Tampere University

COMP.SE.140
CI, CD, DevOps

Kari Systä, 31.10.2023

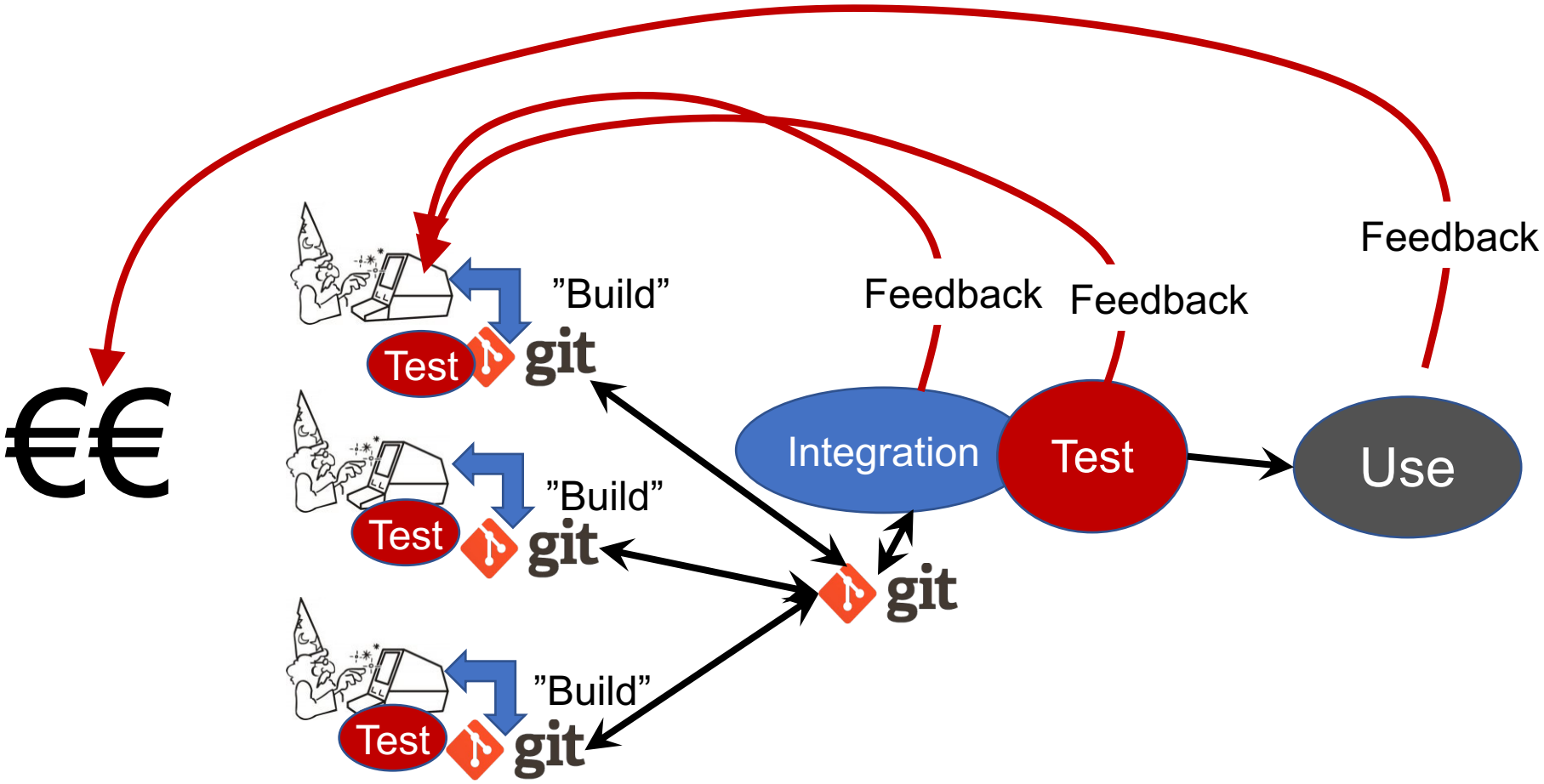
Tampereen yliopisto
Tampere University

What is DevOps (there are several definitions)

- Lucy Lwakatare:
  - DevOps is a concept that embodies a **cultural and mindset change** that is substantiated with a **set of practices** to encourage **cross-disciplinary collaboration between software development and IT operations** within a software company. The main purpose for the collaboration is to enable the **fast release of quality software changes** while simultaneously **operating resilient systems**.
  - From a **socio-technica perspective**, DevOps practices are focused on the **automation practices** o nfrastructure management, specifically a on management and monitoring.

