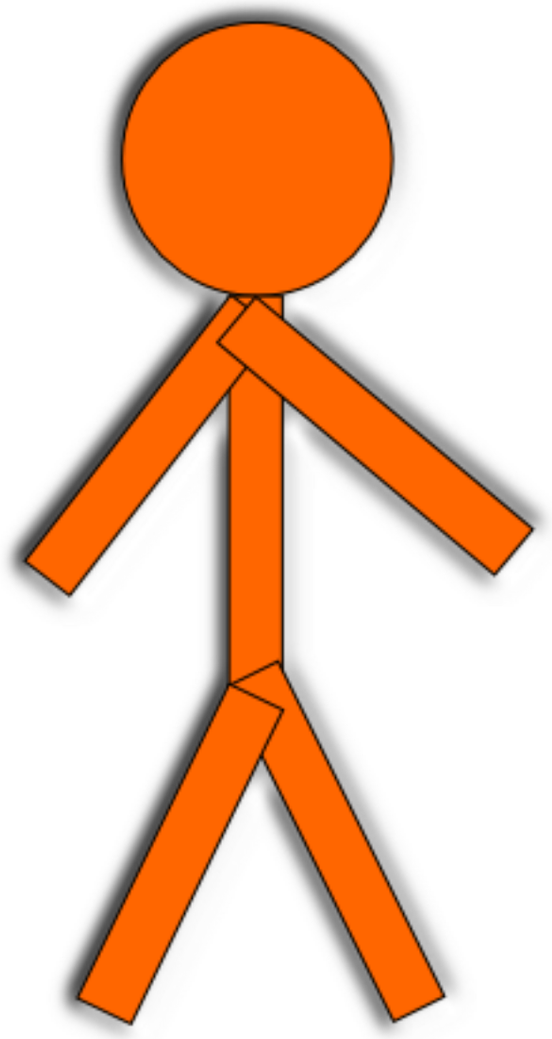
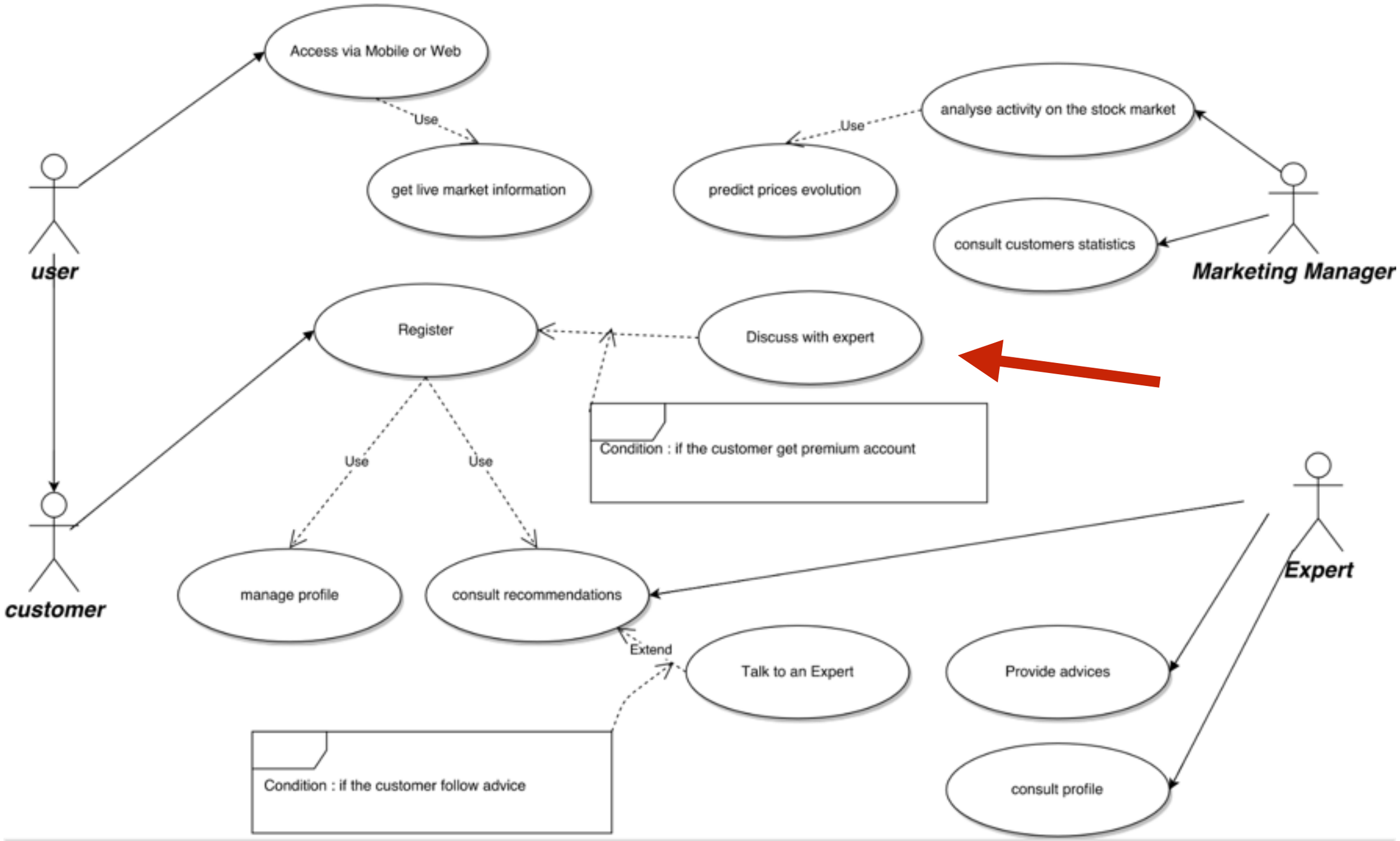
Software Engineering and Architecture
Continuous Delivery

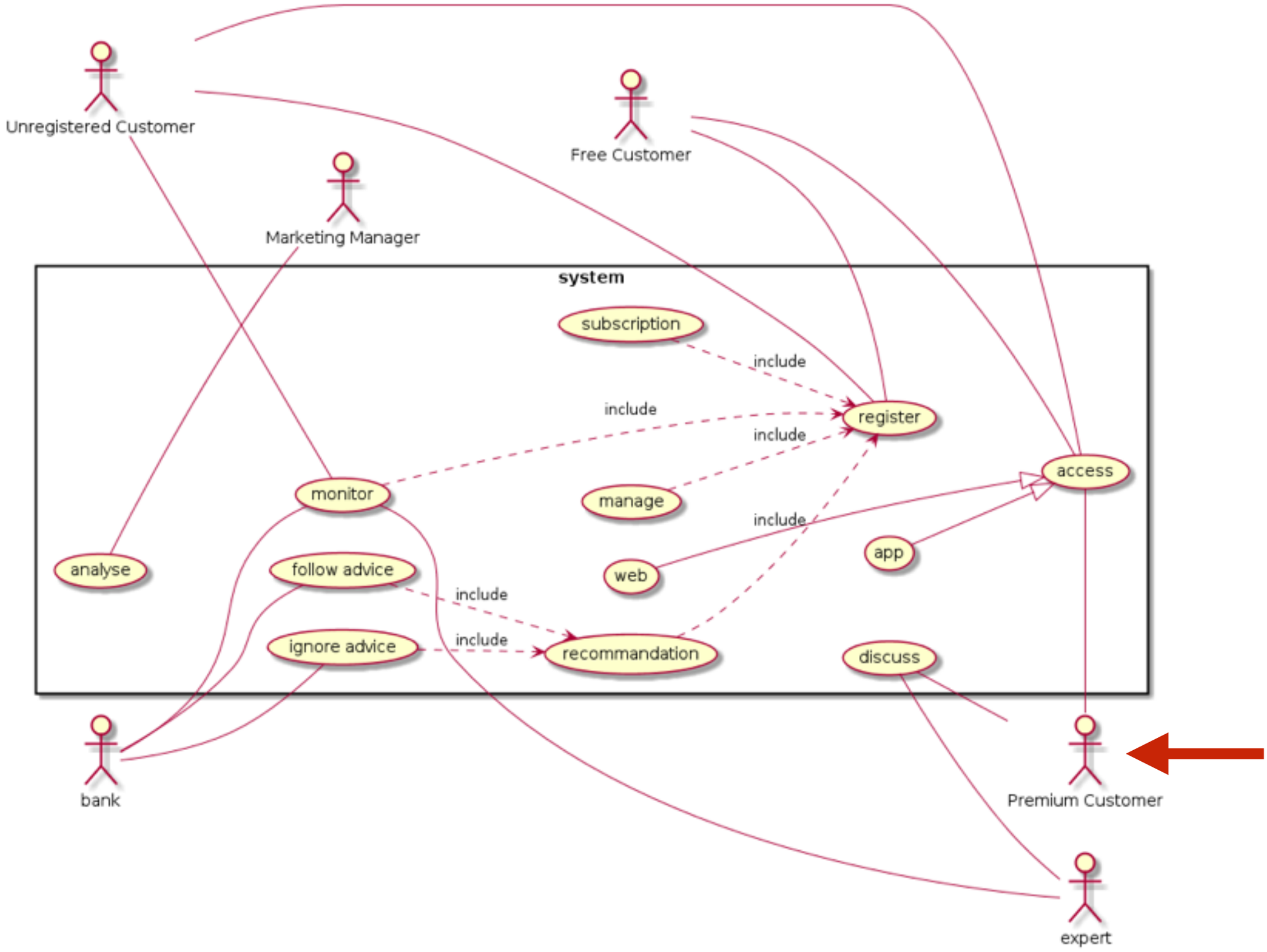
Olivier Liechti
HEIG-VD
olivier.liechti@heig-vd.ch

