Tampere University

Lecture 6: continuous deployment – part 1

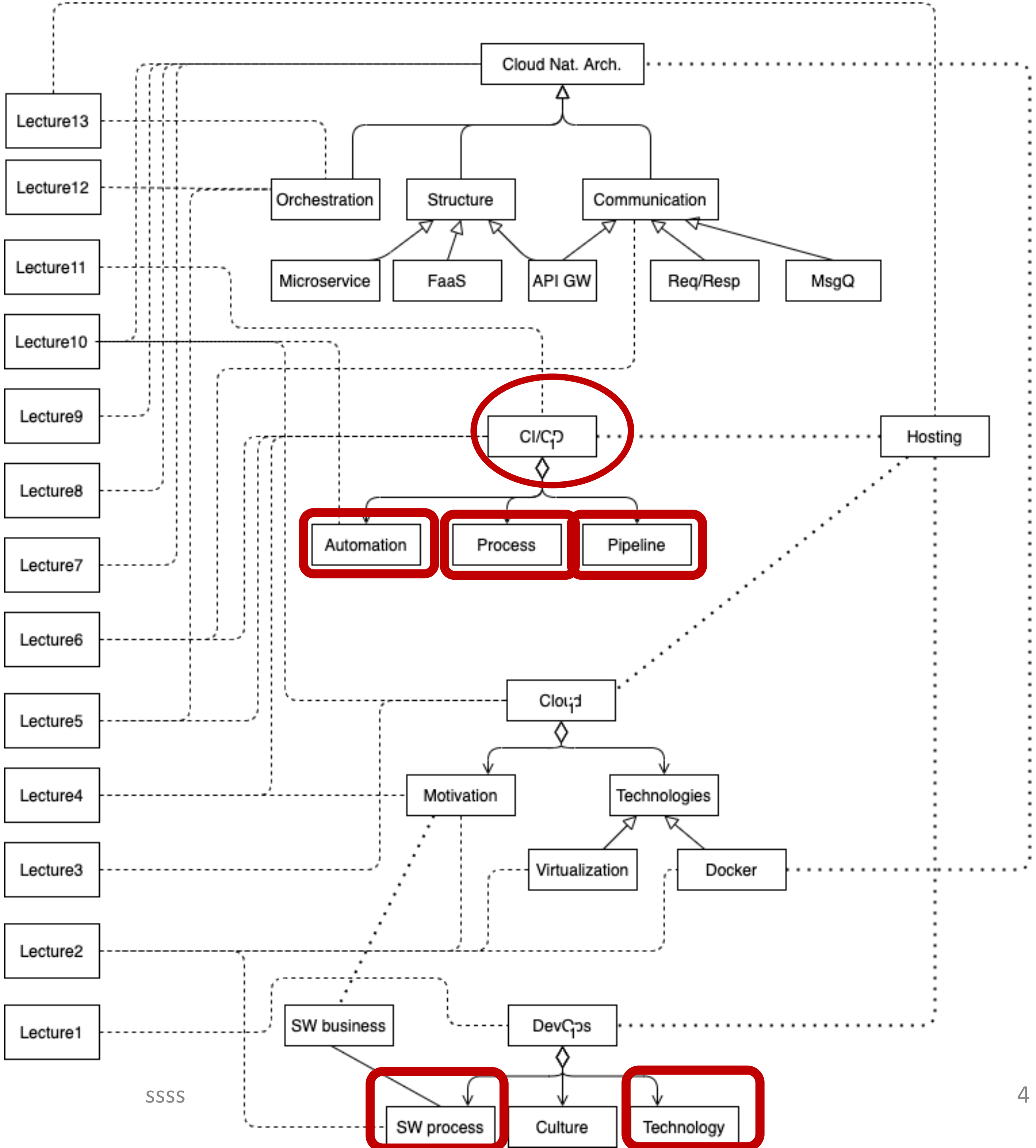
Continuous Delivery and Deployment

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-technical perspective**, DevOps practices are focused on the **automation practices** o nfrastructure management, specifically a on management and monitoring.