Continuous Delivery

Continuous deployment



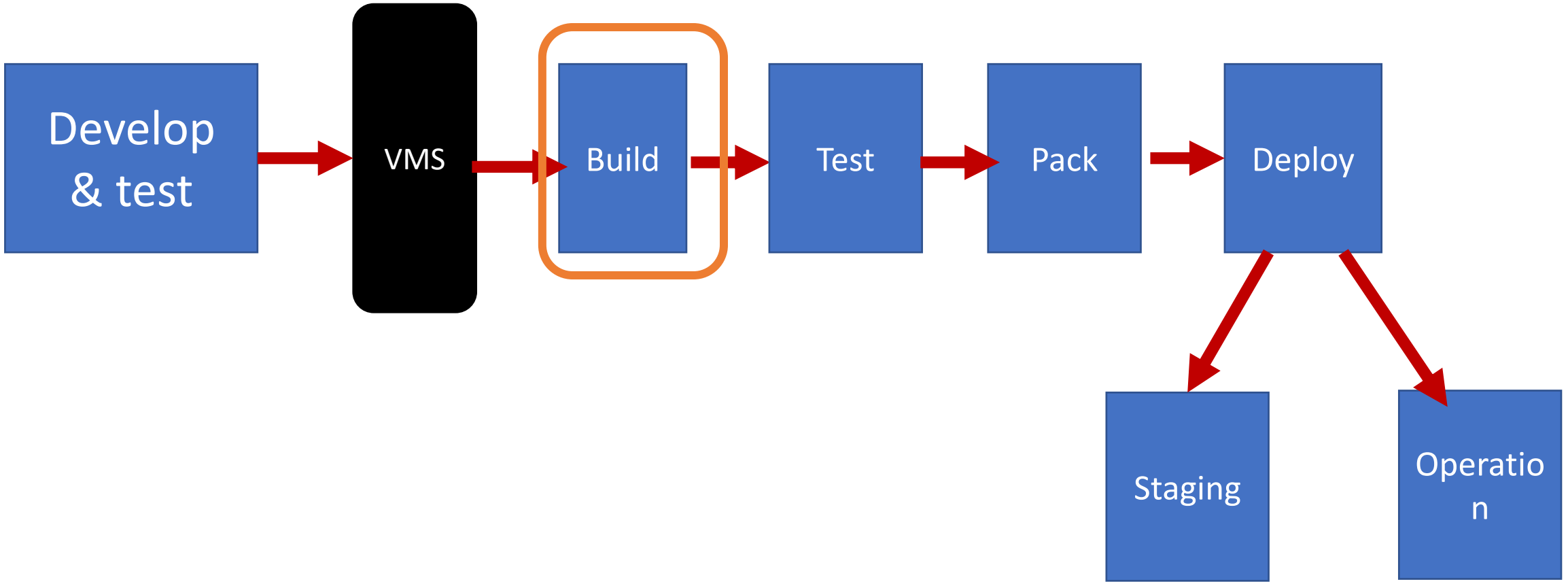
Tampereen yliopisto
Tampere University

Perceived benefits

- **Improved delivery speed** of software changes Improved speed in the development and deployment of software changes to production environment.
- **Improved productivity in operations** work. Decreased communication problems, bureaucracy, waiting overhead due to removal of manual deployment hand-offs and organisational boundaries; Lowered human error in deployment due to automation and making explicit knowledge of operation-related tasks to software development
- **Improvements in quality**. Increased confidence in deployments and reduction of deployment risk and stress; Improved code quality; Improved product value to customer resulting from production feedback about users and usage.
- **Improvements in organisational-wide culture** and mind-set. Enrichment and wider dissemination of DevOps in the company through discussions and dedicated training groups 'communities of practice'