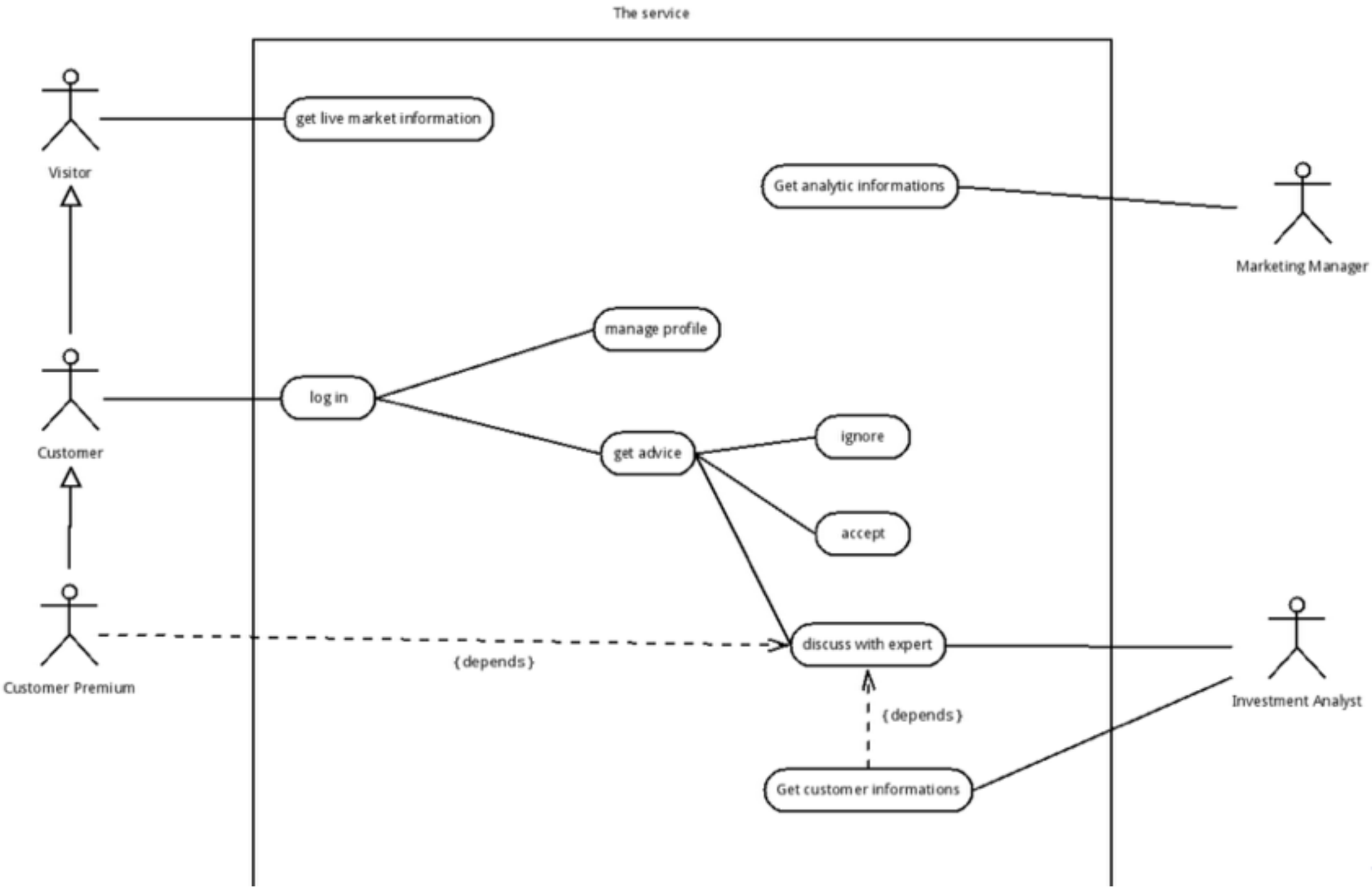
MSE | MASTER OF SCIENCE IN ENGINEERING

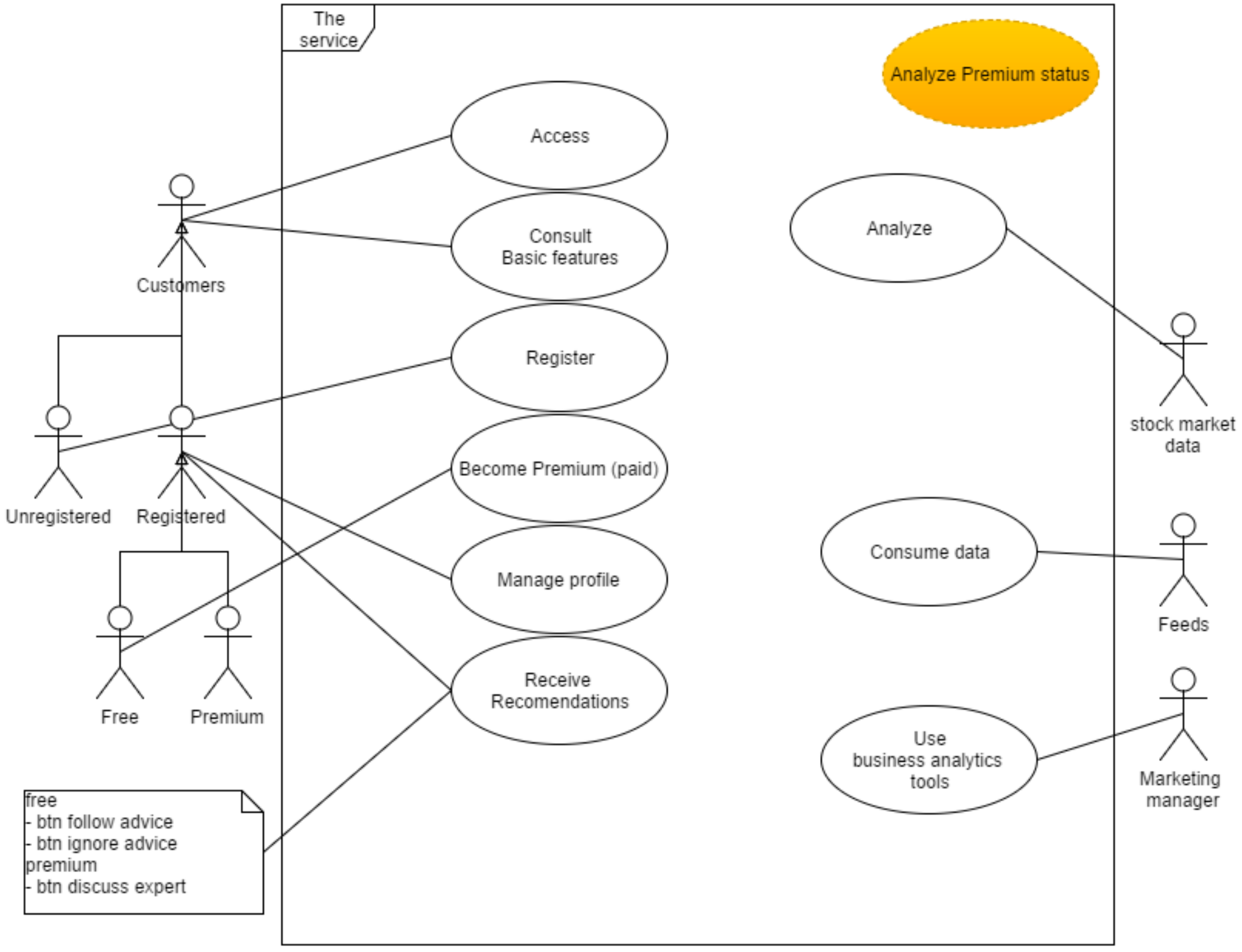


Use Cases


Access via Mobile or Web
Use
get live market information
user
analyse activity on the stock market
Use
predict prices evolution
consult customers statistics
Marketing Manager
Register
Discuss with expert
Condition : if the customer get premium account
Use
Use
manage profile
consult recommendations
customer
Expert
Extend
Talk to an Expert
Provide advices
consult profile
Condition : if the customer follow advice

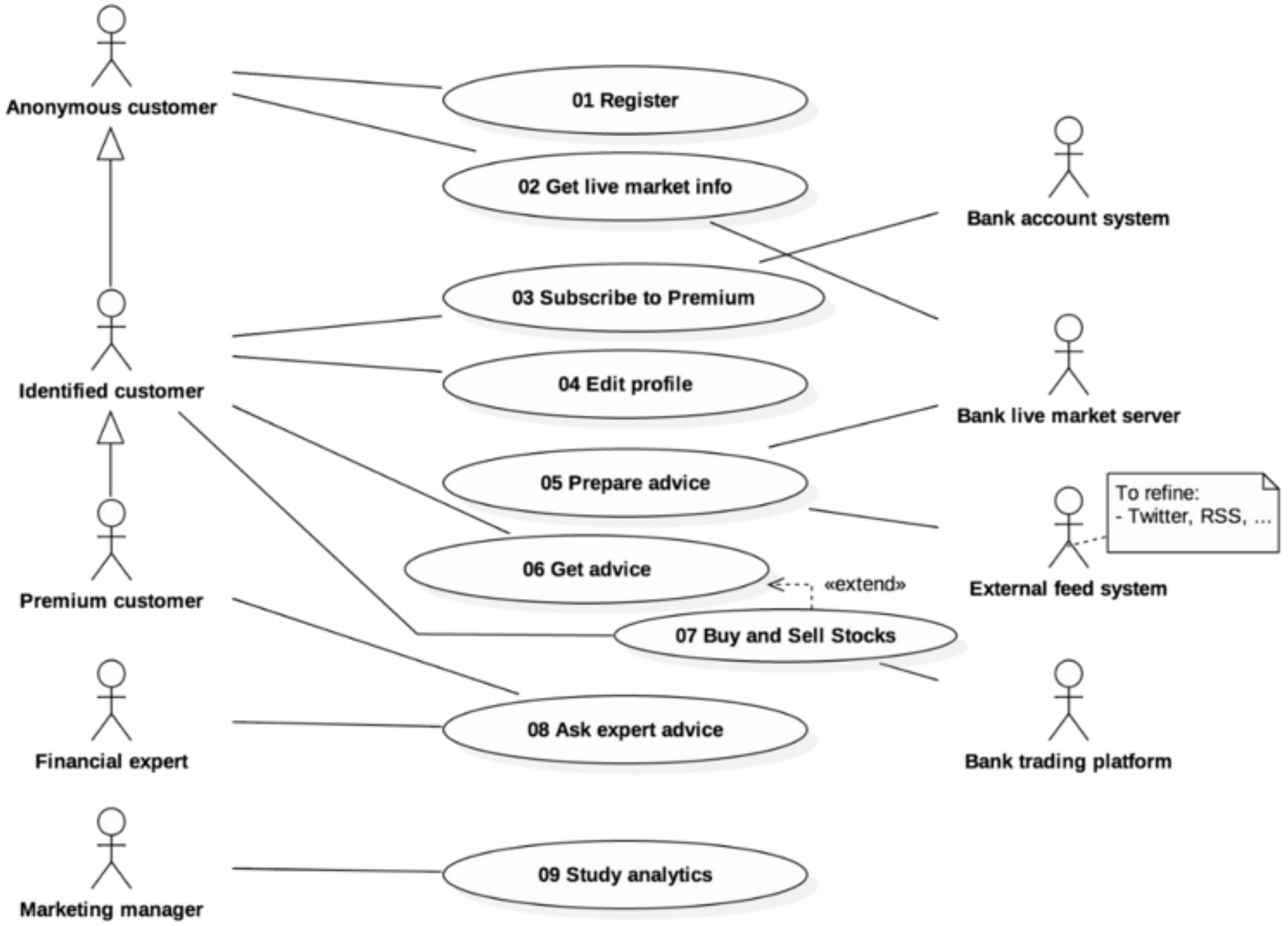

Unregistered Customer
Free Customer
Marketing Manager
system
subscription
include
include
include
register
access
monitor
manage
include
analyse
follow advice
include
web
app
ignore advice
include
recommandation
discuss
bank
Premium Customer
expert

