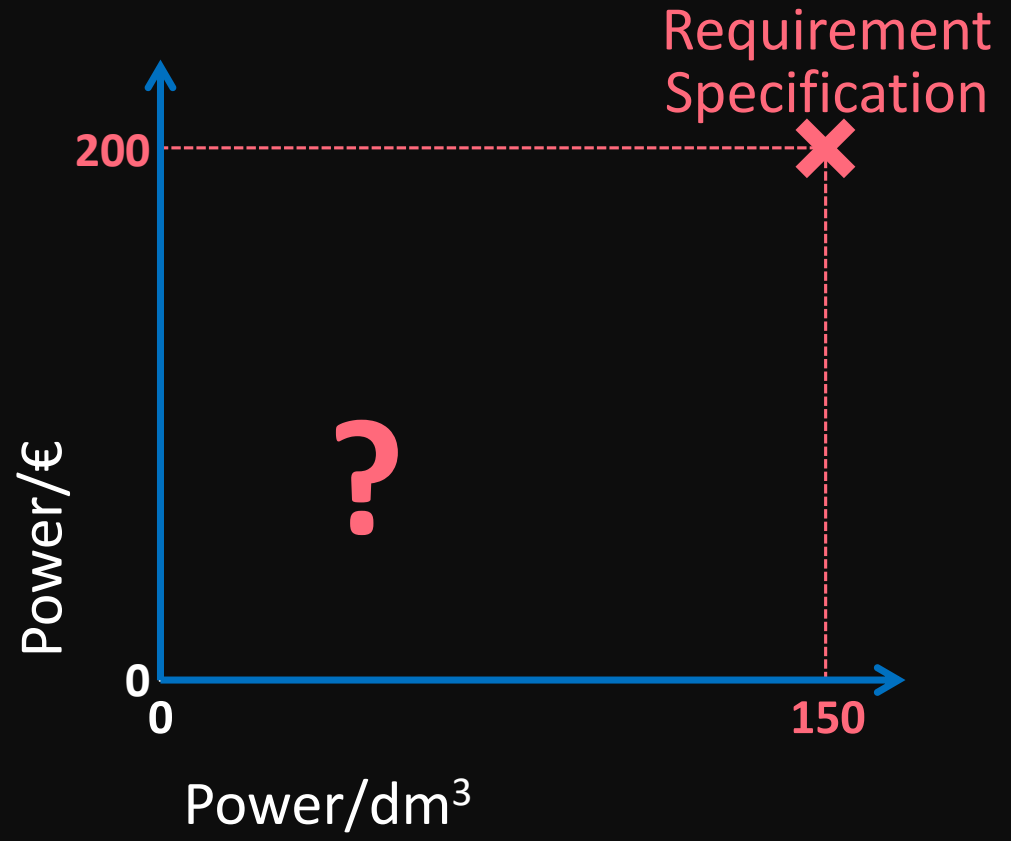
AUFERO

Deliberate Problem Seeking and Benefit From Flow



Leaders and Teams Do Better By Measuring Vector





*Define aspirational **Targets**.*

*Test as simple as possible to quantify **Current Conditions**.*

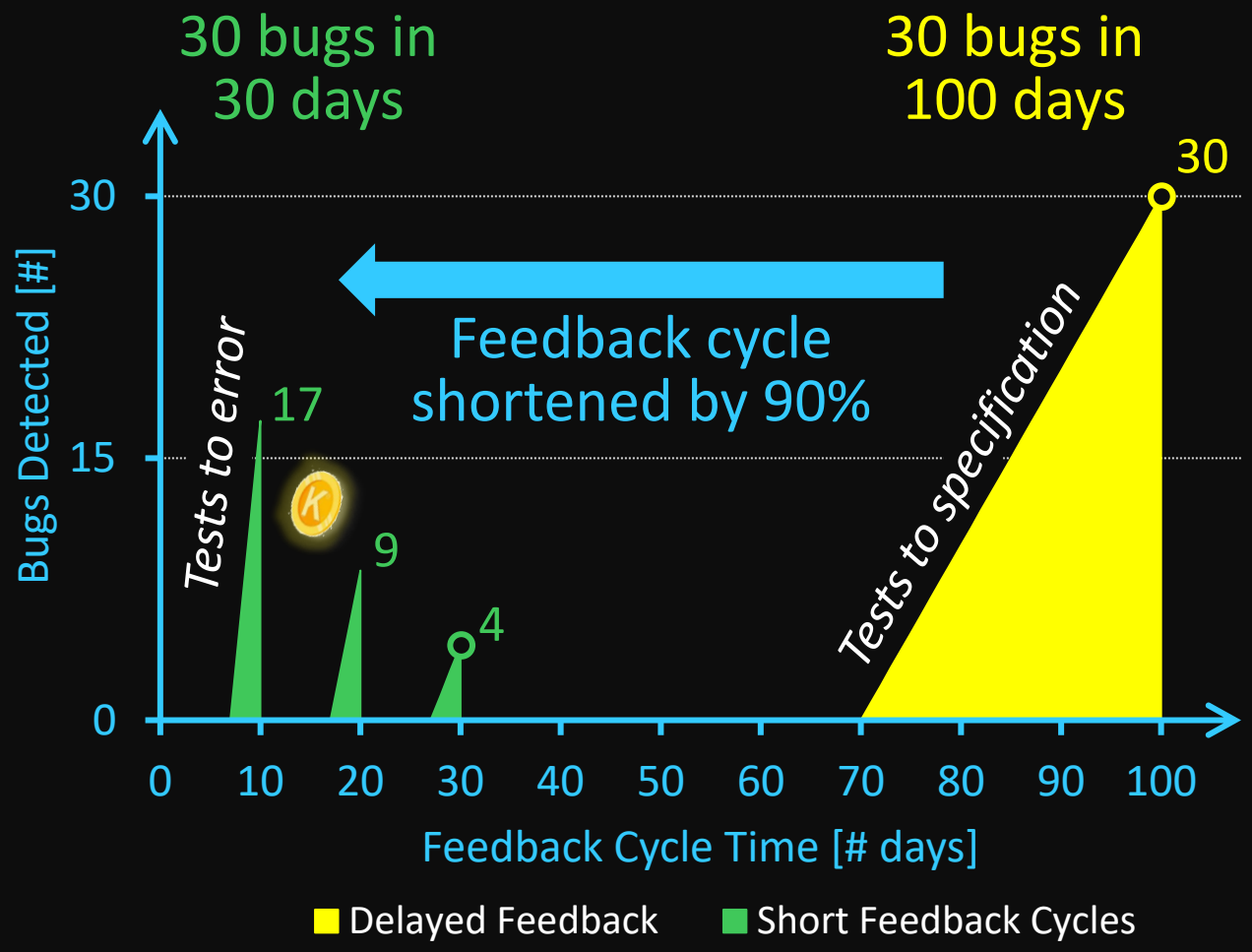
*Quantify the **Knowledge GAP**. Measure vs. the **Vector**.*

Far inferior way of working
*– developing against **Requirements**.*

Test Aggressively

AUFERO

The Feedback Cycle is the Heart of Learning & Development. Where the No. of Cycles Makes a Huge Difference.



For development, the **FEEDBACK CYCLE** is the heart of learning. How to close knowledge gaps. Where the **number of cycles**, and where **tests to error** makes a decisive difference.

