Guest lecture · Hochschule Reutlingen

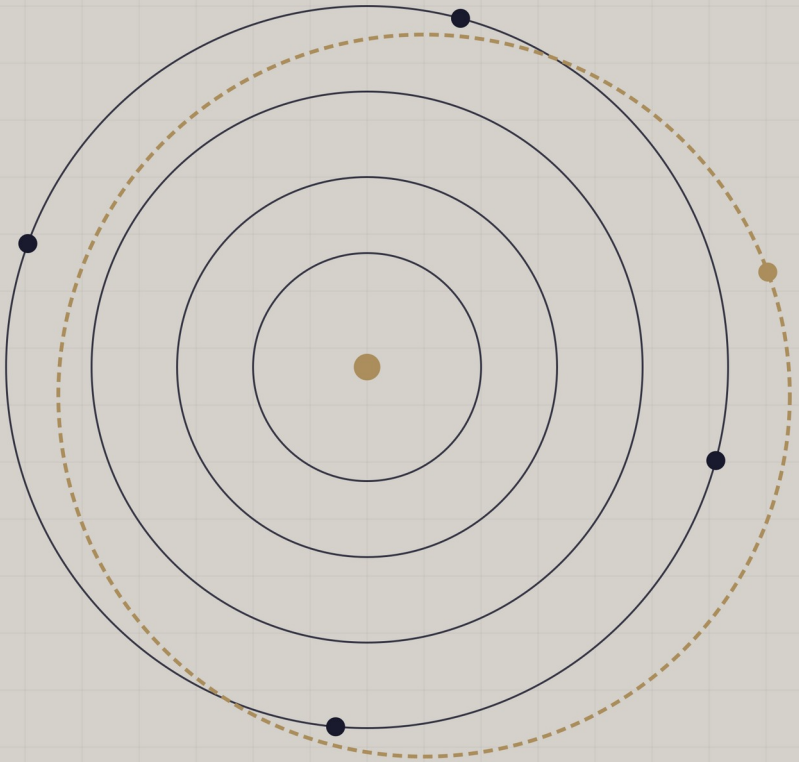
Leading Agile *in Industry*

What you will actually find behind the framework

Kyle Hauslaib

Solution Train Engineer · Robert Bosch GmbH

Author, Leading Agile When No One Agrees



Reutlingen · 2026

Who is talking to you

Mechatronics Engineer (UCT, South Africa).

Eight years at Bosch: from software engineer to running a Solution Train of four ARTs (≈300 people).

Leads the company-wide RTE/STE Guild.

Builds Home Assistant, IoT and LoRa things on weekends. Engineering does not stop at the office door.

What I do at scale

4	Agile Release Trains coordinated
≈300	engineers across countries
9 wk	planning cadence (Program Increment)
1	book on what they don't teach you

Why I am here

Within six months of graduating, you will be on a SAFe team.

Almost every large German engineering employer (Bosch, Daimler, Porsche, ZF, SAP, Siemens) runs scaled agile in some form. You will not get to opt out.

01

The vocabulary

Enough to understand any agile conversation you walk into.

02

The gap nobody mentions

Why agile on paper and agile in practice are different things.

03

Where leadership lives

What good looks like when there is no perfect process.

Agile, in 90 seconds

Started in software, now everywhere. Strip away the jargon and there are four ideas.

1. ## Build in short cycles.

   Two-week sprints instead of two-year plans. Each cycle ends with something usable.

2. ## Show the work.

   At the end of every cycle, demo what is real, not slides about what might be done.

3. ## Decide closer to the work.

   The people doing the engineering decide how to do it. Managers set direction, not method.

4. ## Adapt on a steady drumbeat.

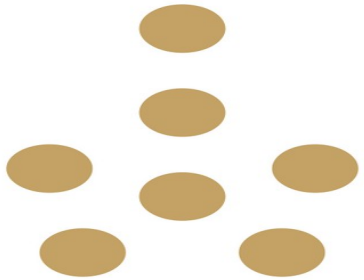
   Plans change. Built into the rhythm: review, retrospective, replan. Repeat.

Five words you will hear in your first week

Sprint	A fixed time-box, usually two weeks, in which a team commits to a set of work.
Backlog	The ordered list of work (features, bugs, refactors) waiting to be picked up.
Stand-up	A 15-minute daily sync. What did I do, what will I do, what is blocking me.
Retrospective	End of every sprint: what worked, what did not, what we will change.
Definition of Done	The team's shared standard for when something counts as finished. Not 'almost'.