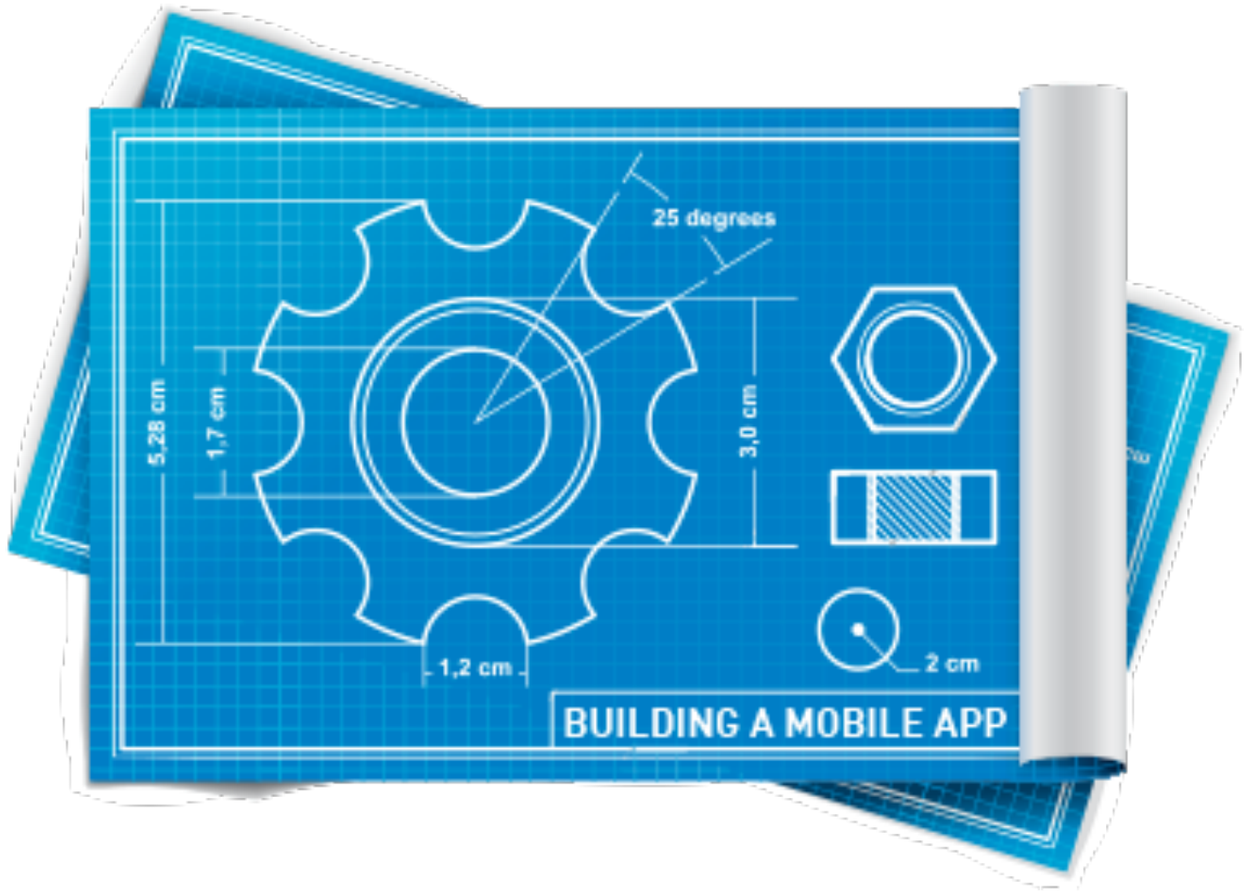

The service
Visitor
get live market information
Get analytic informations
Marketing Manager
manage profile
log in
Customer
get advice
ignore
accept
Customer Premium
{depends}
discuss with expert
Investment Analyst
{depends}
Get customer informations


The service
Analyze Premium status
Access
Consult Basic features
Register
Become Premium (paid)
Manage profile
Receive Recomendations
Analyze
Consume data
Use business analytics tools
Customers
Unregistered
Registered
Free
Premium
stock market data
Feeds
Marketing manager
free
- btn follow advice
- btn ignore advice
premium
- btn discuss expert

Use case analysis

- Left Actors: Humans
- Right Actors: Systems





Architecture

Fun with blocks

Block 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

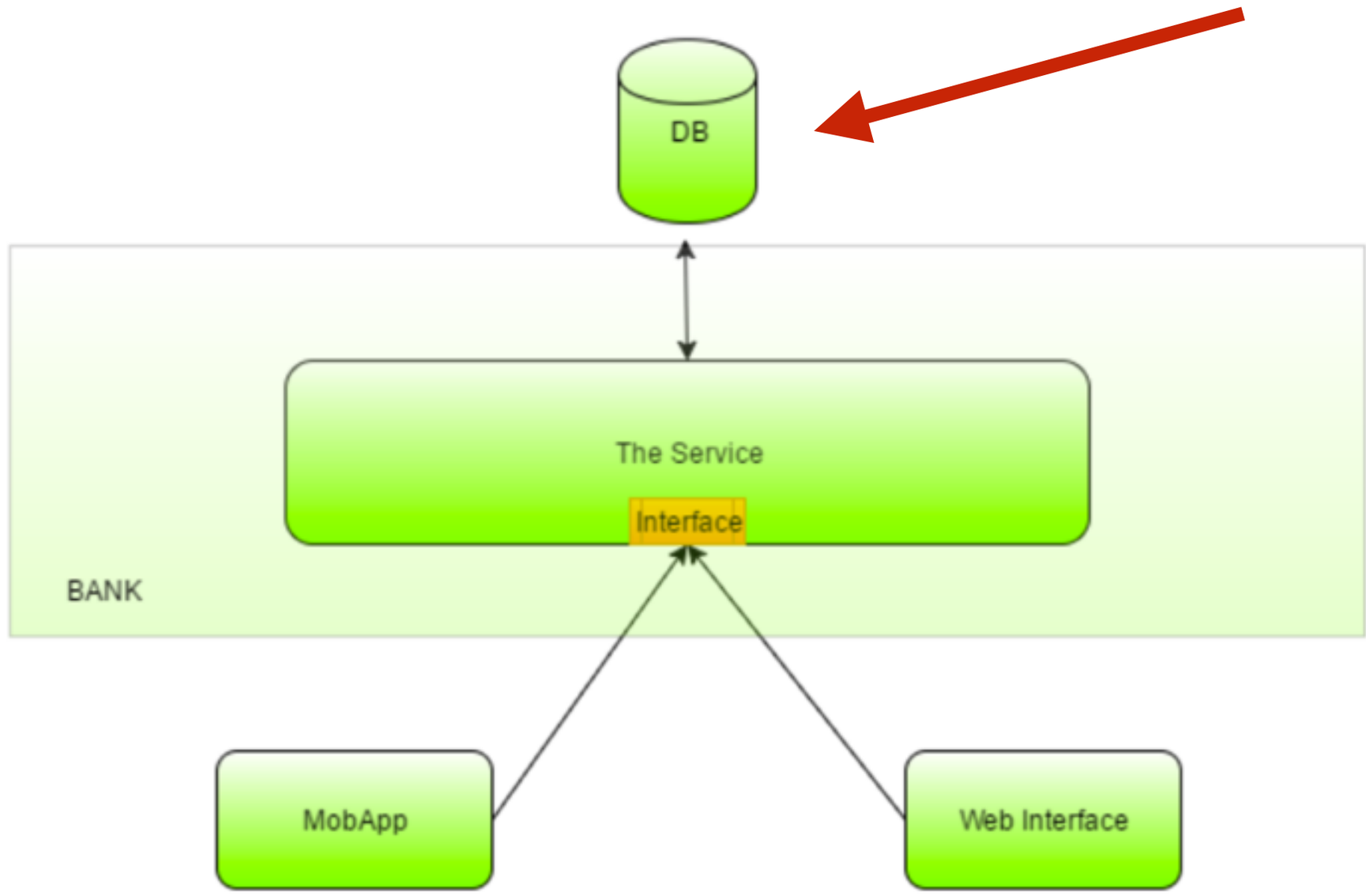
Block 2

Pellentesque sed tellus purus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vestibulum quis magna at risus dictum tempor eu vitae velit.

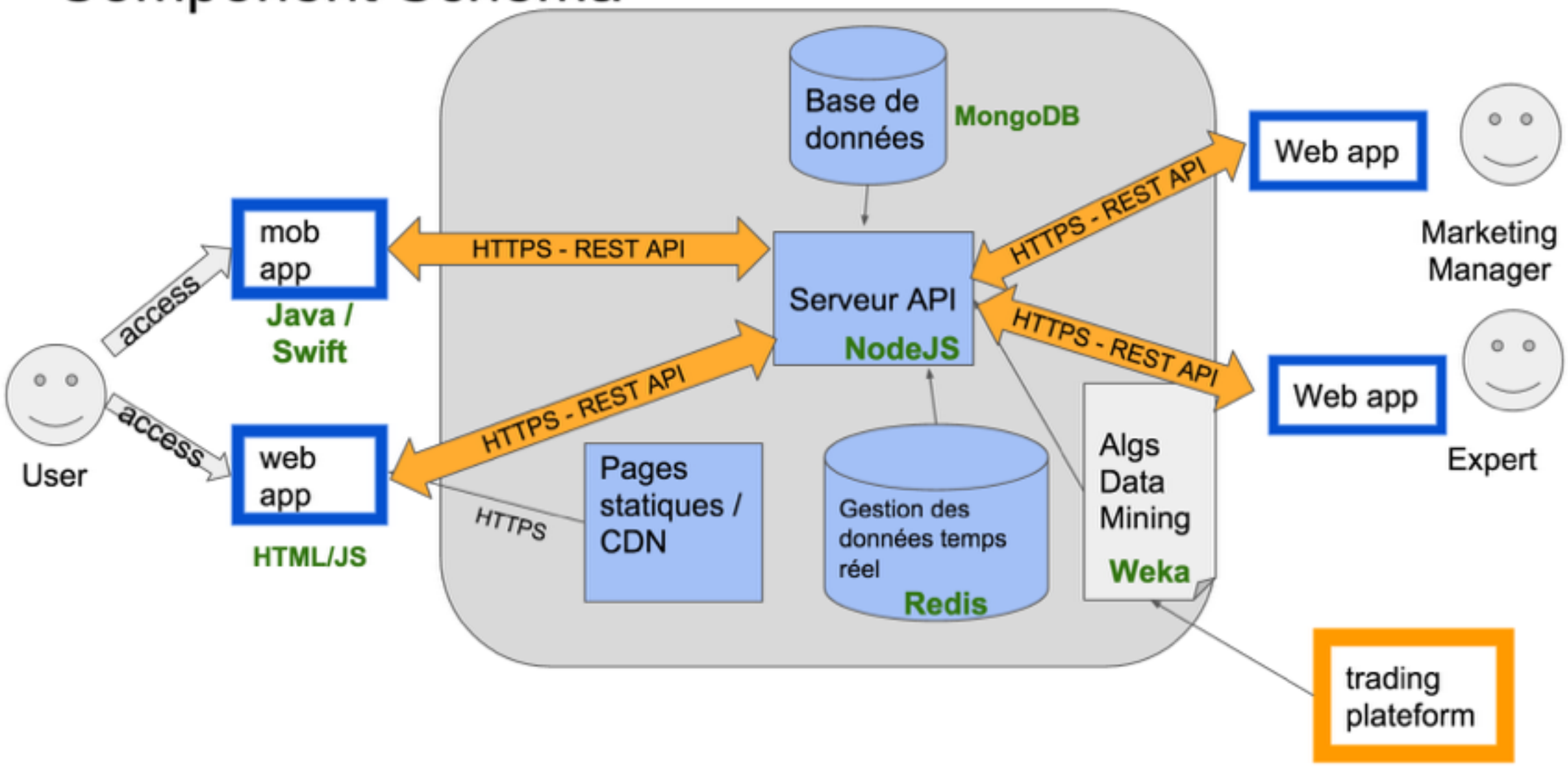
Block 3

Suspendisse tincidunt sagittis gravida. Curabitur condimentum, enim sed venenatis rutrum, ipsum neque consectetur orci, sed blandit justo nisi ac lacus.