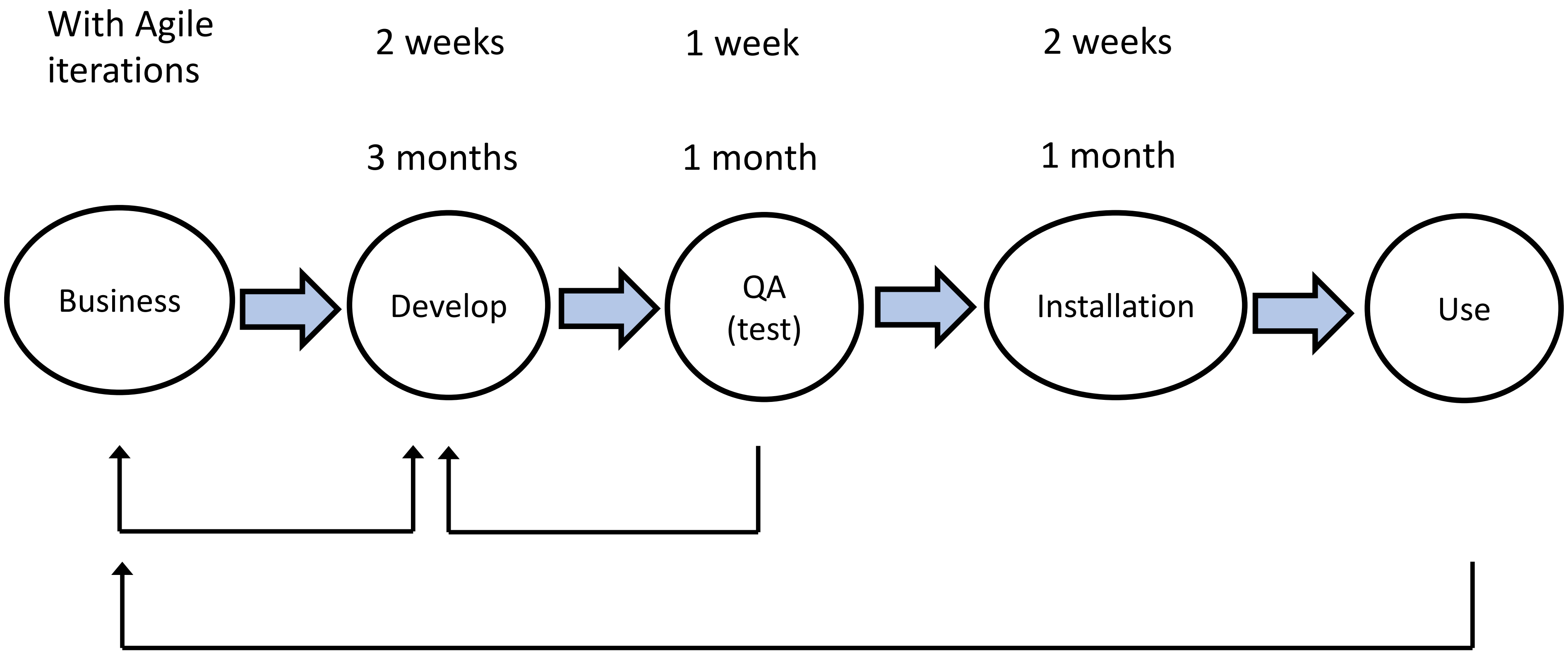
Continuous Delivery

Relation to our content

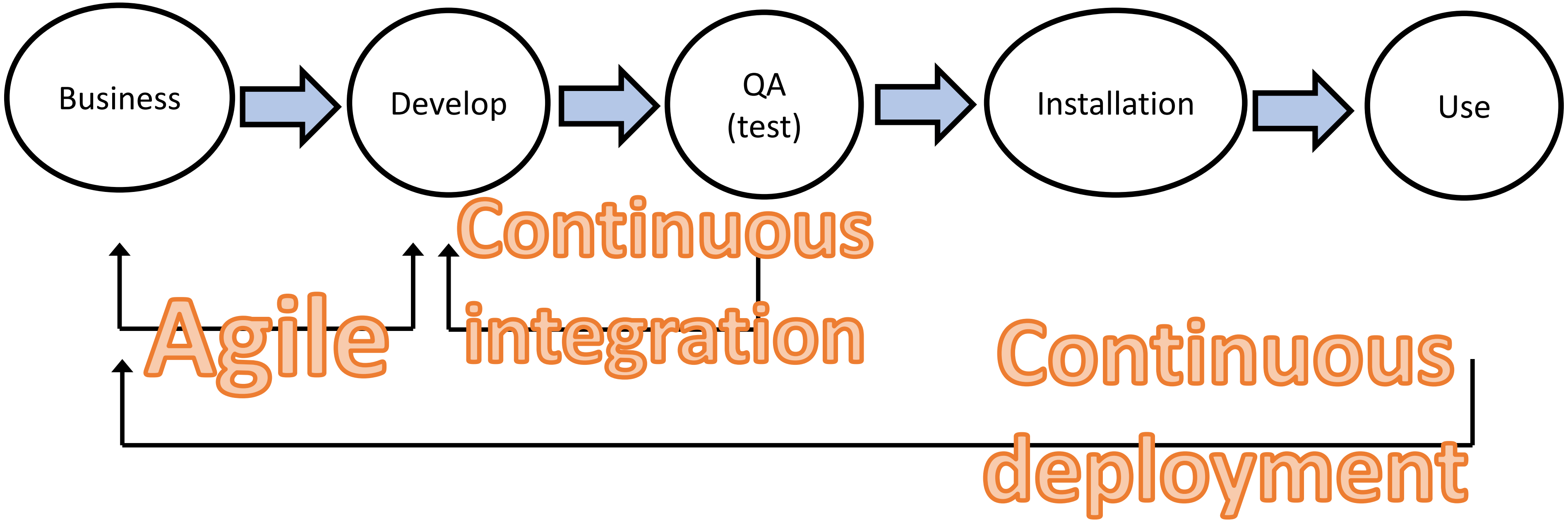


Feedback in traditional development

(Case: Internet-based service; based on slide by Antti Tirilä)

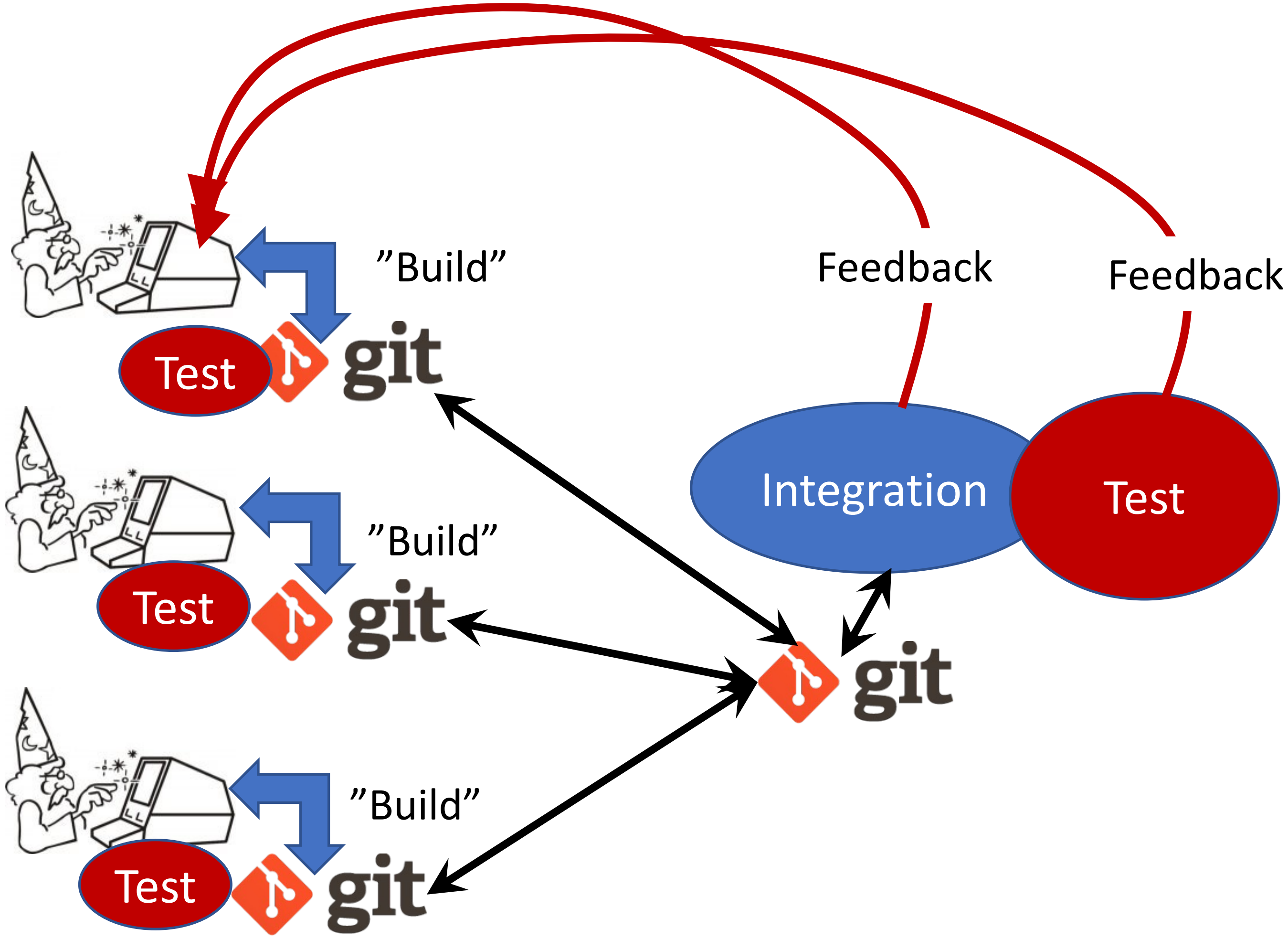


Feedback in traditional development (Case: Internet-based service; based on slide by Antti Tirilä)

