Team collaboration with Test-First Approach

Etienne Pierrot

Senior Software Engineer

Zlata Shtamburg

Senior Software Engineer In Test

checkout.com

Our offices

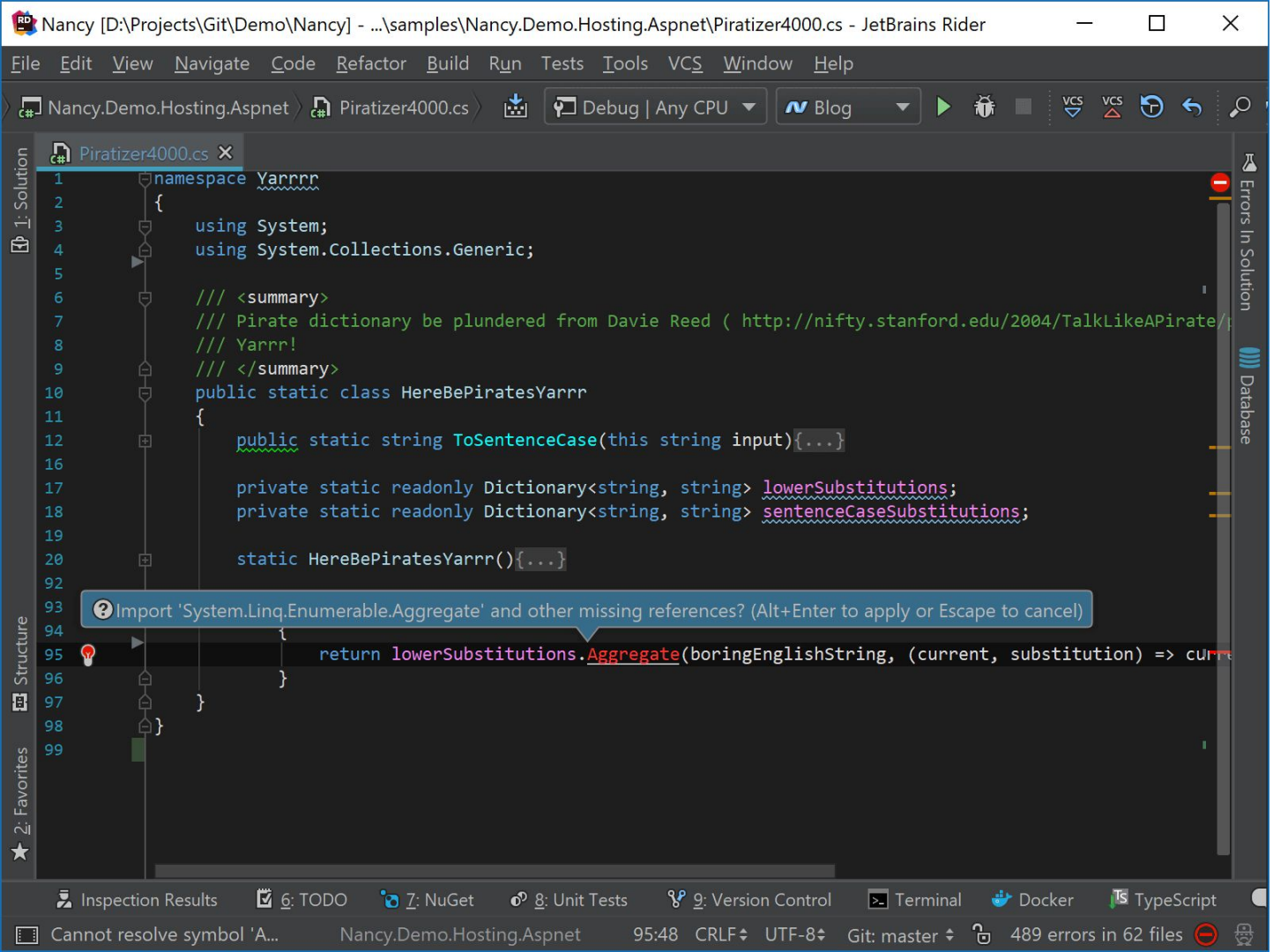
Goals

1. Quick Feedback loop