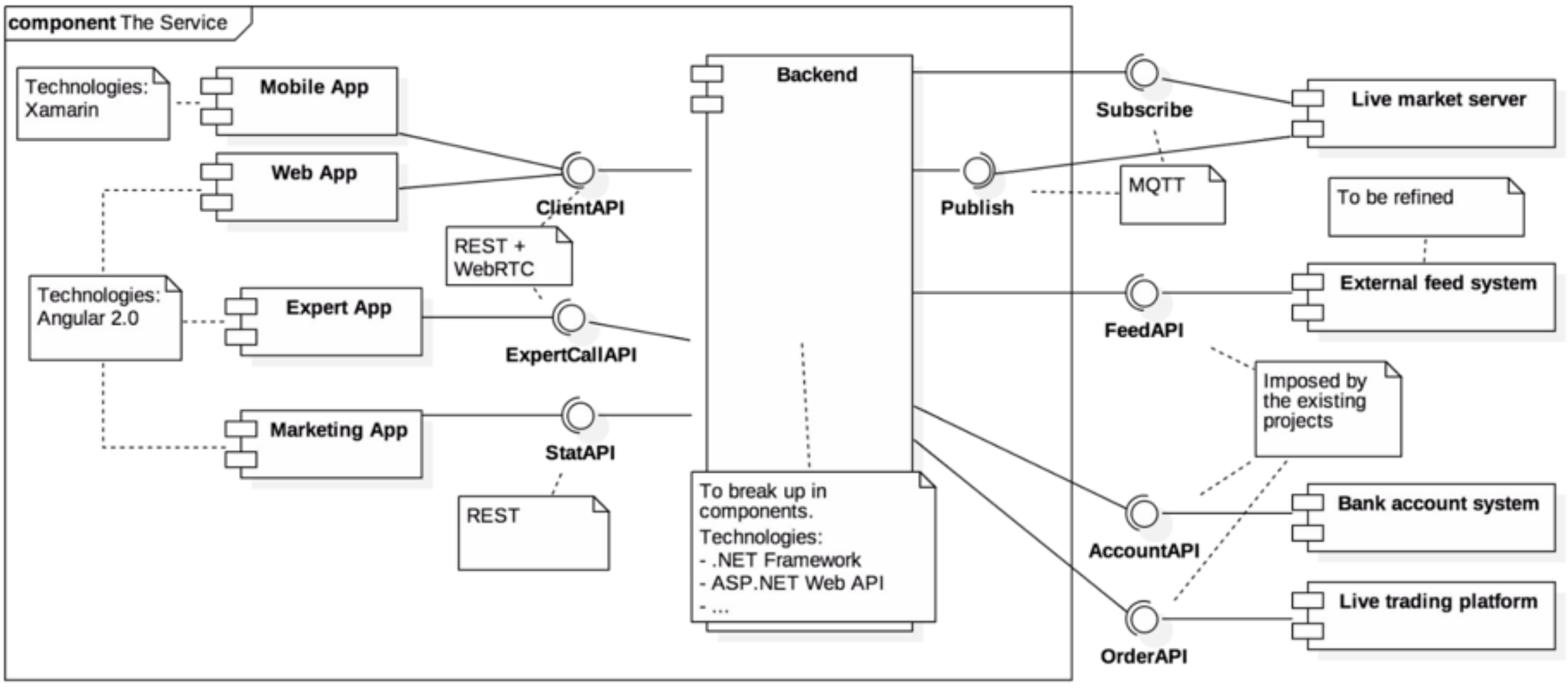
High Level Architecture



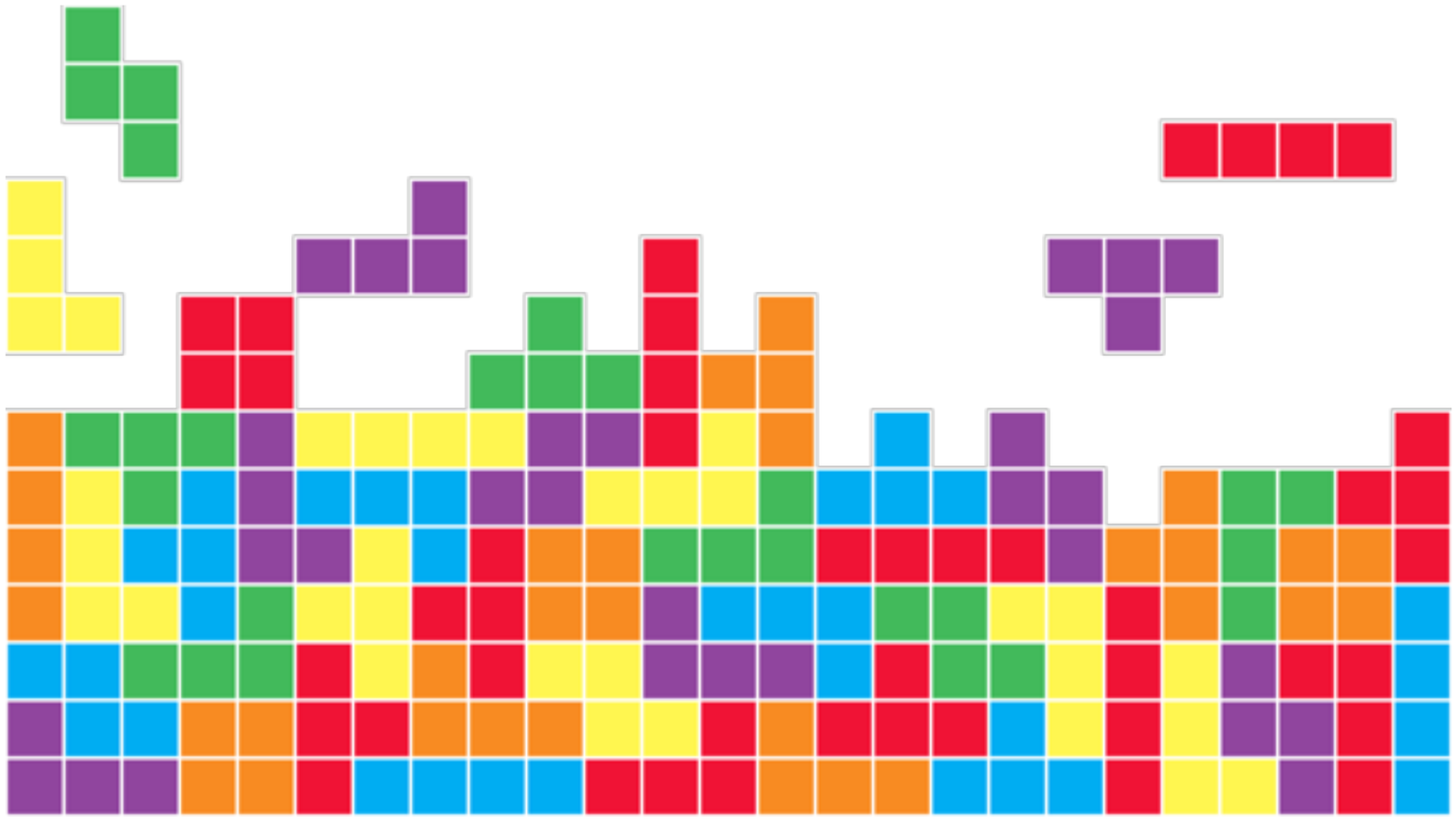
Component Schema



System architecture - Components



MSE | MASTER OF SCIENCE IN ENGINEERING



Planning

We are going to use a scrum methodology to develop The service.

The team is build from 1 project manager, 3 front-end developper, 2 back-end developper and 1 mobile developper.

The following activites need to be done :

- Planning
- Specifications
- Conception
- Implementation
- Testing

The alpha version will be release in 6 months.
The beta version for external user will be release in 8 monthts.

We are going to use Git to keep track of the progress. A bug bounty programm will be create to enhanced the security.

Planning - General

- **Méthode : *Agile***
  - Durée d'un sprint : 2 semaines
  - Velocité de l'équipe : à calculer
  - Nombre d'équipes : 3 équipes de 2
- **Itérations de développement (pour chaque fonctionnalité) :**
  - 1. Analyse du cahier des charges et redéfinitions
  - 2. Estimation du temps et attribution à une équipe
  - 3. Analyse sécuritaire et validation du design
  - 4. Implémentation
  - 5. Tests
  - 6. Acceptation de la fonctionnalité et MEP
- **Deadline v1.0 (distribution clients) :** ?

MSE | MASTER OF SCIENCE IN ENGINEERING

Methodology

- SCRUM
- 5 to 9 people for the team
- Including a Scrum master and a product owner
- Two weeks sprints to keep track of the progress

Team

We need the folowing comptences :

- Java developper (Android App)
- Swift / Objective C developper(iOS App)
- Nodejs / Web Developper (WebApp & NodeJS core)
- Data-mining Expert
- Front End designer
- System Admin (for DB and interactions)
- Scrum Master
- Product Owner

Initial Planning

1. Design architecture of the all system [2 weeks]
2. Build the core [6 weeks]
    a. Developp the REST API [6 weeks]
    b. Install Databases [2 weeks]
    c. Developp the Data-mining algoritms [6 weeks]
3. Developp the Marketing manager and Expert webApp [5 weeks]
4. Developp the client apps [6 weeks]
    a. iOS App [6 weeks]
    b. Android App [6 weeks]
    c. WebApp [5 weeks]
5. Recruit & train experts and marketing managers [4 weeks]

slicing & baby steps

component vs feature team?

1st takeaway

Starting *something* is hard.
Starting something *as a group* is often harder.

Starting this exercise was not that easy for most groups.
Starting a software project is harder.

If you have been "stuck" 20 minutes last week, would you get stuck 20 days in a project?

The first drawing was often the turning point.

*This is one reason why agile methods say that it is important to **deliver early and to deliver often**.*

*It does not only allow you to gather feedback, but it also **“gets you started”** and gives you a **rhythm**.*

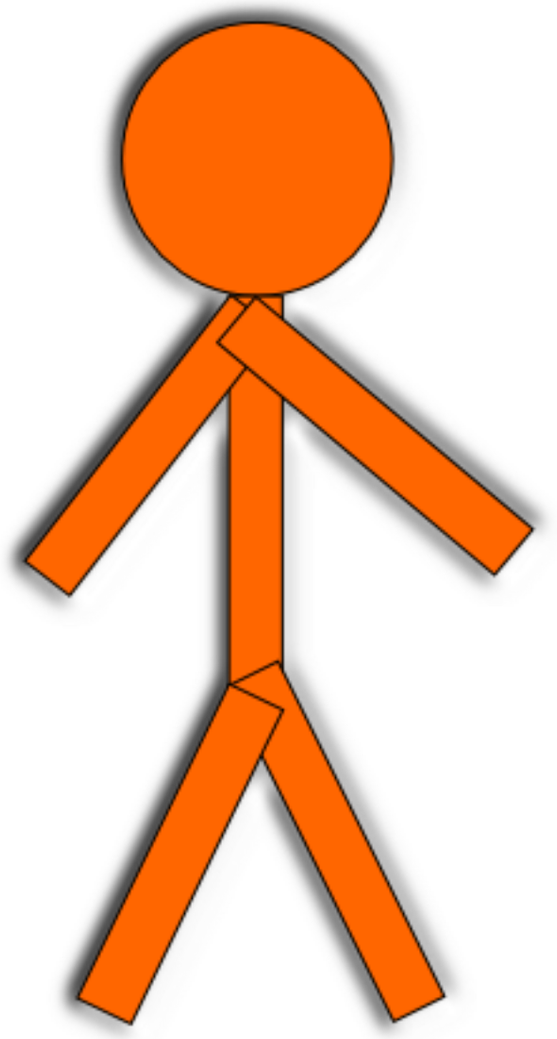
What seemed complex and hard to grasp quickly becomes more concrete and more approachable.

2nd takeaway

Should you remember one thing from this course, this is “it”.