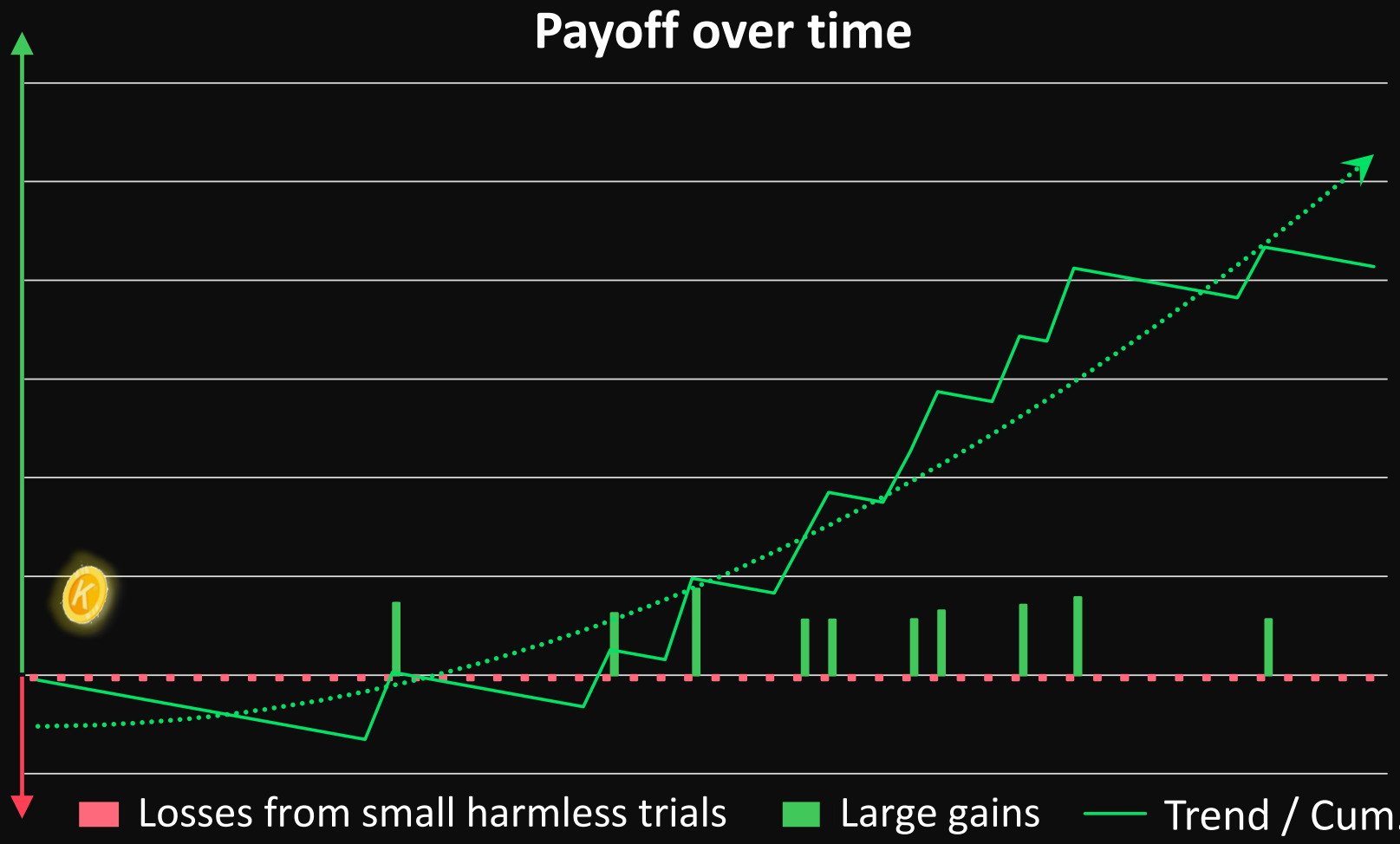
* **Source**: Allen C. Ward, Lean Product and Process Development, 2007, page 55.

AUFERO

Test Small

AUFERO

‘Trials with Small Errors’ the Essence of Set-Based Design*



- Small frequent and harmless errors, but **Big Infrequent Success!**
- **Trial with small errors** work because you’re rational and keep what you found is better than we had before.
- **CREATE OPTIONS**, keep all upsides and unaffected by all small downsides.

