Lessons Learnt from our BI Journey:

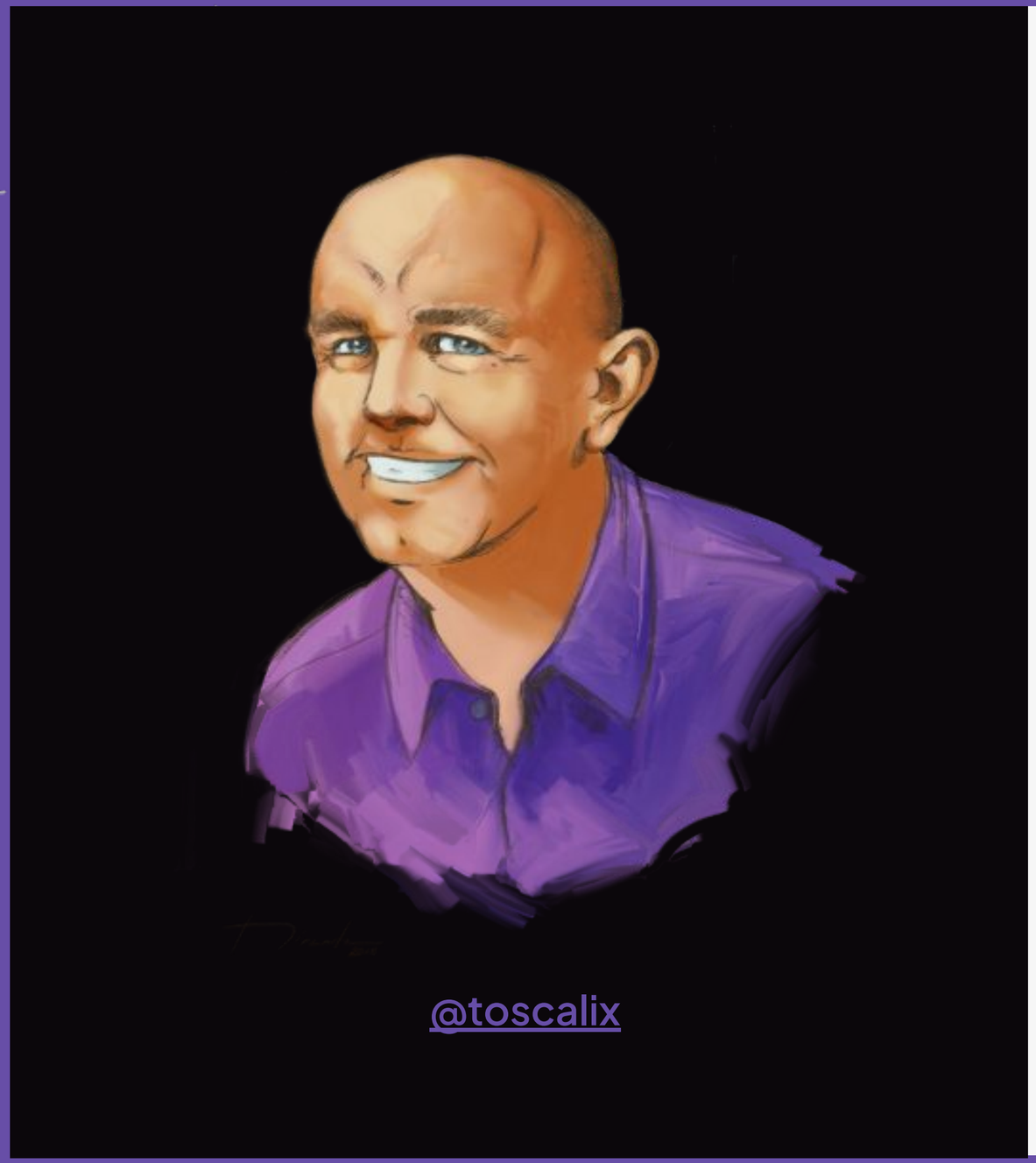
From Open Source to SDV products development

EF SDV - Shifting Gears Webinar
2026-06-03

Agustín Benito Bethencourt
&
Luis Cañas Díaz

Table of contents

1.- Speakers



@toscalix

Agustín Benito Bethencourt (toscalix)

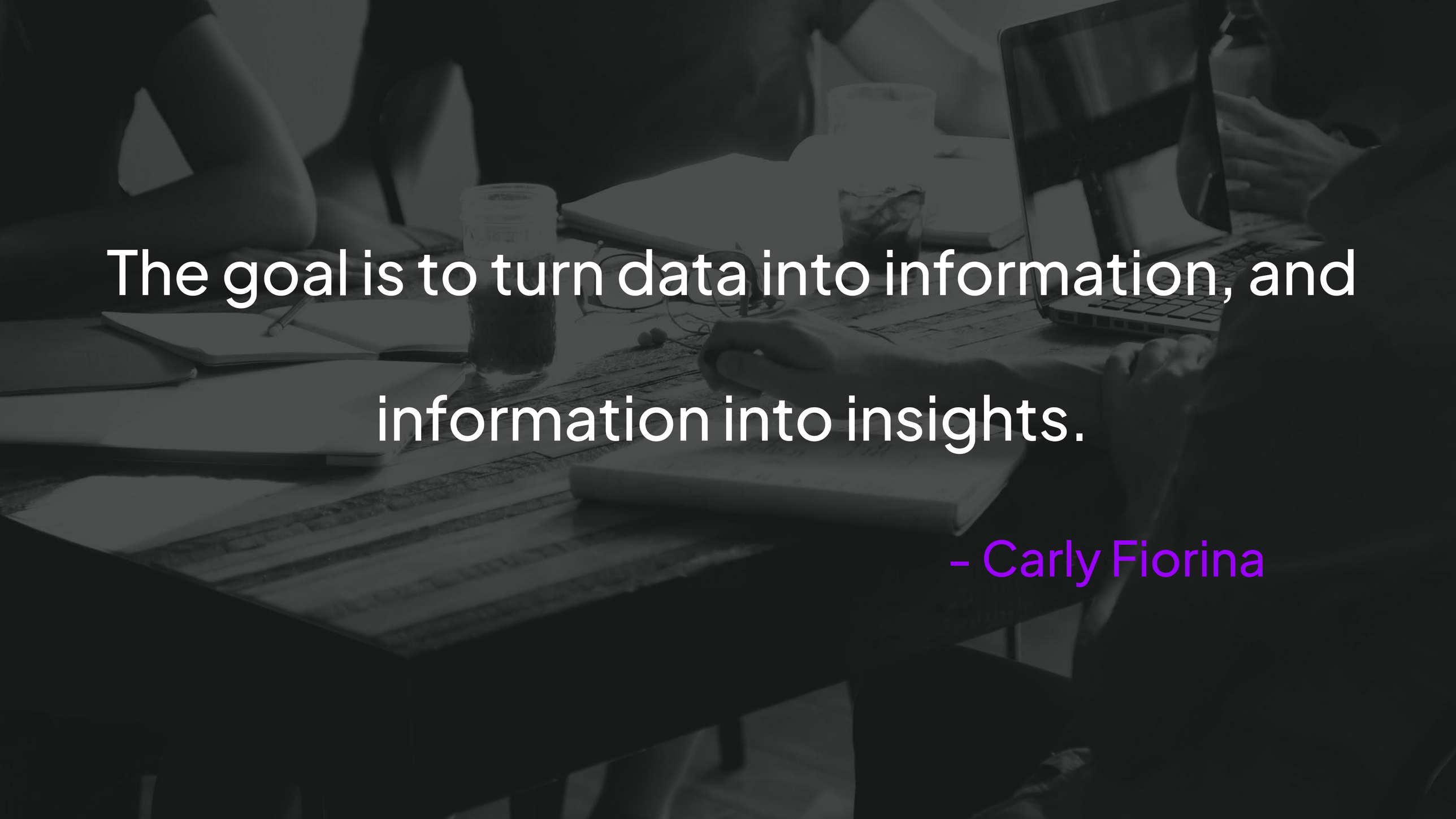
- FLOSS, Continuous Delivery, BI, agility and remote work advocate
- Independent consultant helping companies in two ways:
  - Increasing their organizational performance by becoming good open source citizens
  - Applying advanced data analytics to production environments to increase delivery performance through a service offering called Delivery Performance Analytics
- More about Agustín:
  - Background: entrepreneur, SUSE, Linaro, Codethink, MBition (Mercedes Benz), Eclipse Foundation, ...
  - Open Source: KDE eV Member, SwH Ambassador, SPDX Cryptography Group, KDE España, STF
  - Blog - About - Talks - Contact

@canasdiaz

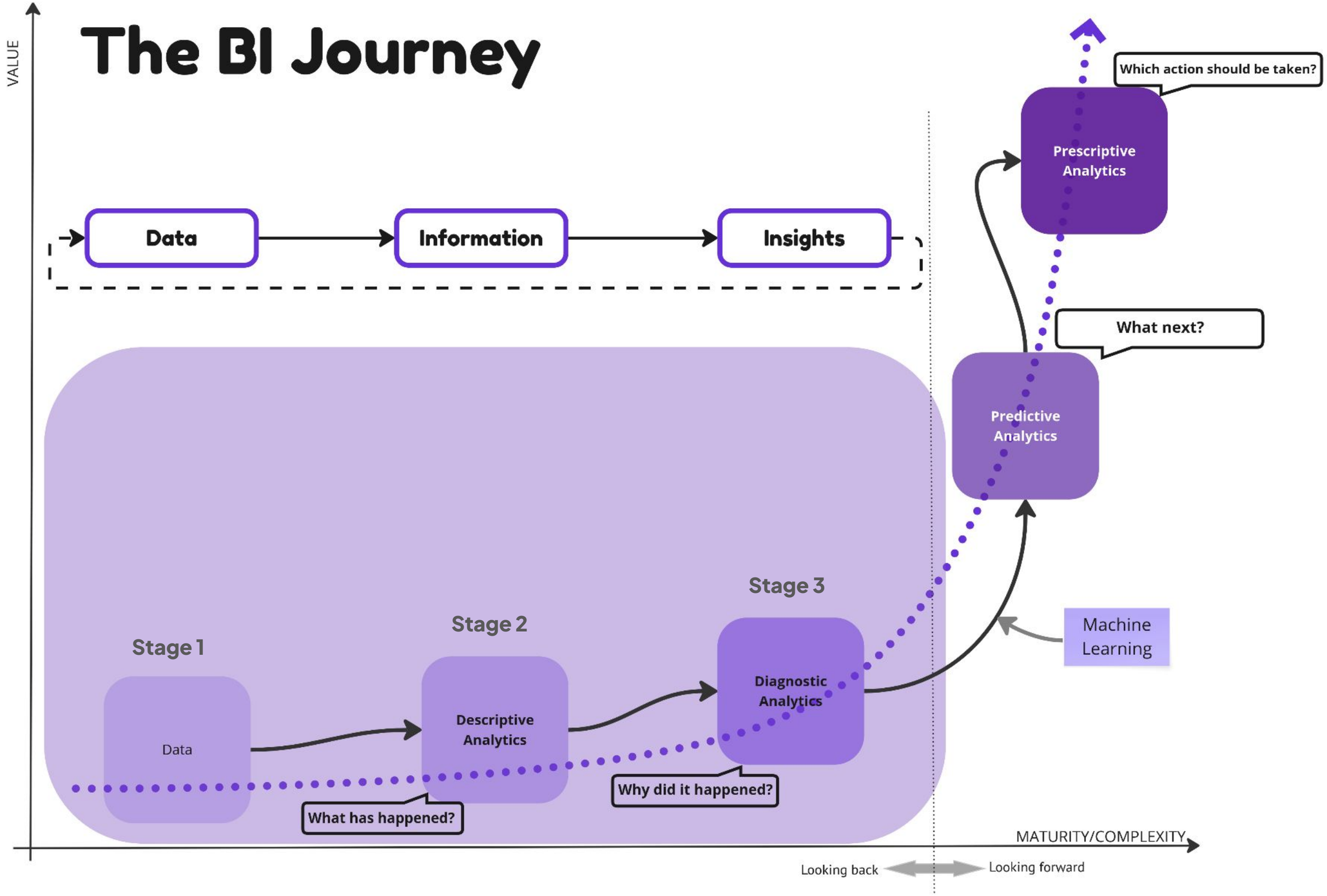
Luis Cañas Díaz (@canasdiaz)

- FLOSS, Open Source Data Insights
- Independent consultant helping companies in two ways:
  - Producing insights from open source ecosystems to evaluate the health of the projects
  - Applying advanced data analytics to production environments to increase delivery performance through a service offering called Delivery Performance Analytics
- Background:
  - Co-Founder of Bitergia, a company specialized in Software Data Analytics in the open source community
  - Blog - About - Contact

2.- The Business Intelligence Journey


The goal is to turn data into information, and
information into insights.
- Carly Fiorina

The BI Journey



Overall Goals

Promote data-supported…

1. Decision-making processes
2. Continuous improvement processes

Foundation for applying ML&AI to predict behaviour of the software production system

Stage 1: data

Goals:

- From raw data to consumable processed data
- Data collection …
  - High-quality data
  - Data collection must be reproducible and auditable
- Datahub creation: consumable processed data at scale

Stage 1: data

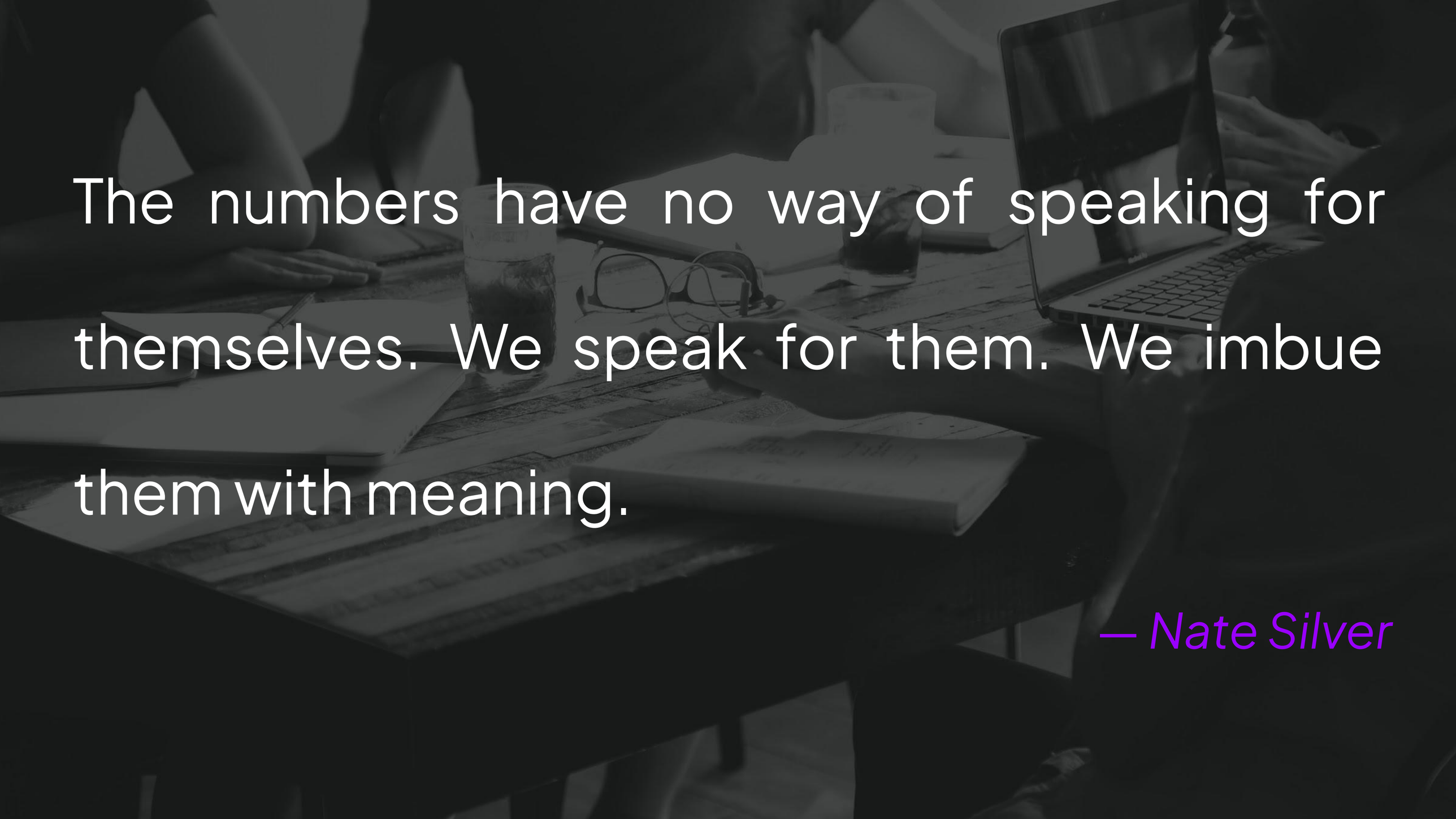
Data sources:

- People: interviews, surveys, reports, workshops…
- Documentation: wiki pages, architecture descriptions, decisions records, diagrams, policies, functional and org. charts, roles descriptions….
- Tools: SCM, code reviews, integration, testing, deployment, communication channels, orchestration, ticketing, reqs…

Stage 1: data

Data workflow:

- Input: raw data from different data types, from different tools.
- Intermediate step: data lake creation.
- Output: a data hub with processed and curated data
  - Purpose: data integration, sharing, interoperability, operational delivery…consumption by users and other tools.
    - Users: business users, dev. teams, tech leads, DevEx, consultants, etc.
    - Other tools: visualization, AI models…



The numbers have no way of speaking for themselves. We speak for them. We imbue them with meaning.

