Maintaining Architectural Quality in Software Teams

Evelyn van Kelle
@EvelynvanKelle

Yiannis Kanellopoulos
@ykanellopoulos

12 million

professional software developers

</> write

120 billion

lines of code per year

of which

15%

needs to be changed each year

Is the rapid digitalization of our modern information society sustainable?

@EvelynvanKelle

Image by Tom Patterson, www.shadedrelief.com.

@ykanellopoulos

The secret for high quality software: Listen to your people.

The way you architect your software mirrors the way your organization thinks.

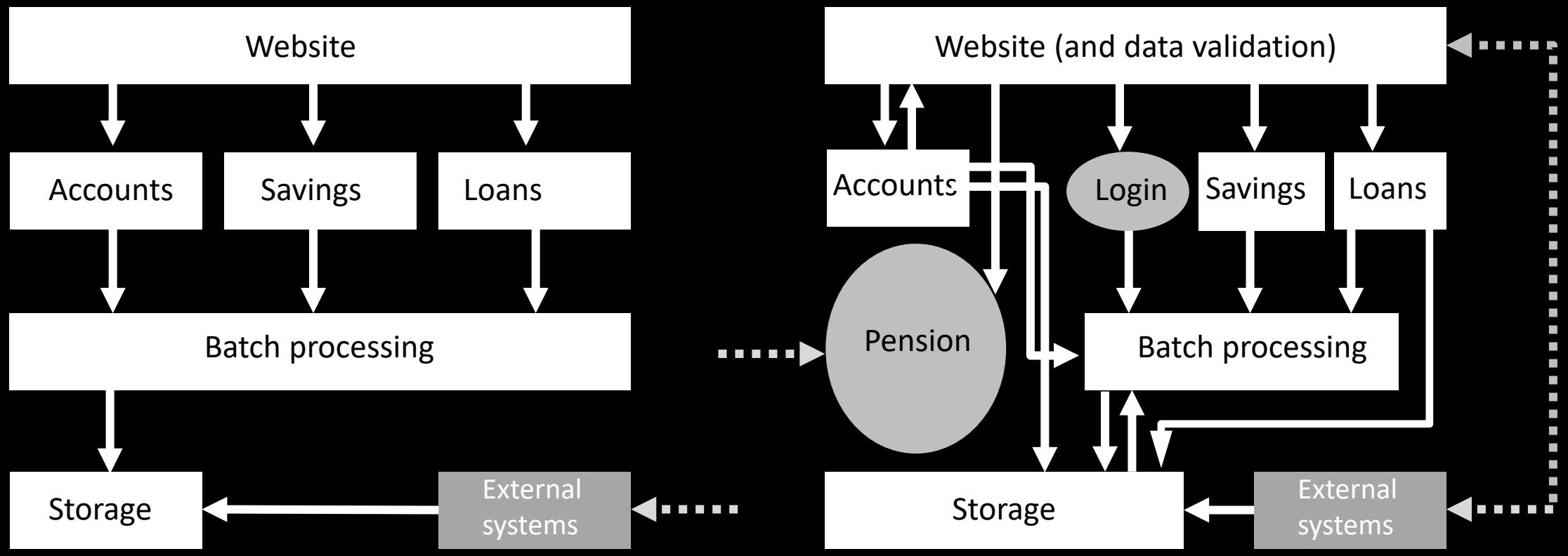
The way you architect your software mirrors the way your organization thinks.

@EvelynvanKelle

@ykanellopoulos

Plan versus Reality

Plan versus Reality



What we want
(or what we designed)

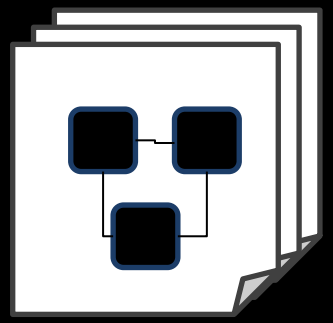
What we ended up with
(or what we implemented)