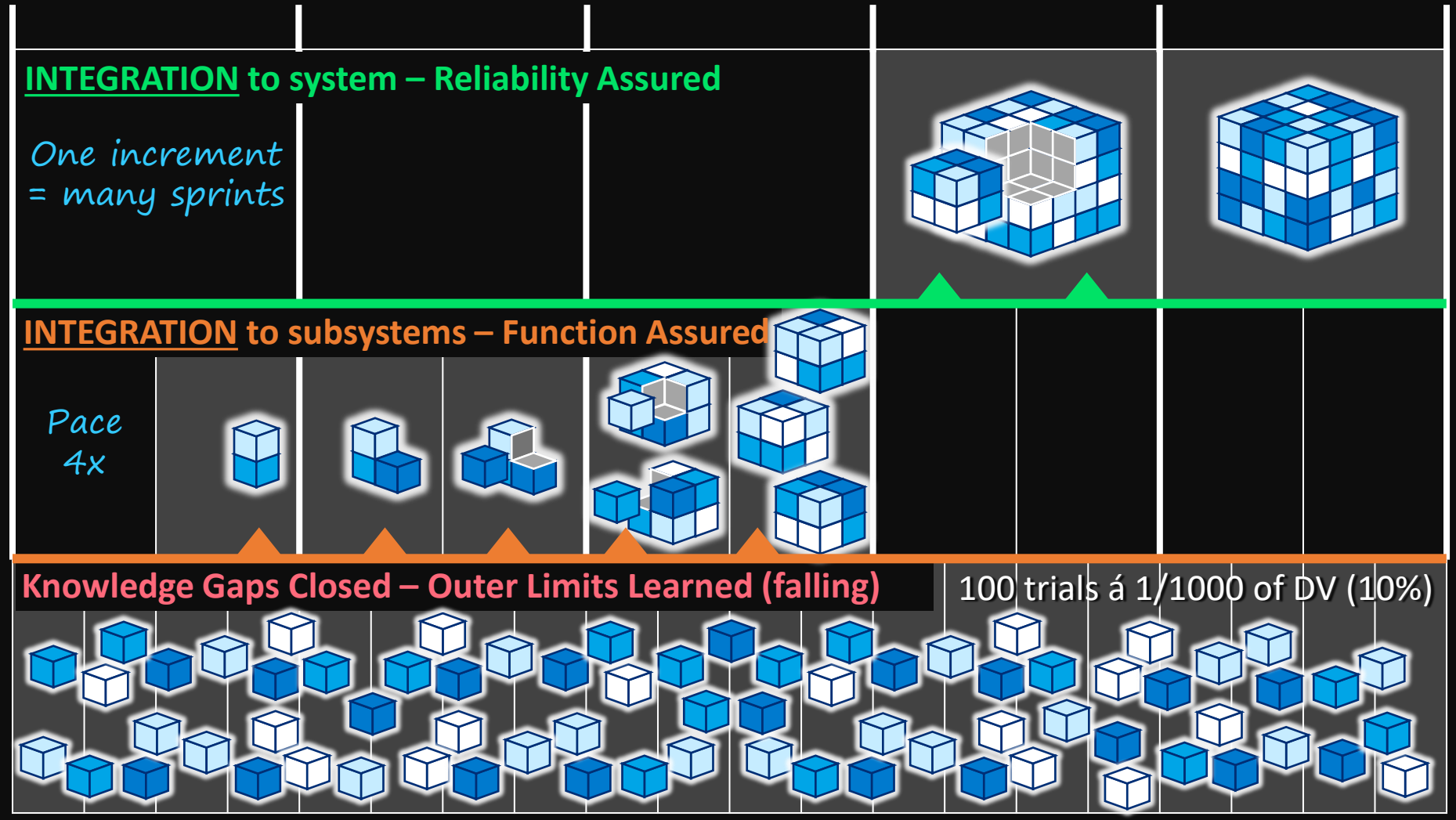
**Set-Based Design; explores alternative sets of design decisions. Contrary to Point-Based, that makes design decision based on one solution.*

How Do You Mitigate Risks In Your Development Activities?