2. Share QA mindset with overall team
3. Microservices testing

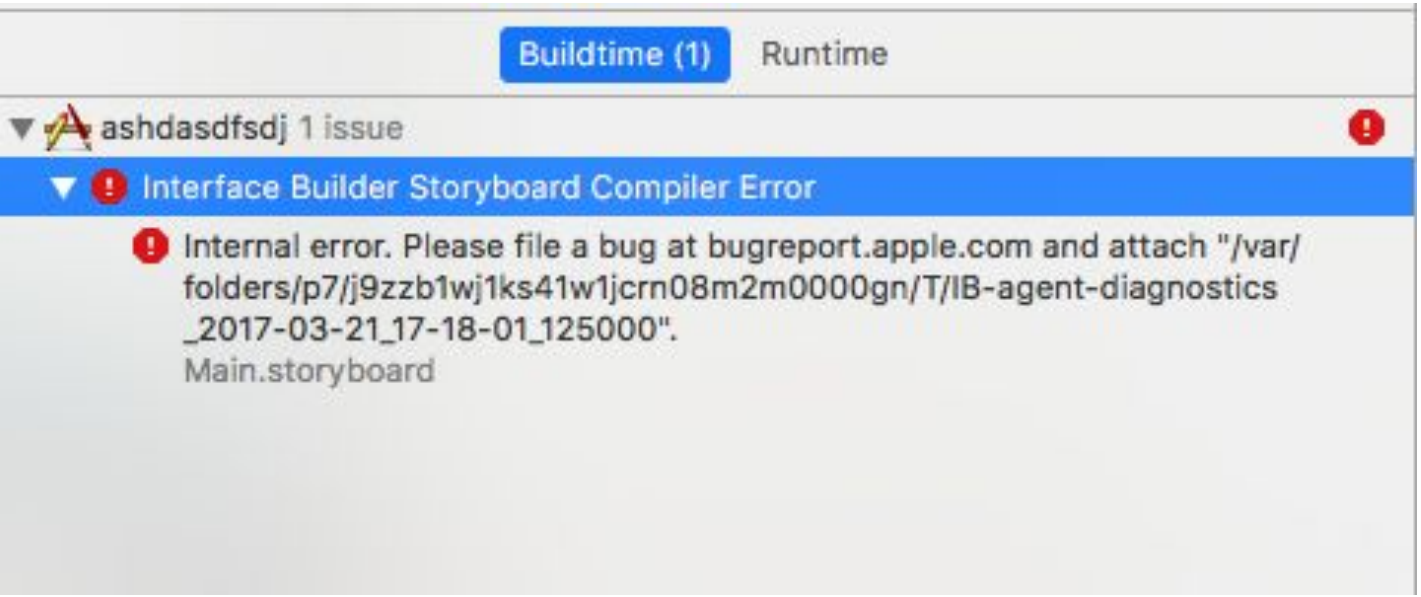
Feedback loop

Feedback is the fuel of the development process

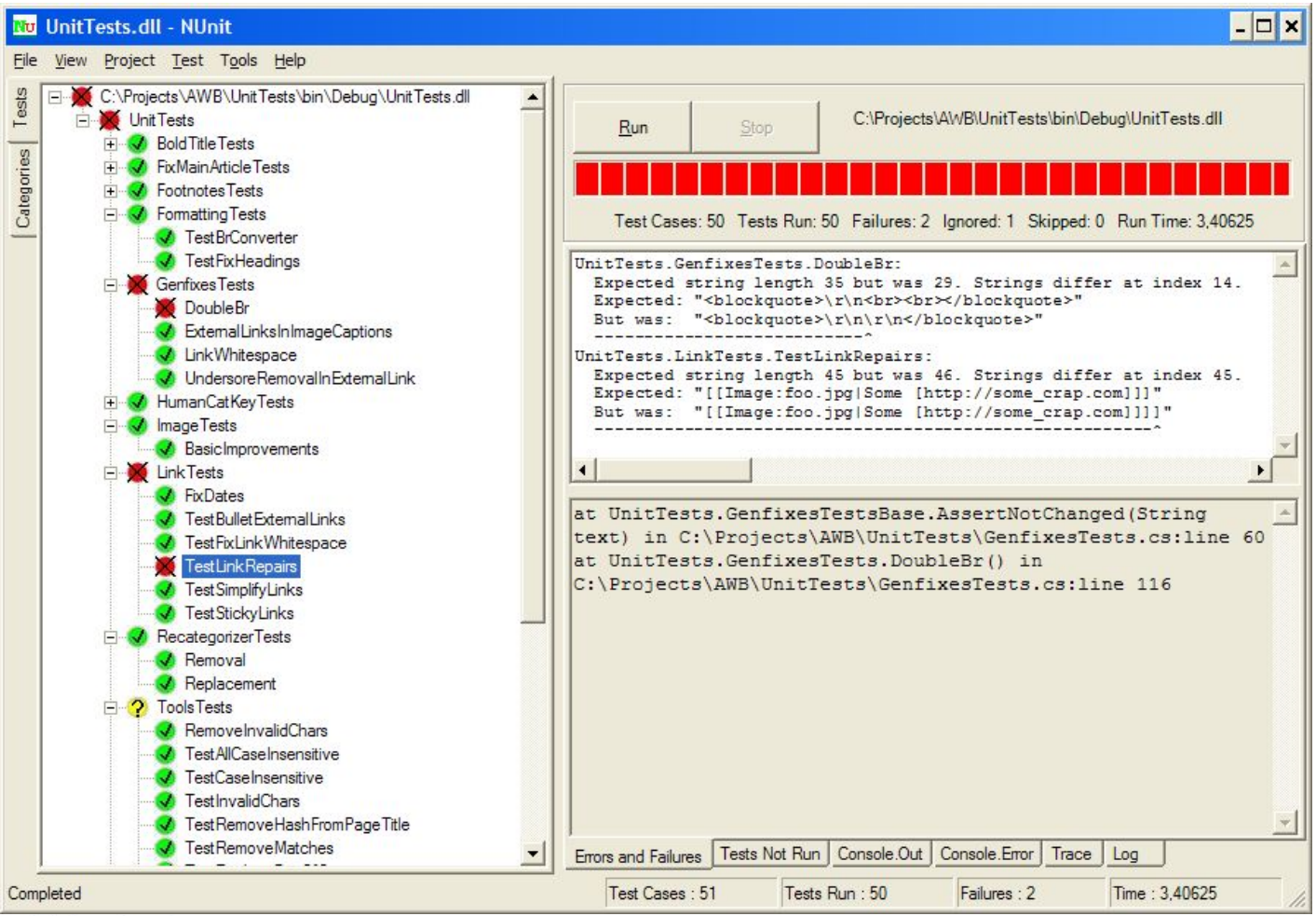
LOOP FEEDBACK