A lot of good things will happen from this simple action.
People will love you and you will live happy forever.

The next time that you draw a use case diagram, don’t forget this actor...



The System Administrator

By doing this, you will naturally and constantly remember that a piece of software is not something that lives on a bookshelf.

It is a system that runs and that needs to be operated. It is like a baby that you need to watch, to take care of, to nurture and to see grow.

You are a parent of this baby. You have a heavy responsibility and you should just assume that strangers will take care of him.

Software Engineering and Architecture
Continuous Delivery

Olivier Liechti
HEIG-VD
olivier.liechti@heig-vd.ch

MSE | MASTER OF SCIENCE IN ENGINEERING

Ops guy

Dev guy

“Here is a USB key with the .war file that you need to deploy on Tomcat. I have written down the configuration steps on this sheet of paper...”

Pissed-off Ops guy

Pissed-off Dev guy

"WTF... why do I get a database connection error when I deploy this damn application in the Glassfish cluster...???"

Worried Ops guy

Worried Dev guy

*“We have deployed the new release... are we **sure** that we are ok?*
By the way, anyone has a clue about what is in the release?”

*“I hate these production rollouts... it’s always so **stressful**.*
What could we do to avoid bad surprises and not be so
***afraid to touch the prod**...?”*

*"Releasing **small chunks** of changes is less scary than releasing big chunks of changes."*

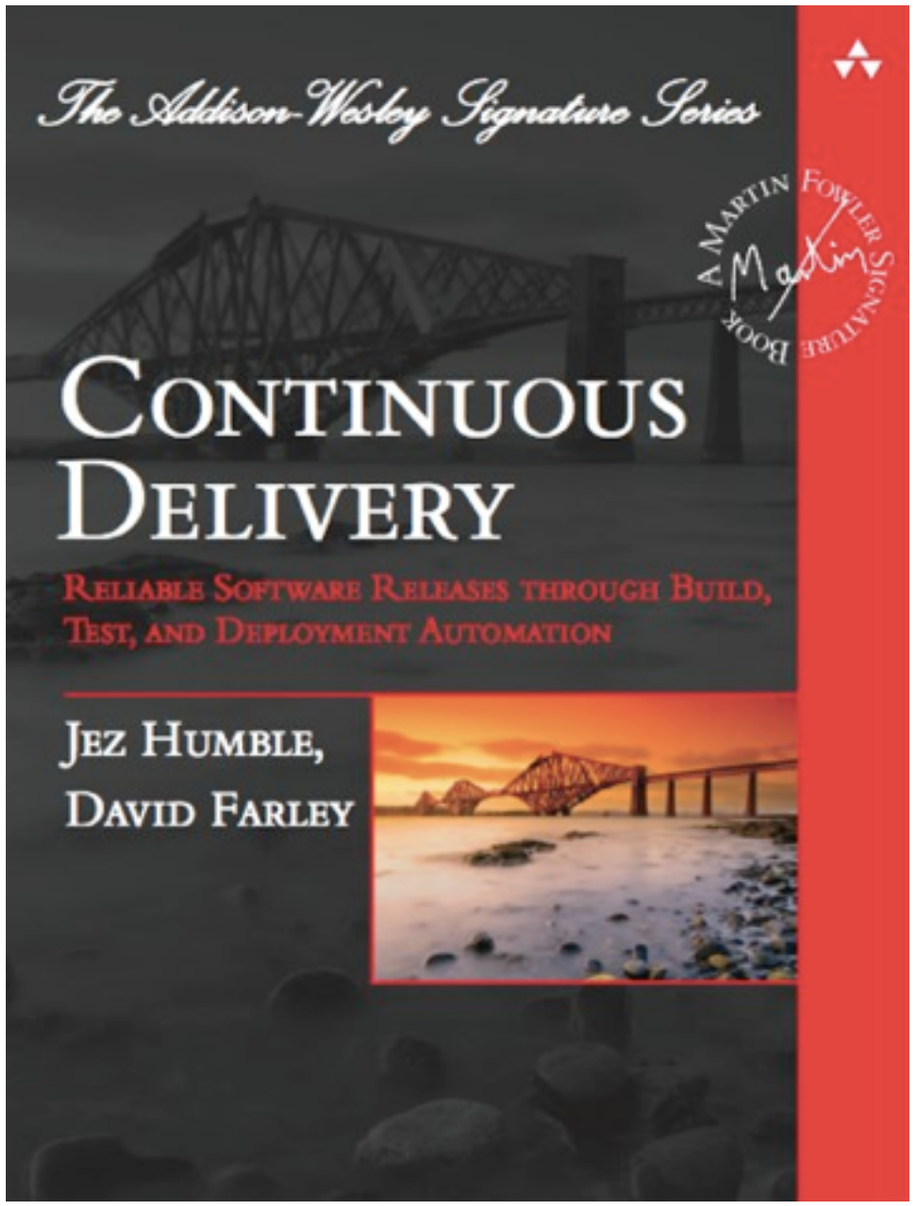
*"Something that you do **often** is less scary than something you do rarely."*

Agile
Robert Redford

*"If something is **automated**, there is now worry that I can make a wrong manipulation."*

*"If I have **evidence** that the system has been **fully tested**, at different levels, I feel secure."*

Continuous Delivery



The slides are based on this book (ISBN:)

Free chapter: http://www.informit.com/articles/article.aspx?p=1621865

Foreword, by Martin Fowler (1)

*"**Continuous Integration** is just the first step.*

Software that's been successfully integrated into a mainline code stream still isn't software that's out in production doing its job.

*Dave and Jez's book pick up the story from CI to **deal with that "last mile"** describing **how to build the deployment pipeline that turns integrated code into production software.**"*

Foreword, by Martin Fowler (2)

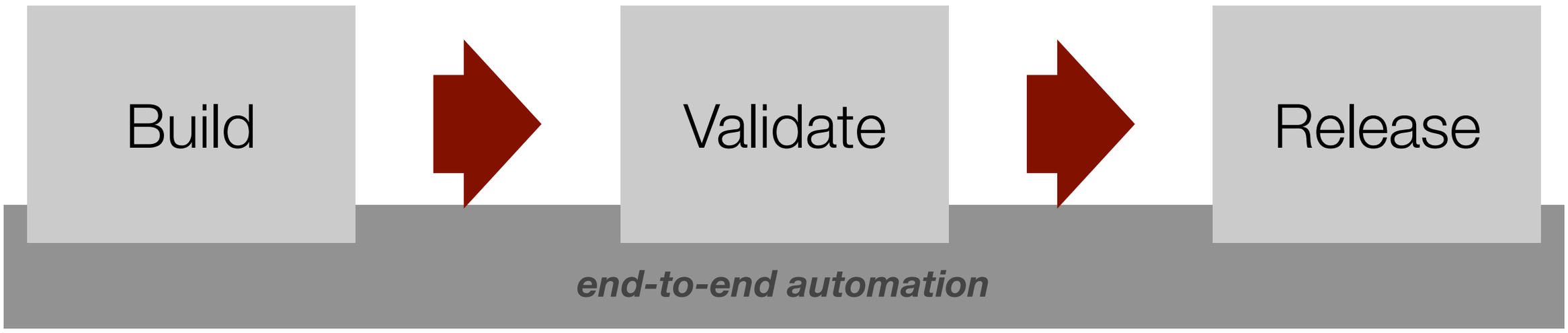
"This kind of delivery thinking has long been a forgotten corner of software development, falling into ***a hole between developers and operations teams****.*

So it's no surprise that the techniques in this book rest upon bringing these teams together—a harbinger of the nascent but growing ***DevOps*** *movement.*

This process also involves ***testers****, as testing is a key element of ensuring error-free releases."*

The Problem of Delivering Software

- How do we transform an **idea** into **something usable by end-users** as **quickly** as possible?
- We need an efficient **software development process**, but that is not enough.
- We need to cover the **complete delivery life-cycle**, until the release into the production environment.



Release Anti-Patterns

- **Anti-Pattern**: Deploying Software Manually
- **Anti-Pattern**: Deploying to a Production-like Environment Only after Development Is Complete
- **Anti-Pattern**: Manual Configuration Management of Production Environments

Principles of Software Delivery

1. Create a **Repeatable, Reliable Process** for Releasing Software
2. **Automate** Almost Everything
3. Keep Everything in **Version Control**
4. **If It Hurts, Do It More Frequently**, and Bring the Pain Forward
5. Build **Quality** In
6. "**Done**" Means Released
7. **Everybody Is Responsible for the Delivery Process**
8. **Continuous Improvement**

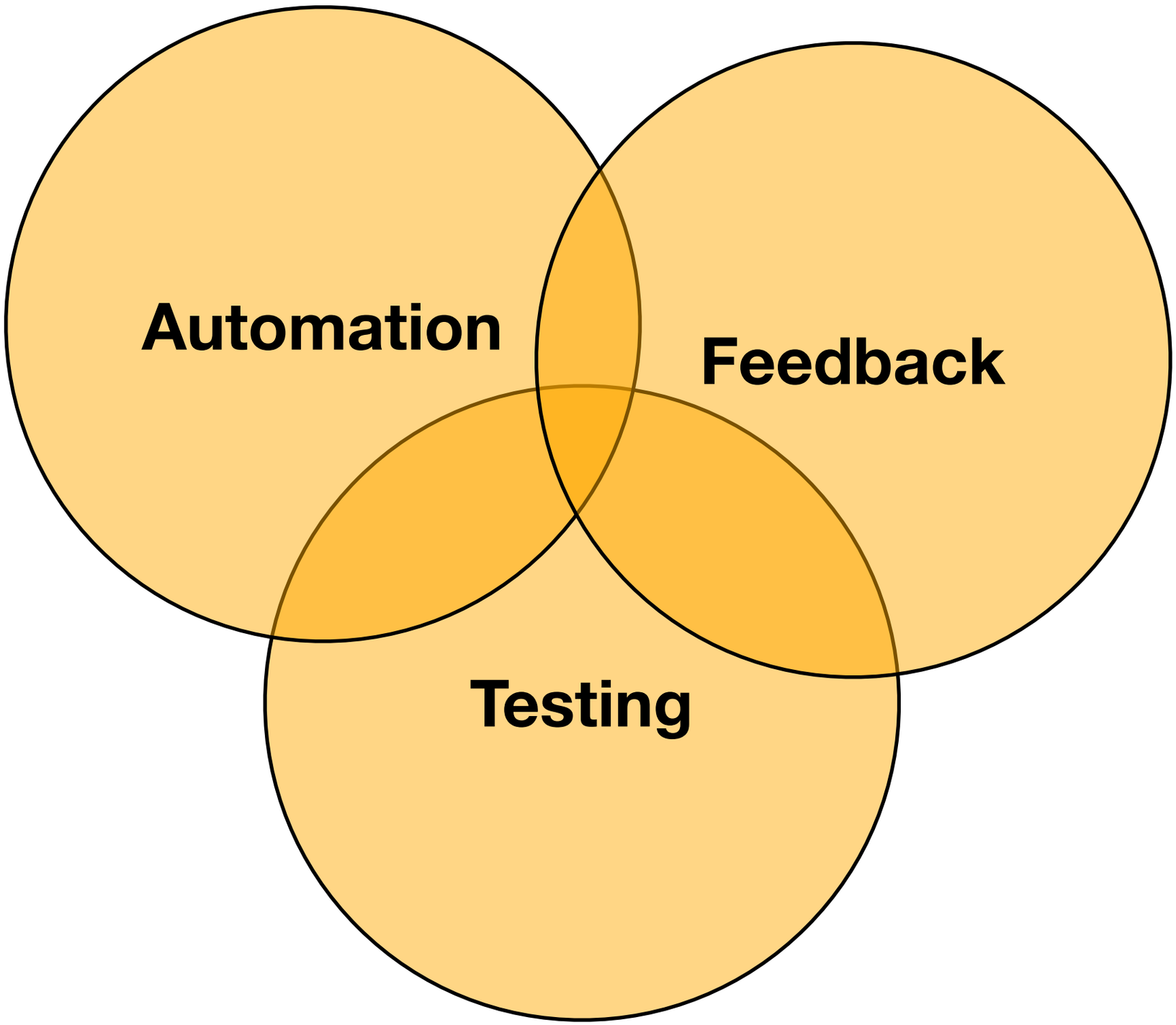
Frequent and Automated Releases

- We, and our fellow practitioners, have discovered that in order to achieve these goals—low cycle time and high quality—**we need to make frequent, automated releases of our software.** Why is this?