Continuous flow of small trials. Early errors are most often small and benign, even reversible and quickly overcome. They are also rich in information. Ensures front-end ‘subtracting’ of what doesn’t work.

Big batch super-system testing, back-end to verify. Late detections of failure forces ‘adding’ to make it work.

Small Teams Run a Continuous Flow of Small Trials



Keep it Simple
No Premium for Complication

- Test first >> Design first
- Test small:
  – Modularization
  – Sub-systems
- Test aggressively:
  – Learn edge cases

- Tiny short iterations:
  – Stays the course
  – Measure the Vector
- Right-sized missions
- Accelerate feedback cycles

- Quantify **knowledge** gaps
- Close **knowledge** gaps
- Build **knowledge** bottom-up

≠ Fail fast

¼

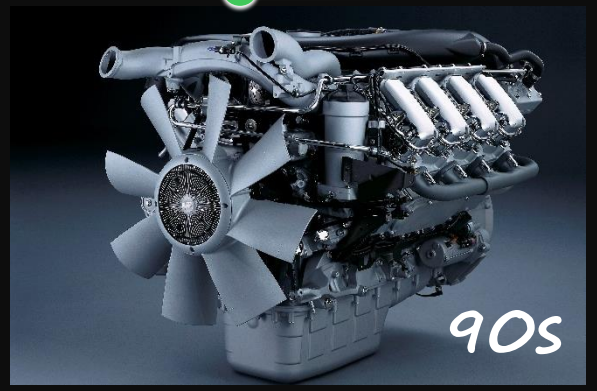
AUFERO

Christer Lundh

Funder & Owner of AUFERO AB

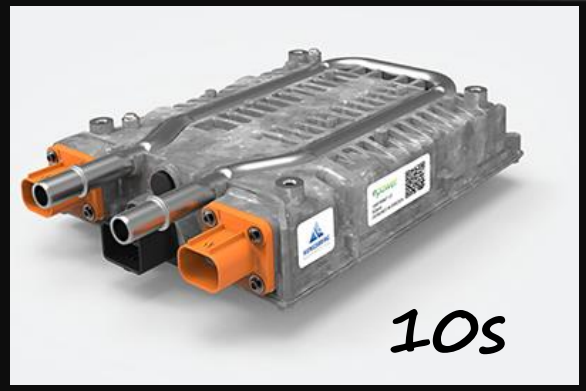
Senior leader consultant.
Led large business transformations, and product, process and team development for more than 25 years.
Led a Lean start-up from start to its growth take-off.
Works embedded, provides transformational and servant leadership to business executives and teams.

Lean and Agile Development

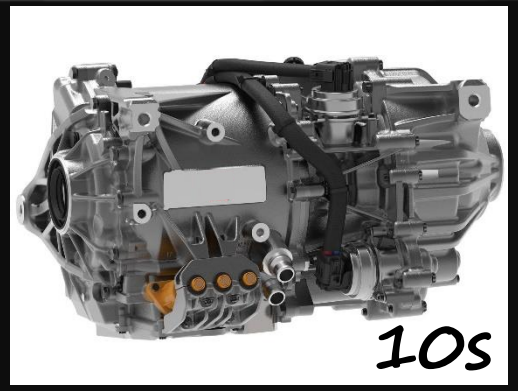


Internal Combustion Engines

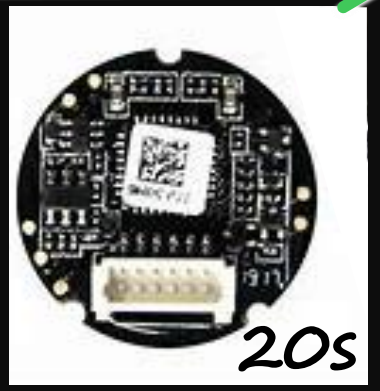
Gear shifters



Power Electronics



Electric Motors



IoT Sensor

AUFERO AB | clu.christerlundh@gmail.com| +46 72 016 80 16