LOOP FEEDBACK EVERYWHERE



Feedback duration:
almost instant



Feedback duration:
few seconds

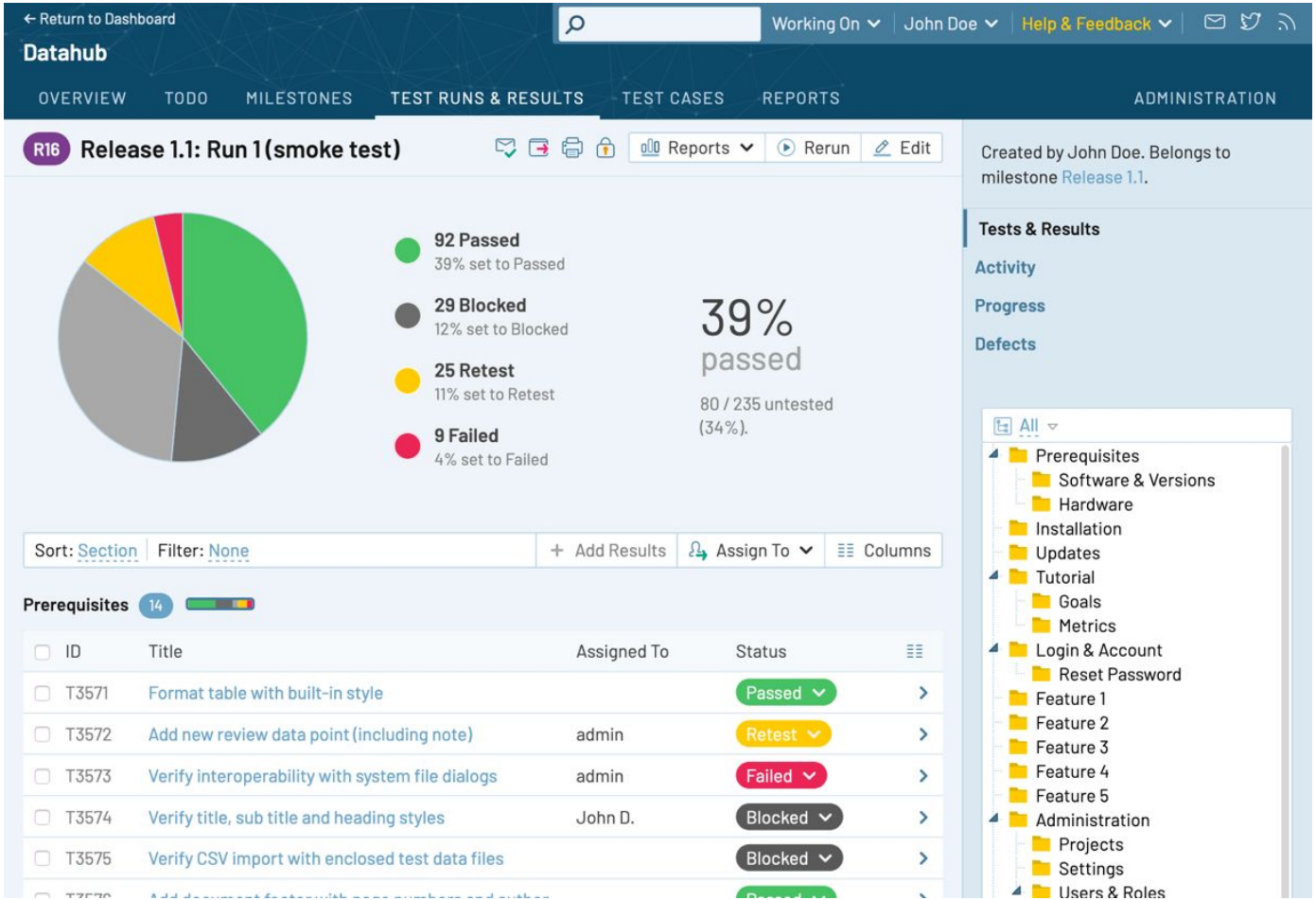


Feedback duration:
few seconds/minutes

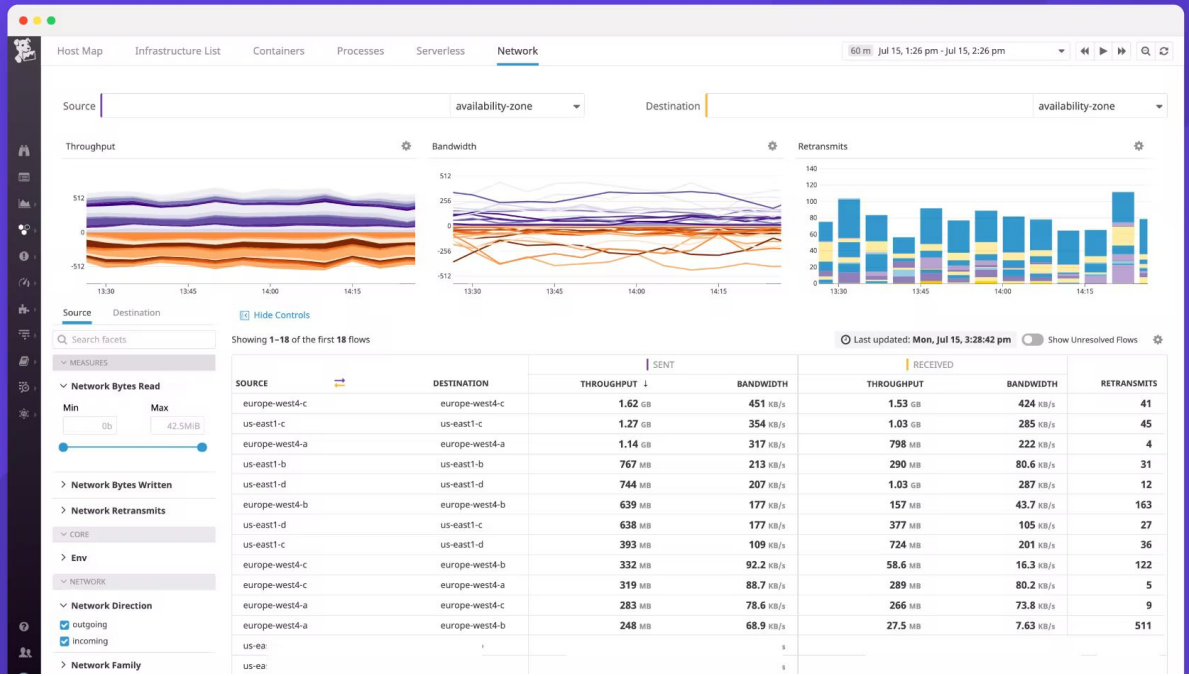
Feedback duration: hour(s)

Jenkins

Feedback duration:
minutes - hour(s)



Feedback duration: minutes - weeks



Feedback duration:
hours - month

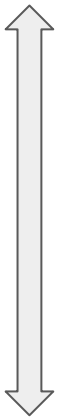
checkout.com

Feedback duration:
days - years

Summary

IDE Syntax lighting	almost instant
Compilation	seconds/minutes
Unit tests	seconds/minutes
Peer review	hour(s)
Continuous integration	minutes - hour(s)
QA validation	minutes - weeks
Production logs and metrics	hours - month
Consumers feedback	days - years