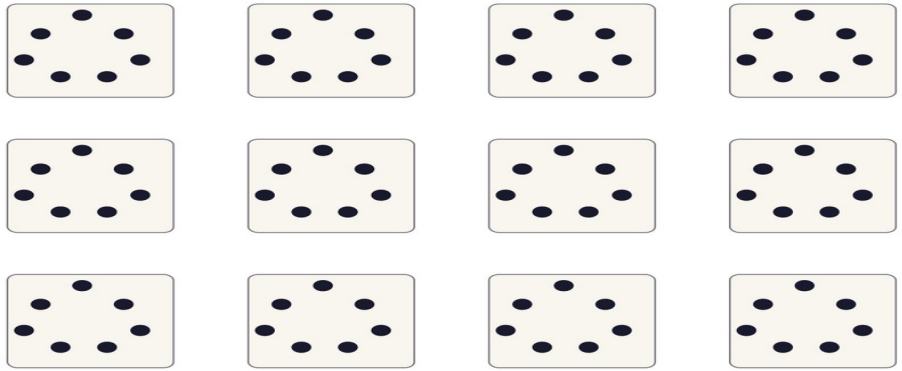
Most of what you've heard about agile assumes a small team.

7 people, one team
What Scrum was designed for



vs

~120 people, 12 teams
An Agile Release Train



An engine control unit needs hardware, firmware, calibration, validation and safety. That is hundreds of people who must ship the same product at the same time. Scrum was not designed for that. SAFe, the Scaled Agile Framework, was.

SAFe, stripped to the essentials

TEAM

Each team runs Scrum or Kanban

5–9 people, two-week sprints, their own backlog. Same as before.

TRAIN

10–12 teams form an Agile Release Train

≈100–125 people working on one product. They plan together every 9 weeks. This event is called PI Planning. Two days. Everyone in one room.

SOLUTION

Multiple trains form a Solution Train

When the product is big enough (a vehicle, a platform, an aircraft), trains synchronise across hundreds of engineers. This is the level I work at.

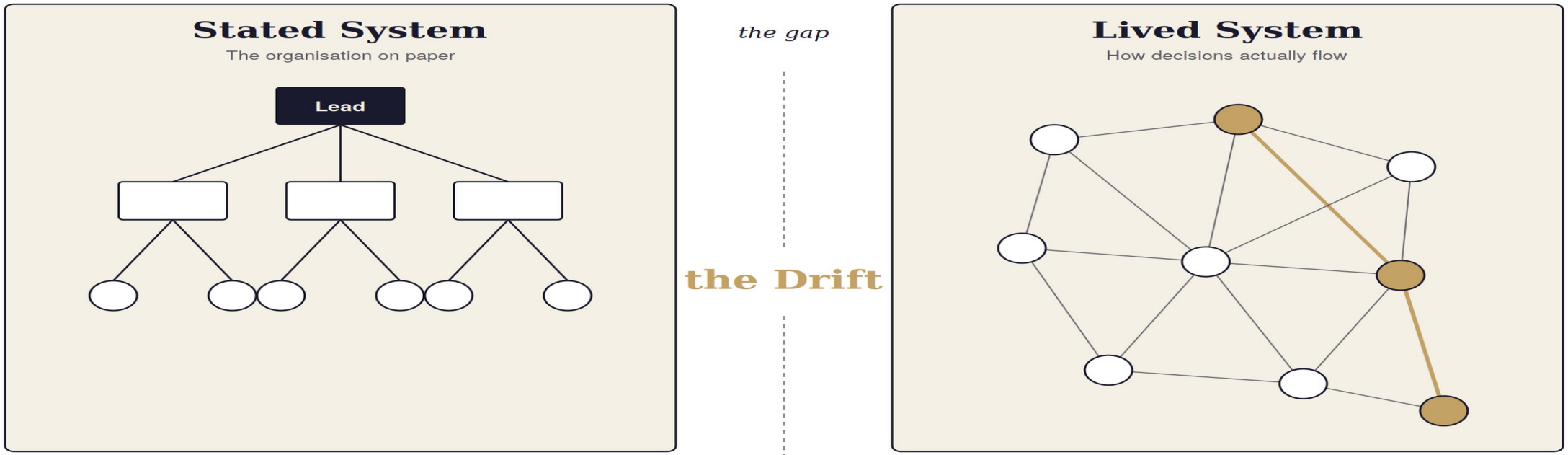
You will sit at the team level. But your work depends on how the train and solution layers behave.

Part Two

So far, the textbook.

Now the part nobody puts on the slides.