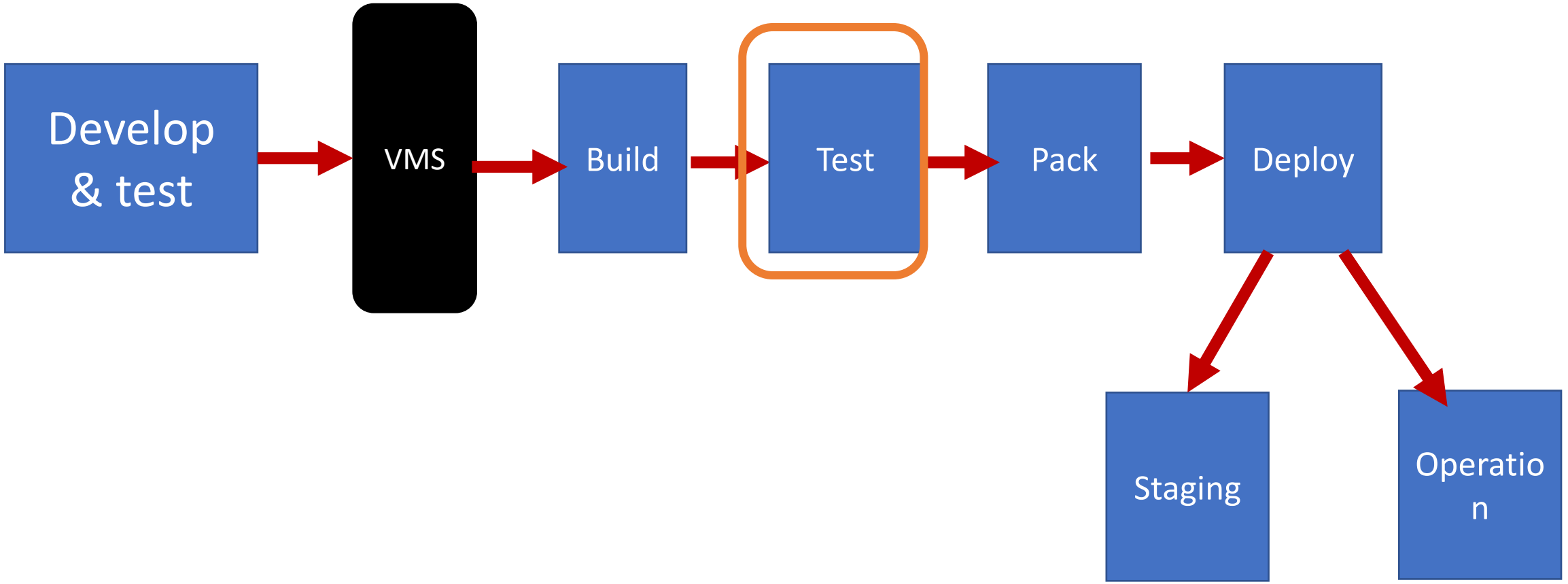
Tampereen yliopisto
Tampere University

Simplified pipeline



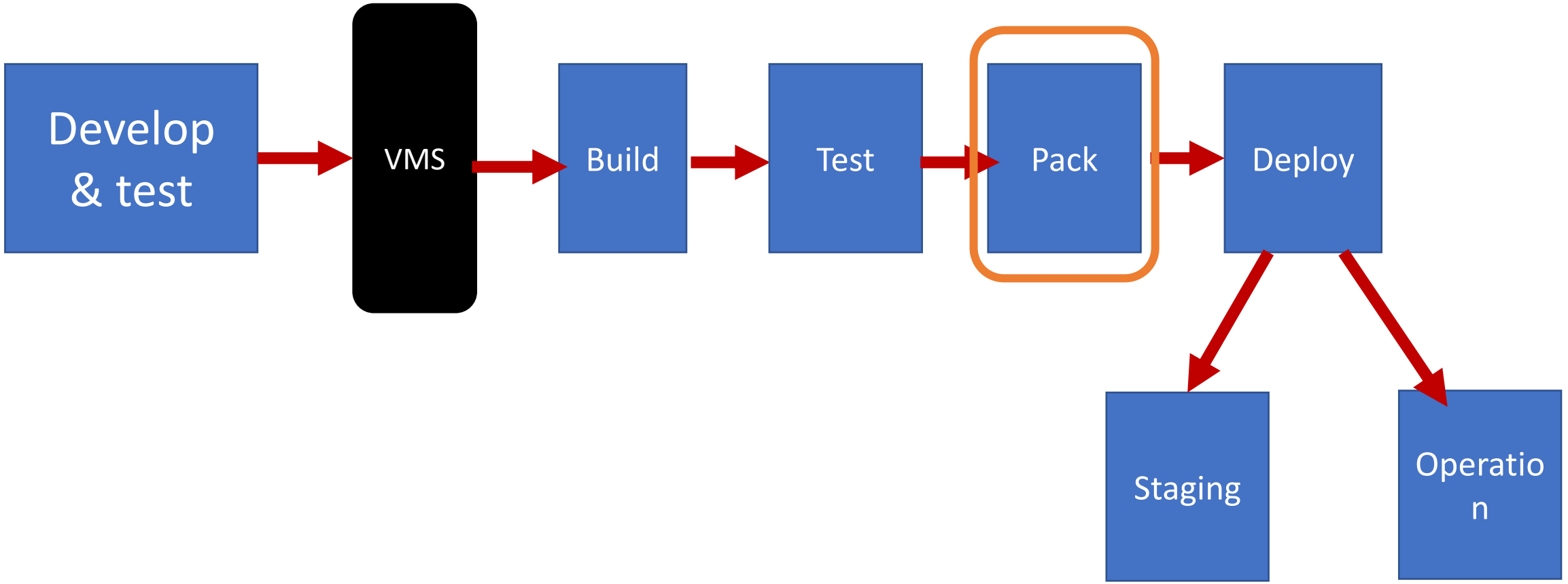
Tampereen yliopisto
Tampere University

Simplified pipeline



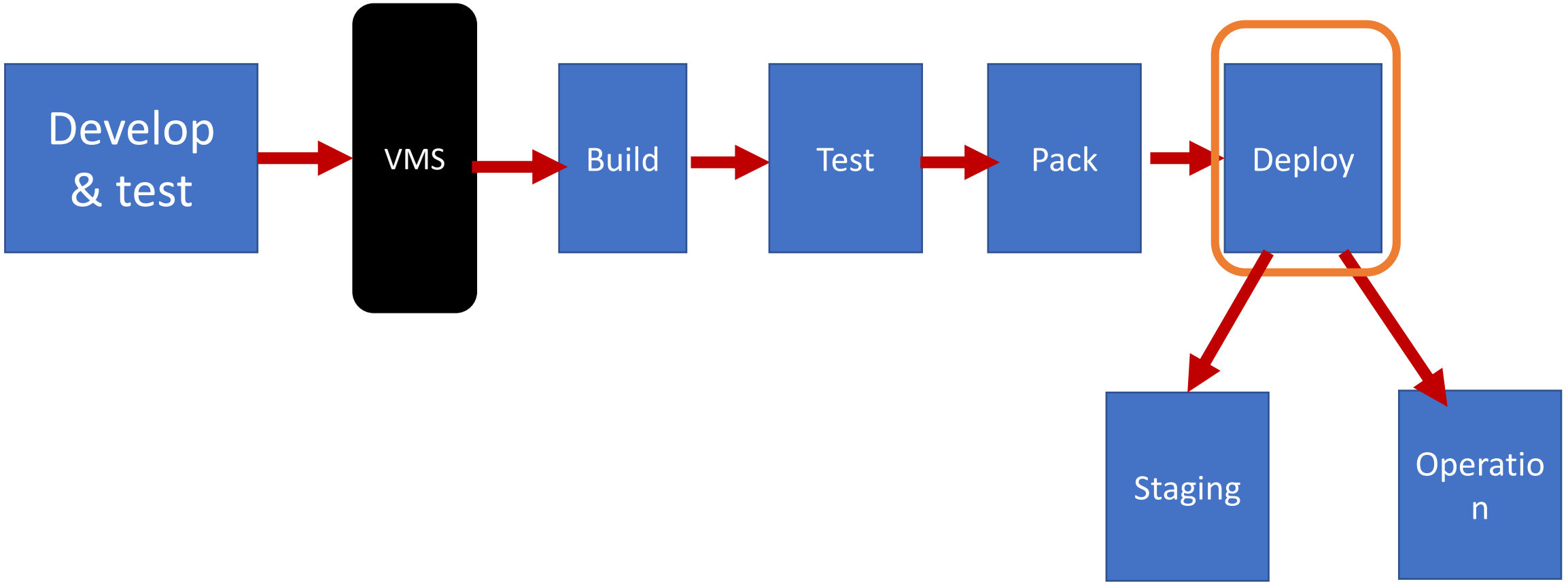
Tampereen yliopisto
Tampere University

Simplified pipeline



Tampereen yliopisto
Tampere University

Simplified pipeline



Tampereen yliopisto
Tampere University

Testing

- Automate, automate, automate
- Know any tools?

Business

AUTOMATED (functional acceptance tests)	MANUAL Showcase Usability testing Exploratory testing
Unit tests Integration tests System tests AUTOMATED	Nonfunctonal acceptance tests MANUAL/AUTOM.