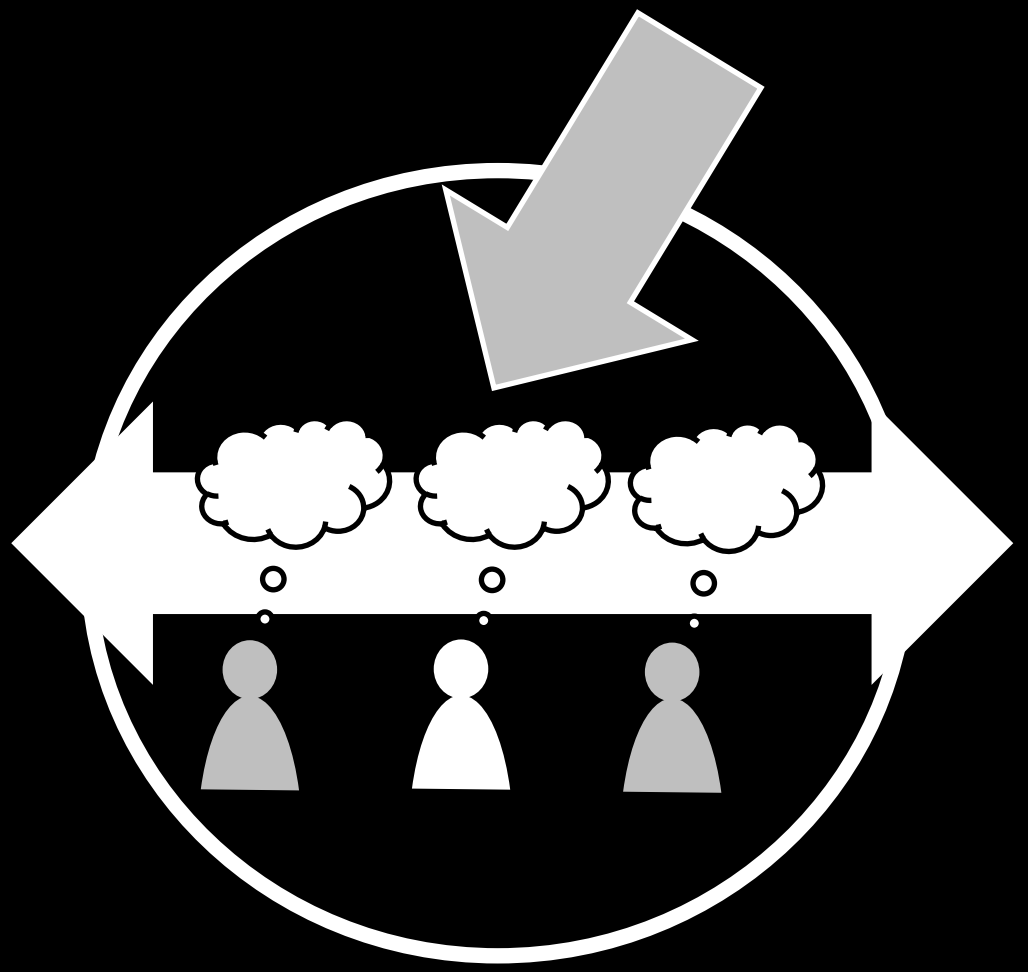
@EvelynvanKelle

@ykanellopoulos

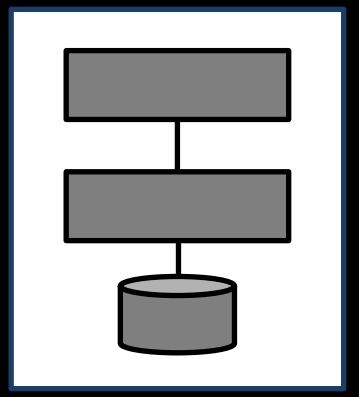
Where did it go wrong?

Designed architecture





Implemented architecture



@EvelynvanKelle

@ykanellopoulos

The future is all about people.

Top 10 causes of Unhappiness, Categories, and Frequency

CAUSE	CATEGORY	
Being stuck in problem solving	Software developer's own being	186
Under performing colleagues	External Causes -> people -> colleague	71
Bad decision making	External Causes -> Process	42
Time pressure	External Causes -> Process	152
Feel inadequate with work	Software developer's own being	63
Imposed limitation on development	External Causes -> artifact and working with Artifact -> code and coding	40
Bad code quality and coding practice	External Causes -> artifact and working with Artifact -> code and coding	107
Mundane or repetitive task	External Causes -> Process	60
Unexplained broken code	External Causes -> artifact and working with Artifact -> code and coding	57
Personal issues – Not work related	Software developer's own being	39

Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamson 2017. On the Unhappiness of Software Developers. In Proceedings of 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, June 15–16 2017 (EASE '17)