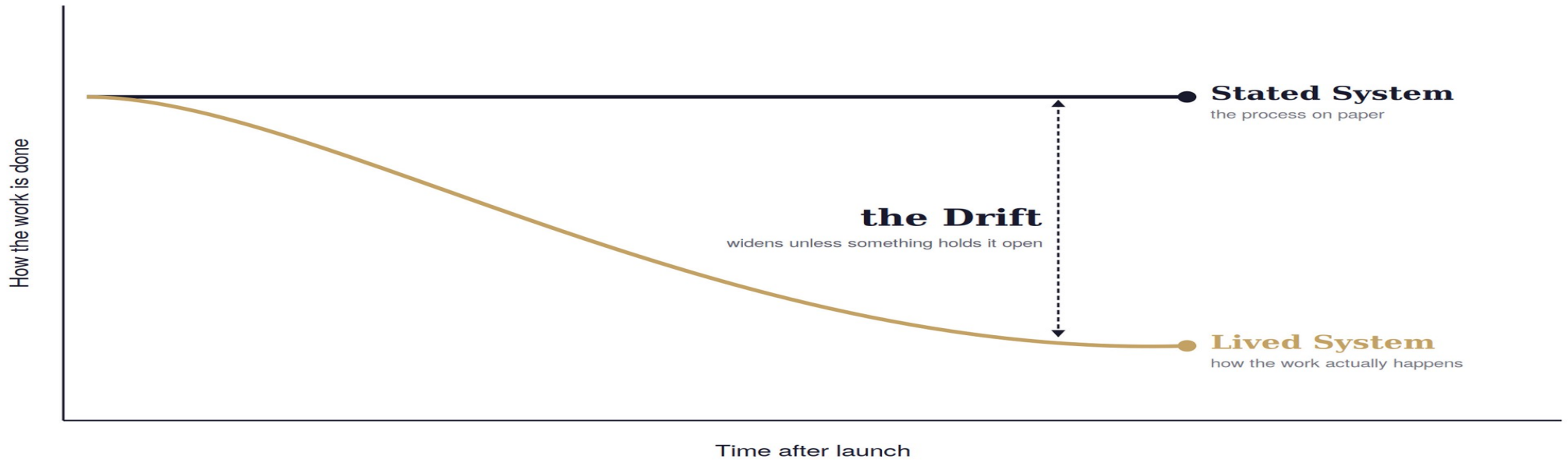
Every organisation runs two systems at once



They are never identical. The interesting question is how far apart they are, and whether anyone is allowed to say so.

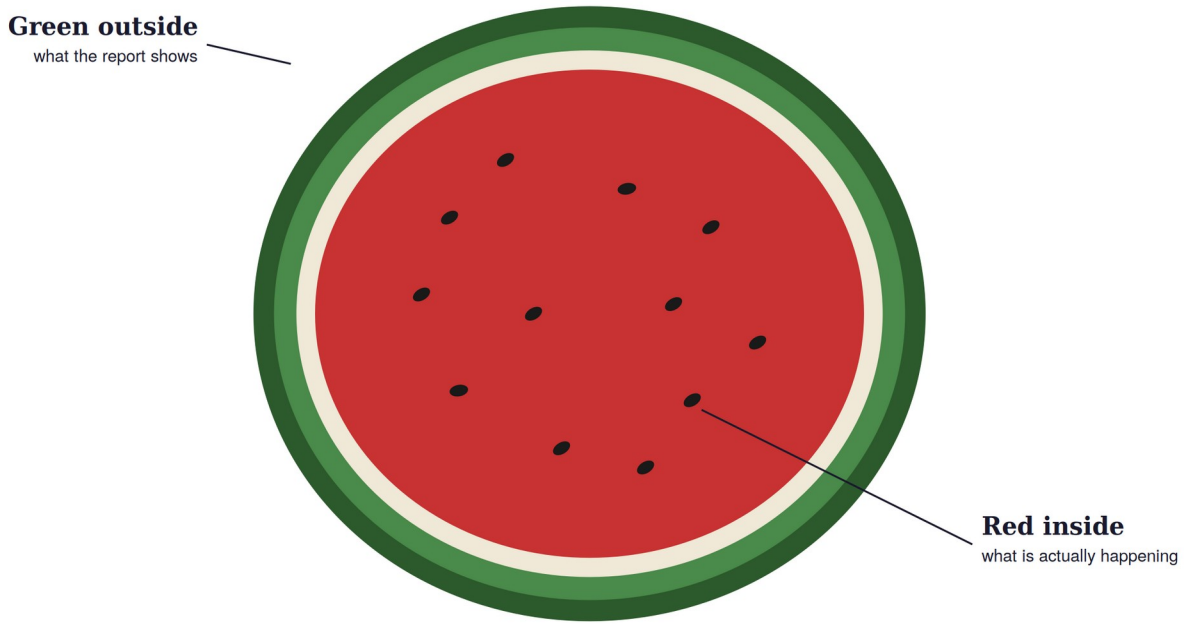
The gap, over time

the Drift



The Stated System stays where the org chart left it. The Lived System keeps adapting to reality. Without active work, the two pull apart.