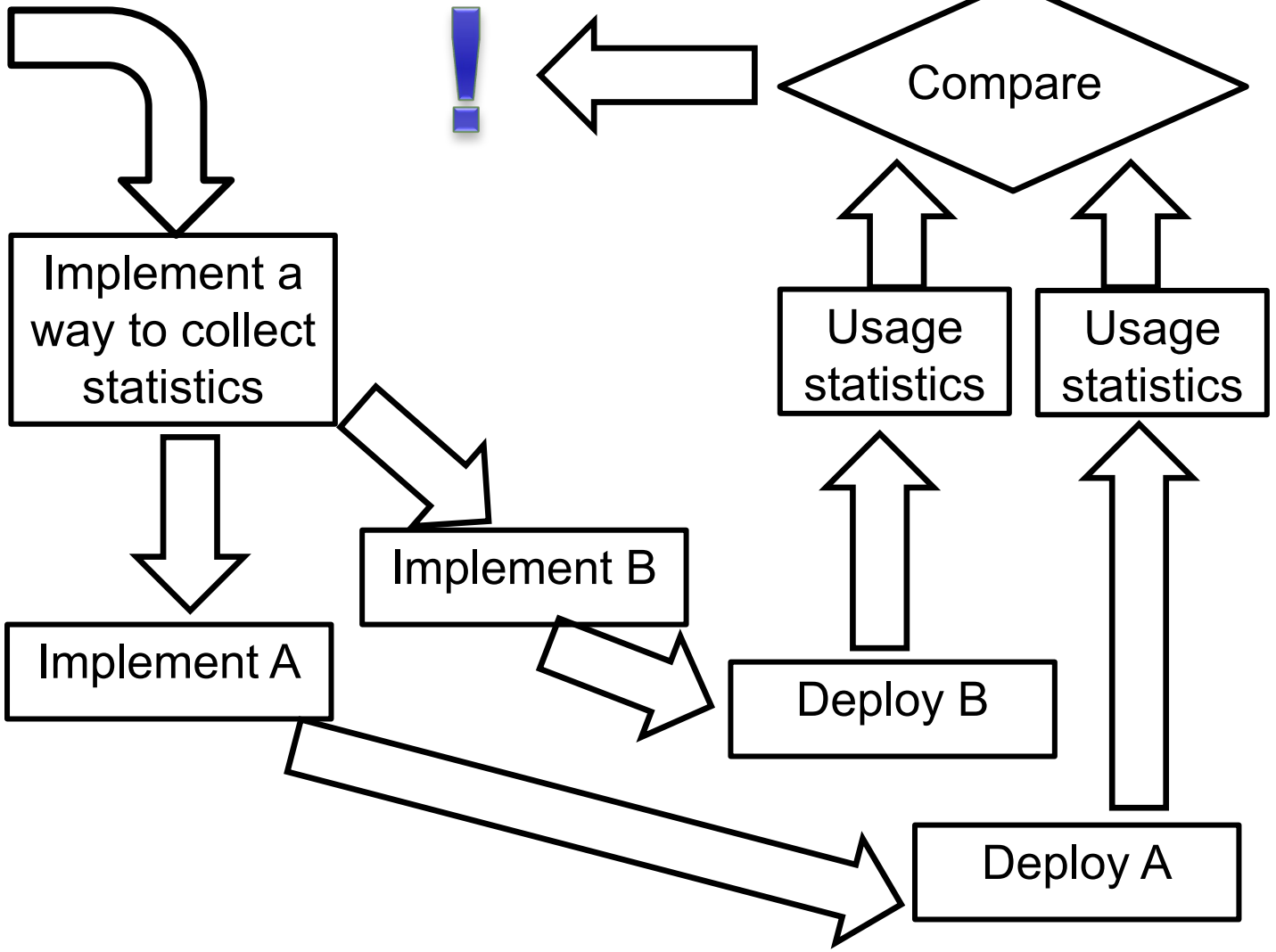
Support coding

Critique project

Technology

A/B Testing

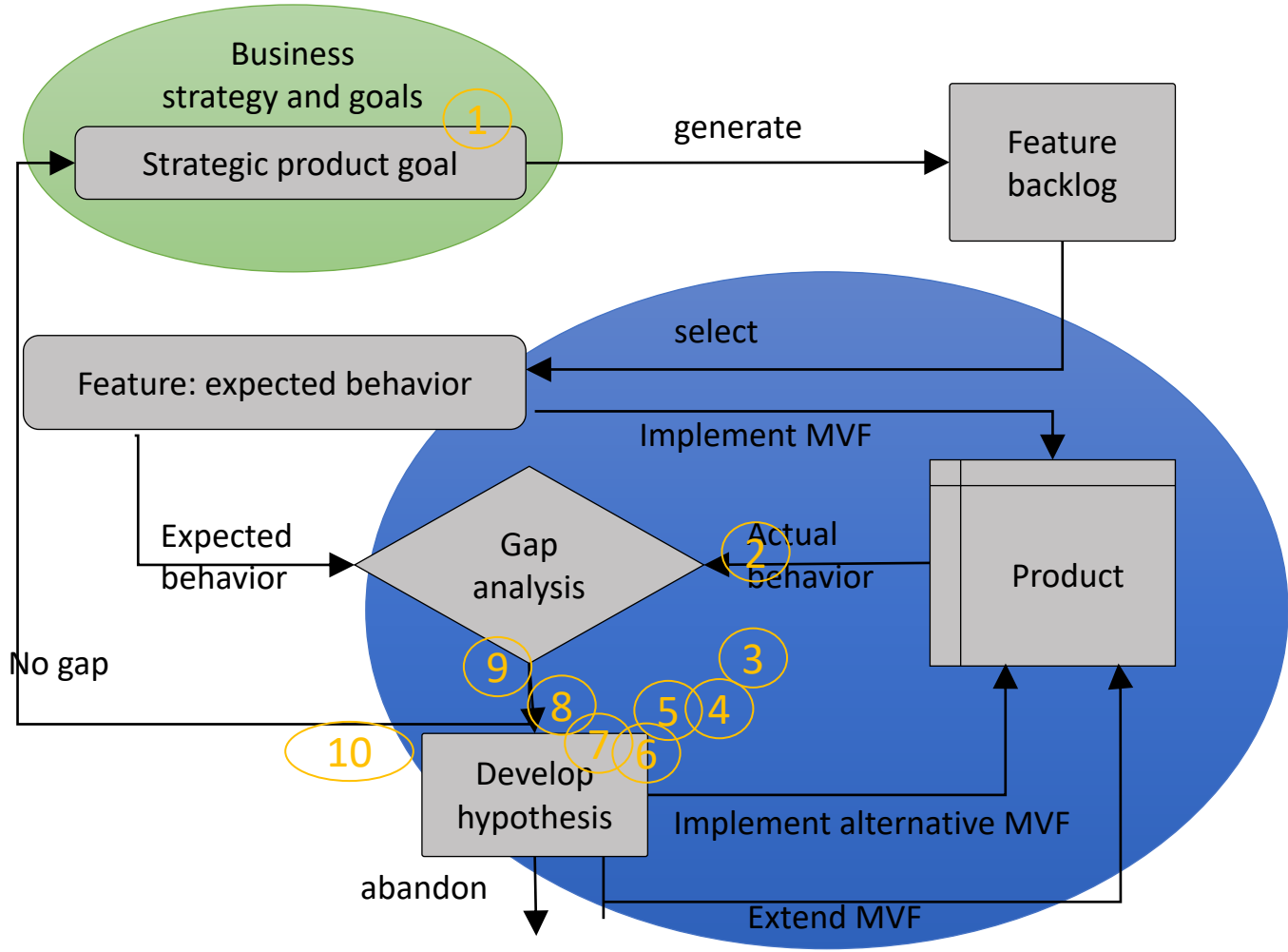
Should our project have A or B?