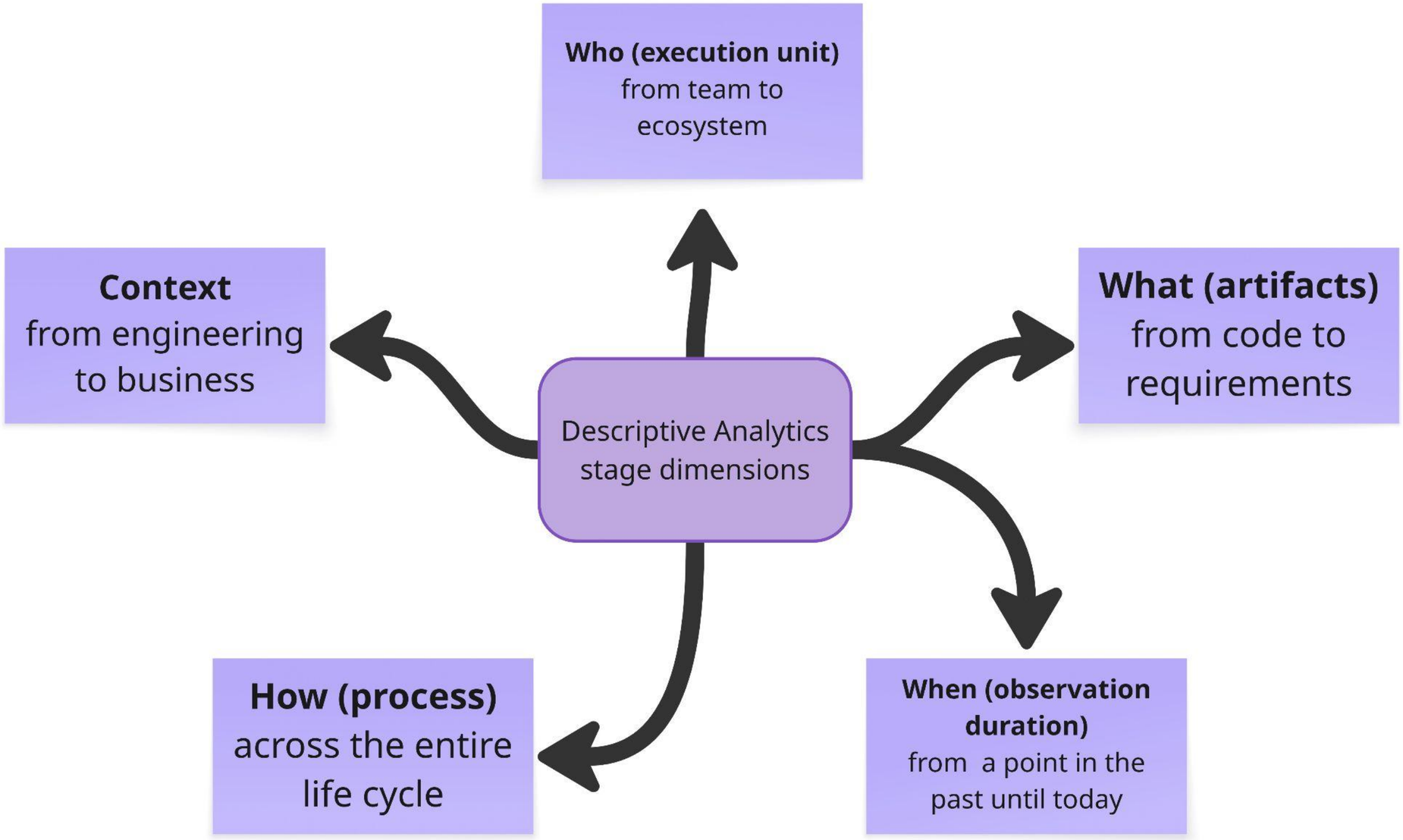
— *Nate Silver*

Stage 2: descriptive analytics

Goals:

- Describe end-to-end the software production environment through data
- From consumable data at scale to information

Stage 2: descriptive analytics



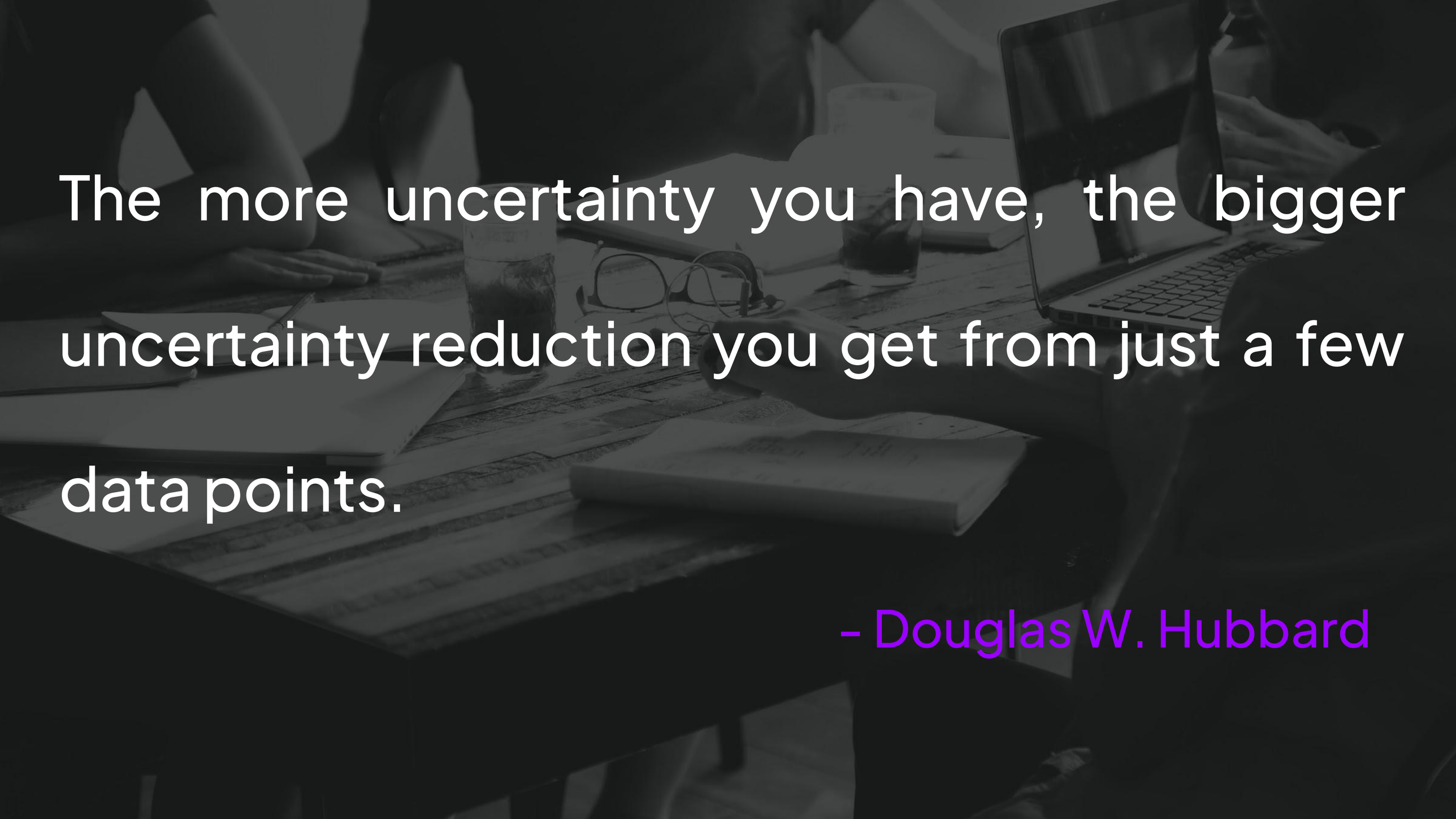
Stage 2: descriptive analytics

Extensive and horizontal approach vs *intensive and vertical*

Stage 2: descriptive analytics

The output of this stage:

- Data-supported end-to-end view of the SW production system and its environment through
  - Simple metrics and dashboards
- Reports highlighting the most relevant aspects
  - Targeting execution units up to business
  - Inconsistencies between information and experience
    - It should generate questions that require further investigations: why?
    - Work as context and guidance for the diagnostic analytics stage.


The more uncertainty you have, the bigger uncertainty reduction you get from just a few data points.
- Douglas W. Hubbard

Stage 3: diagnostic analytics

Goals:

- Get to the root causes (constraints) that limit the software production system performance
- From information to insights

Stage 3: diagnostic analytics

- Identify and characterize potential delivery and organizational performance sinks:
  - Identify and select the highest impact symptoms to investigate
  - Analyse them and get to their root causes.
  - Build a common understanding around those root causes (constraints)
- Identify and propose potential remedies (exploits) as well as best practices to prevent them.

Stage 3: diagnostic analytics

Priorization: How do we select the symptoms to work on?

- Cluster and prioritise the questions (whys) based on overall expected impact for:
  1. Organizational/project performance
  2. SW delivery performance
  3. Well-being

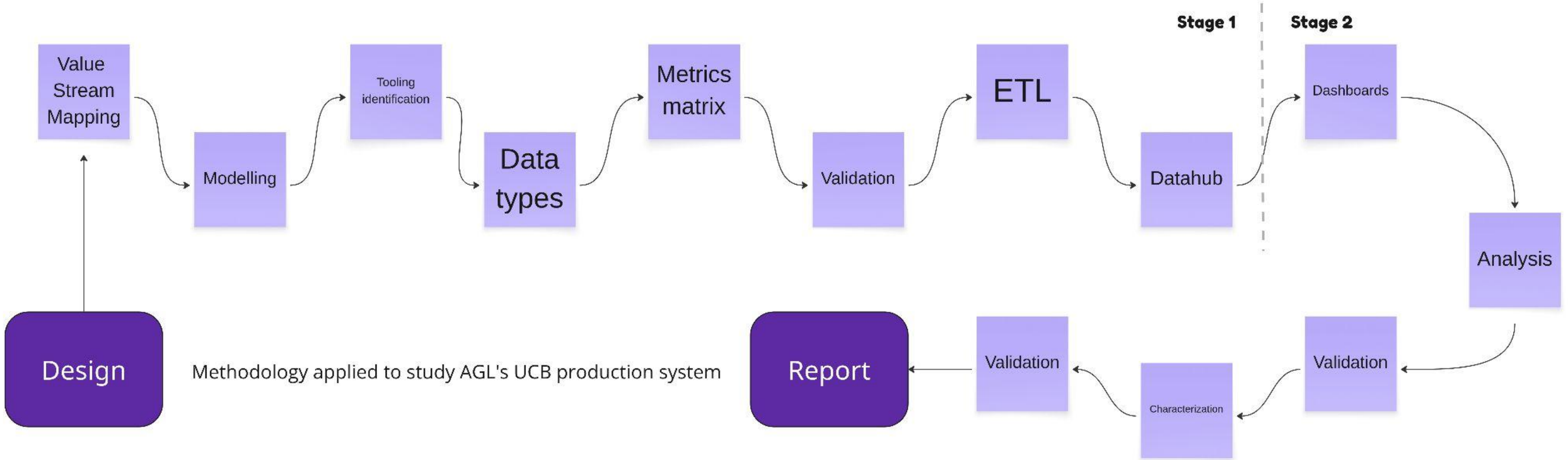
Stage 3: diagnostic analytics

Outputs of this stage:

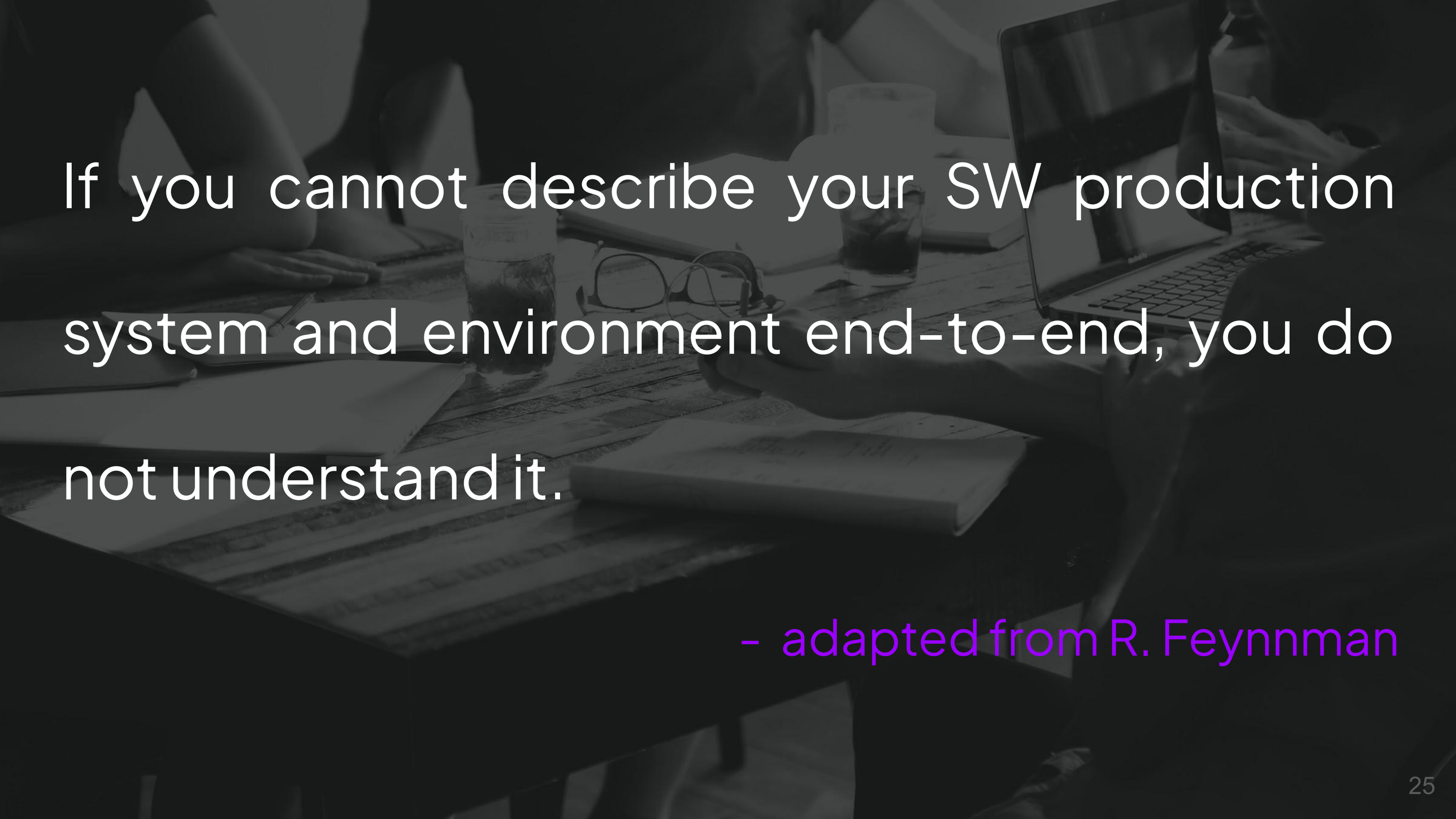
- Workshops to discuss the early results with the experts on the ground about root causes (constraints)
- Reports including potential SW production performance gains
  - Those potential gains become inputs for continuous improvements processes