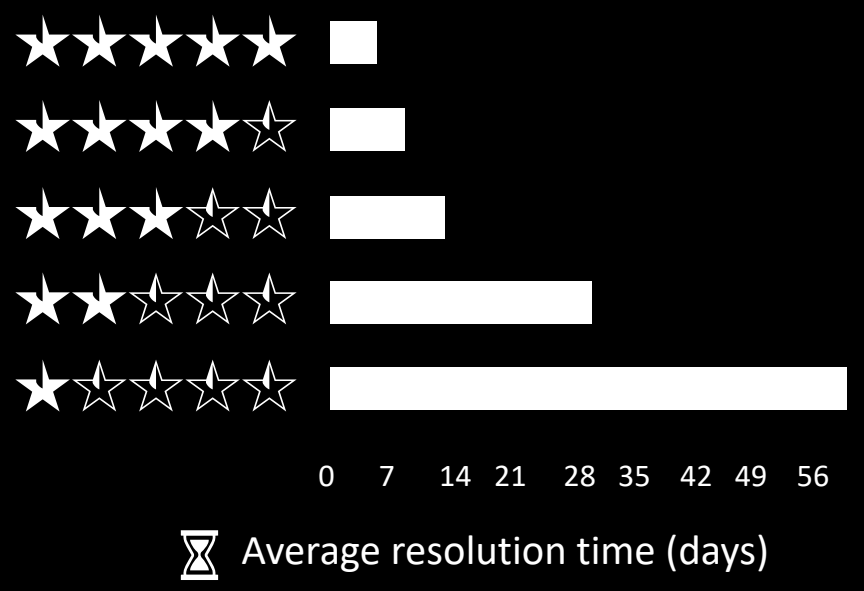
@EvelynvanKelle

@ykanellopoulos

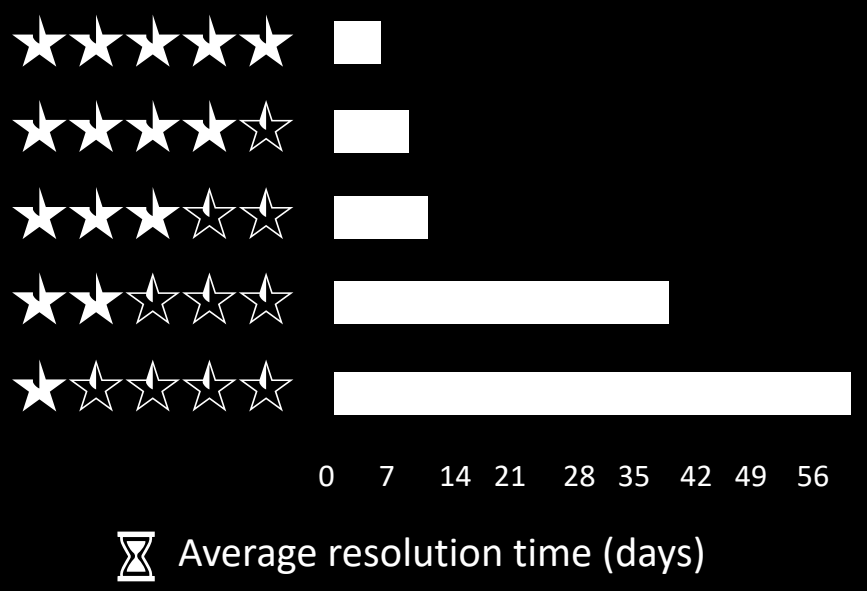
A higher software product quality leads to:

- The faster implementation of improvements and the solution of defects
- The throughput rate improves by factor 3.5 to 4.0 between 2 and 4 stars

Time needed to implement enhancements



Time needed to fix bugs



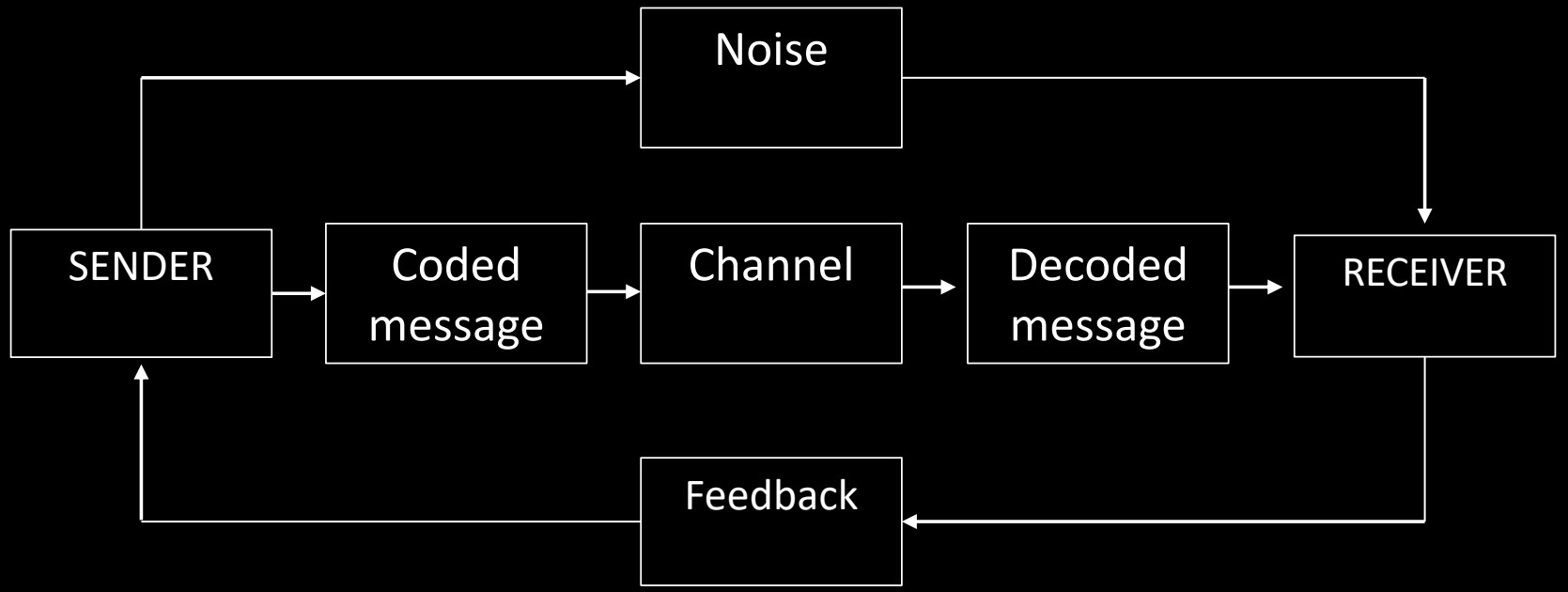
Source: "Faster issue resolution with higher technical quality of software", Software Quality Journal, 2011