Watermelon reporting

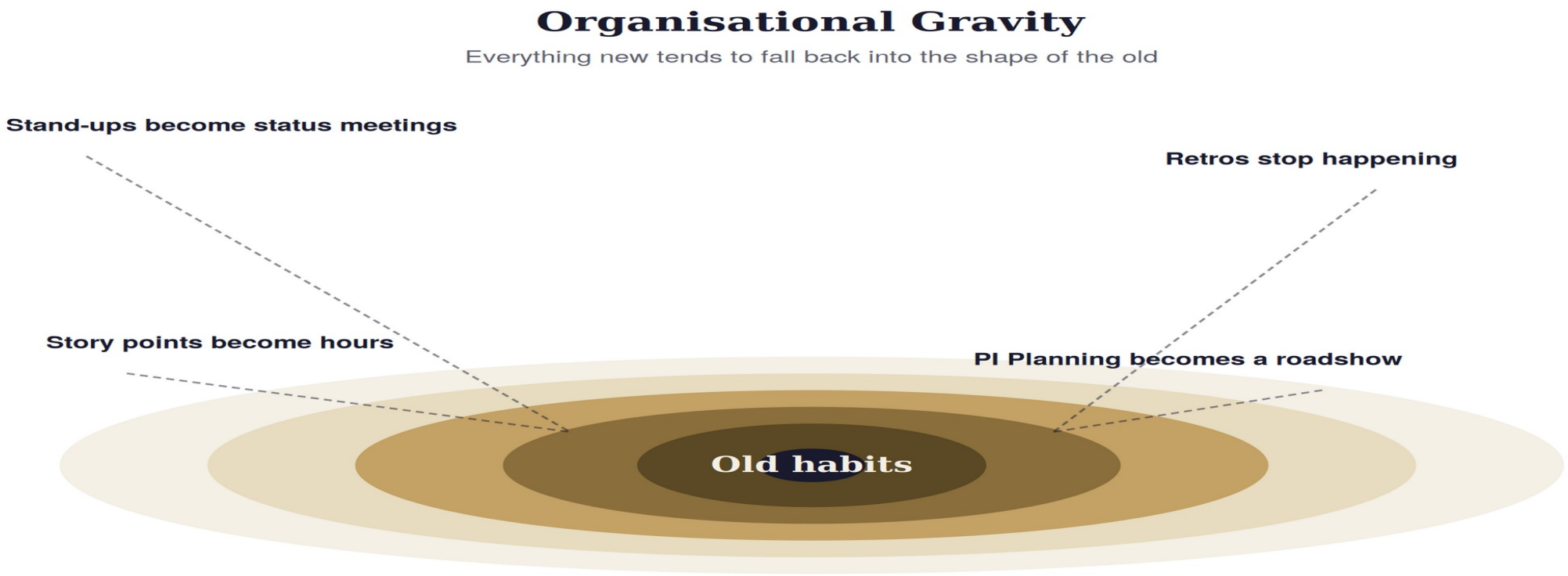


Every status report is green.

Until the launch.

- Teams know things are slipping.
- Middle managers smooth the message upward.
- Senior leadership sees only the smoothed message.
- Surprise arrives close to the deadline.

In my book I call this the failure mode of every reporting culture that punishes red.



Old habits have mass. New practices need active force to stay in orbit. Stop pushing, and they fall.

Leading agile is holding the gap open

Not pretending it isn't there. Not closing it with paperwork.

1 Name what is actually happening.

If the team is doing waterfall in sprint clothing, say so. Out loud. Without blame.

2 Make red safe.

If reporting bad news is punished, you will only ever hear good news. That is how launches surprise people.

3 Defend the cadence.

Retros, demos and PI Planning are the first things sacrificed when things get busy. They are the last things you should give up.

4 Translate, both directions.

Engineers want to ship working things. Management wants predictability. Your job is to keep both honest.

Three habits that will set you apart

01 Watch the Lived System.

In your first weeks, do not study the process documents. Study who actually talks to whom, who decides what, and where information stops moving. That is the system you will work in.

02 Be the one who tells the truth in retros.

Most people stay quiet. The team that says what isn't working (calmly, specifically, without blame) improves faster than every team around it. Be on that team. Better still, start it.

03 Learn to write a clear status.

Green / yellow / red, one sentence each, on time, every time. People who can do this end up in rooms others never reach. It is the cheapest leadership skill to acquire and the most reliably noticed.

Take this with you

Two ideas worth keeping

Every org runs two systems

The Stated System is the diagram. The Lived System is reality. Your job is to know the gap and refuse to pretend it doesn't exist.

Leadership is the force that resists the Drift

Not the org chart, not the framework, not the certification. The work of staying honest when honesty is uncomfortable.

Read more

Leading Agile When No One Agrees

Kyle Hauslaib · 2026

kylehauslaib.com

LinkedIn · /in/kylehauslaib

Q & A

Hardest question wins.