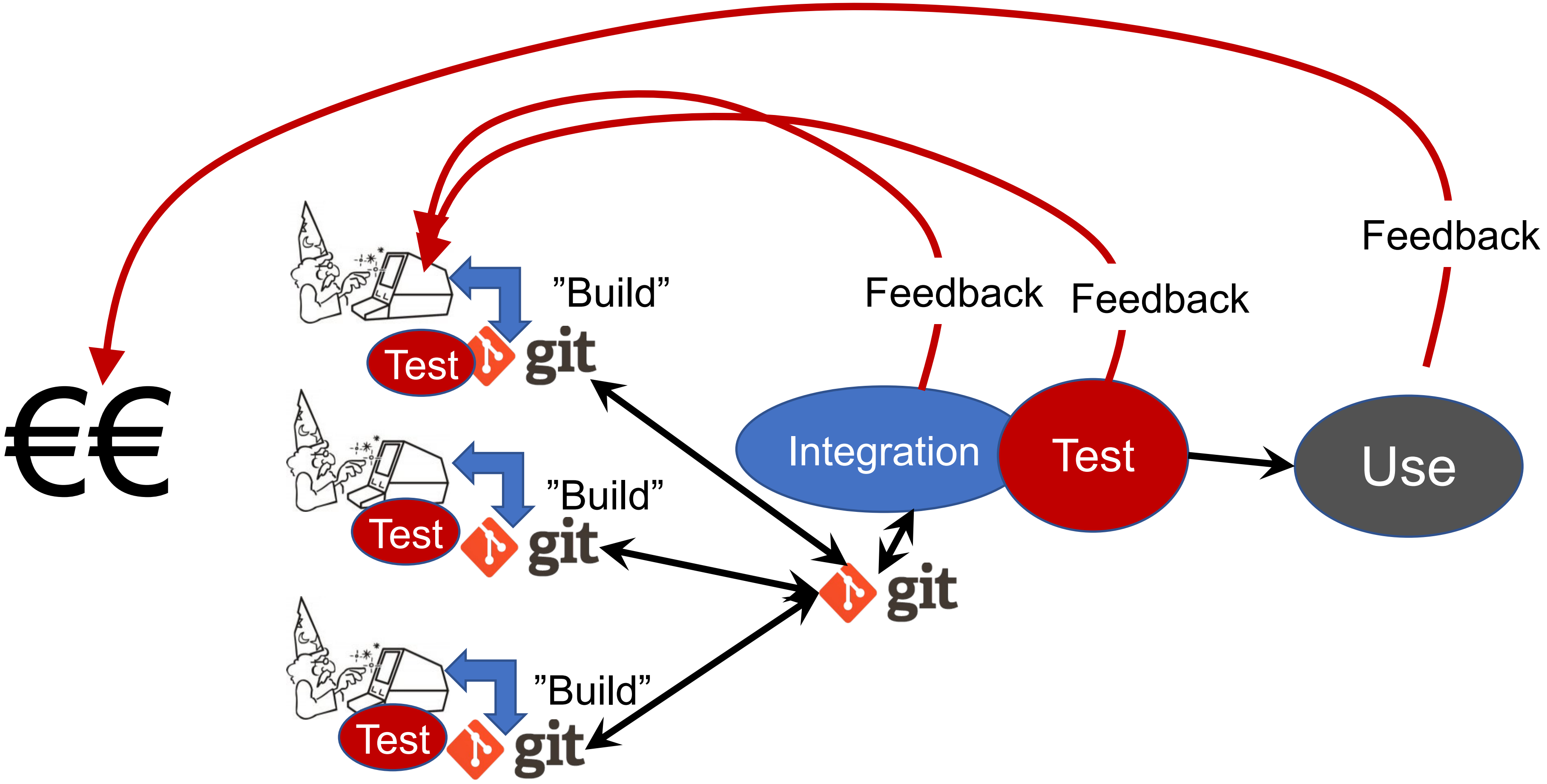


Tampereen yliopisto
Tampere University

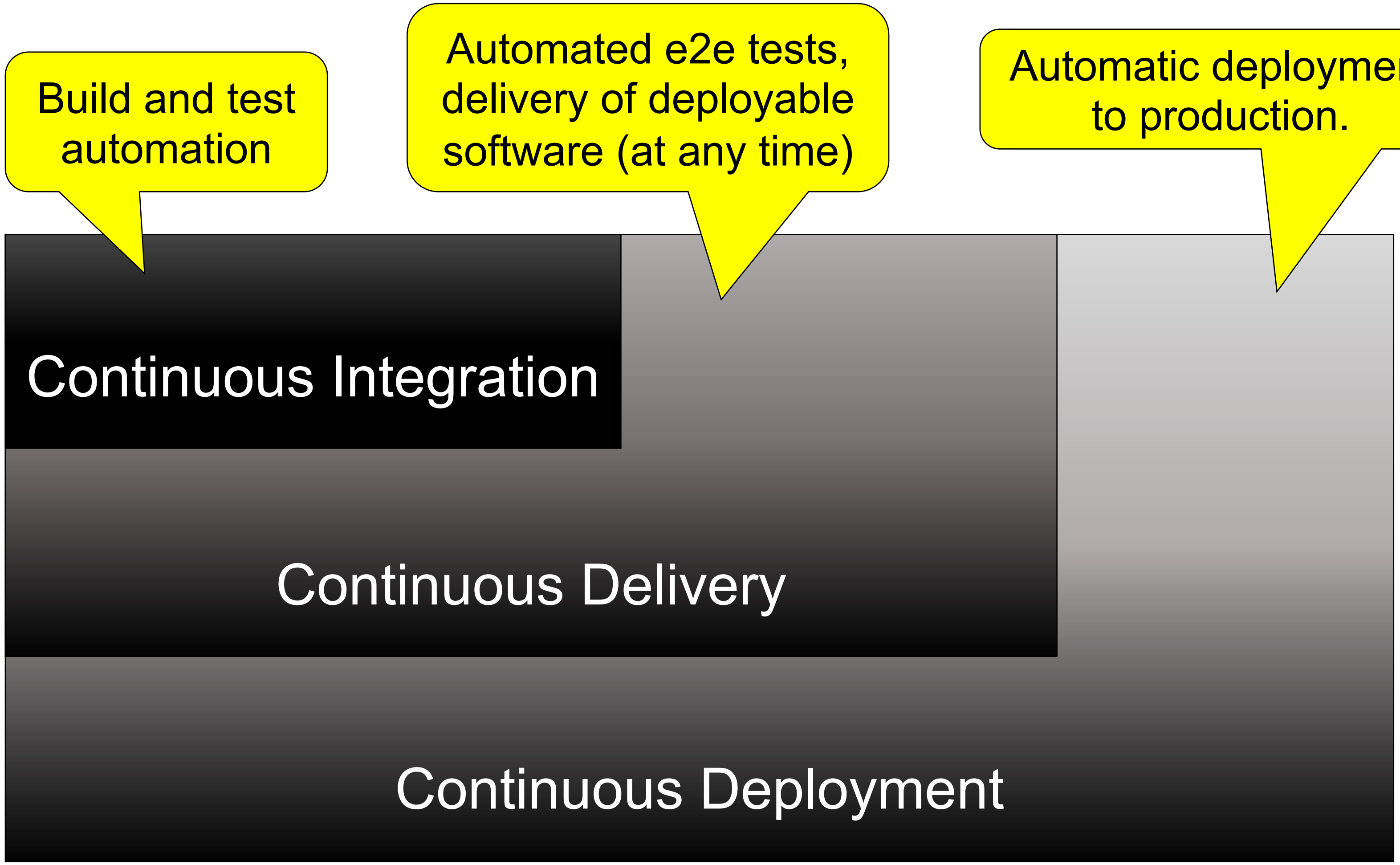
Continuous integration



Continuous deployment



Continuous X



Build and test automation

Automated e2e tests, delivery of deployable software (at any time)

Automatic deployment to production.

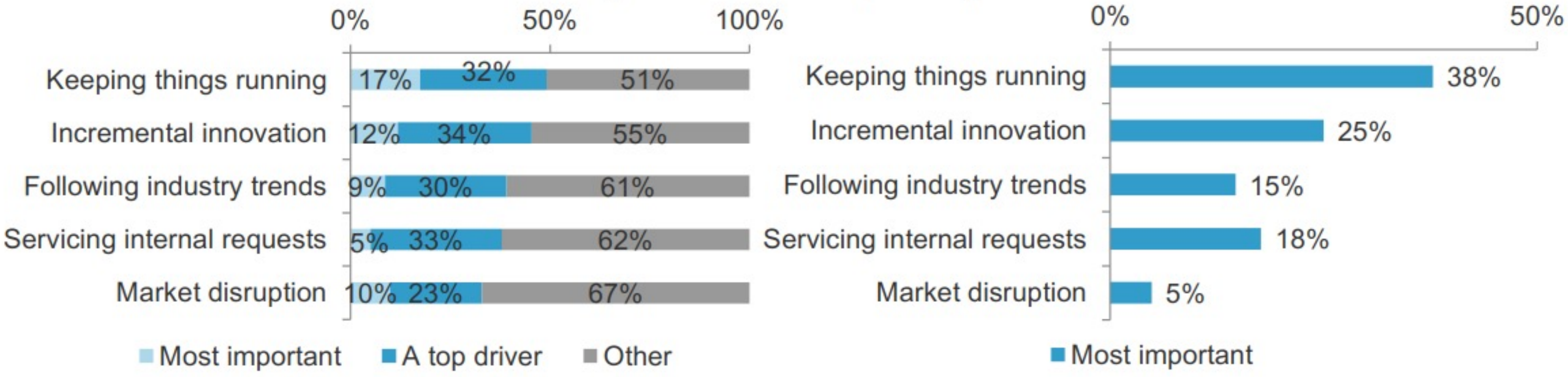
Continuous Integration

Continuous Delivery

Continuous Deployment

From Forrester report: Continuous Delivery: A Maturity Assessment Model: Building Competitive Advantage With Software Through A Continuous Delivery Process, 2013