3.- The methodology

Process overview



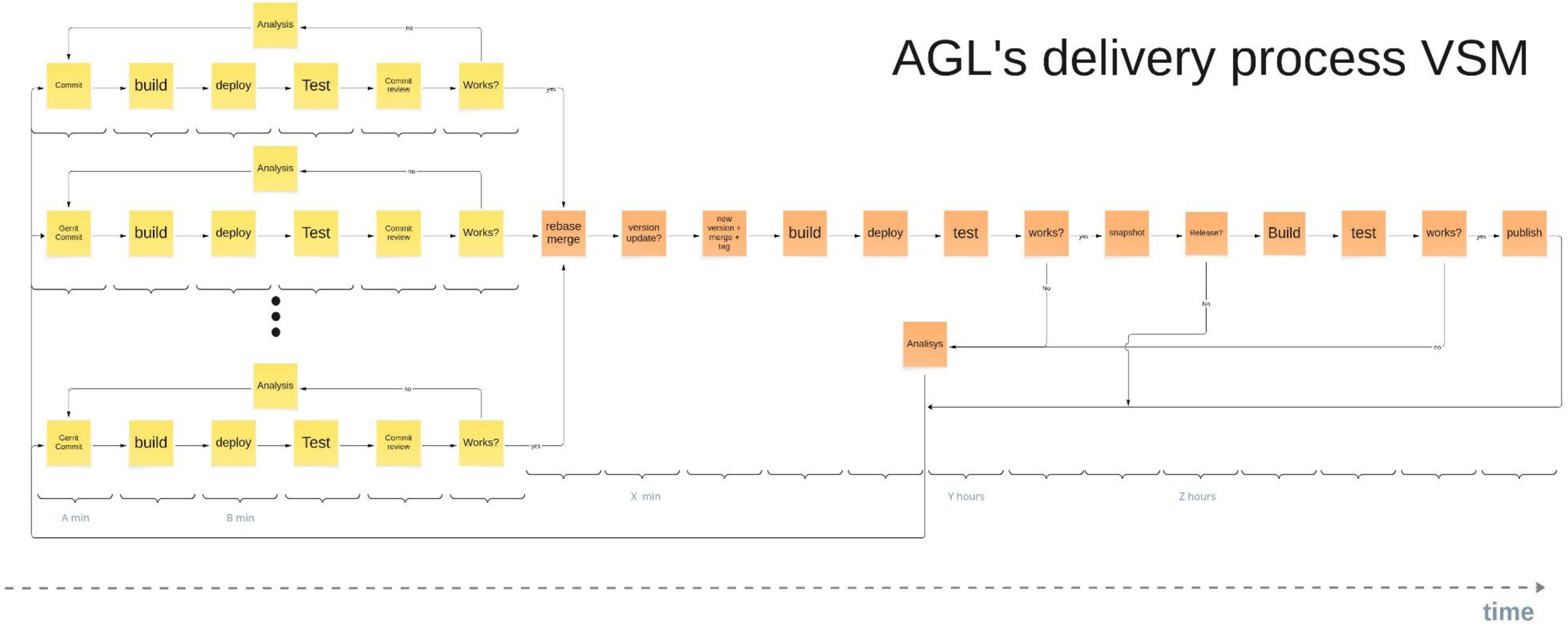
Process followed for the BI analysis of AGL's UCB (stage 1 and 2 only)



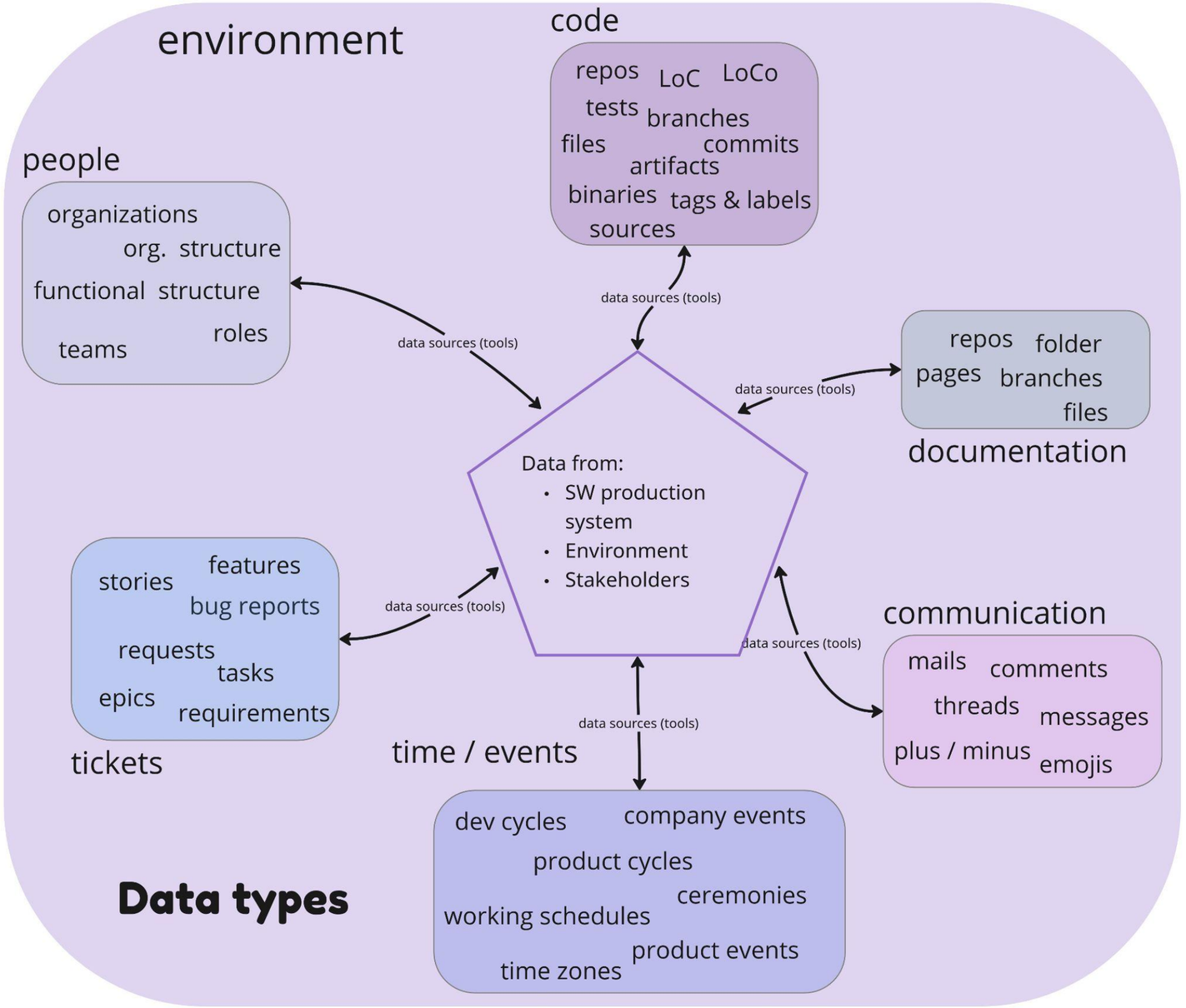
If you cannot describe your SW production system and environment end-to-end, you do not understand it.

- adapted from R. Feynnman

VSM

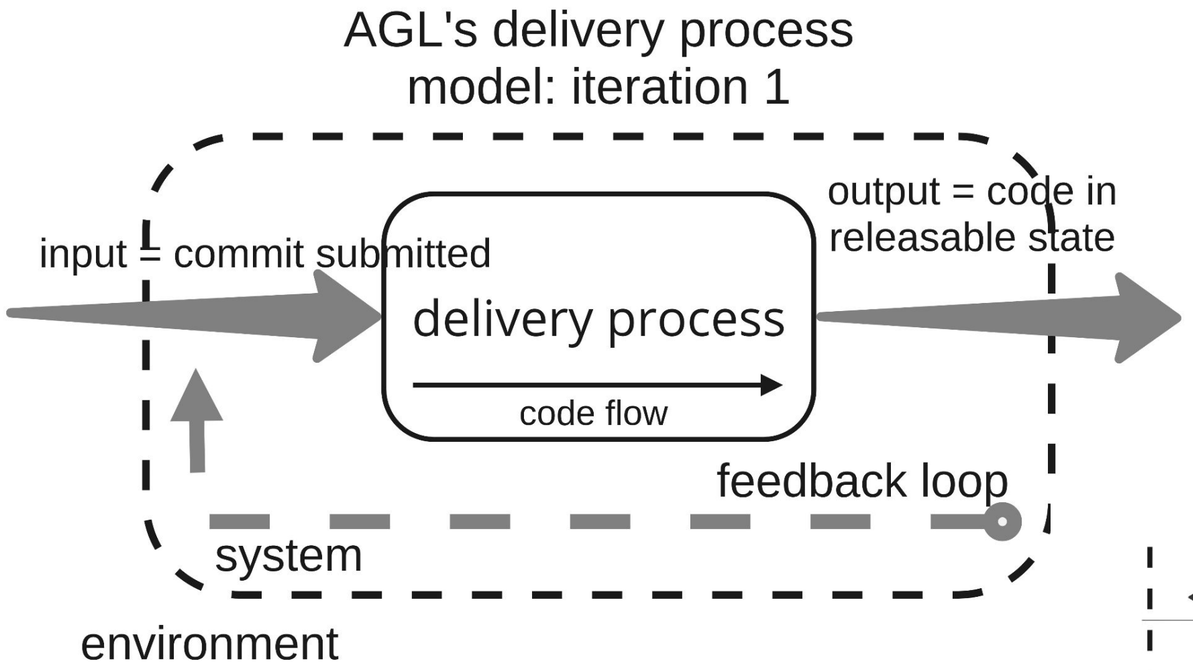

AGL's delivery process VSM
Analysis
no
Commit
build
deploy
Test
Commit review
Works?
yes
Gerrit Commit
rebase merge
version update?
new version + merge + tag
test
works?
snapshot
Release?
Build
publish
Analisys
A min
B min
X min
Y hours
Z hours
time

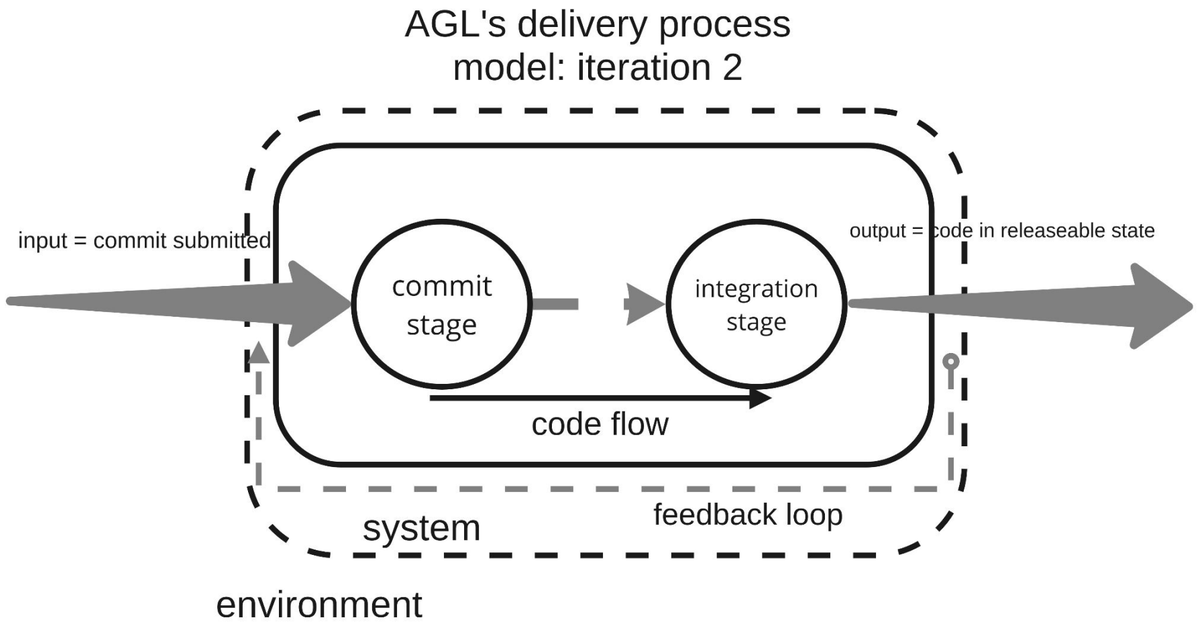
Data types and tooling

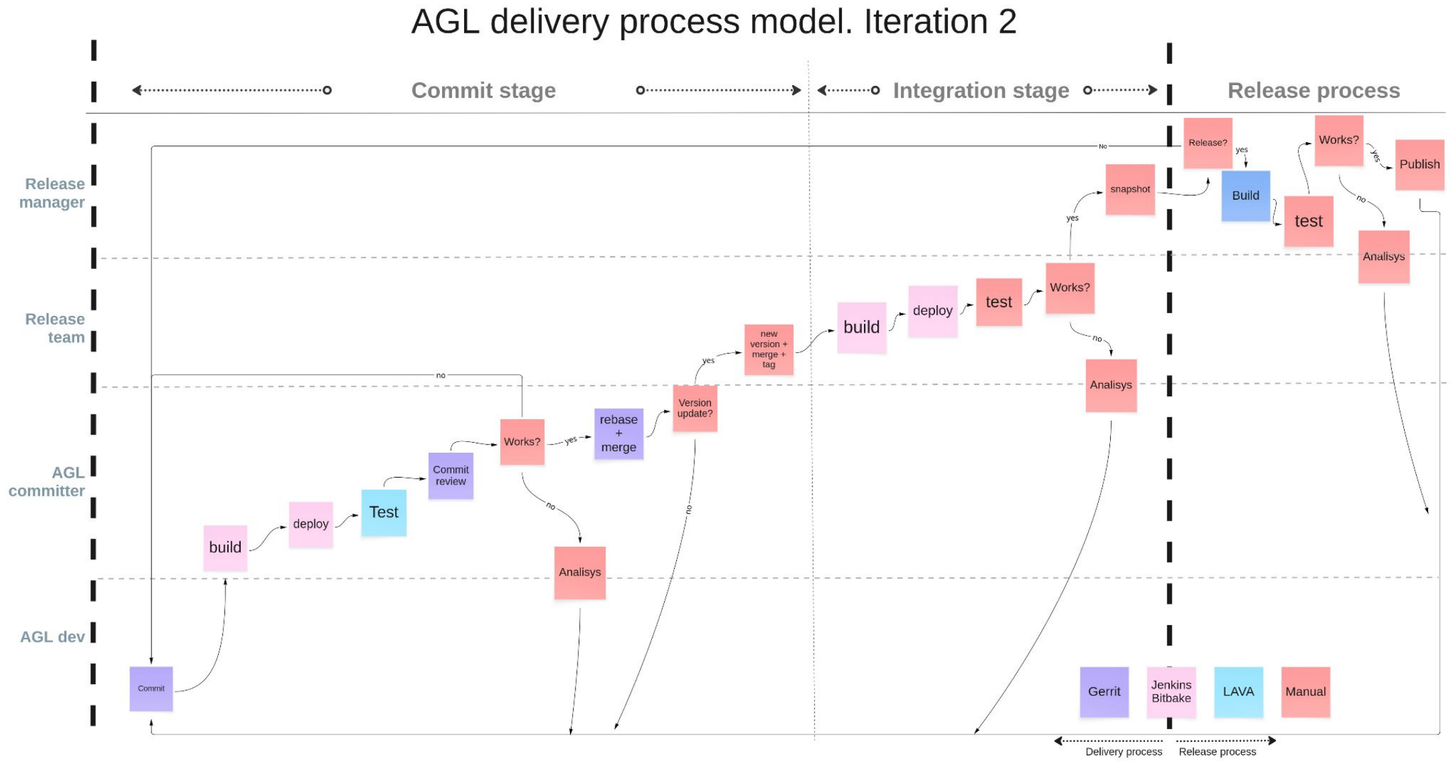


- Requirements management tools
- Ticketing systems
- SCMs and all-in-one solutions
- Building and Integrations
- Code review management
- Orchestration tools
- Artifacts management
- Software configuration tools
- HW provisioning and management systems
- Testing frameworks and tools
- Communication tools
- Wikies and CMSs
- SW Composition Analysis (SCA) tools
- etc.