COMMUNICATION

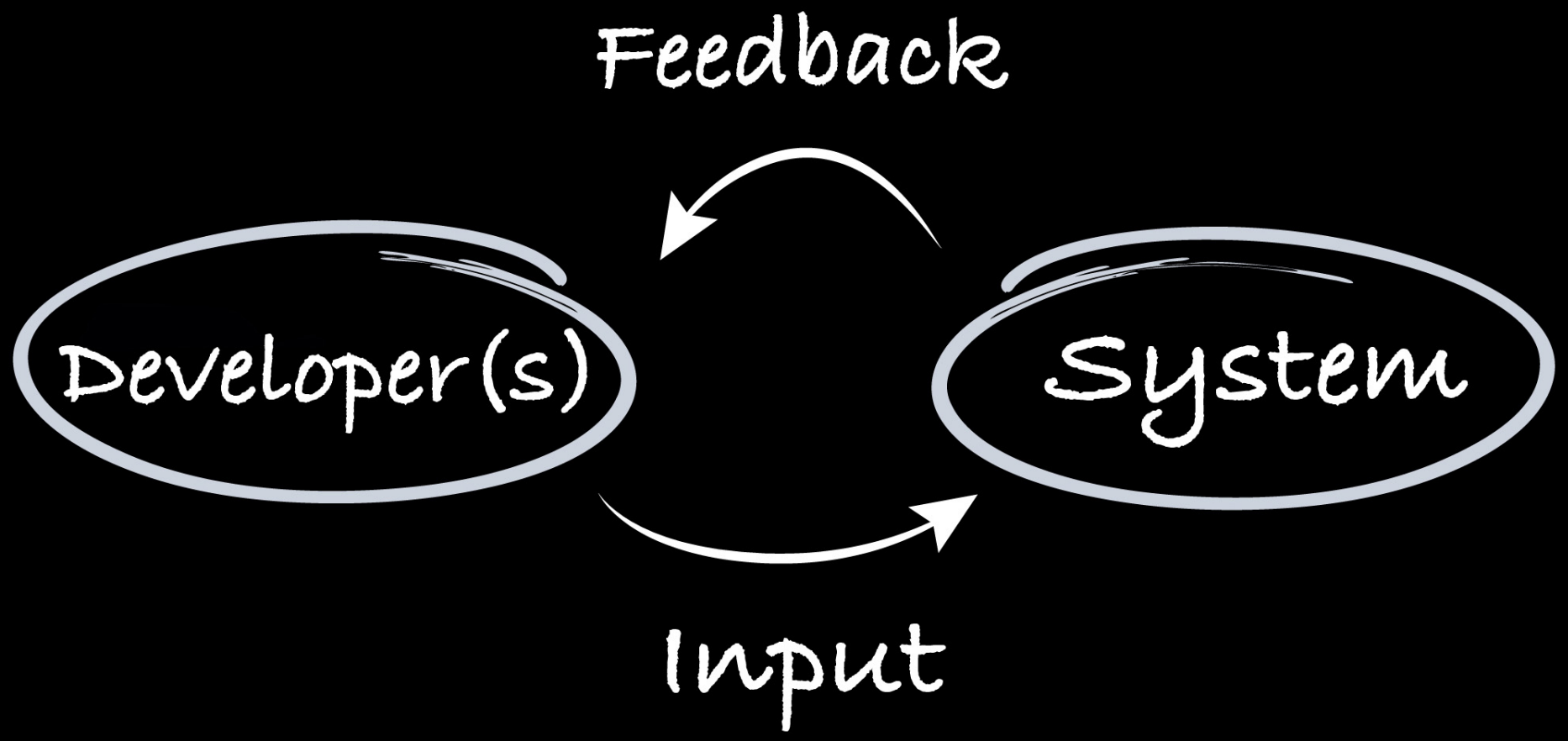
The artifacts of your ideas

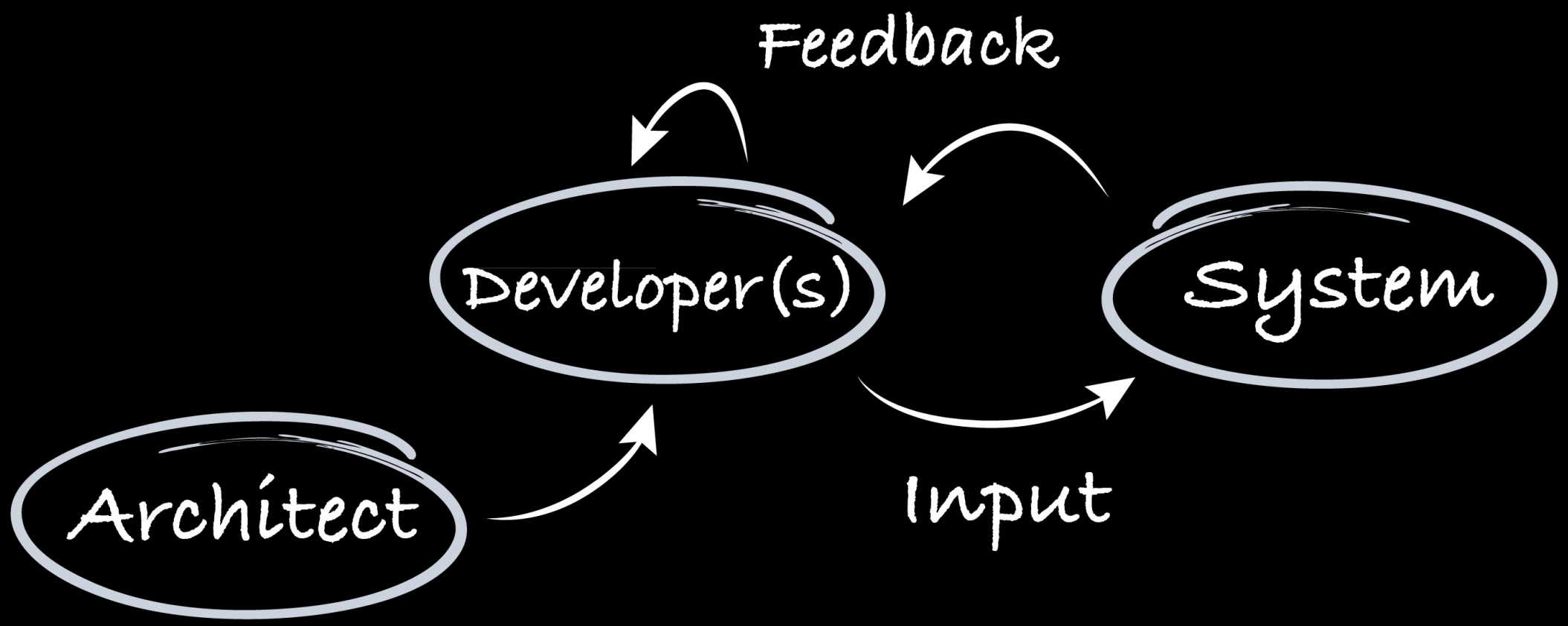
COMMUNICATION

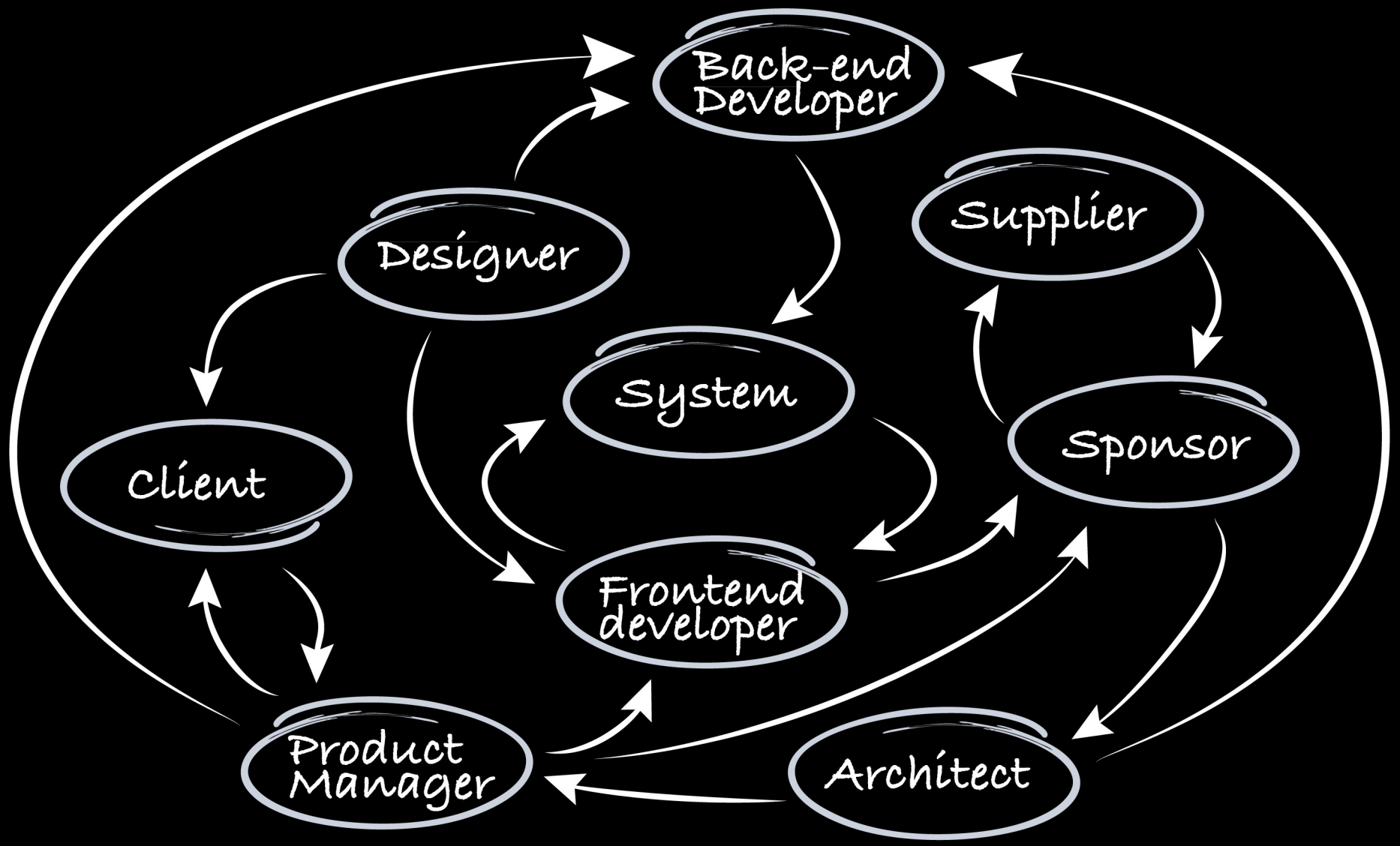
If it goes well…



Feedback Loops


Feedback
Developer(s)