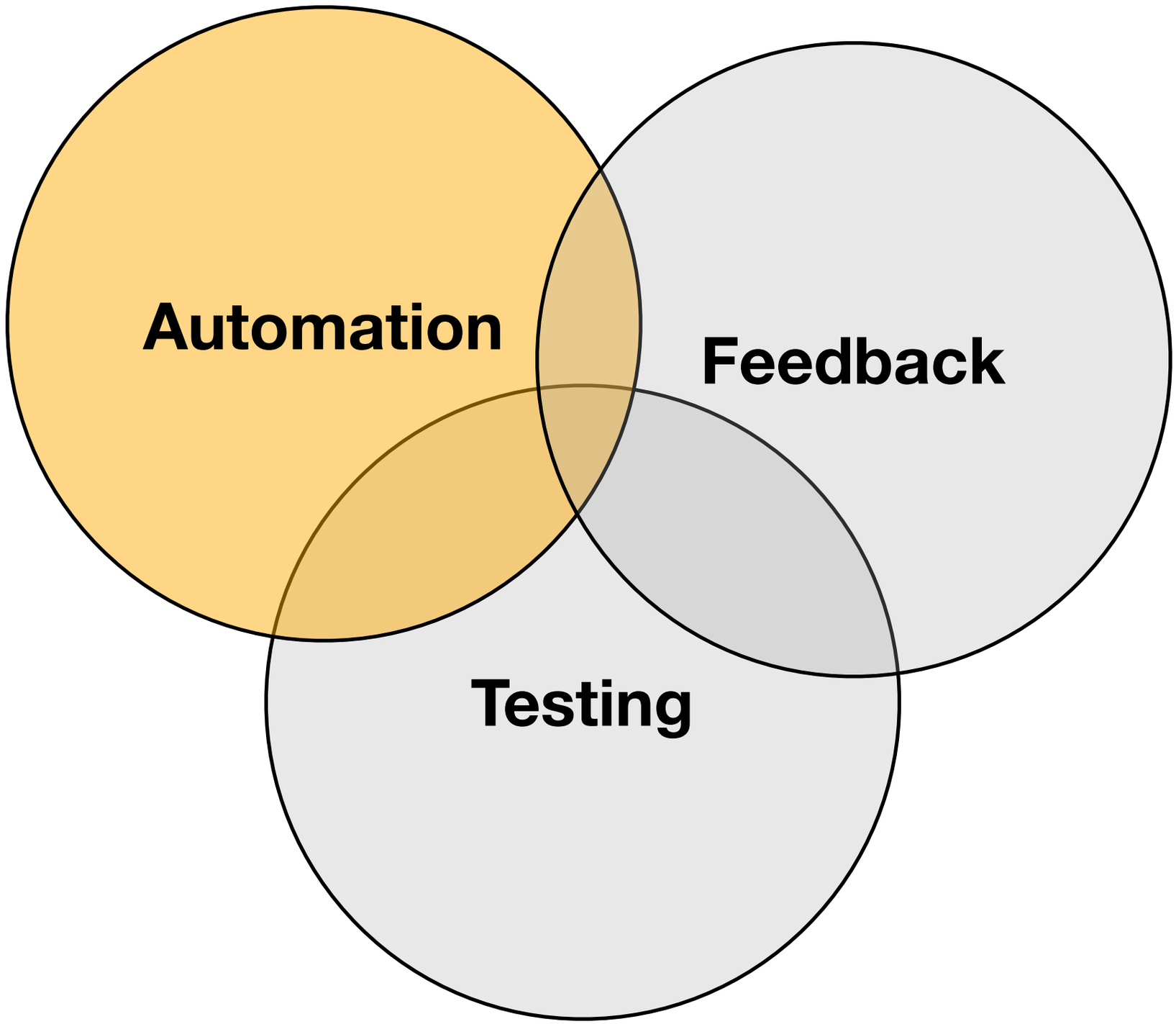
- **Automated.** If the build, deploy, test, and release process is not automated, it is not repeatable. Every time it is done, it will be different, because of changes in the software, the configuration of the system, the environments, and the release process. Since the steps are manual, they are error-prone, and there is no way to review exactly what was done. This means there is no way to gain control over the release process, and hence to ensure high quality. **Releasing software is too often an art; it should be an engineering discipline.**

- **Frequent.** If releases are frequent, the delta between releases will be small. This significantly reduces the risk associated with releasing and makes it much easier to roll back. Frequent releases also lead to faster feedback—indeed, they require it. Much of this book concentrates on getting feedback on changes to your application and its associated configuration (including its environment, deployment process, and data) as quickly as possible.

The Pillars of Continuous Delivery



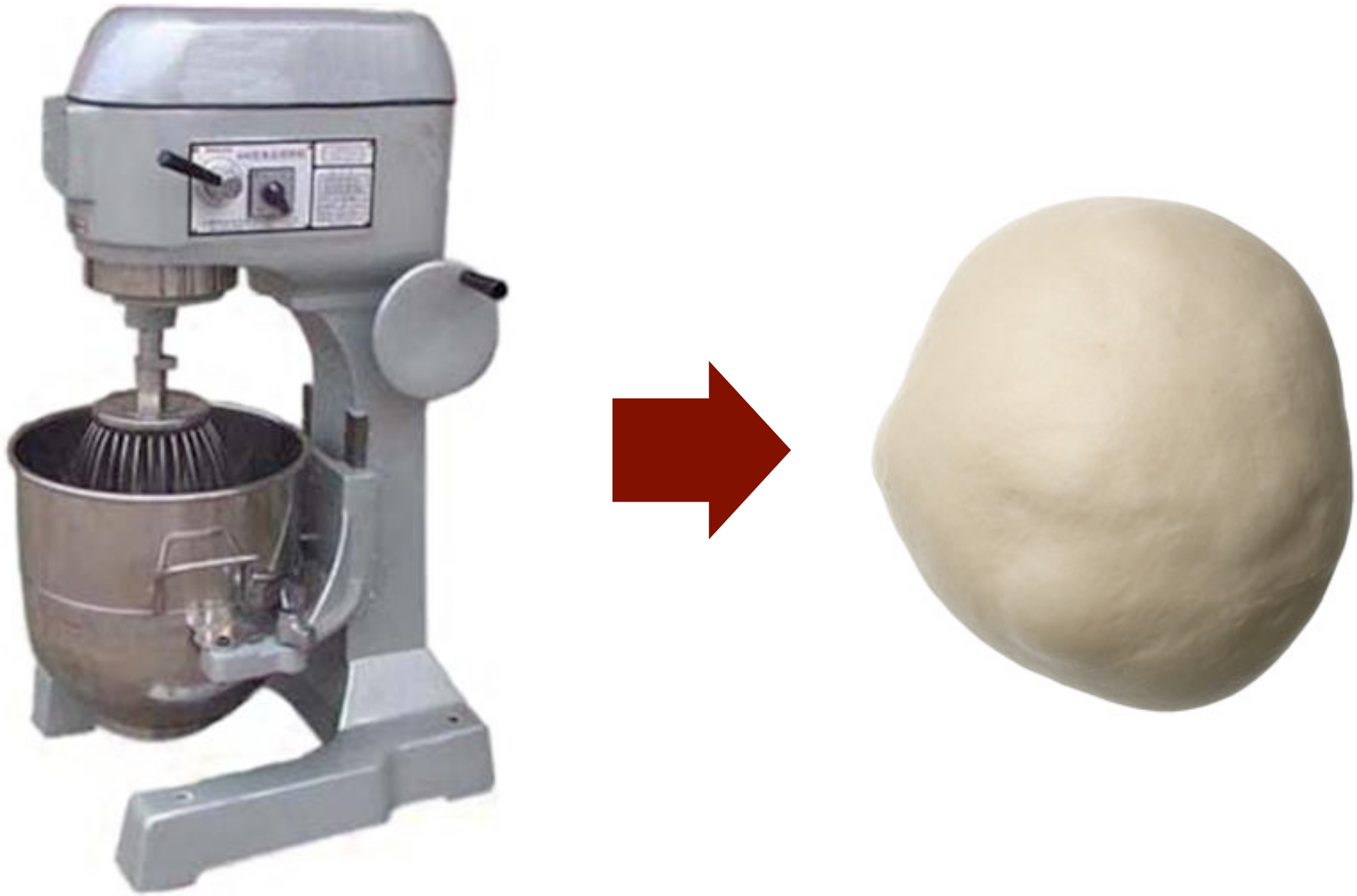
The Pillars of Continuous Delivery



The Deployment Pipeline

"The deployment pipeline has its foundations in the process of continuous integration and is in essence ***the principle of continuous integration taken to its logical conclusion****."*





Continuous Integration

Continuous Delivery

The Deployment Pipeline

Figure 1.1 *The deployment pipeline*

The Deployment Pipeline

"Every change that is made to an application's configuration, source code, environment, or data, ***triggers the creation of a new instance of the pipeline****.*

One of the first steps in the pipeline is to ***create binaries and installers****.*

The rest of the pipeline ***runs a series of tests*** *on the binaries to prove that they can be released. Each test that the release candidate passes gives us more confidence that this particular combination of binary code, configuration information, environment, and data will work.*

If the ***release candidate*** *passes all the tests, it can be* ***released****."*