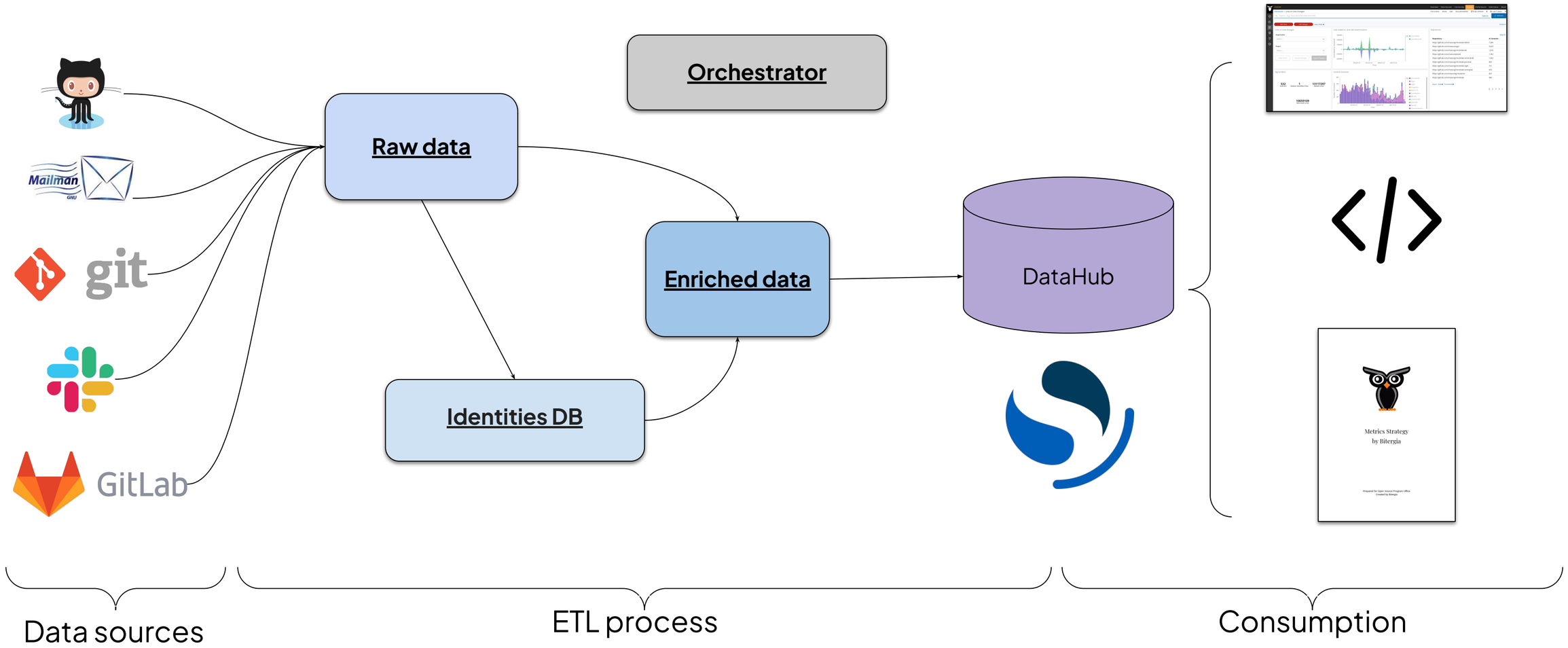
Modelling


AGL's delivery process
model: iteration 1
input = commit submitted
delivery process
code flow
output = code in
releasable state
feedback loop
system
environment


AGL's delivery process
model: iteration 2
input = commit submitted
commit
stage
integration
stage
output = code in releaseable state
code flow
feedback loop
system
environment

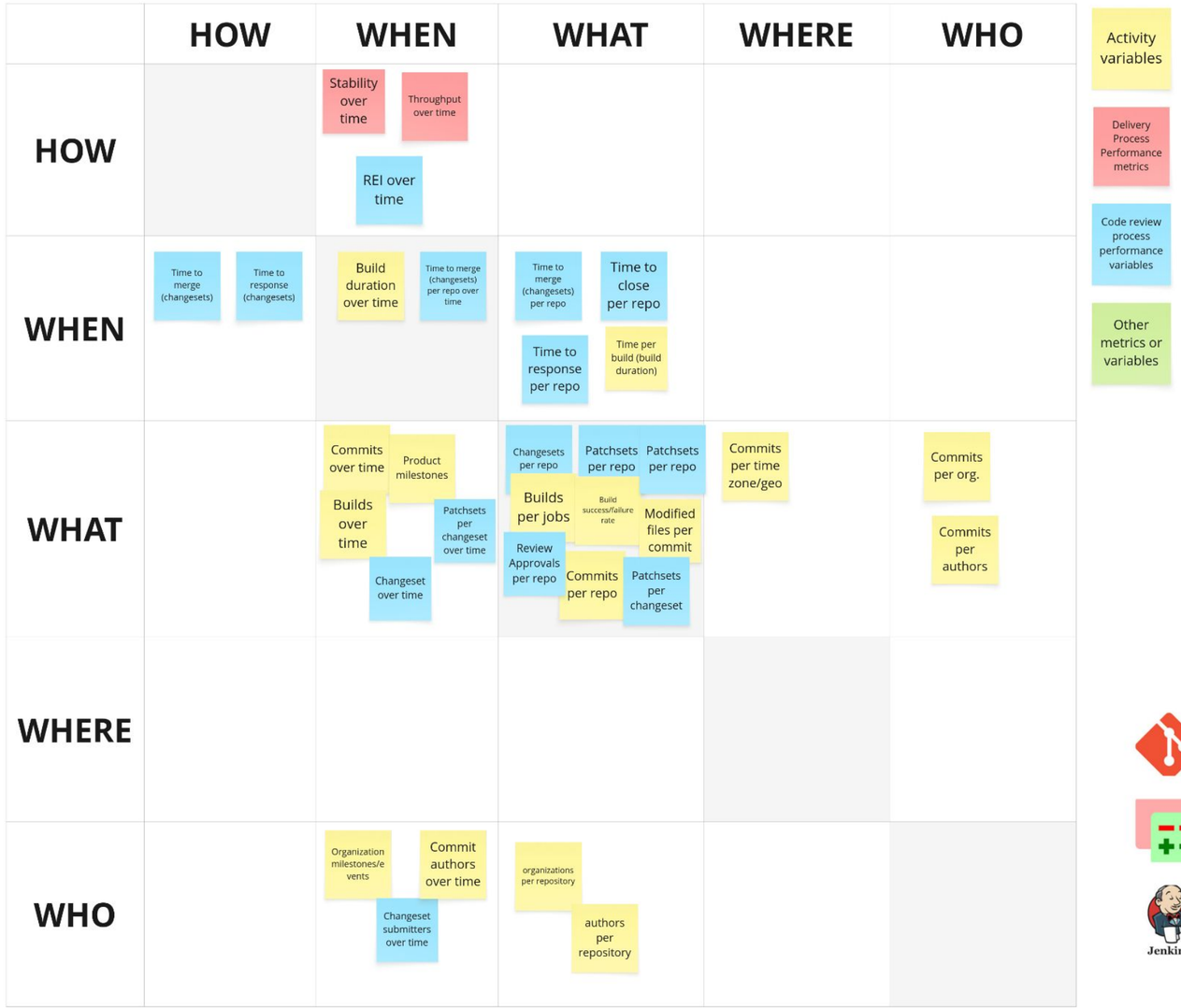

AGL delivery process model. Iteration 2
Commit stage
Integration stage
Release process
Release manager
Release team
AGL committer
AGL dev
Commit
build
deploy
Test
Commit review
Works?
rebase + merge
Version update?
new version + merge + tag
Analisys
build
deploy
test
Works?
snapshot
Analisys
Release?
Build
test
Works?
Publish
Analisys
Gerrit
Jenkins Bitbake
LAVA
Manual
Delivery process
Release process

Bitergia Analytics: based on GrimoireLab



Metrics matrix

Metrics/variables Characterization matrix



REI

Git commit logs

Gerrit changesets + pathsets

Jenkins jobs + builds

Visualizations

Visualization matrix: iteration 1

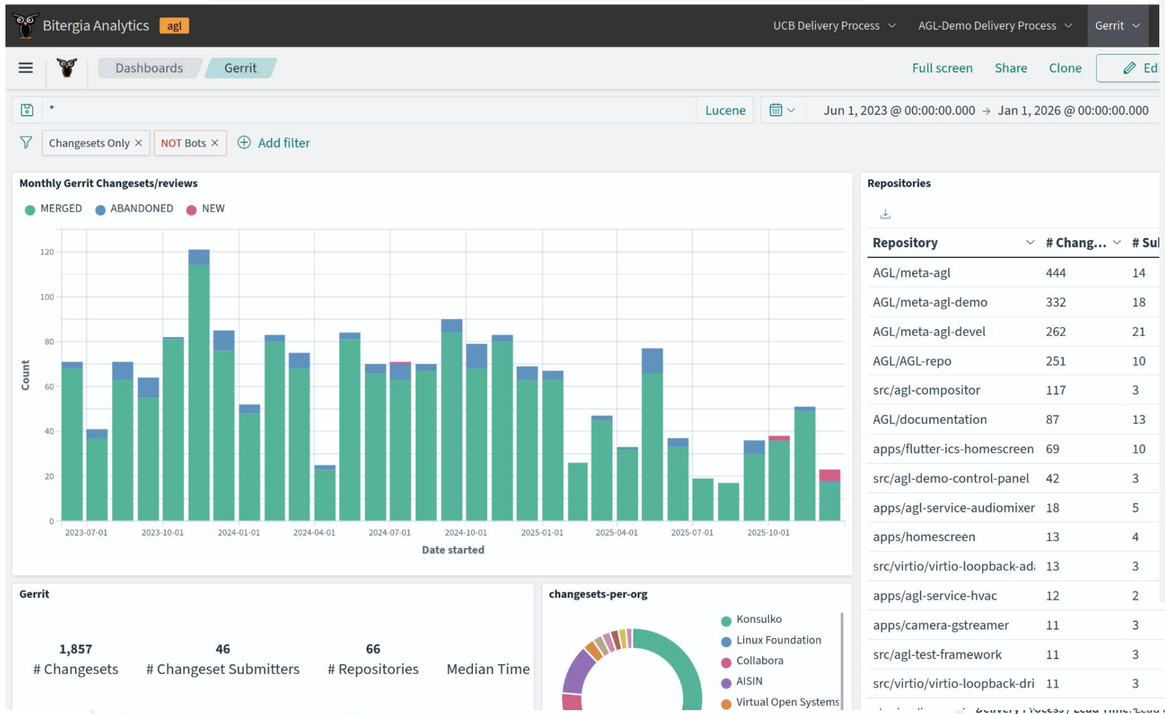
Product / Pipeline	Goal	Metric	Measurement	Delivery Process	Release Process	Maintenance Process
meta-AGL	Evaluate Delivery Process Performance	Throughput	Lead time	✔		
			Time interval	✔		
		Stability	Failure Recovery Time	✔		
			Change Failure rate	✔		

Visualizations

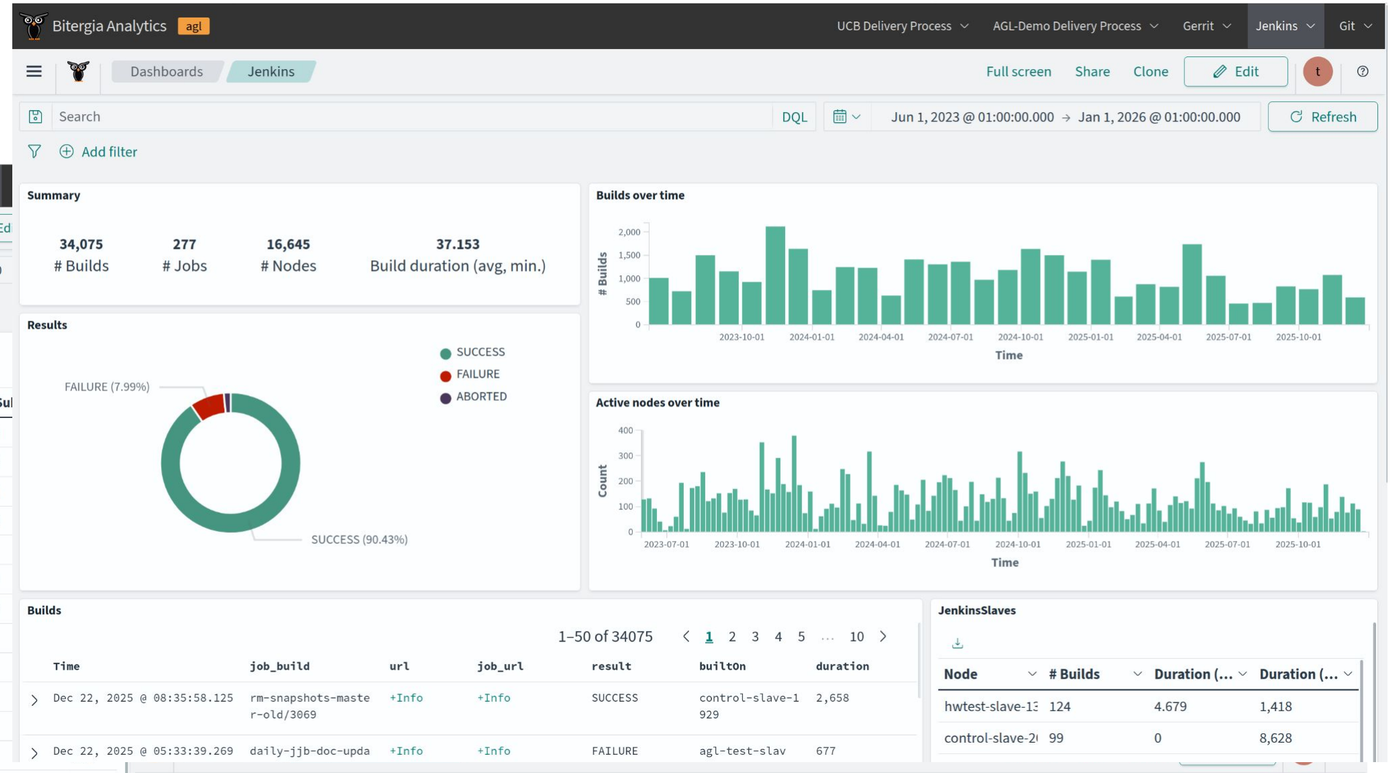
Visualization matrix: iteration 2

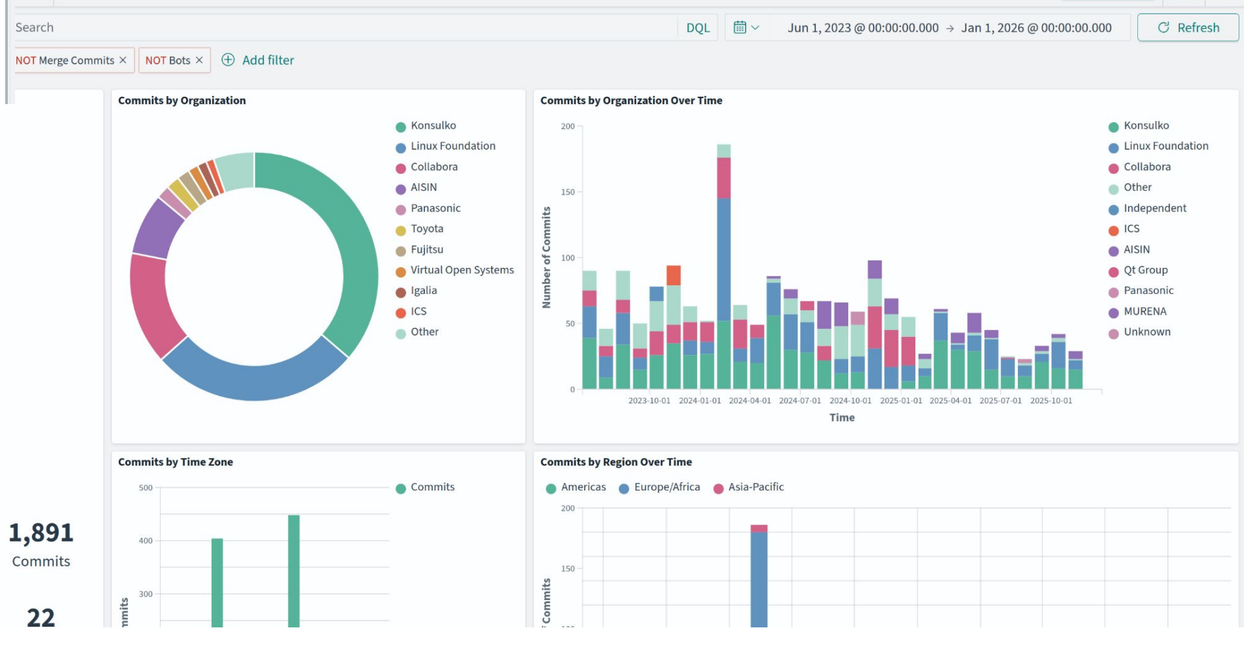
Product / Pipeline	Goal	Metric	Variables	Delivery Process Stages		Release Process	Maintenance Process
				Commit	Integration		
meta-agl	Evaluate Delivery Process Performance	Throughput	Lead time	✔	✔		
			Time interval	✔	✔		
		Stability	Failure Recovery Time	✘	✔		
			Change Failure rate	✘	✔		