Tampereen yliopisto
Tampere University

Data-driven software development

1. Planning of the data collection
2. Deployment of data collection
3. Monitoring of the applications
4. Picking up the relevant data
5. Pre-processing – filtering and formatting – the data
6. Sending and/or saving the data
7. Cleaning and unification of the data
8. Storing the data
9. Visualizations and analysis
10. Decision making



Tampereen yliopisto
Tampere University

DevOps and critical systems

Tampereen yliopisto
Tampere University

Some claims from

https://www.beyondtrust.com/blog/entry/devops-security-best-practices

- **DevOps' focus on speed often leaves security teams flat-footed and reactive**
- **Cultural resistance to security:** There's a widespread perception that introducing security will slow or derail the development process.
- **DevOps and cloud environments:** The typical DevOps environment relies on cloud deployments, thereby sharing many cloud security considerations. DevOps teams often leverage new, open-source or immature tools
- **Containers and other tools carry their own risks: I**s container a powerful black box?
- **Unmanaged secrets and poor privileged access controls open dangerous backdoors**

Tampereen yliopisto
Tampere University

DevSecOps

- DevSecOps is seen as a necessary expansion to DevOps, where the purpose is to integrate security controls and processes into the DevOps software development cycle and that it is done by promoting the collaboration between security teams, development teams and operations teams.

- H. Myrbakken, R. Colomo-Palacios, DevSecOps: A Multivocal Literature Review., in: A. Mas, A. Mesquida, R. O'Connor, T. Rout, A. Dorling (eds) Software Process Improvement and Capability Determination. SPICE 2017. Communications in Computer and Information Science, vol 770. Springer, Cham, 2017, pp. 17-20. https://doi.org/10.1007/978-3-319-67383-7_2.

Tampereen yliopisto
Tampere University

Principles

- Culture: DevSecOps means to include collaboration with the security team as well as promote a culture where operations and development also work on integrating security in their work.
- Automation: DevSecOps promotes a focus on automating security, to be able to keep up with the speed and scale achieved by DevOps. The aim should be 100% automation of security controls, where the controls can be deployed and managed without manual interference.
- Measurement: DevSecOps promotes the use and development of metrics that track threats and vulnerabilities throughout the software development process
- Sharing: DevSecOps promotes the inclusion of the security team in the sharing promoted in a DevOps environment.
- Shift security to the left: This means that security teams are involved from the very first planning step and is part of planning every iteration of the development cycle

Tampereen yliopisto
Tampere University

DevSecOps

the mindset that "*everyone is responsible for security*"

Manifesto

- **Leaning in** over Always Saying "No"
- **Data & Security Science** over Fear, Uncertainty and Doubt
- **Open Contribution & Collaboration** over Security-Only Requirements
- **Consumable Security Services with APIs** over Mandated Security Controls & Paperwork
- **Business Driven Security Scores** over Rubber Stamp Security
- **Red & Blue Team Exploit Testing** over Relying on Scans & Theoretical Vulnerabilities
- **24x7 Proactive Security Monitoring** over Reacting after being Informed of an Incident
- **Shared Threat Intelligence** over Keeping Info to Ourselves
- **Compliance Operations** over Clipboards & Checklists

Tampereen yliopisto
Tampere University

Practices

- Threat modeling and risk assessments:
- Continuous testing:
- Monitoring and logging:
- Security as code:
- Red-Team and security drills:

Tampereen yliopisto
Tampere University

Some practical tips from

https://www.redhat.com/en/topics/devops/what-is-devsecops

- **Standardize and automate the environment**. Each service should have the least privilege possible to minimize unauthorized connections and access.
- **Centralize user identity and access control capabilities**. Tight access control and centralized authentication mechanisms are essential for securing microservices, since authentication is initiated at multiple points.
- **Isolate containers running microservices from each other and the network.** This includes both in transit and at rest data, since both can represent high-value targets for attackers.
- **Encrypt data between apps and services**. A container orchestration platform with integrated security features helps minimize the chance of unauthorized access.
- **Introduce secure API gateways**. Secure APIs increase authorization and routing visibility. By reducing exposed APIs, organizations can reduce surfaces of attacks.

Tampereen yliopisto
Tampere University

Some practical tips from

https://www.redhat.com/en/topics/devops/what-is-devsecops

- **Integrate security scanners for containers**. This should be part of the process for adding containers to the registry.
- **Automate security testing in the CI process.** This includes running security static analysis tools as part of builds, as well as scanning any pre-built container images for known security vulnerabilities as they are pulled into the build pipeline.
- **Add automated tests for security capabilities into the acceptance test process.** Automate input validation tests, as well as verification authentication and authorization features.
- **Automate security updates, such as patches for known vulnerabilities.** Do this via the DevOps pipeline. It should eliminate the need for admins to log into production systems, while creating a documented and traceable change log.
- **Automate system and service configuration management capabilities.** This allows for compliance with security policies and the elimination of manual errors. Audit and remediation should be automated as well.