The Deployment Pipeline

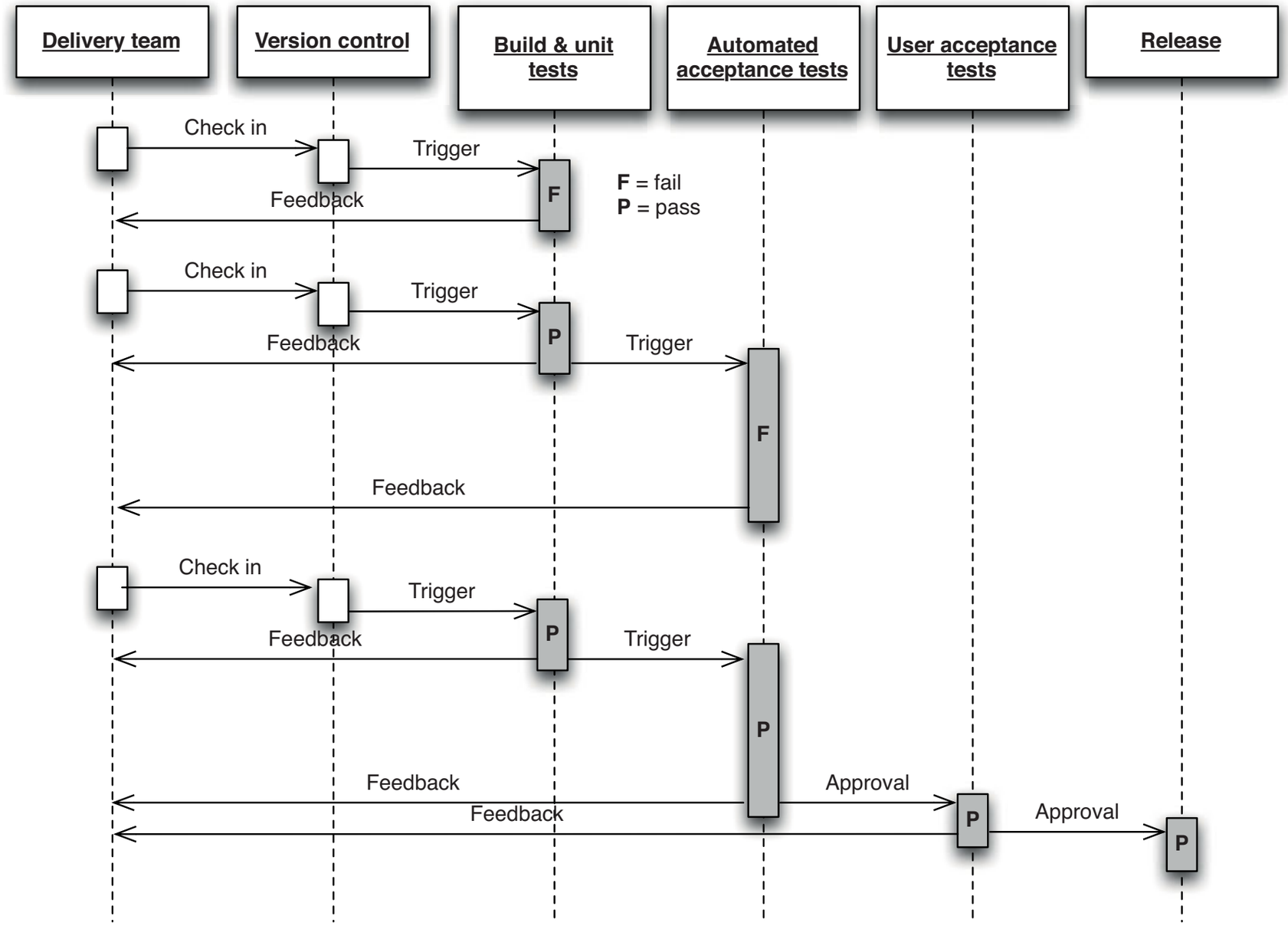


Figure 5.2 *Changes moving through the deployment pipeline*

The Deployment Pipeline

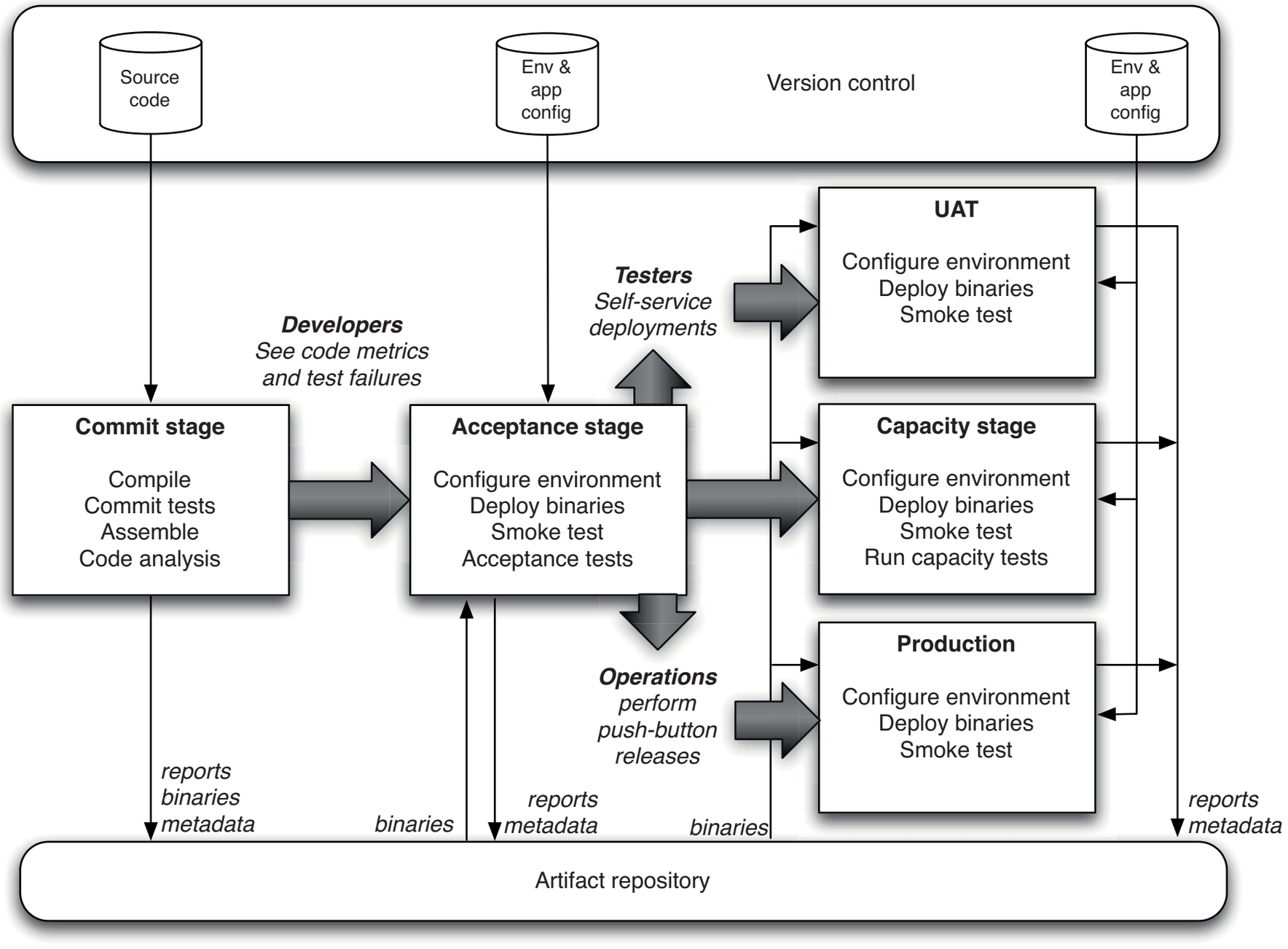


Figure 5.4 *Basic deployment pipeline*

The Commit Stage

- The commit stage begins with a change to the state of the project—that is, a **commit to the version control system**.
- It **ends with either a report of failure or**, if successful, **a collection of binary artifacts and deployable assemblies** to be used in subsequent test and release stages, as well as reports on the state of the application.
- Ideally, a commit stage should **take less than five minutes to run**, and certainly no more than ten.
- The commit stage represents, in more ways than one, **the entrance into the deployment pipeline**. Not only is it the point at which a new release candidate is created; it is also where many teams start when they begin to implement a deployment pipeline. **When a team implements the practice of continuous integration, it creates a commit stage in the process of doing so.**

Implementing a Deployment Pipeline

- **Take an agile approach**. Start simple, iterate and improve on a continuous basis.
- Consider the following elements:
  - Automate the **build** and **deployment** process.
  - Automate **unit tests** and **code analysis**.
  - Automate **acceptance tests**.
  - Automate **releases**.

Tools

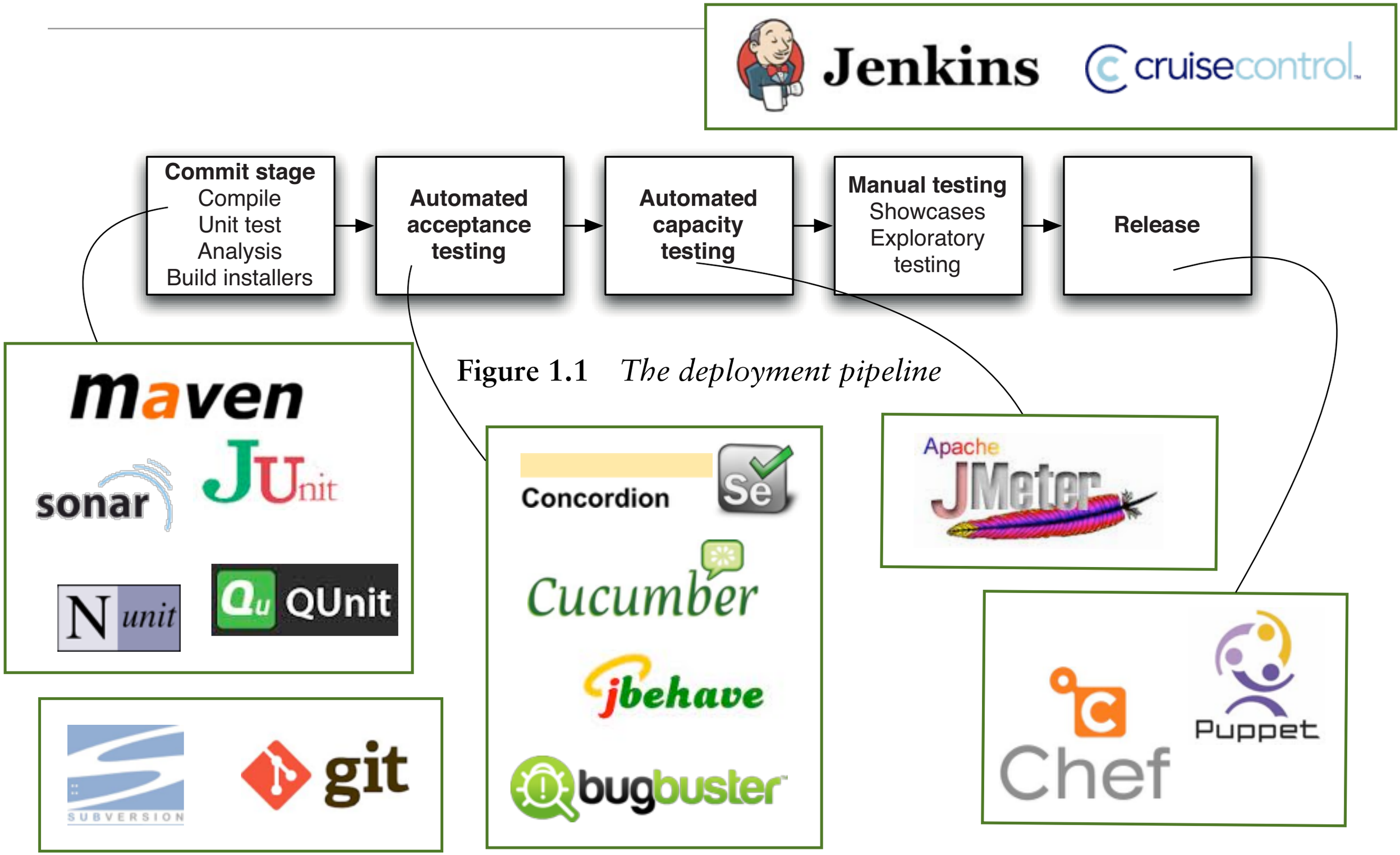
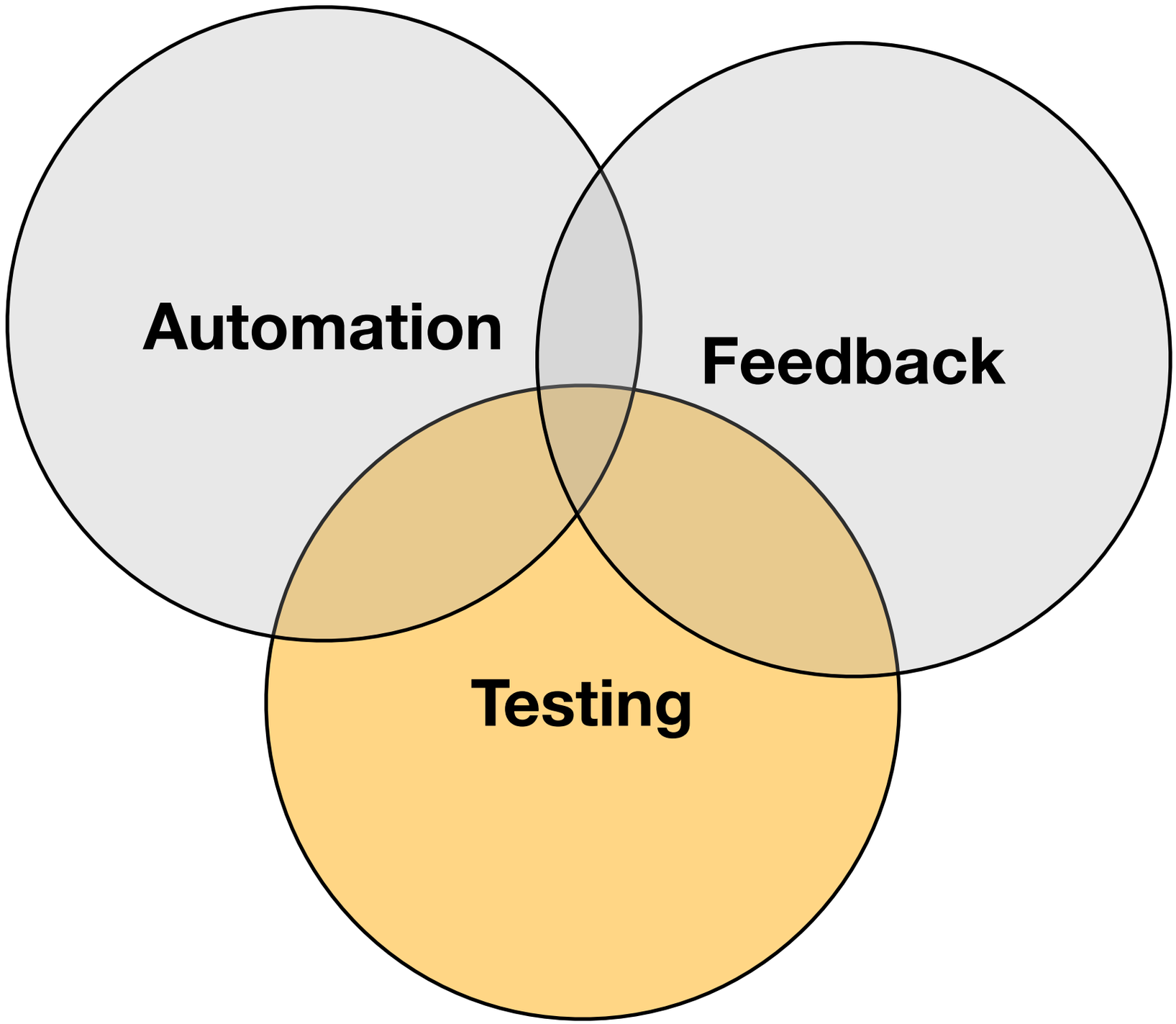