Dashboards


Bitergia Analytics
agl
UCB Delivery Process
AGL-Demo Delivery Process
Gerrit
Dashboards
Gerrit
Full screen
Share
Clone
Lucene
Jun 1, 2023 @ 00:00:00.000 → Jan 1, 2026 @ 00:00:00.000
Changesets Only
NOT Bots
Add filter
Monthly Gerrit Changesets/reviews
MERGED
ABANDONED
NEW
Count
Date started
Repositories
Repository
Chang...
Sul
AGL/meta-agl 444 14
AGL/meta-agl-demo 332 18
AGL/meta-agl-devel 262 21
AGL/AGL-repo 251 10
src/agl-compositor 117 3
AGL/documentation 87 13
apps/flutter-ics-homescreen 69 10
src/agl-demo-control-panel 42 3
apps/agl-service-audiomixer 18 5
apps/homescreen 13 4
src/virtio/virtio-loopback-ad 13 3
apps/agl-service-hvac 12 2
apps/camera-gstreamer 11 3 2.63
src/agl-test-framework 11 3 3.16
src/virtio/virtio-loopback-dri 11 3 0.71
Gerrit
1,857
Changesets
46
Changeset Submitters
66
Repositories
Median Time
changesets-per-org
Konsulko
Linux Foundation
Collabora
AISIN
Virtual Open Systems

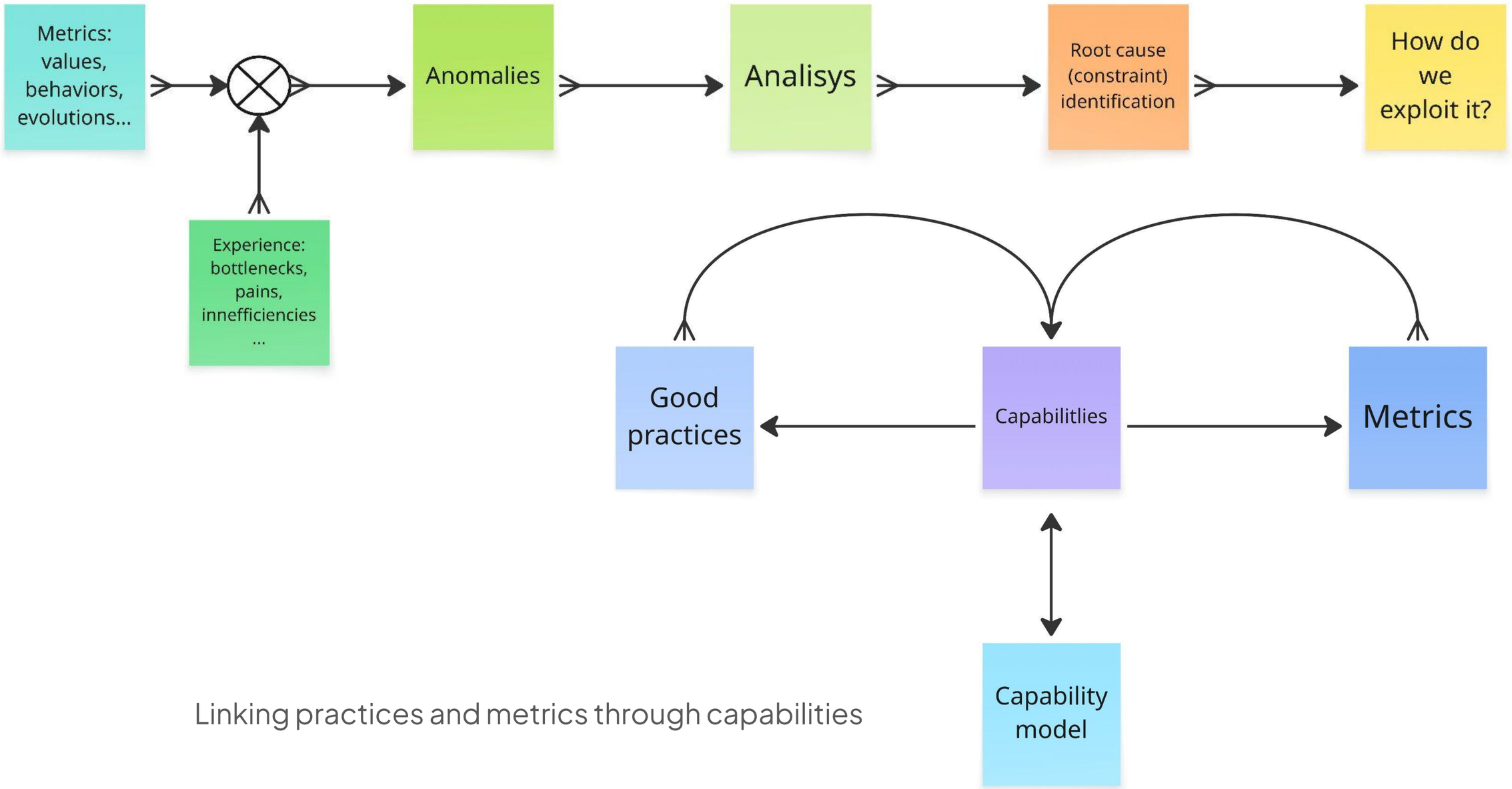

25th percentile of Leac
50th percentile of Leac
75th percentile of Leac
Lead Time (days)
Time (end date grouped by 90 days)
provides an idea of the time that source code takes until it is included in a valid snapshot. It measures the time since a Git commit in the master branch until a successful snapshot.
NOTE: the data for the builds that generate the development snapshot starts in June 2023.
UCB Delivery Process: Time Interval
25th percentile of Tim
50th percentile of Tim
75th percentile of Tim
Time Interval (days)
Time (sampling period 3 months)
Delivery Process / Time Interval: It's the time interval between successive successful outputs. In this case, it measure the time interval between successive successful development or pre-release snapshots.
NOTE: the data for the builds that generate the development snapshot starts in June 2023.
UCB Delivery Process: Failure Recovery Time
25th percentile of Fail
50th percentile of Fail
75th percentile of Fail
Delivery Process / Failure Recovery Time: It's the time interval between a failure and its remediation. In this case the time between a failed delivery process (because it failed with an


Bitergia Analytics
agl
UCB Delivery Process
AGL-Demo Delivery Process
Gerrit
Jenkins
Git
Dashboards
Jenkins
Full screen
Share
Clone
Edit
Search
DQL
Jun 1, 2023 @ 01:00:00.000 → Jan 1, 2026 @ 01:00:00.000
Refresh
Add filter
Summary
34,075
Builds
277
Jobs
16,645
Nodes
37.153
Build duration (avg, min.)
Builds over time
Builds
Time
Results
SUCCESS
FAILURE
ABORTED
FAILURE (7.99%)
SUCCESS (90.43%)
Active nodes over time
Count
Time
Builds
1–50 of 34075
Time
job_build
url
job_url
result
builtOn
duration
Dec 22, 2025 @ 08:35:58.125
rm-snapshots-master-old/3069
+Info
+Info
SUCCESS
control-slave-1 929
2,658
Dec 22, 2025 @ 05:33:39.269
daily-jjb-doc-upda
+Info
+Info
FAILURE
agl-test-slav
677
JenkinsSlaves
Node
Builds
Duration (...
Duration (...
hwtest-slave-1
124
4.679
1,418
control-slave-2
99
0
8,628


Search
DQL
Jun 1, 2023 @ 00:00:00.000 → Jan 1, 2026 @ 00:00:00.000
Refresh
NOT Merge Commits
NOT Bots
Add filter
Commits by Organization
Konsulko
Linux Foundation
Collabora
AISIN
Panasonic
Toyota
Fujitsu
Virtual Open Systems
Igalia
ICS
Other
Commits by Organization Over Time
Number of Commits
Time
Other
Independent
ICS
AISIN
Qt Group
Panasonic
MURENA
Unknown
1,891
Commits
22
Commits by Time Zone
Commits
Commits by Region Over Time
Americas
Europe/Africa
Asia-Pacific

Analysis

High level view of the process at the diagnostic stage



Linking practices and metrics through capabilities

Continuous improvement

Data-supported, continuous improvement process

- Step 1: pilots at small scale.
  - Validate the process.
  - Deploy successful experiments
- Step 2: scale up the process
- Step 3: deployment of successful improvements at scale
  - Track the evolution of the target metrics

Our role (zero vendor-lock-in policy):

  - Drivers ⇨ coaches ⇨ level 3 support ⇨ hand over

4.- Analysis examples

Analysis (diagnosis) examples

- Use case 01:
  - Symptom: developers wait too long for build results
- Use case 02:
  - Symptom: code review communication density is increasing
- Use case 03:
  - Symptom: random periods of time where failure recovery time values go way beyond thresholds (lower stability)
- Use case 04:
  - Symptom: a significant increase in development capacity in a functional group did not bring the expected time interval reductions (throughput increases)

USE CASE 01

Delivery process: performance analysis

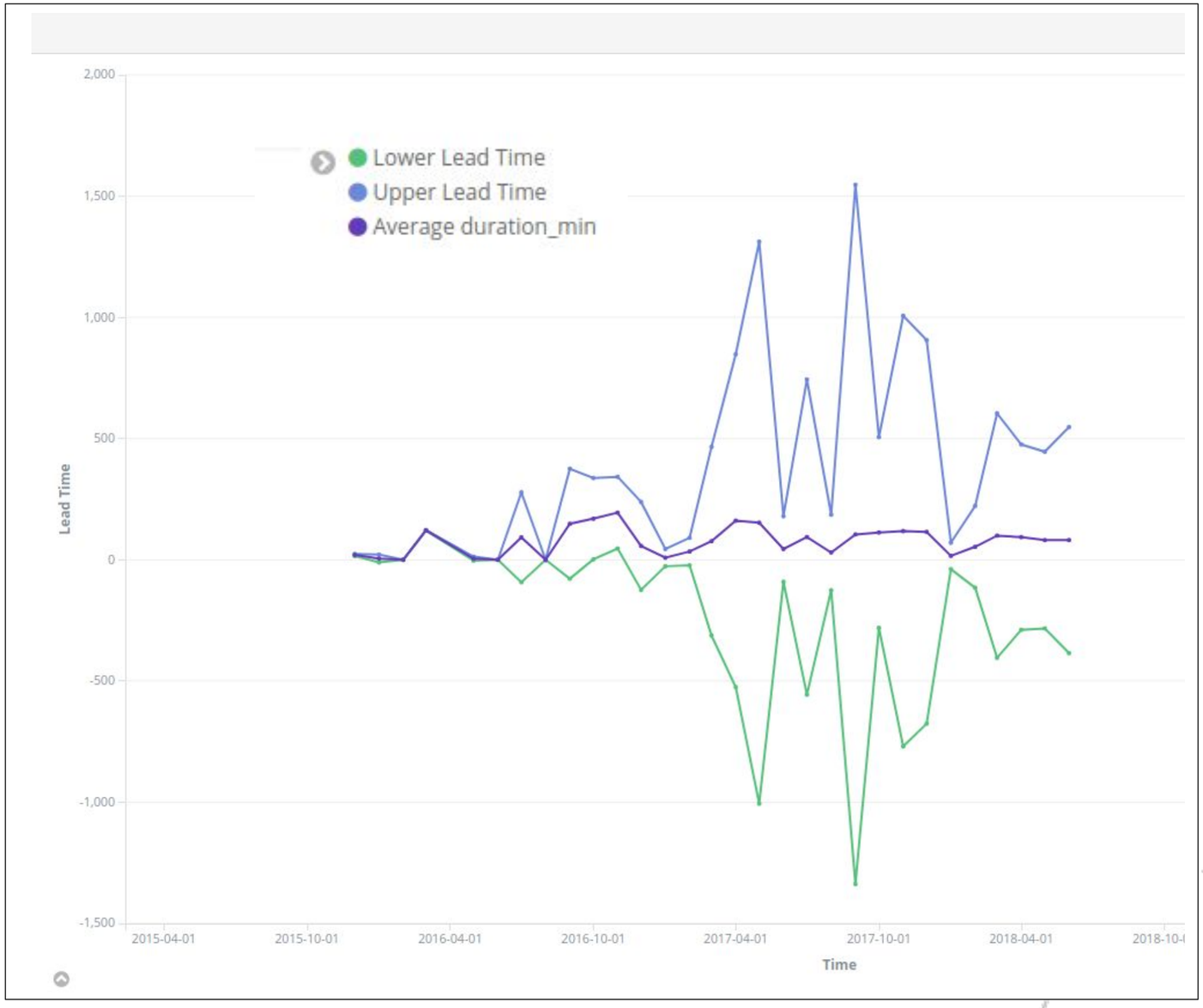
Describing the production system in general, and the delivery process in particular, is a necessary step to support further investigations and analysis to detect points of improvements and strengths.

Here are some real use cases on how such description can lead to interesting questions about symptoms that may or may not be evident to some.

01

Integration Stage

Integration Lead Time – the integration lead time shows a clear degrading of the performance of the delivery process on this stage. The graphic show that some measures were taken at some point but the the degrading came back again.

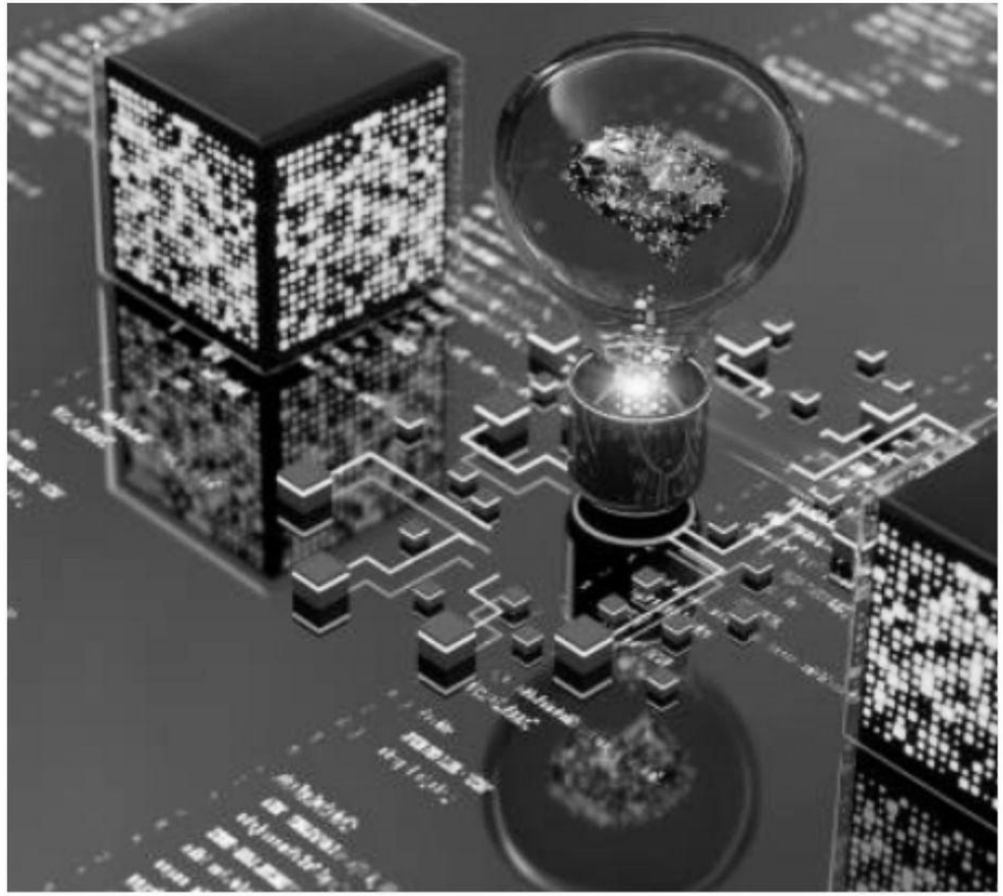


USE CASE 02

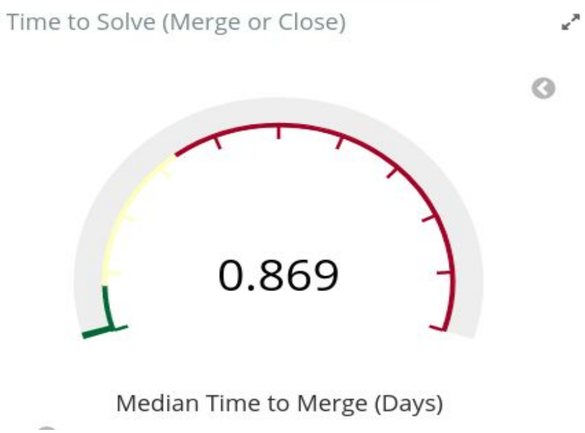
Performance Analysis: Code Review

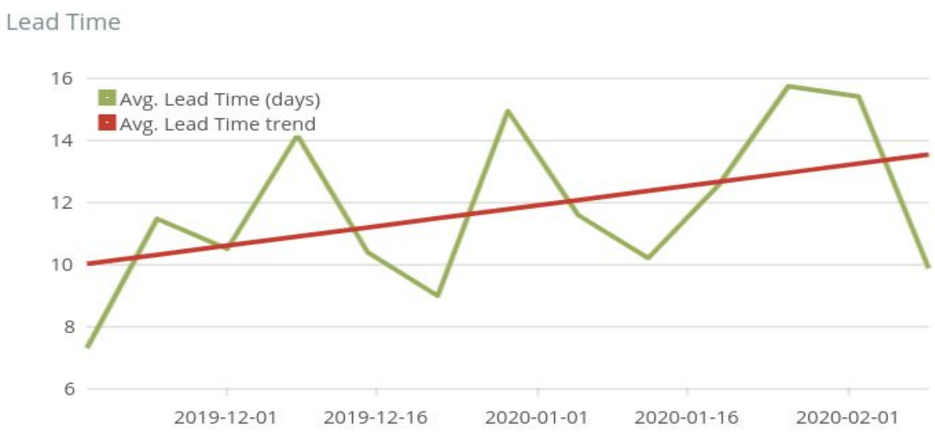
Code review analysis is relevant, not just to optimise the process itself, but also to predict good development practices among engineering teams, as well as antipatterns. Good development practices have a positive impact on the delivery process performance.