Regulated software development

Example: medical systems

Examples of requirements/standards

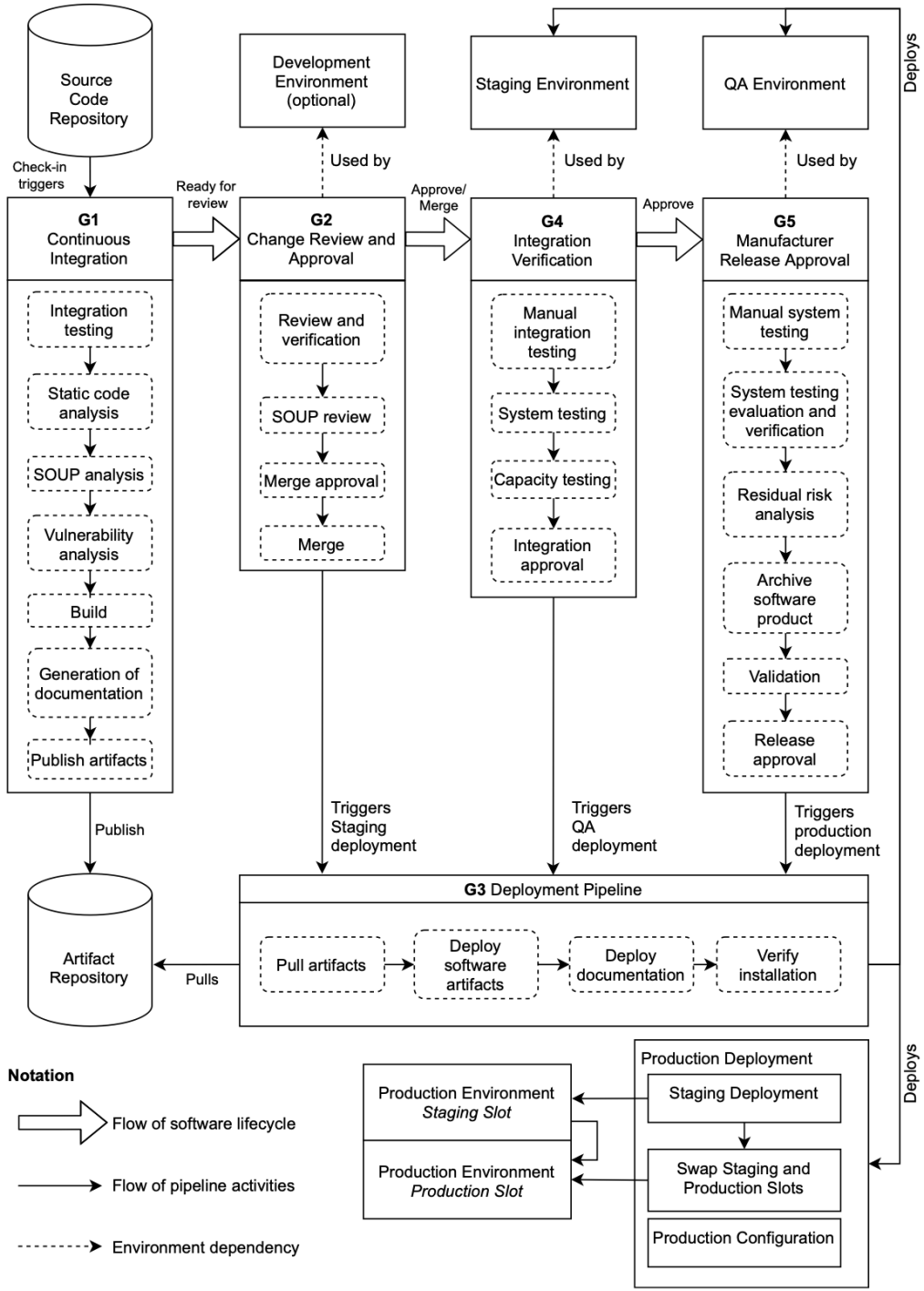
- general requirements for health software product safety (IEC 82304-1),
- software life cycle process (IEC 62304),
- risk management process (ISO 14971),
- usability engineering (IEC 62366-1),
- quality management system requirements (ISO 13485), and
- security activities in the product life cycle (IEC 81001-5-1)