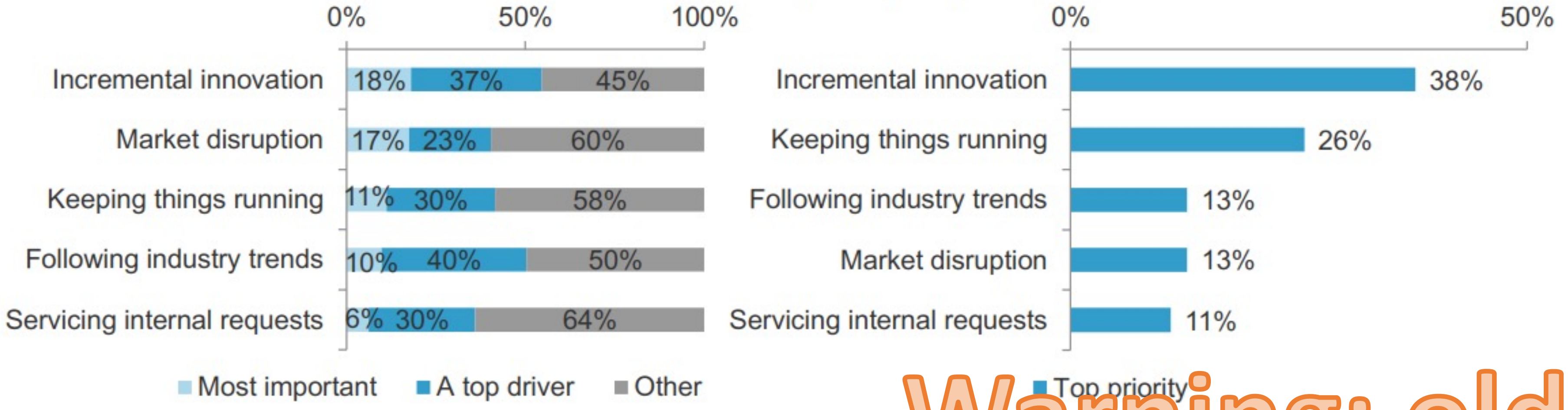
"Please rank the importance of the business drivers behind your software development investments within your business area (current)."



Base: 161 business decision-makers

Base: 164 IT executives and managers

"Please rank the importance of the business drivers behind your software development investments within your business area (in two years)."



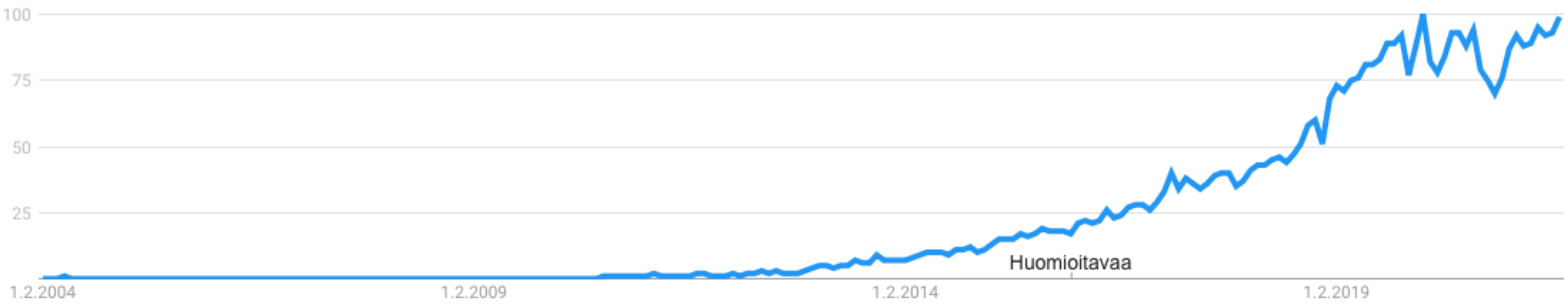
Warning: old stuff

Base: 161 business decision-makers

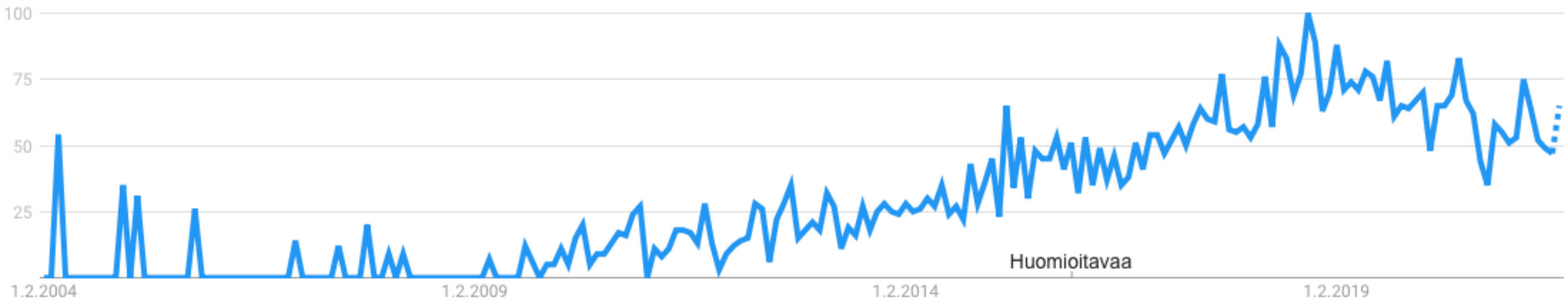
Base: 164 IT executives and managers

Source: A commissioned study conducted by Forrester Consulting on behalf of Thoughtworks, September 2012

Google trends after that 2003



DevOps

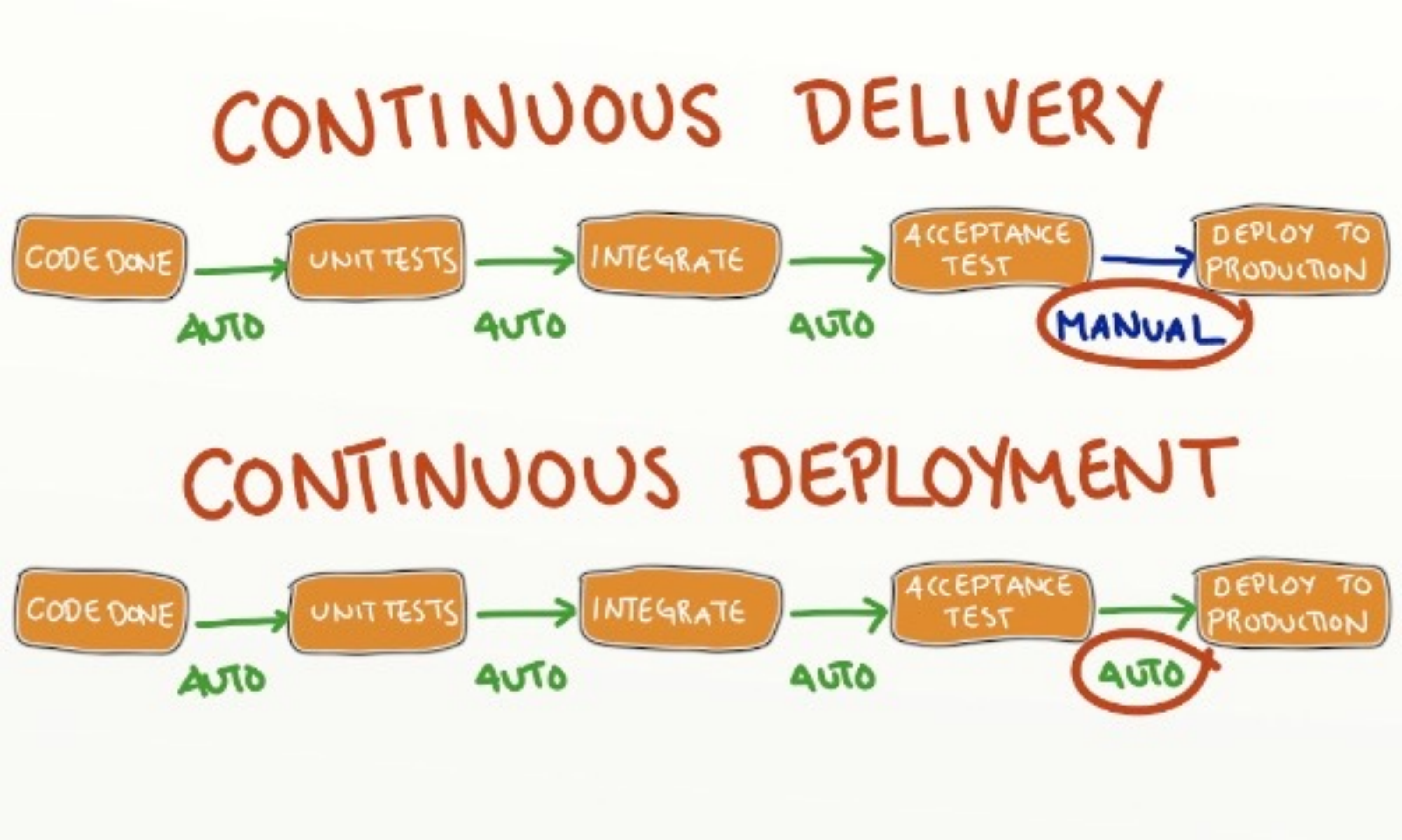


Continuous deployment

Tampereen yliopisto
Tampere University

Continuous delivery and deployment

(http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment)



Main principles (https://continuousdelivery.com/principles/)

- Build quality in
- Work in small batches
- Computers perform repetitive tasks, people solve problems
- Relentlessly pursue continuous improvement
- Everyone is responsible

Sound familiar from somewhere?