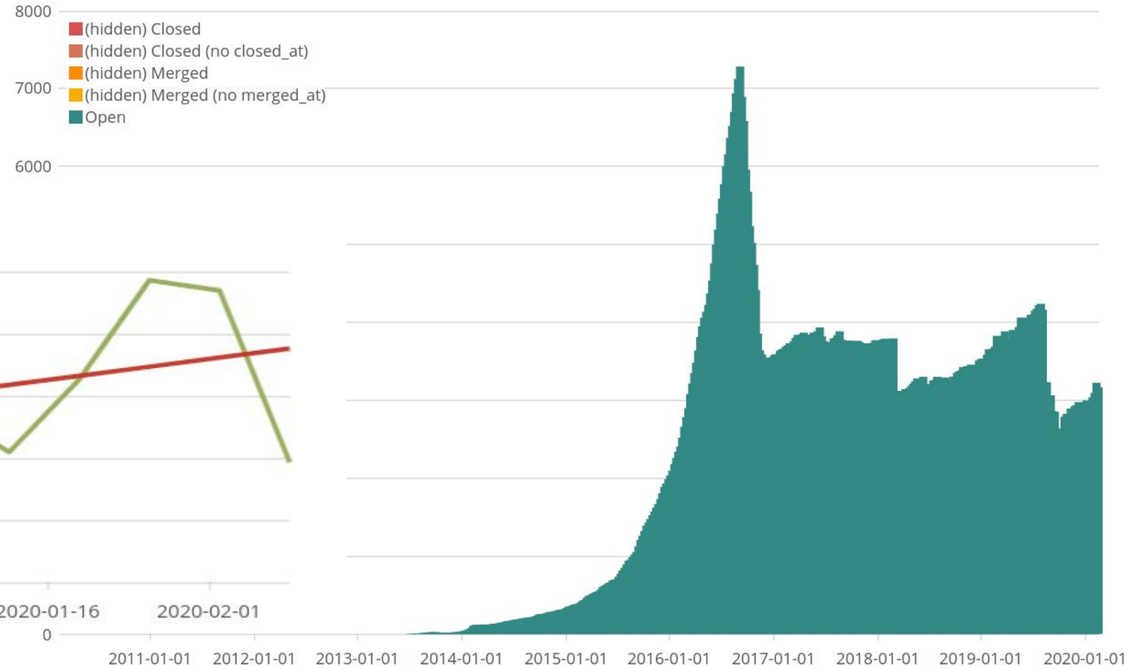
Code review is just one of the practices targeting code quality and knowledge transfer among developers. Its analysis needs to be expanded to include the study of constraints associated with it.

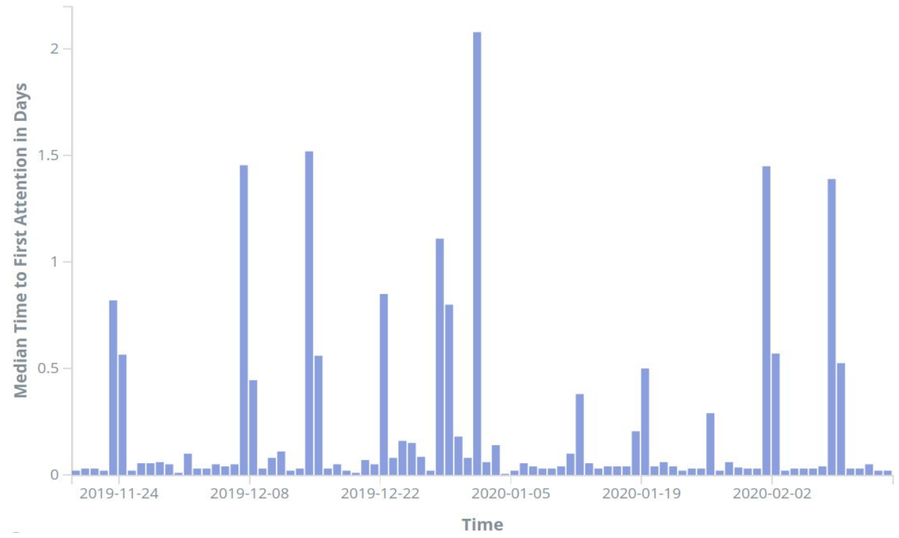


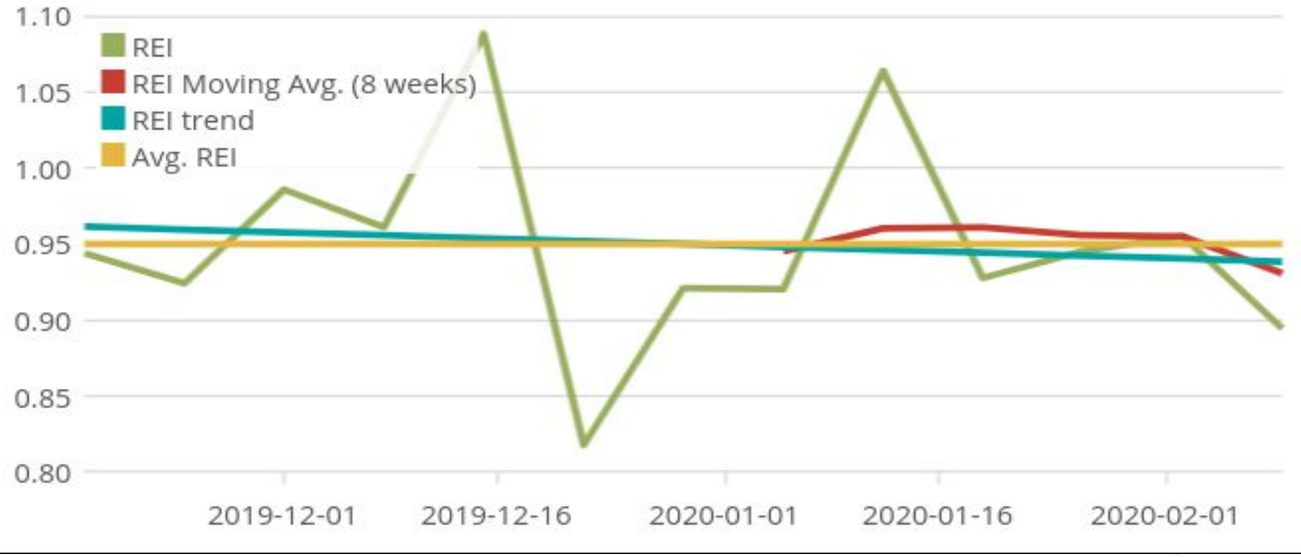
02 Code Review





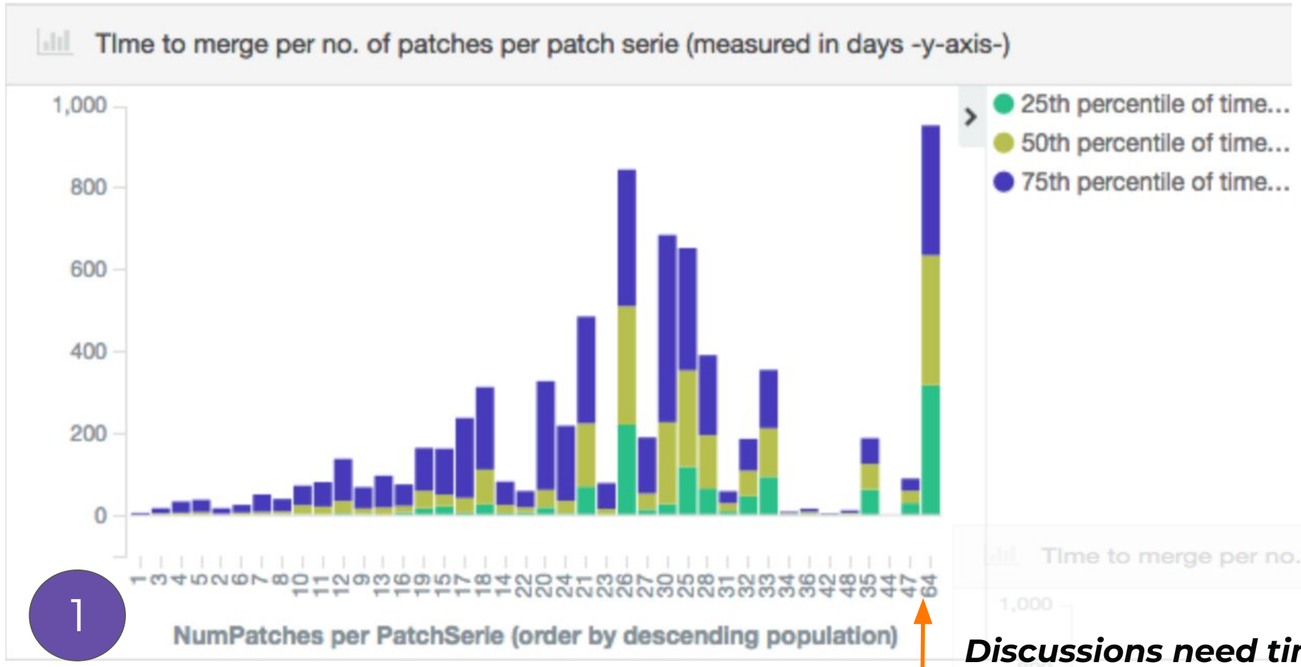




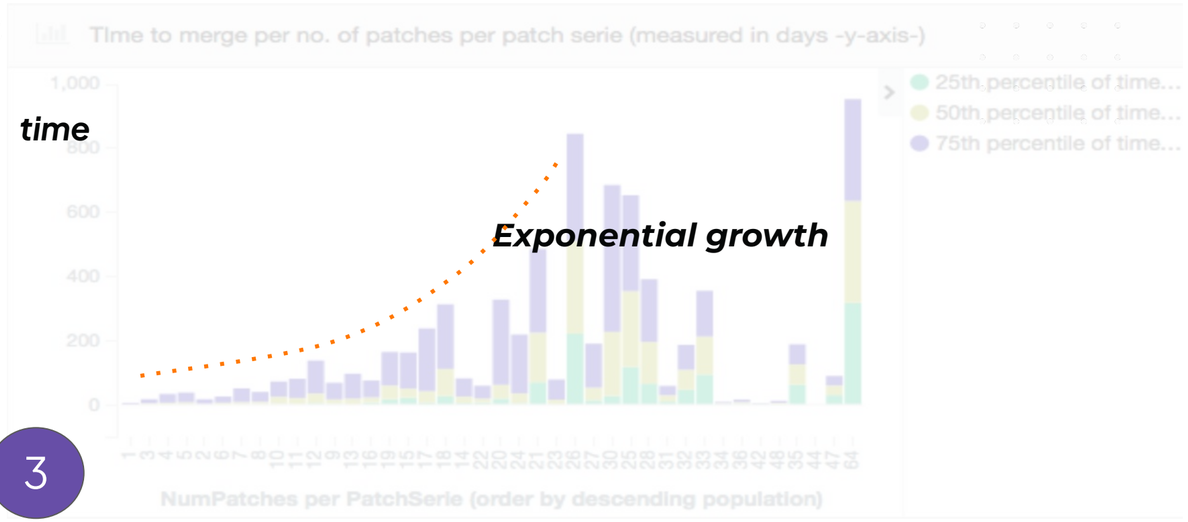


02 Code Review

D. Izquierdo, J. M. Gonzalez-Barahona, L. Kurth and G. Robles, "**Software Development Analytics for Xen: Why and How**," in *IEEE Software*, vol. 36, no. 3, pp. 28-32, May-June 2019, doi: 10.1109/MS.2018.290101357



1

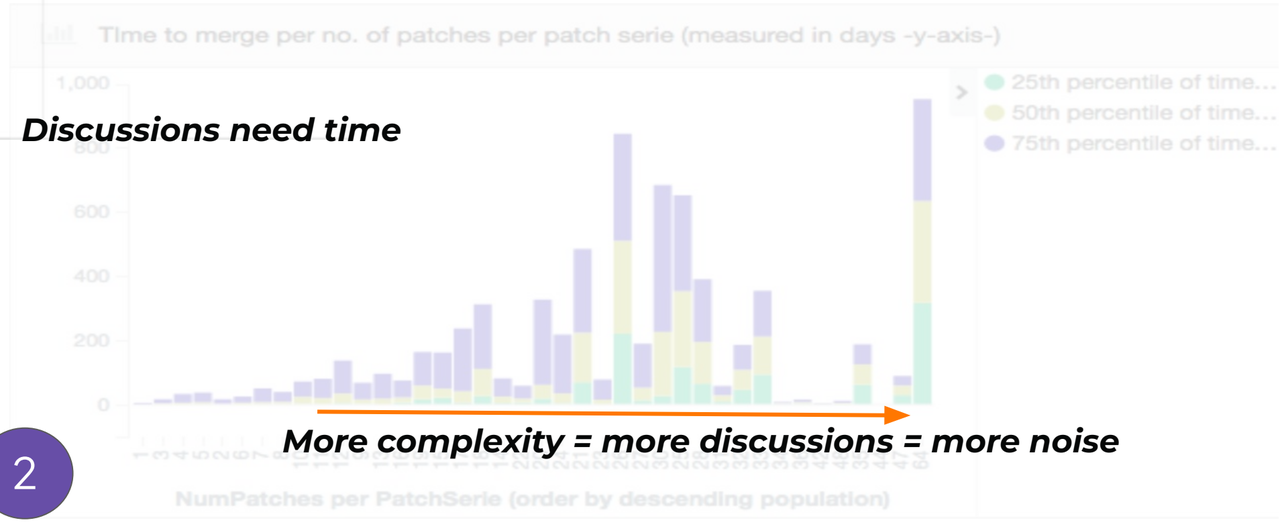


3

hypothesis (for the improvement kata): split patch series > 20 patches into smaller patch series

4

Discussions need time



2

More complexity = more discussions = more noise

USE CASE 03

Analysis of Complex Topics: Variability

Once the production system in general, and the delivery process in particular, have been characterised and described, we can start digging into the root causes for the most impactful symptoms. High levels of variability is frequently one of those symptoms. The sources of such variability needs to be determined.

03

Variability: Batch Sizes and Branch Lifespans

We can see noticeable differences among organizations in batch sizes and branch lifespans. These differences increase variability, potentially degrading the delivery process in later stages. Different cultures, practices, contract terms, access to baselines or HW, etc. might be behind these differences.