System
Input


Feedback
Developer(s)
System
Architect
Input

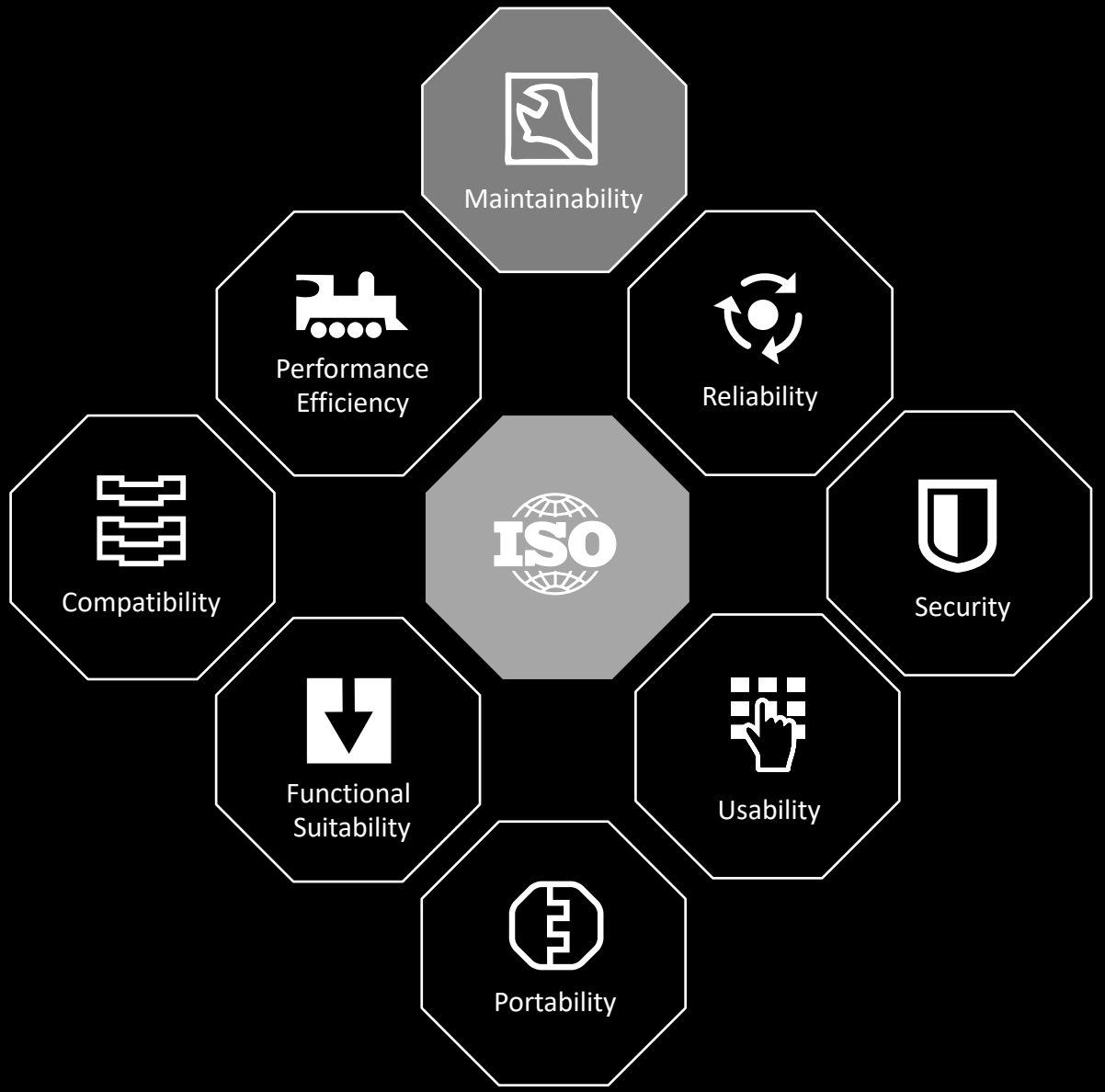


@EvelynvanKelle

@ykanellopoulos

Value Congruence

Non-functionals



@EvelynvanKelle

@ykanellopoulos

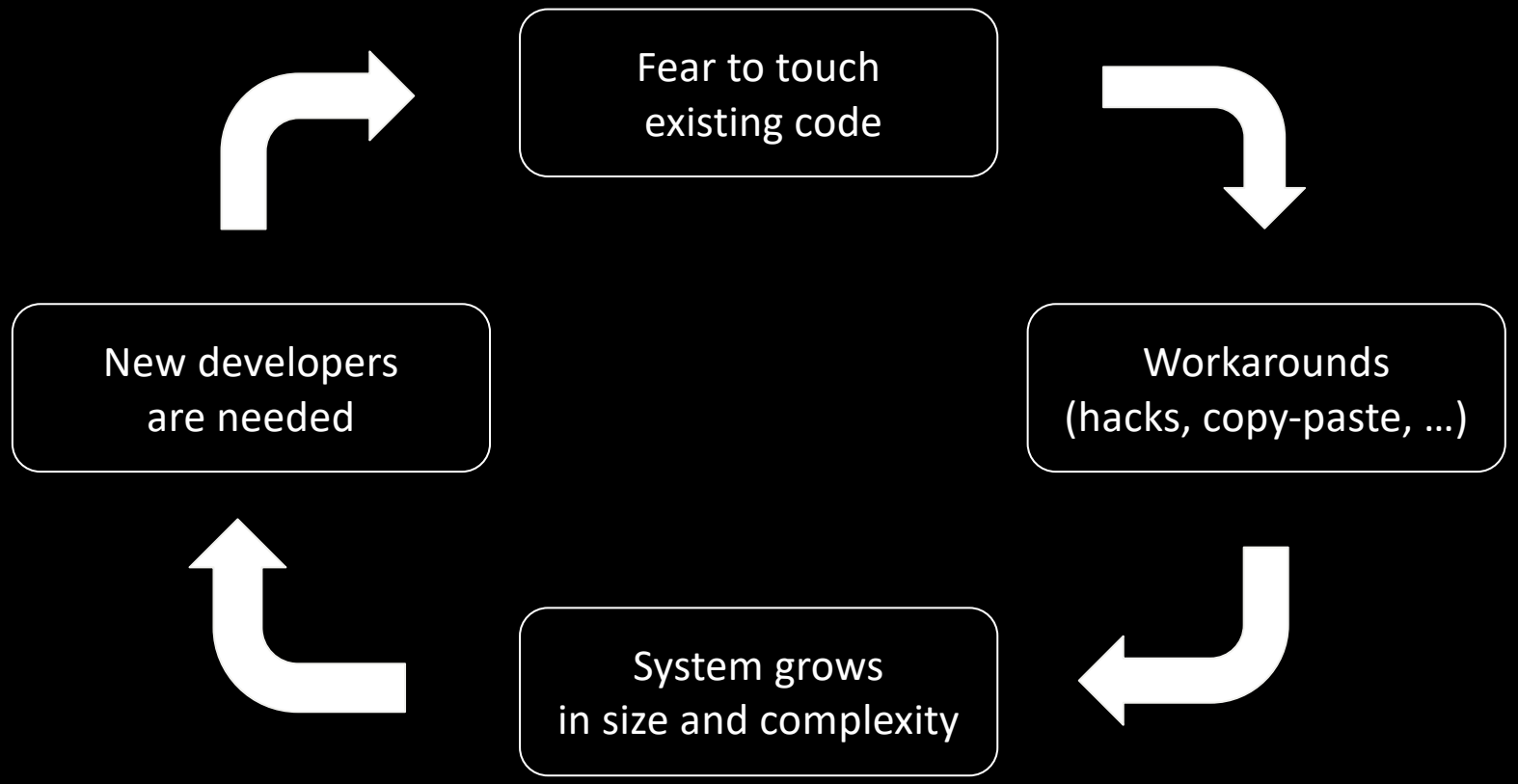