CI – essential practices
(according to Humbley and Farley)

- Don’t check in on a broken code
- Always run all commits tests locally before committing, or get your CI server to do it for you
- Wait for commit tests to pass before moving on
- Never go home on a broken build
- Always be prepared to revert to the previous revisions
- Time-box fixing before reverting
- Don’t comment out failing tests
- Take responsible for all breakages that result from your changes
- Test-driven development

Deployment essential pract.
(according to Humbley and Farley)

- Only build your binaries once
- Deploy the same way to every environment
- Smoke-test your deployments
- Deploy to copy of production
- Each change should propagate through the pipeline instantly
- If any part of pipeline fails, stop the line

Reported HP case-study
(https://continuousdelivery.com/evidence-case-studies/)

They had three high-level goals:

- Create a single platform to support all devices
- Increase quality and reduce the amount of stabilization required prior to release
- Reduce the amount of time spent on planning

A key element in achieving these goals was implementing continuous delivery, with a particular focus on:

- The practice of continuous integration
- Significant investment in test automation
- Creating a hardware simulator so that tests could be run on a virtual platform
- Reproduction of test failures on developer workstations

Reported HP case-study

(https://continuousdelivery.com/evidence-case-studies/)

They

- C
- I
- F

A l
a p

- T
- S
- C
- F

Results:

- Overall development costs were reduced by ~40%.
- Programs under development increased by ~140%.
- Development costs per program went down 78%.
- Resources driving innovation increased eightfold.

Let’s speculate the contribution of each

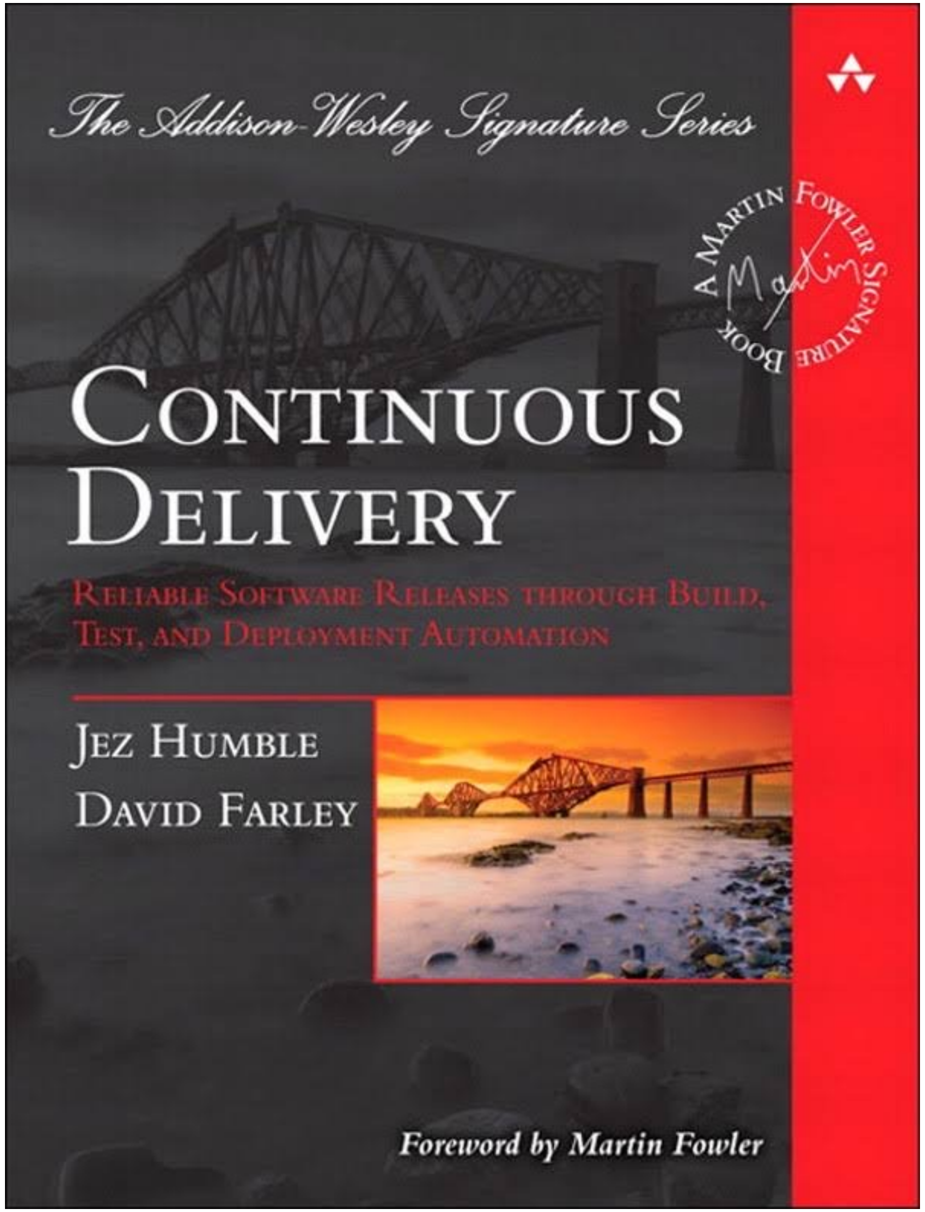
They had three high-level goals:

- Create a single platform to support all devices
- Increase quality and reduce the amount of stabilization required prior to release
- Reduce the amount of time spent on planning