Figure 1.1 *The deployment pipeline*

The Pillars of Continuous Delivery



Testing

“In many projects, testing is treated as a distinct phase carried out by specialists.

However, high-quality software is only possible if ***testing becomes the responsibility of everybody involved in delivering software*** *and is practiced right from the beginning of the project and throughout its life.”*

Testing

*"Testing is primarily concerned with **establishing feedback loops** that drive development, design, and release.*

*Any plan that defers testing to the end of the project is broken because it removes the feedback loop that generates higher quality, higher productivity, and, most importantly of all, any **measure of how complete the project is**."*

Types of Tests

- **Unit** Tests
- **Integration** Tests
- **Automated Acceptance** Tests
- **Systemic Quality** Tests
  (performance, capacity, security)
- **User Acceptance** Tests

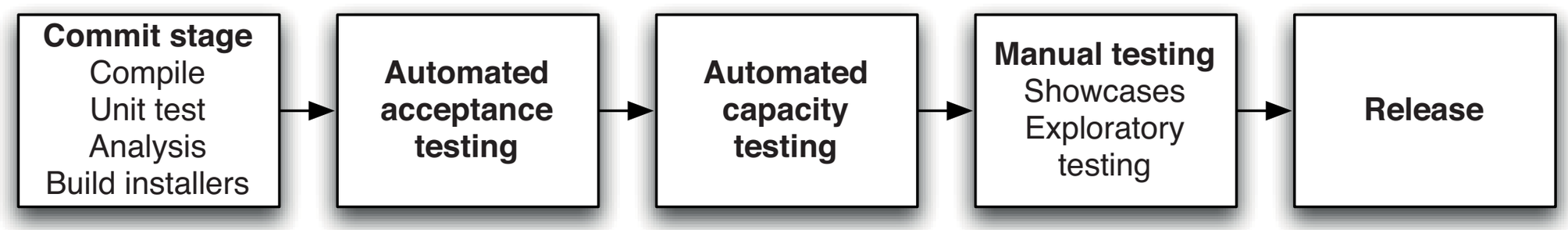
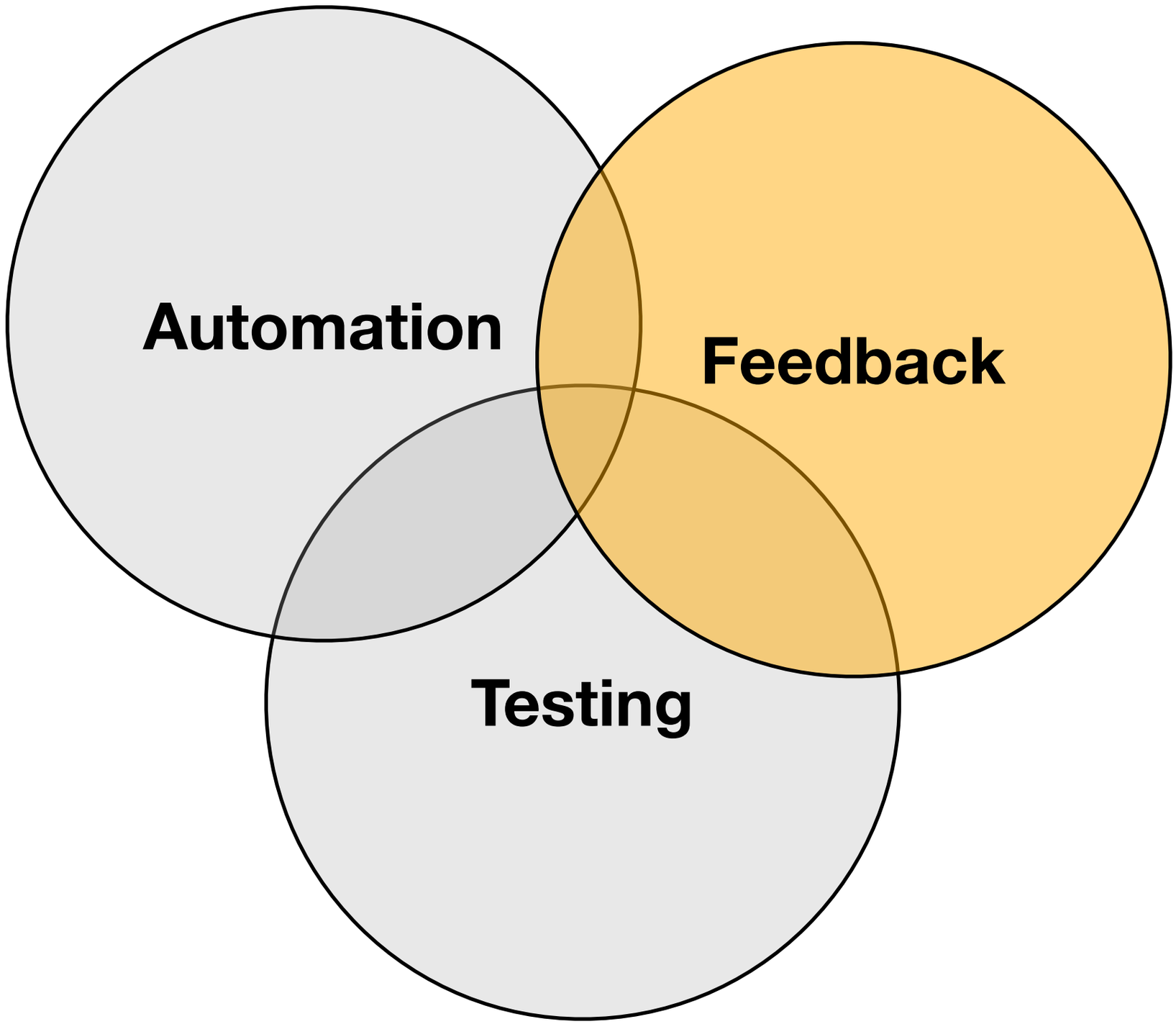


Figure 1.1 *The deployment pipeline*

The Pillars of Continuous Delivery



Feedback

- Feedback is essential to frequent, automated releases. There are three criteria for feedback to be useful:
  - Any change, of whatever kind, needs to **trigger** the feedback process.
  - The feedback must be **delivered as soon as possible**.
  - The delivery team must **receive feedback** and then **act on it**.

Broadcasting Feedback

*"You should **measure continually and broadcast the results** of the measurements in some hard-to-avoid manner, such as on a very visible poster on the wall, or on a computer display dedicated to showing bold, big results.*

*Such devices are known as **information radiators**."*

What to Measure?

*"What you choose to measure will have an enormous **influence on the behavior** of your team. According to the lean philosophy, it is essential to **optimize globally, not locally**.*

*So it is important to have a global metric that can be used to determine if the **delivery process as a whole** has a problem."*

Is it a good idea to measure lines of code?
Or the number of bugs fixed?

What to Measure?

*"For the software delivery process, the most important global metric is **cycle time**. This is the time between deciding that a feature needs to be implemented and having that feature released to users."*

If the pipeline implementation makes it possible how the cycle time is spent on the successive phases, then it is possible to find bottlenecks and to fix them.

Happy
DevOps Family

Summary

*"There is **no one-size-fits-all solution** to the complex problem of implementing a deployment pipeline.*

*The crucial point is to create a system of record that **manages each change from check-in to release**, providing the **information** you need to discover problems as early as possible in the process."*

Summary

*"Having an implementation of the deployment pipeline can then be used to **drive out inefficiencies** in your process so you can make your feedback cycle faster and more powerful, perhaps by adding more automated acceptance tests and parallelizing them more aggressively, or by making your testing environments more production-like, or by implementing better configuration management processes."*

Summary

"A deployment pipeline, in turn, depends on having some ***foundations*** *in place:* ***good configuration management, automated scripts for building and deploying your application, and automated tests*** *to prove that your application will deliver value to its users.*

It also requires ***discipline****, such as ensuring that only changes that have passed through the automated build, test, and deployment system get released."*

Resources

- http://continuousdelivery.com