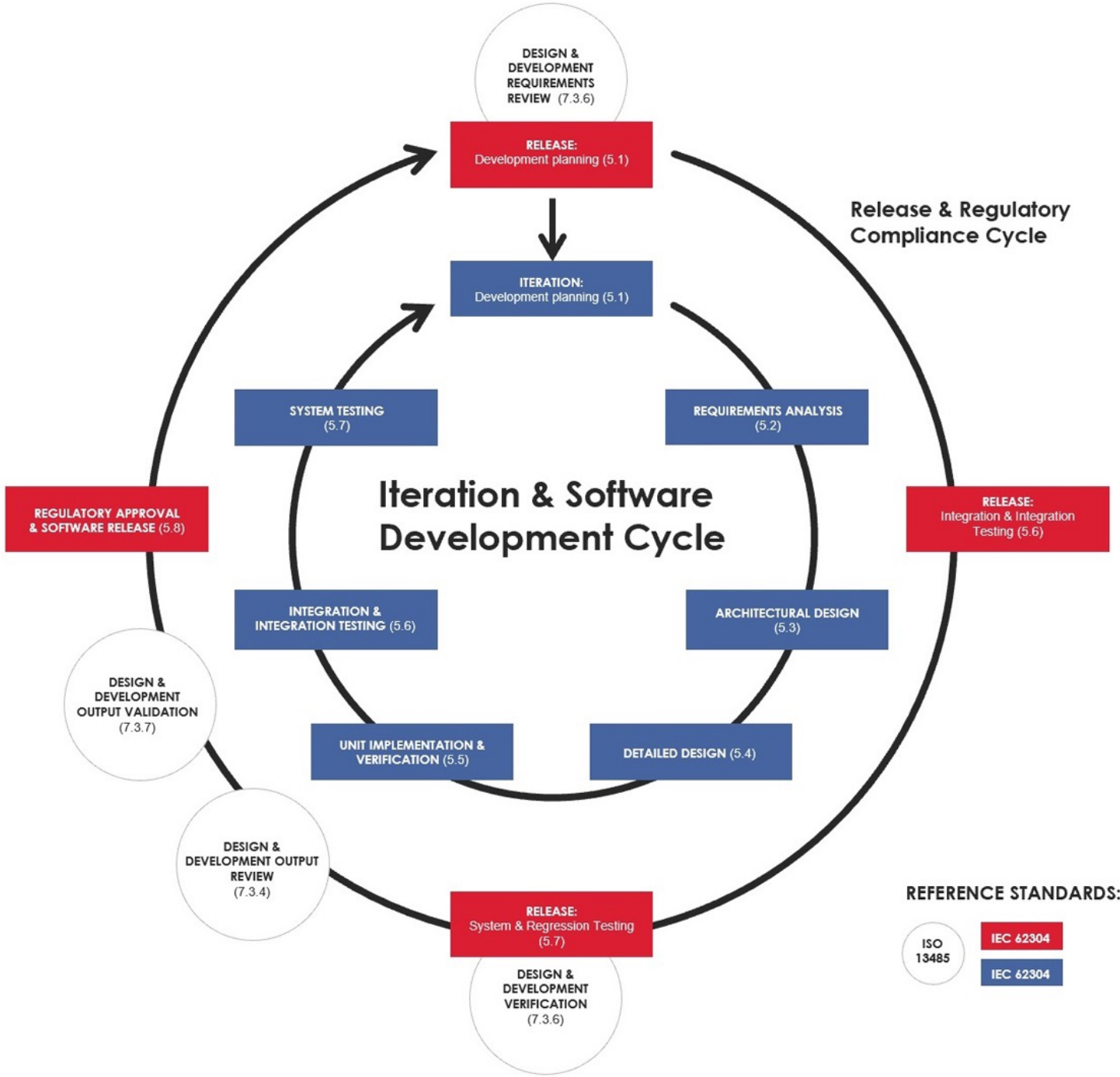
These slow down process lead time

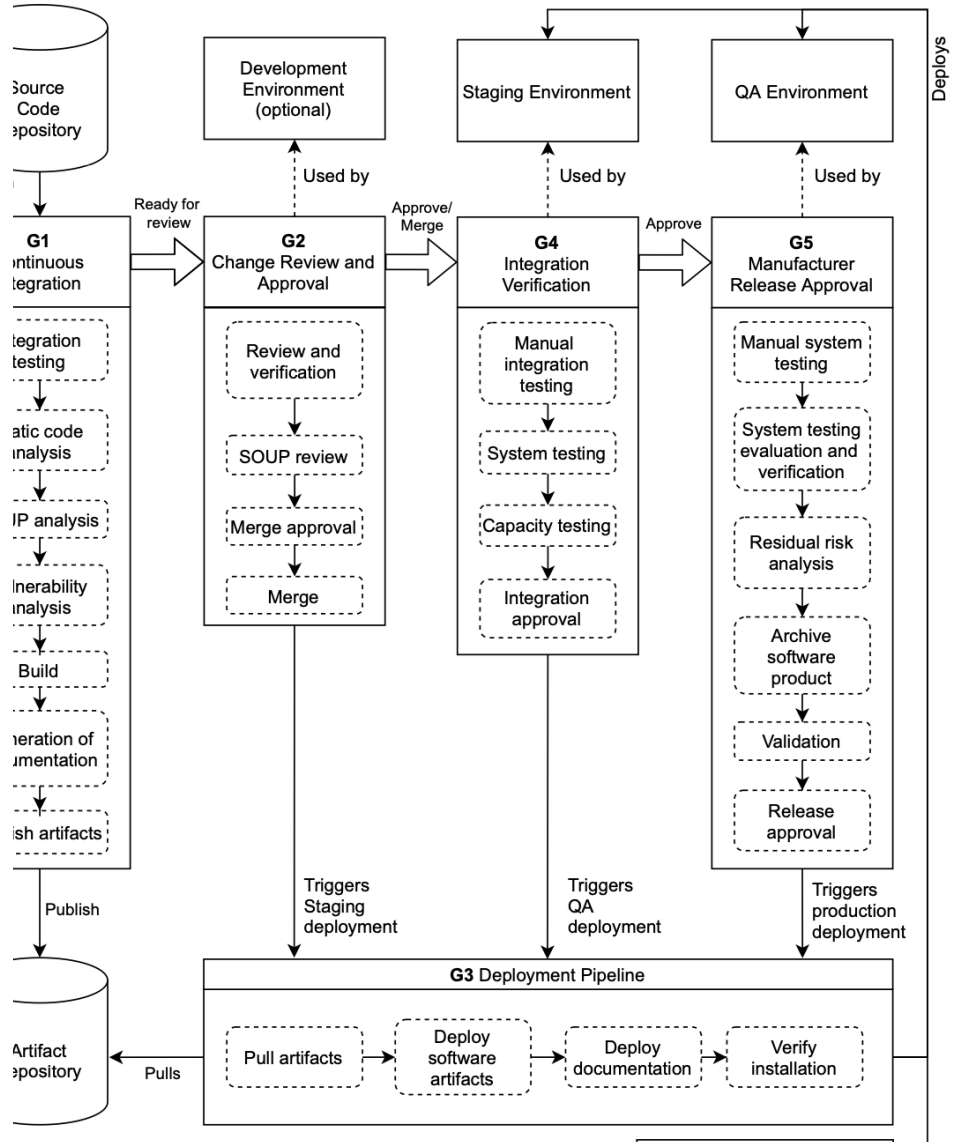
DevOps is about making it faster

Tampere University

Toivakka, H., Granlund, T., Poranen, T., Zhang, Z. (2021). Towards RegOps: A DevOps Pipeline for Medical Device Software. Proc od Product-Focused Software Process Improvement. PROFES 2021. Lecture Notes in Computer Science(), vol 13126. Springer, Cham. https://doi.org/10.1007/978-3-030-91452-3_20







RegOps – diving into the dilemma of agile software development in regulated industry https://www.solita.fi/blogs/regops-diving-into-the-dilemma-of-agile-software-development-in-regulated-industry/