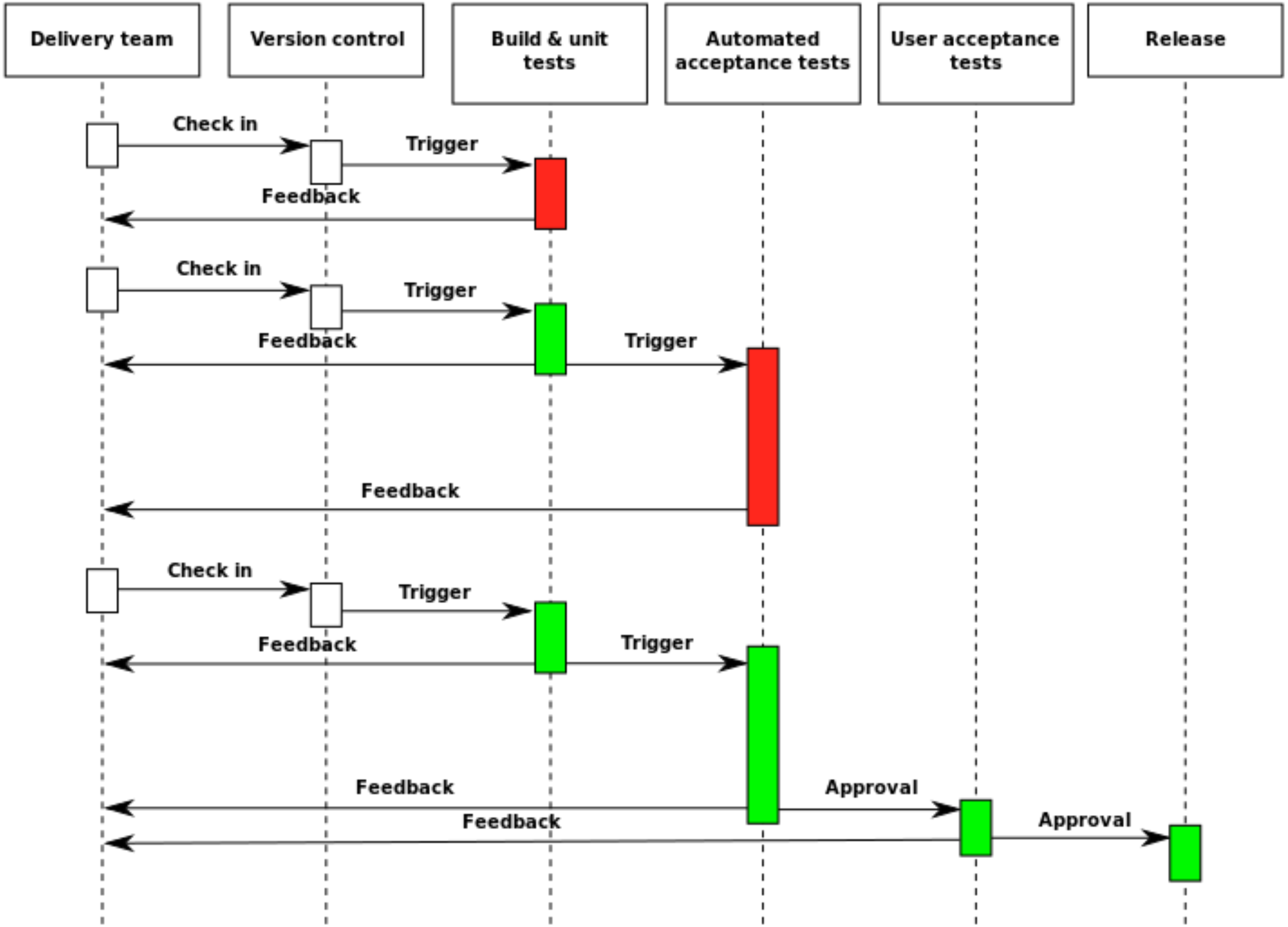
A key element in achieving these goals was implementing continuous delivery, with a particular focus on:

- The practice of continuous integration
- Significant investment in test automation
- Creating a hardware simulator so that tests could be run on a virtual platform
- Reproduction of test failures on developer workstations

CD: Some technical material

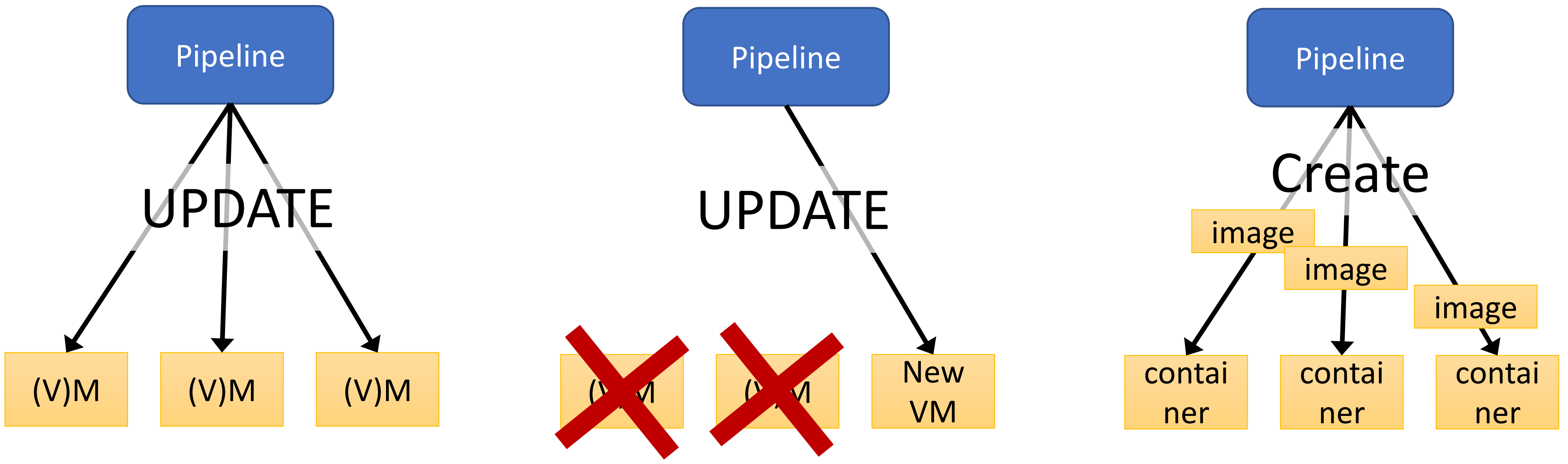

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Martin
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

Deployment pipeline (a possible example)