Search

Organization	Commits	Authors	Touched Files	Added Lines	Removed Lines	Projects	Repositories	Avg. Lines/Commit
org. 1	397	8	1,461	25055	6204	1	14	78.738
org. 2	313	1	566	21412	3480	1	5	79.527
org. 3	100	3	246	1044	1529	1	4	25.73
org. 4	33	1	65	11741	9493	1	1	643.455
org. 5	33	1	95	512	239	1	2	22.758
org. 6	33	1	212	1106	176	1	2	38.848
org. 7	27	3	67	976	180	1	3	42.815
org. 8	17	3	381	8950	7971	1	4	995.353
org. 9	11	4	24	715	1279	1	1	181.273
org. 10	10	1	10	106	6	1	2	11.2

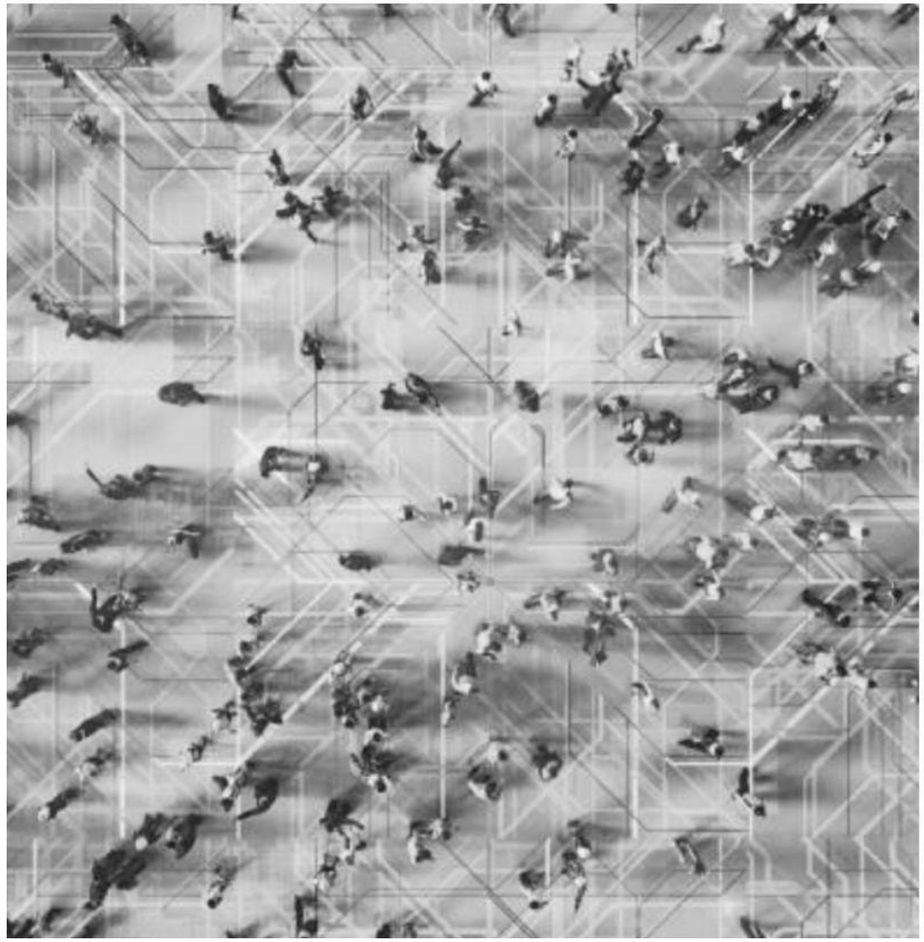
Organization	Merge Requests	Submitters	Repositories	Avg. Open Days
org. 1	95	1	5	11.513
org. 2	88	6	14	27.781
org. 3	61	3	5	66.873
org. 4	15	1	4	74.803
org. 5	14	2	3	88.722
org. 6	3	1	2	3.47
org. 7	2	1	1	0.46
org. 8	2	1	1	9.79

USE CASE 04

Analysis of Complex Topics: Dependencies

Dependencies might be hard to detect and predict when producing software-defined products at scale. They are the root cause of numerous symptoms that affect the delivery process performance.

These visualizations target the detection of undesired or unexpected dependencies between orgs., teams, bug reports, repositories and specific code files.

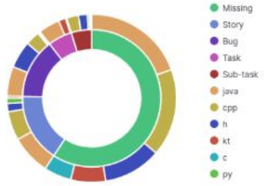


Limitation: at the time this is updated, 1/3 of the commits could not be classified. That means that the number of bugs is *underestimated*.

The tables below show that [redacted] fixes bugs at a higher rate than team [redacted]

More info: The two tables on the left show the type of activity per repository and company. One of them is focused on the number of commits and their link with a bug report. The other one goes a bit deeper, as it focuses on the modifications over files (1..n modifications per commit). The column on the right calculates the diff between the proportion of activity for each team, it is in yellow when the result is the expected one ar[redacted] investing less effort in working on bugs.

Modified file extensions per issue type



File modifications per company

[redacted] # file updates)	[redacted] Bugs)	[redacted] # file updates)	[redacted] % Bugs)	Diff Activity
26,074	7	34,309	19	12

Commits per company

[redacted] (# commits)	[redacted] (% Bugs)	[redacted] # commits)	[redacted] (% Bugs)	Diff Activity
3,420	13	3,067	25	12

File modifications per company and repository

Repository	SE (# file updates)	SE (% Bugs)	CN (# file updates)	CN (% Bugs)	Diff Activity
[redacted]	2,042	29	1,312	52	23
	80	23	2,466	11	-12
	2,446	6	0		
	1,523	19	755	57	37
	4	0	1,619	4	4
	795	36	318	10	-26
	168	0	1,418	15	15
	1,391	0	359	8	8
	40	90	0		
	8	13	3,183	3	-10
	447	2	0		
	28	8	342	25	17
	535	1	0		
	921	7	2	0	-7
	0		995	3	
	520	6	0		
	1,207	3	0		
	20	0	5,212	6	6
	1,332	2	0		
	431	9	26	46	37

Export: Raw Formatted

1 2 3 4 5 … 6 »

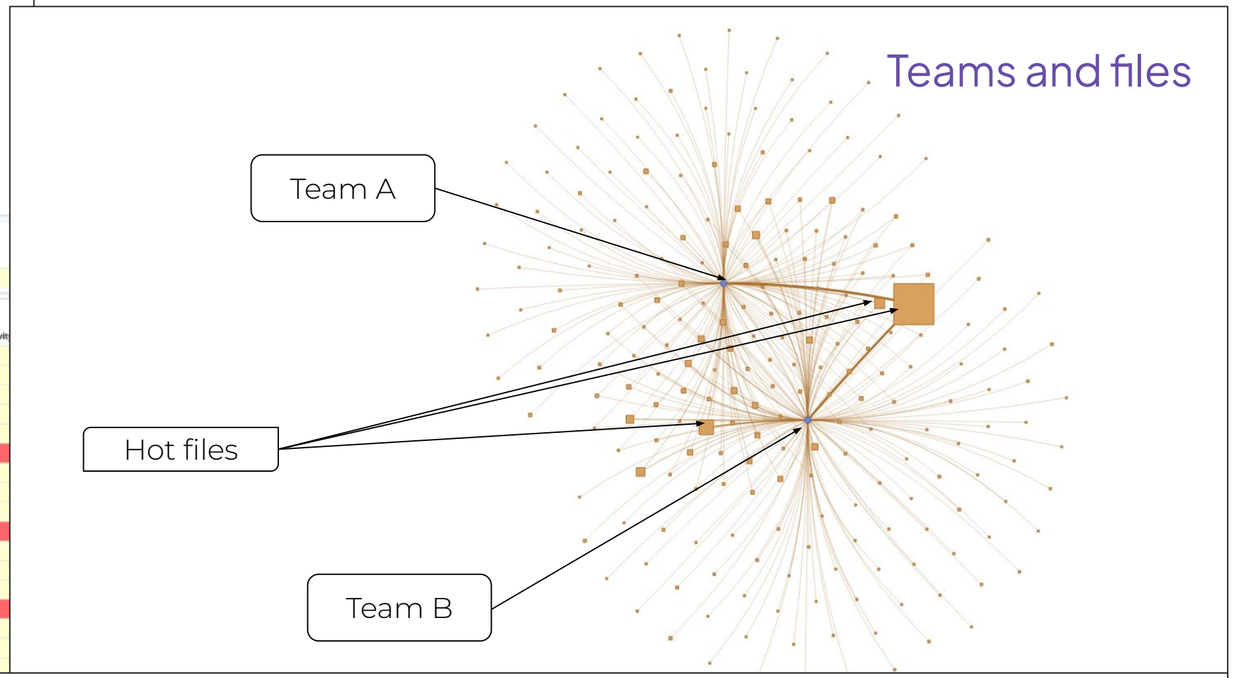
Commits per company and repository

Repository	SE (# commits)	SE (% Bugs)	CN (# commits)	CN (% Bugs)	Diff Activit
[redacted]	431	34	419	64	30
	22	9	692	14	5
	295	7	0		
	222	31	167	66	35
	2	0	290	7	7
	131	37	71	11	-25
	3	0	198	20	20
	124	0	66	21	21
	9	78	0		
	3	33	146	21	-13
	149	3	0		
	13	8	133	27	19
	108	1	0		
	123	11	1	0	-11
	0		116	7	
	63	16	0		
	112	4	0		
	6	0			
	109	6			
	90	10			

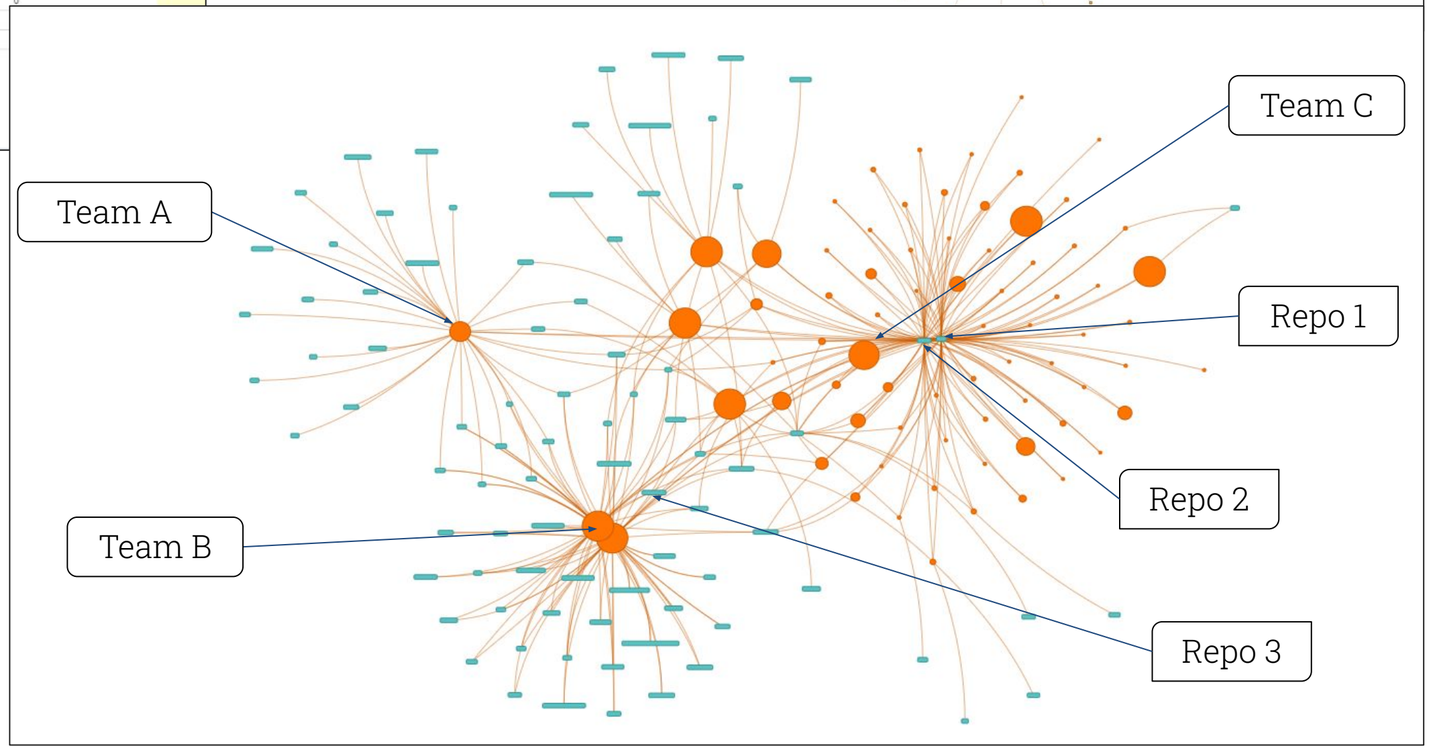
Export: Raw Formatted

Orgs. and modified repos/files

Teams and files



Teams and repos



Example of Stage 1 and stage 2 applied to AGL

Check the report and the full study in AGL's CIAT WG wiki

- Structure of the study
  - Full Study ToC
- Measurements and plots: link
  - Community and activity: link
  - Code review process: link
  - Delivery process: link
- Analysis: link

5.- Experiences

Open Source vs commercial

	Topic	Commercial env.	OSS Project
1	Tools	More tools. Many proprietary.	Less tools. Mostly open source.
2	Tools' observability	Improving fast	Need to improve
3	Tool's purpose	Used for the original purpose	Some tools pushed beyond their original intend
4	Data	Restricted and often close-to-impossible to access	Open and fairly accessible
5	Min. execution unit	Teams. Simpler to profile (persona)	People (harder to profile/characterise)
6	Capacity	Known and predictable. Abrupt evolution	Mostly known but unpredictable. Organic evolution
7	Metrics	Higher number. Req-to-code is relevant	Lower number. Community health is relevant
8	Ecosystem	Fewer orgs. with structured relations	Many orgs. with unstructured relations
9	Focus	More focus on team and delivery performance as well as business-impact aspects	More focus on contributors well-being and cost-impact aspects

5.1.– Experiences: data and tools

About tools

- Data lock-in is a real risk. As part of the procurement process, evaluate…
  - The observability capabilities and limitations through external tools
  - Data models should be documented
  - Data storage and management, especially identities
- Different groups of data consumers require different reporting tools.
  - Focus on the dataHub
  - For our efforts, we do not require complex visualizations

About data

- Data engineers need context to process the data correctly.
  - Understanding the SW production system is also key to them.
  - SW Production system data should be part of the overall data strategy
- Do not trust the data by default
  - Look for anomalies in the graphs/tables, constantly.
  - The quality of “current data” decays over time when, faster than software.

    Plan for maintaining it.

About data

- Do not underestimate the value of understanding and selecting the right data types for your study
- Drowning in lower-impact data/graphs is a real risk
- Do not overload the platforms used to get the data from

5.2.- Experiences: metrics

About metrics

- Avoid discussing about metrics, discuss about goals and questions instead
  - Use the Goal-Question-Metric approach when discussing about metrics

About metrics

- Extensive approach
  - Start simple:
    - simple metrics, simple graphs… simple everything.
    - one metric at a time
  - Walking skeleton first: end-to-end metrics first, then metrics that can only be applied locally
- Implementing simple metrics usually take you further than complex metrics

5.3.- Experiences: visualizations

About visualizations

- The collaboration between data engineers and consultants is essential:
  - To narrow what is available down to what matters
  - To validate the dashboards with users, constantly, but…
  - Fight against *adding more*… unless your outputs lead to higher impact decisions/actions by the users…
  - … to prevent losing focus on the key goals

About dashboards/graphs/tables

- Extensive vs intensive approach, for a bigger impact
- Fewer outputs shared among more teams/roles
- Visualizations that provide limited information about any metric applied end-to-end will have more impact

5.4.- Experiences: insights

Insights

When identifying and analysing the root causes (constraints)…

- Attack the highest priority questions first, one by one
  - The highest risk is to put focus on identifying low impact root causes to highly visible symptoms: bigger flames get more attention
- Analyse in iterations:
  - Partial insights might be valuable, saving time and effort
  - Digging beyond required is a common trap

Insights

- Without reaching consensus on the insights (identified constraints/root cause), do not move onto the next question…
  - … or you will pay the price later on, when there might not be a coming back
- Insights are more useful to customers/contributors when they are actionable

Insights

- Do not underestimate the power of confirming experience through data.
  - Reality is poliedric and it is perceived differently by different stakeholders.
  - As consultants, do not try playing smart.Those on the ground are the real experts.
- If consultants conclusions contradict, or doesn’t support, the customer’s experience…
  - … double check: the data ⇒ metrics ⇒ dashboards ⇒ insights…
  - …and only then, question their experience (interpretation)