Tampere University

Material

- Laukkarinen, T., Kuusinen, K., Mikkonen, T.: DevOps in Regulated Software Development: Case Medical Devices. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE-NIER), pp. 15-18. IEEE (2017)
- Laukkarinen, T., Kuusinen, K., Mikkonen, T.: Regulated software meets DevOps. Information and Software Technology, vol. 97, pp. 176—178. (2018)
- Toivakka, H., Granlund, T., Poranen, T., Zhang, Z. (2021). Towards RegOps: A DevOps Pipeline for Medical Device Software. In: Ardito, L., Jedlitschka, A., Morisio, M., Torchiano, M. (eds) Product-Focused Software Process Improvement. PROFES 2021. Lecture Notes in Computer Science(), vol 13126. Springer, Cham. https://doi.org/10.1007/978-3-030-91452-3_20

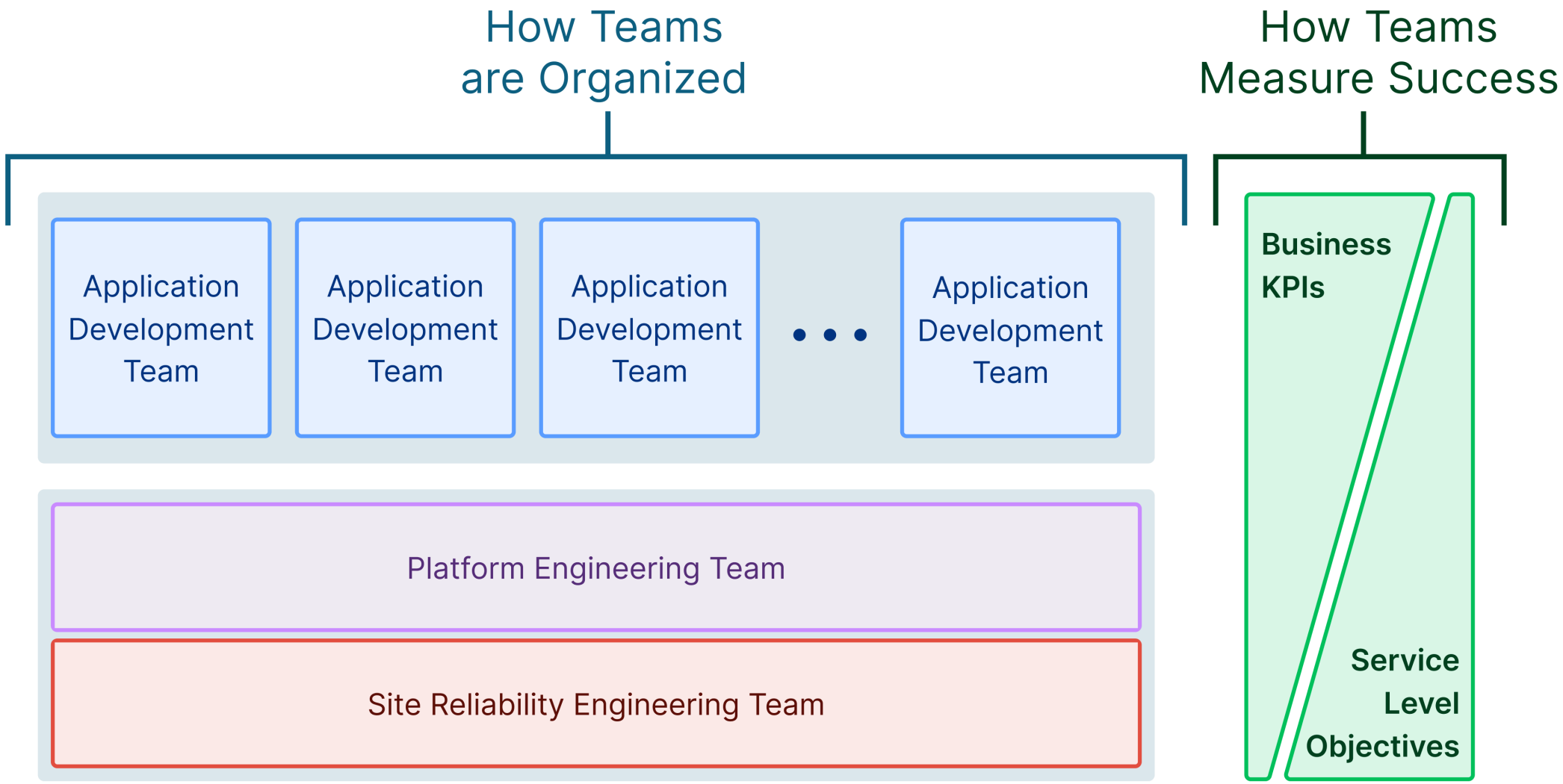
Site reliability engineer

https://aws.amazon.com/what-is/sre/

- Application monitoring
  - service-level agreements (SLAs), service-level indicators (SLIs), and service-level objectives (SLOs)
- Gradual change implementation
  - SRE practices encourage the release of frequent but small changes to maintain system reliability
- Automation for reliability improvement
  - policies and processes that embed reliability principles in every step of the delivery pipeline
- *SRE is the practical implementation of DevOps.*

Tampere University

https://www.getambassador.io/resources/rise-of-cloud-native-engineering-organizations



Example
DORA metrics for DevOps

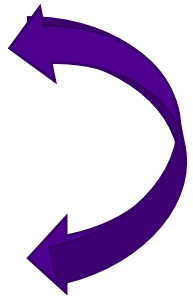
DevOps

- A modern development paradigm for cloud applications

- Cornerstones
  - Continuous deployment: new features are delivered to end-users as soon as possible
  - Close collaboration between development and operations – often even the same team is responsible
  - Empasis on quality – also to cope with the concerns raised by the two previous
  - Extensive use of automation

DORA metric for DevOps

- **Deployment Frequency**
- **Lead Time for Changes**
- **Mean Time to Recovery**
- **Change Failure Rate**



X = 1/ Y ?

DORA metric for DevOps

Metric	Explanation
Deployment Frequency	Refers to the frequency of successful software releases to production.
Lead Time for Changes	Captures the time between a code change commit and its deployable state.
Mean Time to Recovery	Measures the time between an interruption due to deployment or system failure and full recovery.
Change Failure Rate	Indicates how often a team's changes or hotfixes lead to failures after the code has been deployed.

xhttps://www.leanix.net/en/wiki/vsm/dora-metrics