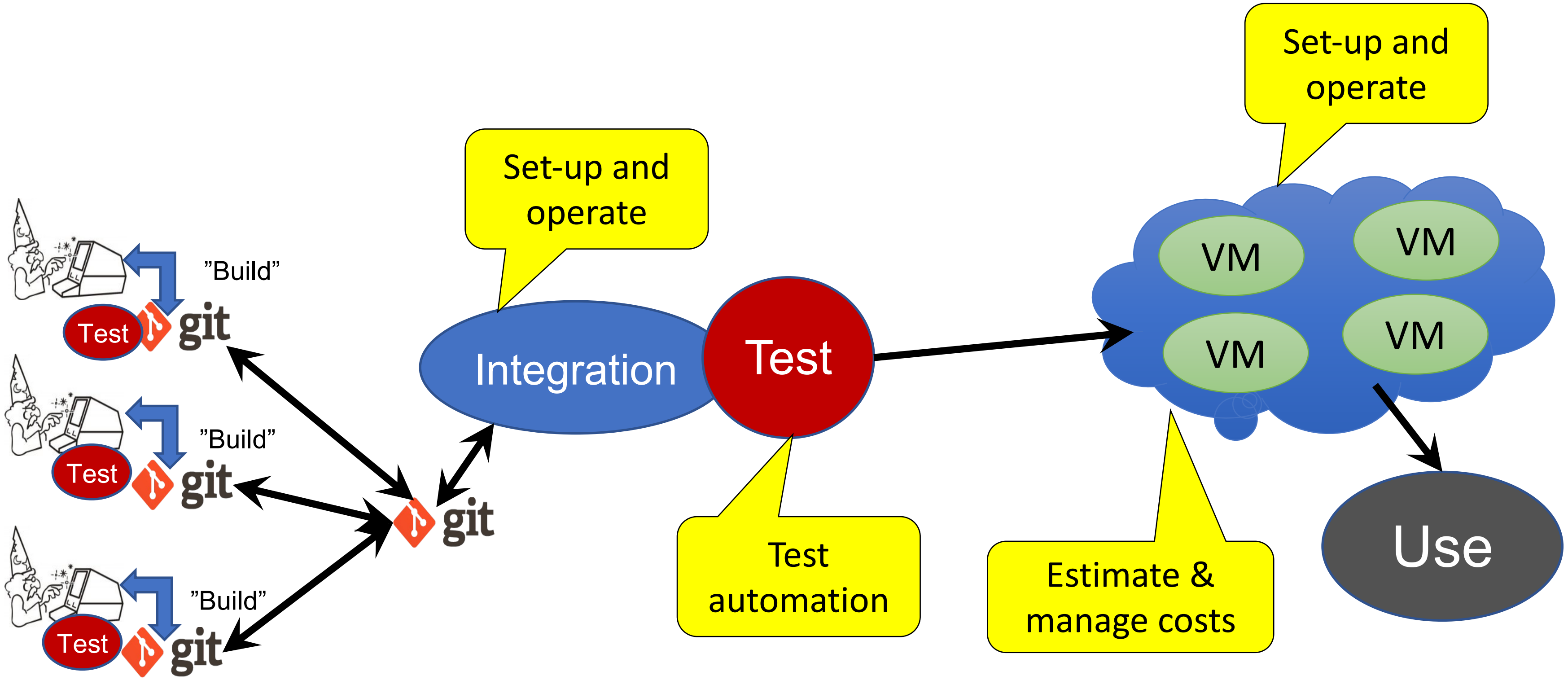


Tampereen yliopisto
Tampere University

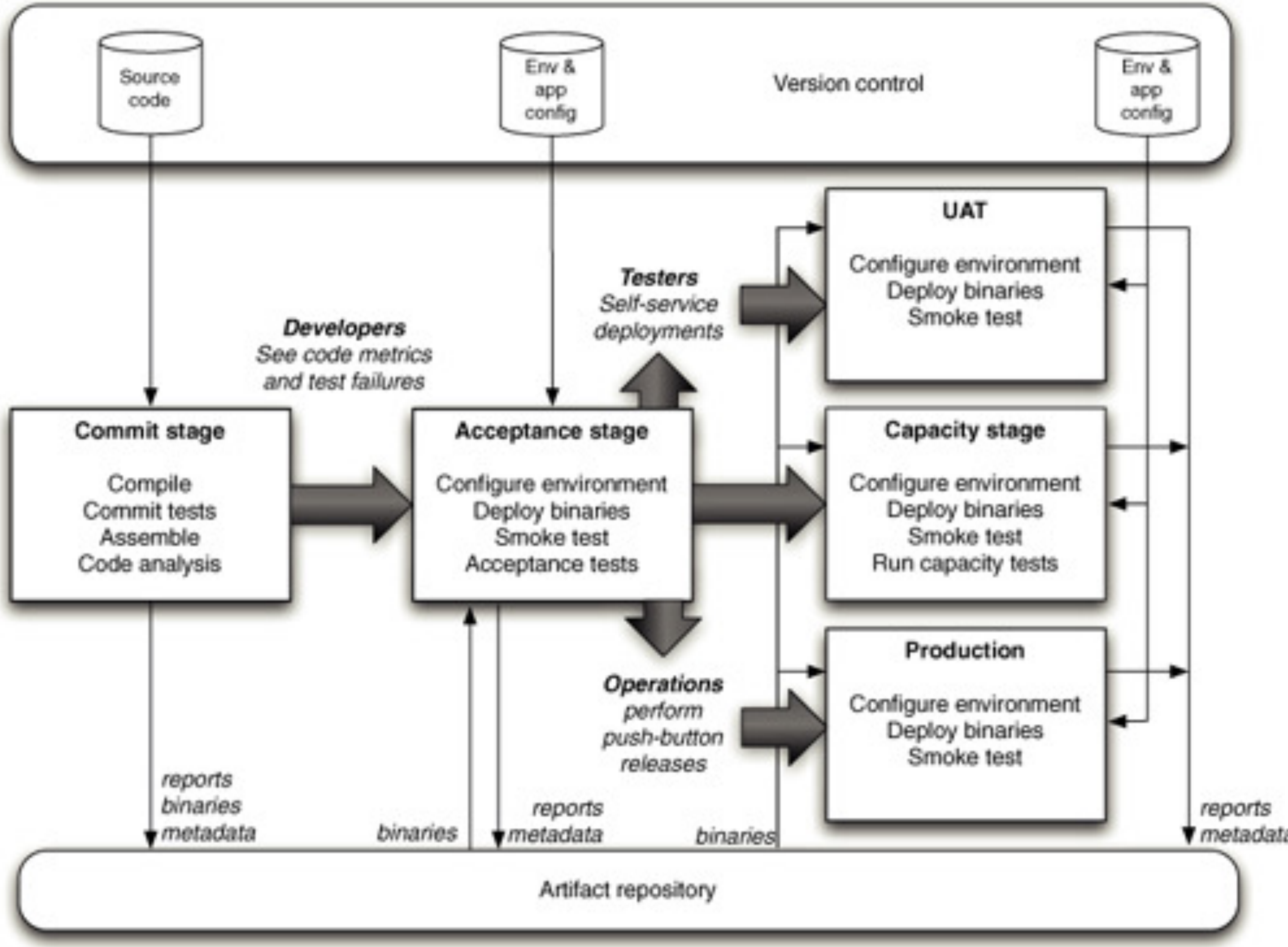
Hypothesis of possible approaches



What does it really take to run CD?

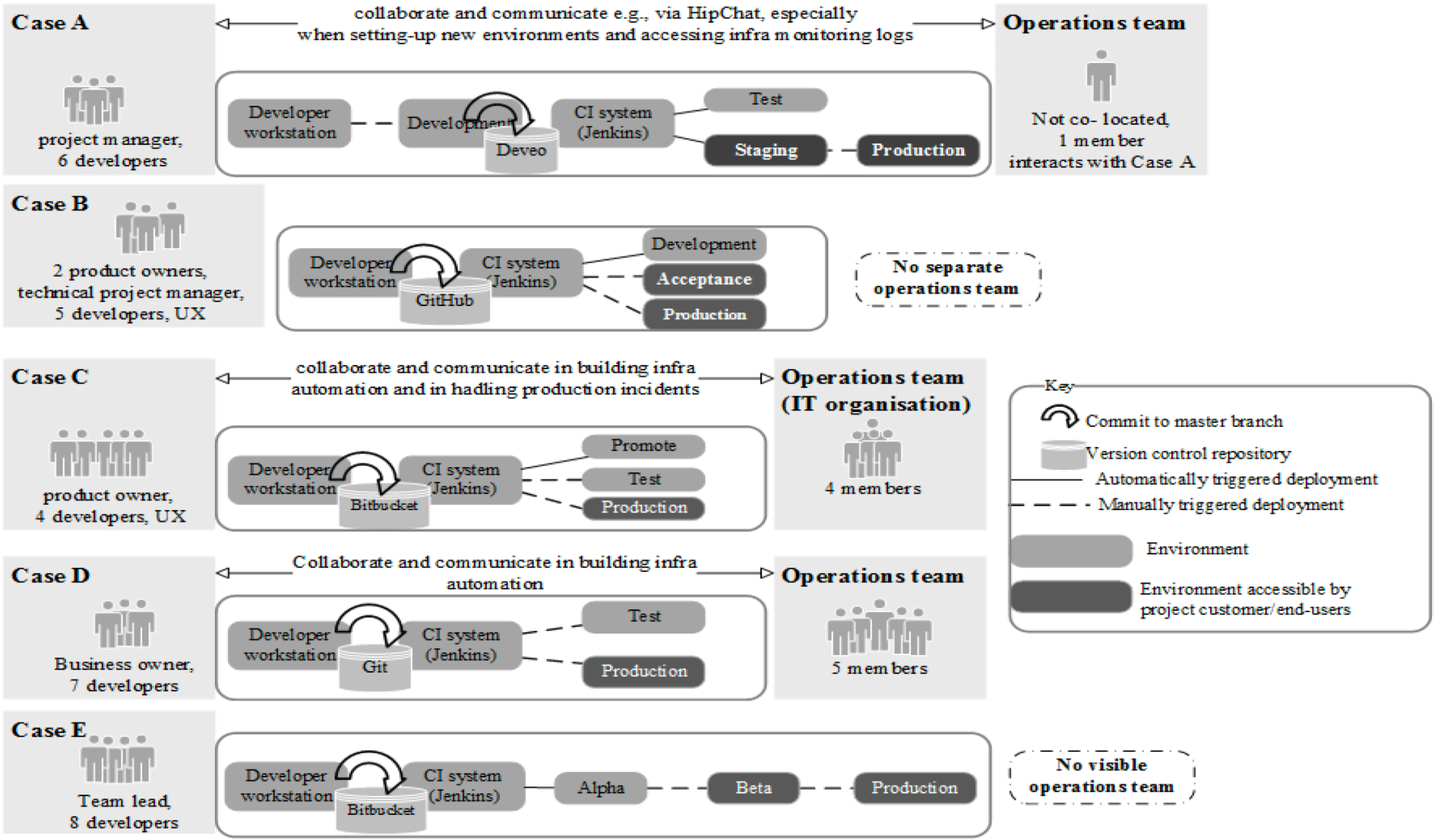


Artifact repository



Couple of Finnish studies

Lwakatare , Kilamo , Karvonen, Sauvola , Heikkilä, Itkonen, Kuvaja, Mikkonen, Oivo & Lassenius:
DevOps in practice : A multiple case study of five companies, Information and Software Technology , vol. 114 , pp. 217-230 .
https://doi.org/10.1016/j.infsof.2019.06.010