Without explicit focus, non-functionals become an afterthought

Ideal approach: continuous understream to ensure non-functionals remain at desired level

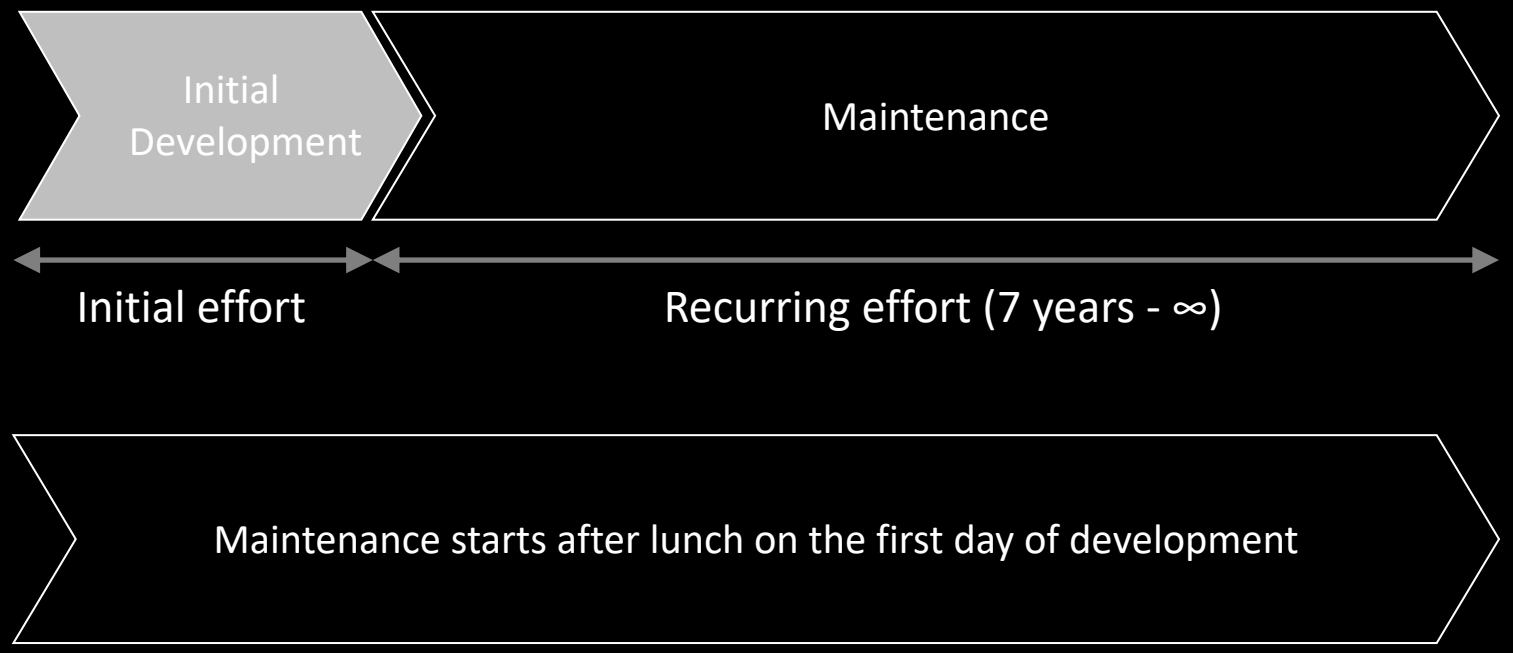
Too much pressure on functionality without focus on non-functionals eventually erodes quality and causes entire iterations dedicated to "restoration", losing the momentum

Software Maintainability

The Vicious Cycle of Unsustainable Software Development



Why maintainable software?