Insights

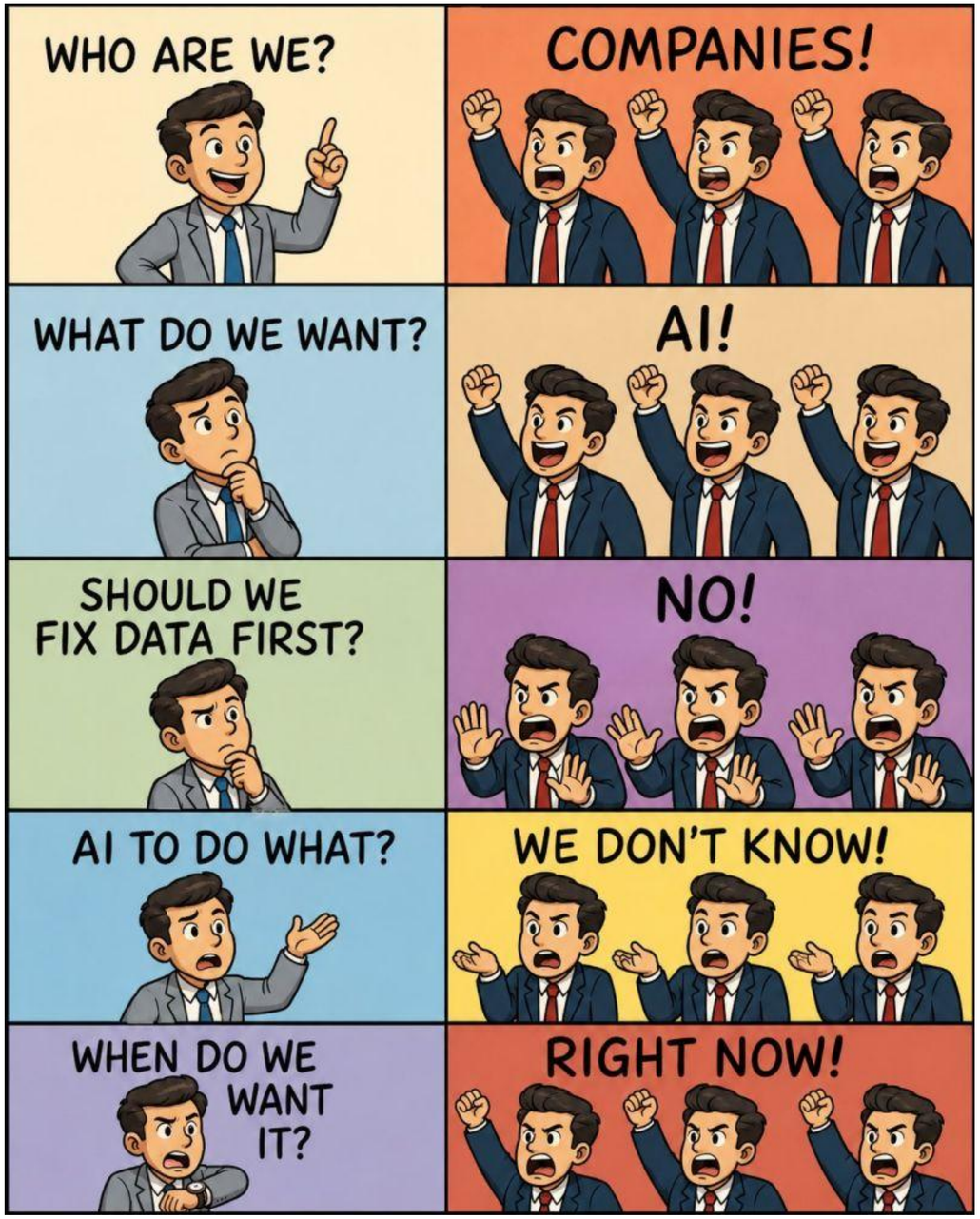
- Common language is required to communicate symptoms, root causes, target expected improvements, experiments' results….
  - Be effective: adapt to the customer's/project's language
- Remedies/improvements directly impacting the entire SW production system (end-to-end) should be prioritised over local ones

5.5.- Experiences: other

open source matters

- Open Source projects are:
  - Far more than just data sources and showcasing environments
  - Rich software production environments with complex production systems
  - Superb learning ecosystems
    - Code review or risks models, for instance, are areas where we built our expertise in open source environments
- No staff implication, no success: collaboration
- Open Source values matter
  - Transparency brings trust and reduces friction, also in commercial env.
  - The zero-vendor-lock approach to services is way more than a differentiator

AI: do your homework

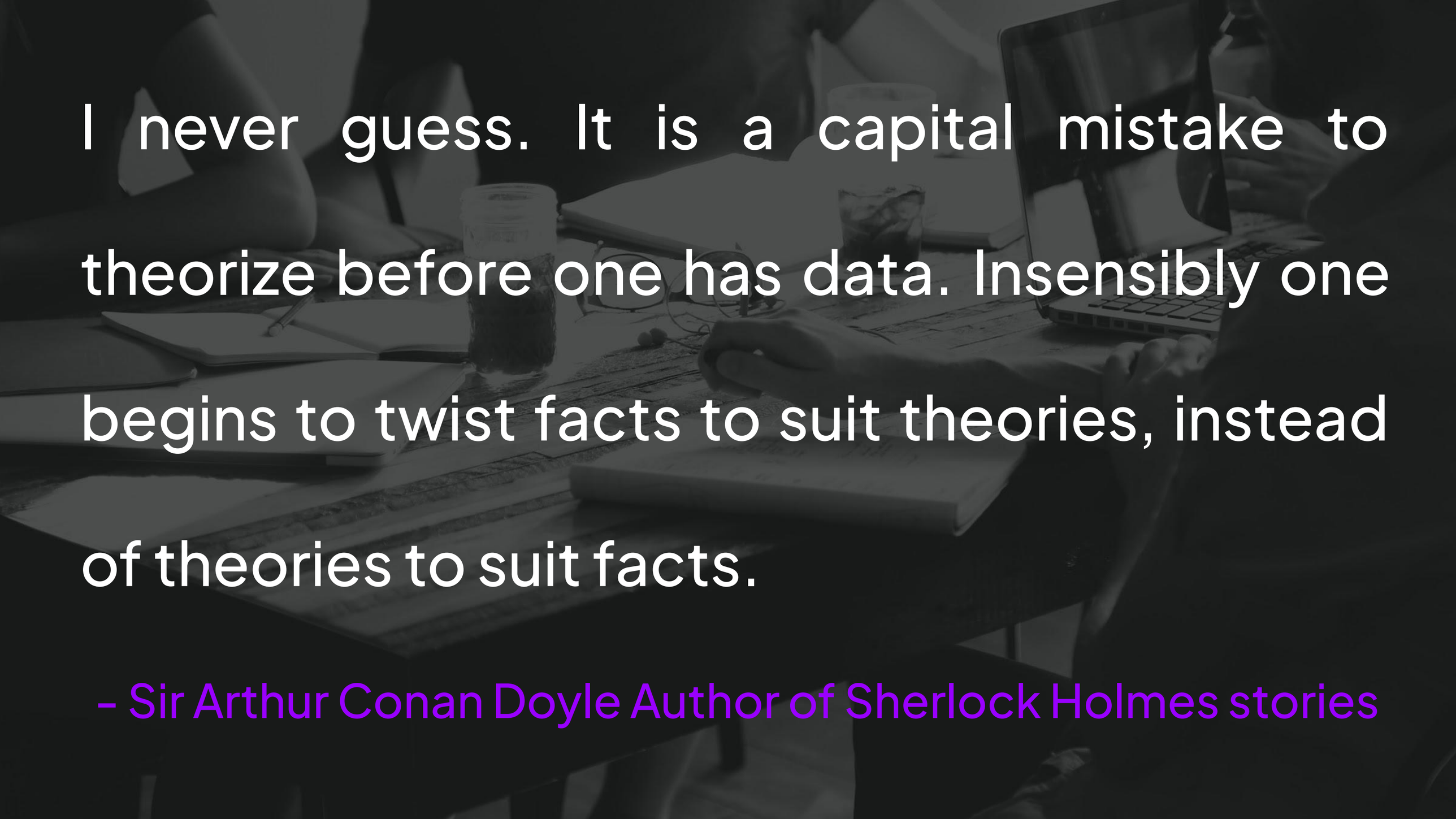


- Creating a data-driven, holistic view of your production system is essential to understand it
- The most effective way to understand your SW production system's behaviour is by applying a systematic approach to improving it
- You will then be ready to apply ML & AI, getting meaningful and sustainable results

Do not lose focus

The ultimate goal for the customer is to balance:

- Time-to-market reduction and…
- … product quality, innovation rates and staff's well-being increase…
- …through data-supported, continuous improvements, and decision making processes …
- …across the software production system



I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

- Sir Arthur Conan Doyle Author of Sherlock Holmes stories

Do not lose focus

The ultimate goal is not:

- Accuracy
- The confirmation of predefined theories
- The creation of a dashboards-as-a-service group within the company to:
  - Feed (project) managers with data for daily usage
  - Create pretty pictures for champions to include in their presentations for executives

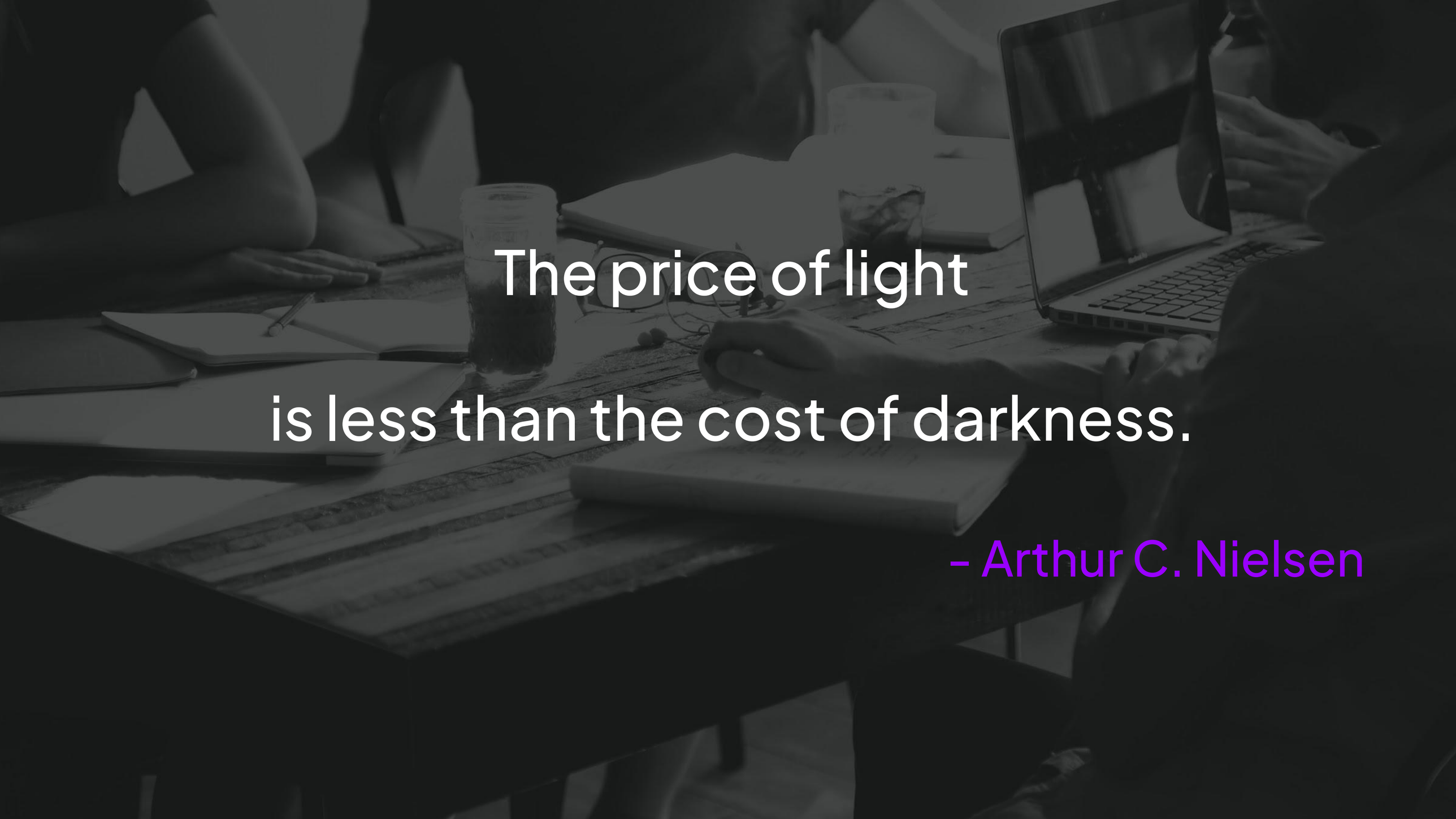

When a measure becomes a target,
it ceases to be a good measure.
– Goodhart’s Law

What are you measuring?

We are measuring the SW production system performance, not the execution units performances.

- Avoid using the information extracted from the analysis of the SW Production System to evaluate min. execution unit's performance, or even worse, to compare performances against each other.
- If you do, your organization's culture will suffer
  - In not-so-extreme cases, irreversibly

6.- Takeaways


The price of light
is less than the cost of darkness.
- Arthur C. Nielsen

Takeaway 1

Applying BI to increase software production and organizational (project) performance, while increasing staff/contributors wellbeing, works for both, open source projects and commercial organizations

Takeaway 2

Introduce the SW production system data, as part of the BI Journey, into your organization’s data strategy

Takeaway 3

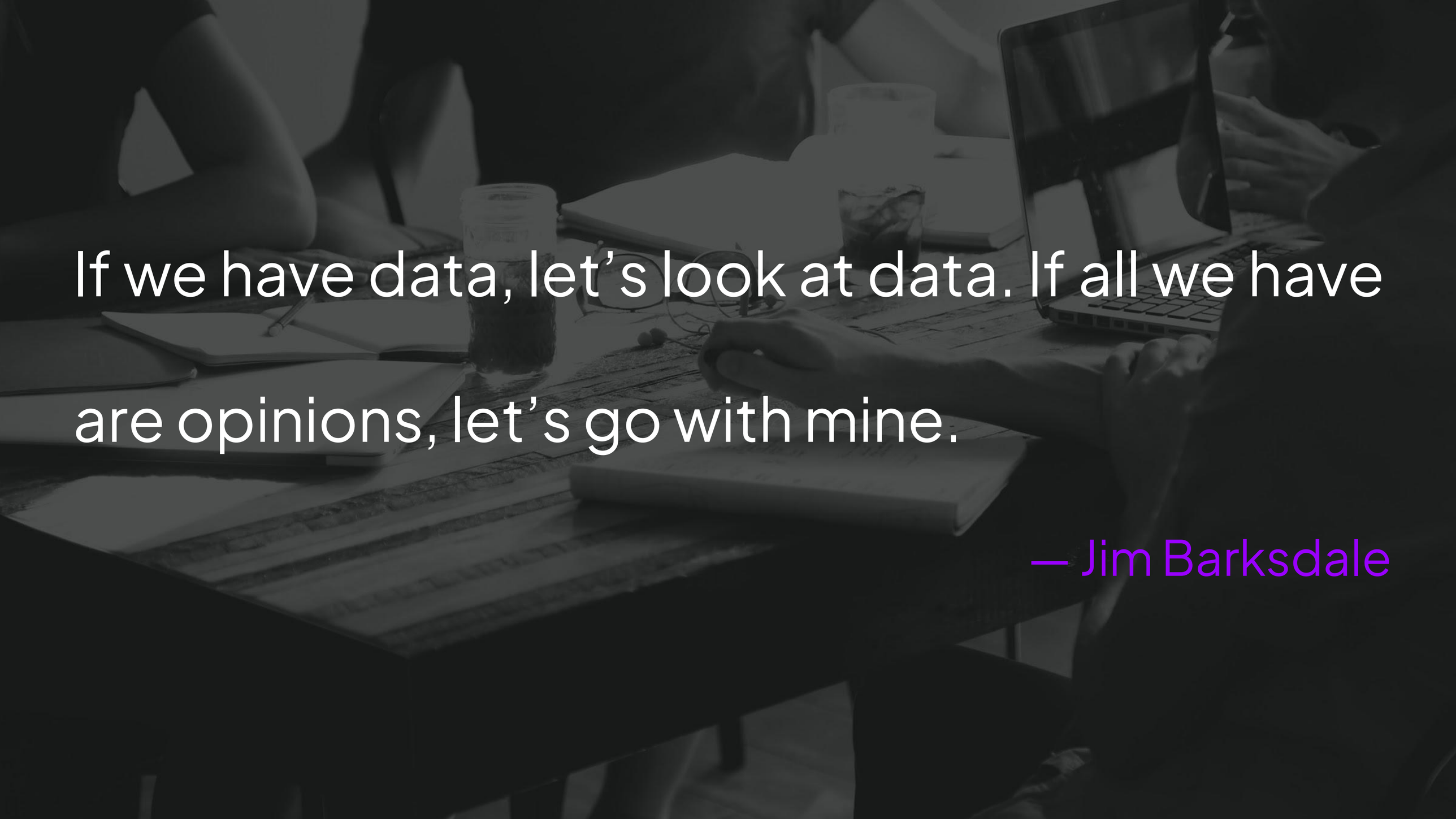
Keep focus on the key goals

Takeaway 4

Simplicity and end-to-end are essential concepts, at every stage of the BI Journey

Takeaway 5

Do not start the house by the roof. Stages 1, 2 and 3 of the BI Journey are pre-requisites to apply ML and AI successfully in the predictive and prescriptive stages



If we have data, let’s look at data. If all we have are opinions, let’s go with mine.

— Jim Barksdale

Thank You

Q&A

Lessons Learnt from our BI Journey:

From Open Source to SDV products development

EF SDV - Shifting Gears Webinar
2026-06-03

Agustín Benito Bethencourt
&
Luis Cañas Díaz