Case A
project manager, 6 developers
collaborate and communicate e.g., via HipChat, especially when setting-up new environments and accessing infra monitoring logs
Operations team
Not co-located, 1 member interacts with Case A
Developer workstation
Development
Deveo
CI system (Jenkins)
Test
Staging
Production
Case B
2 product owners, technical project manager, 5 developers, UX
Developer workstation
GitHub
CI system (Jenkins)
Development
Acceptance
Production
No separate operations team
Case C
product owner, 4 developers, UX
collaborate and communicate in building infra automation and in hadling production incidents
Operations team (IT organisation)
4 members
Developer workstation
Bitbucket
CI system (Jenkins)
Promote
Test
Production
Case D
Business owner, 7 developers
Collaborate and communicate in building infra automation
Operations team
5 members
Developer workstation
Git
CI system (Jenkins)
Test
Production
Case E
Team lead, 8 developers
Developer workstation
Bitbucket
CI system (Jenkins)
Alpha
Beta
Production
No visible operations team
Key
Commit to master branch
Version control repository
Automatically triggered deployment
Manually triggered deployment
Environment
Environment accessible by project customer/end-users

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'

Perceived challenges

- Insufficiencies in infrastructure automation
- High demand for skills and knowledge
- Project and resource constraints
- Difficulties in monitoring, especially for microservice-based applications and in determining useful metrics
- Difficulties in determining a right balance between the speed of new functionality and quality.

Summary of the findings

(i) software development team attaining ownership and responsibility to deploy software changes in production is crucial in DevOps.

(ii) toolchain usage and support in deployment pipeline activities accelerates the delivery of software changes, bug fixes and handling of production incidents. (ii) the delivery speed to production is affected by context factors, such as manual approvals by the product owner

(iii) steep learning curve for new skills is experienced by both software developers and operations staff, who also have to cope with working under pressure.

Leppänen, Mäkinen, Pagels, Eloranta, Itkonen, Mäntylä, Männistö
The highways and country roads to continuous deployment,
IEEE Software, vol. 32, no. 2, pp. 64-72, Mar.-Apr. 2015.
doi: 10.1109/MS.2015.50

” Interviews with 15 information and communications technology companies revealed the benefits of and obstacles to continuous deployment. Despite understanding the benefits, none of the companies adopted a fully automatic deployment pipeline.”

State of the practice (2014)

- Only one company had completely automatic pipeline to deployable product; no one really to production
- Fastest time from code change to production
  - 5min – 4 weeks
    (for web application developers longest time was 1 day)
- Cycle-time to potentially deployable software
  - 20min – 1 months
- Full deployment cycle
  - 1 hour – 1.5 years

Perceived benefits 1/2

- Faster feedback
  - to development
  - From users to decision making
- More Frequent Releases
  - ” less waste because the features weren’t waiting in the development pipeline to be released.”
- Improved Quality and Productivity
  - robust automated deployment with a comprehensive test suite
  - reduced scope for each release

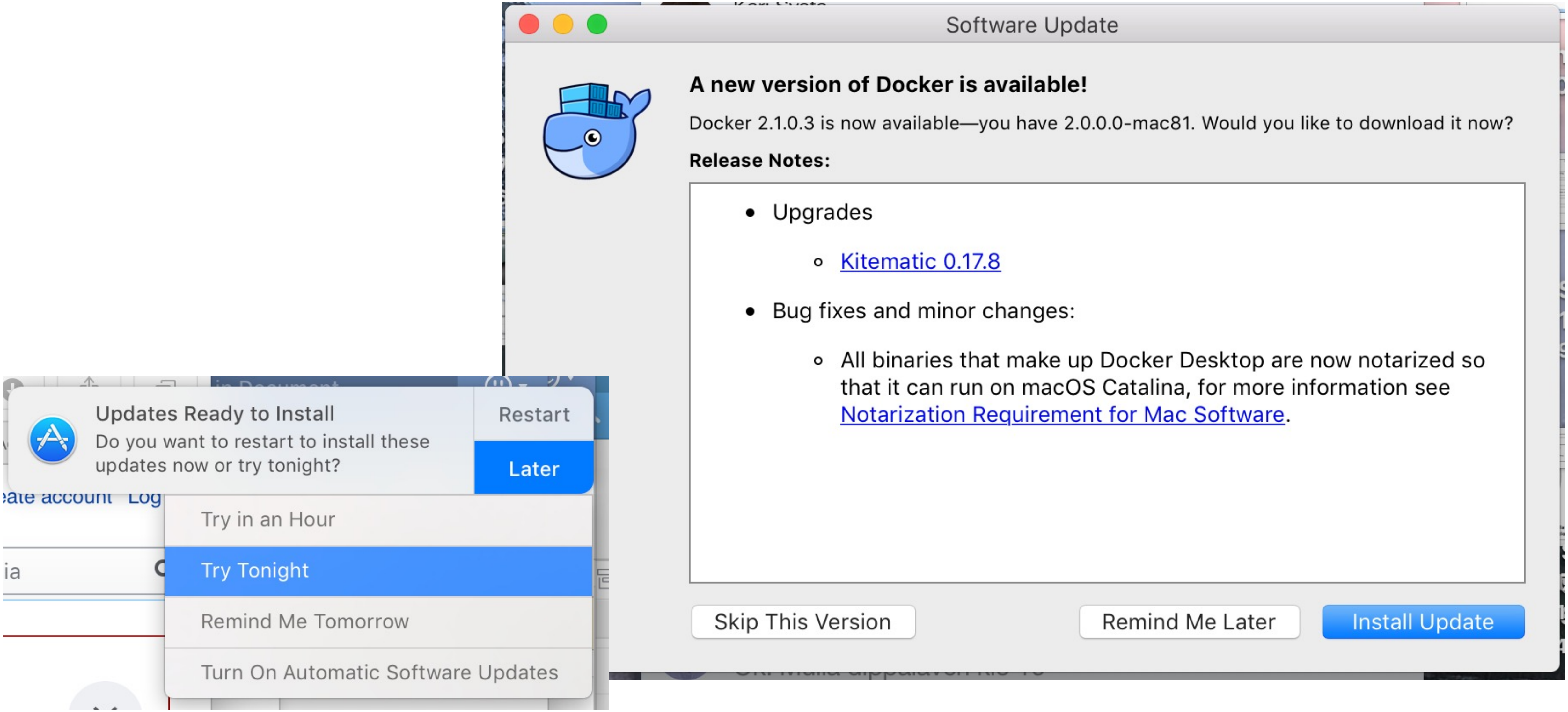
Perceived benefits 2/2

- Improved Customer Satisfaction
  - new product features provided better customer service
  - (reported by 5 out of 15 interviewed organisations)
- Effort Savings
  - three interviewees reported
  - automation saved time
- Closer Connection between Development and Operations
  - only one reported !

Obstacles 1/2

- Resistance to Change
  - Organization culture, management, social relations, …
- Customer Preferences
  - Might be reluctant to deal with more frequent releases
- Domain Constraints
  - Telecom, Medical, Embedded, …
  - Distribution channels
- Developer Trust and Confidence
  - Proficiency and knowledge of typical continuous-deployment practices
  - Reliable automated testing (… even browser-bases apps)

About resistance



Software Update

A new version of Docker is available!

Docker 2.1.0.3 is now available—you have 2.0.0.0-mac81. Would you like to download it now?

Release Notes:

- Upgrades
  - Kitematic 0.17.8
- Bug fixes and minor changes:
  - All binaries that make up Docker Desktop are now notarized so that it can run on macOS Catalina, for more information see Notarization Requirement for Mac Software.

Updates Ready to Install

Do you want to restart to install these updates now or try tonight?

Obstacles 2/2

- Legacy Code Considerations
  - Quality has decreased over time
  - Not be designed to be automatically tested
- Duration, Size, and Structure
  - Effort to create the pipe-line and tests is big
  - In big projects the execution of tests will also take time
- Different Development and Production Environments
  - Especially ”embedded”
- Manual and Nonfunctional Testing