How to measure software quality?

Giving feedback on software products: personal versus tool-based

Personal feedback	Tool-based feedback
Specific for your project	Faster feedback loop
More sensitive to context	Allows for scalability by iteration
Concrete suggestions to improve	More objective

So which one is better?

There is a false dichotomy between full automation and human intervention. Successful quality control combines tool-based measurement with manual review and discussion.

Pitfalls in using measurements

One-track metric

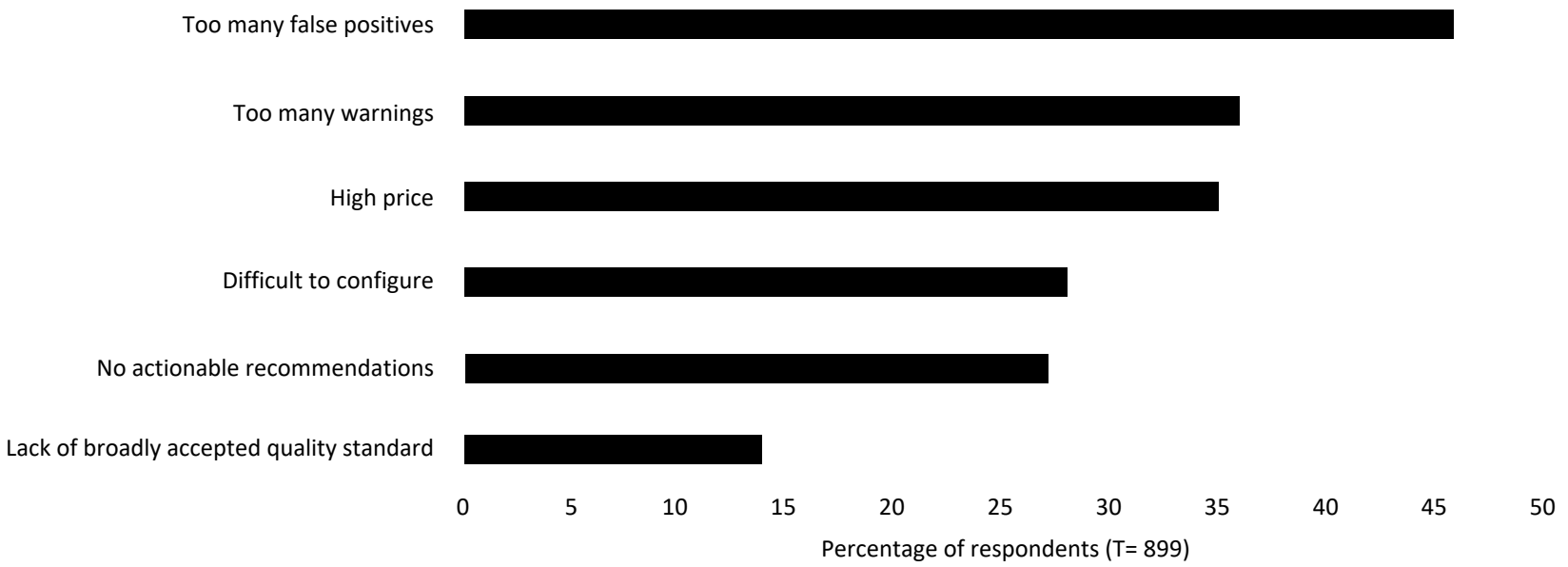
Metrics Galore

Treating the metric

Metric in a bubble

Static code analysis challenges

What are the biggest pitfalls of code quality tools



10 guidelines for future-proof code

Code

- Write small units of code
- Write simple units of code
- Write code once
- Keep unit interfaces small

Architecture

- Separate concerns in modules
- Couple architecture components loosely
- Keep architecture components balanced
- Keep your codebase small

Way of working

- Automate tests
- Write clean code

@EvelynvanKelle

@ykanellopoulos

Use measurable standards: make guidelines quantified and actionable

Code

- Limit units to 15 lines of code
- Limit branch points per unit to 4
- Do not copy code longer than 6 lines
- Limit parameters per unit to 4

Architecture

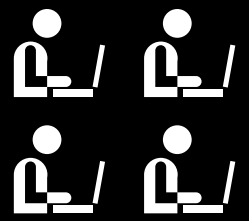
- Avoid modules larger than 400 lines of code
- Hide classes from other components, no cycles
- Aim for 6-12 top-level components
- Keep codebase below 200,000 lines of code

Way of working

- Write automated tests that cover all code
- Stick to the seven "boy scout rules"

@EvelynvanKelle

@ykanellopoulos



How to build effective software teams?

10 best practices for effective software development

- Derive Metrics from Your Measurement Goals
- Make Definition of Done Explicit
- Control Code Versions and Development Branches
- Control Development, Test, Acceptance, and Production Environments
- Automate Tests
- Use Continuous Integration
- Automate Deployment
- Standardize the Development Environment
- Manage Usage of Third-Party Code
- Document Just Enough

@EvelynvanKelle

@ykanellopoulos

QSD

Quality Software Development

Better Code Hub

@EvelynvanKelle

@ykanellopoulos

Software Improvement Group

Evelyn van Kelle
e.vankelle@sig.eu
@EvelynvanKelle

Yiannis Kanellopoulos
y.kanellopoulos@sig.eu
@ykanellopoulos