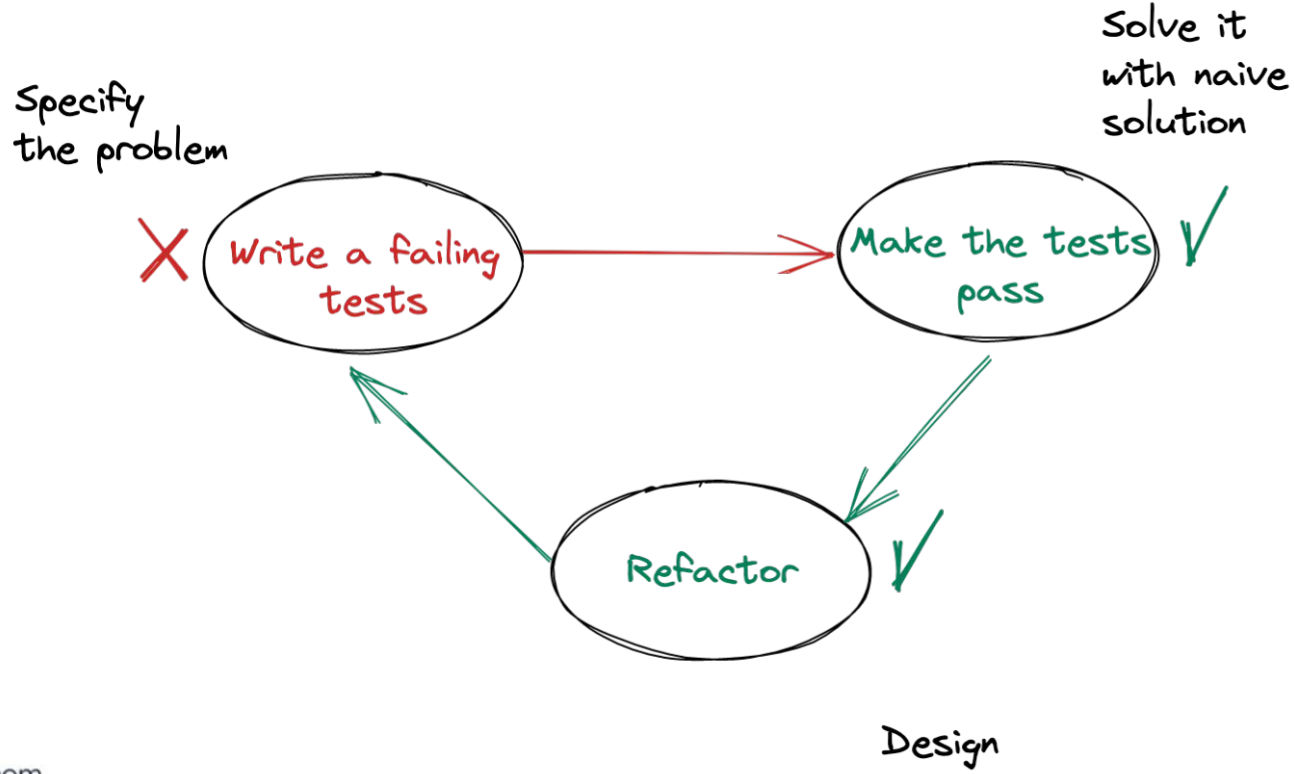
Summary



IDE Syntax lighting	almost instant
Compilation	seconds/minutes
Unit tests	seconds/minutes
Peer review	hour(s)
Continuous integration	minutes - hour(s)
QA validation	minutes - weeks
Production logs and metrics	hours - month
Consumers feedback	days - years

Test First approach

The TDD loop

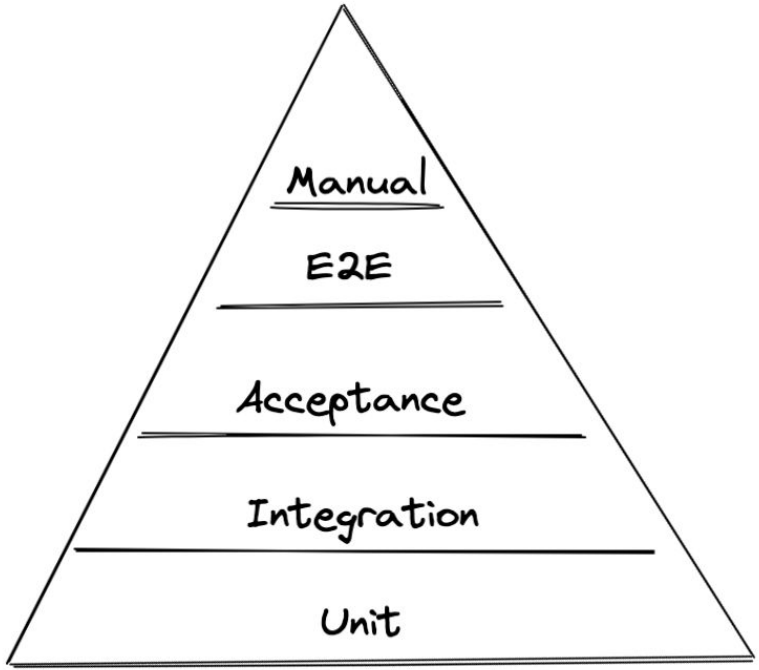

Specify the problem
Write a failing tests
Make the tests pass
Solve it with naive solution
Refactor
Design

TDD

- Push developer to thinking more about the **problem** to solve before jumping to implementation
- Produce **minimalistic** solution focus on the problem to solve
- Progress with **baby step**
- Allow developer to have **confident** about refactoring and improving design

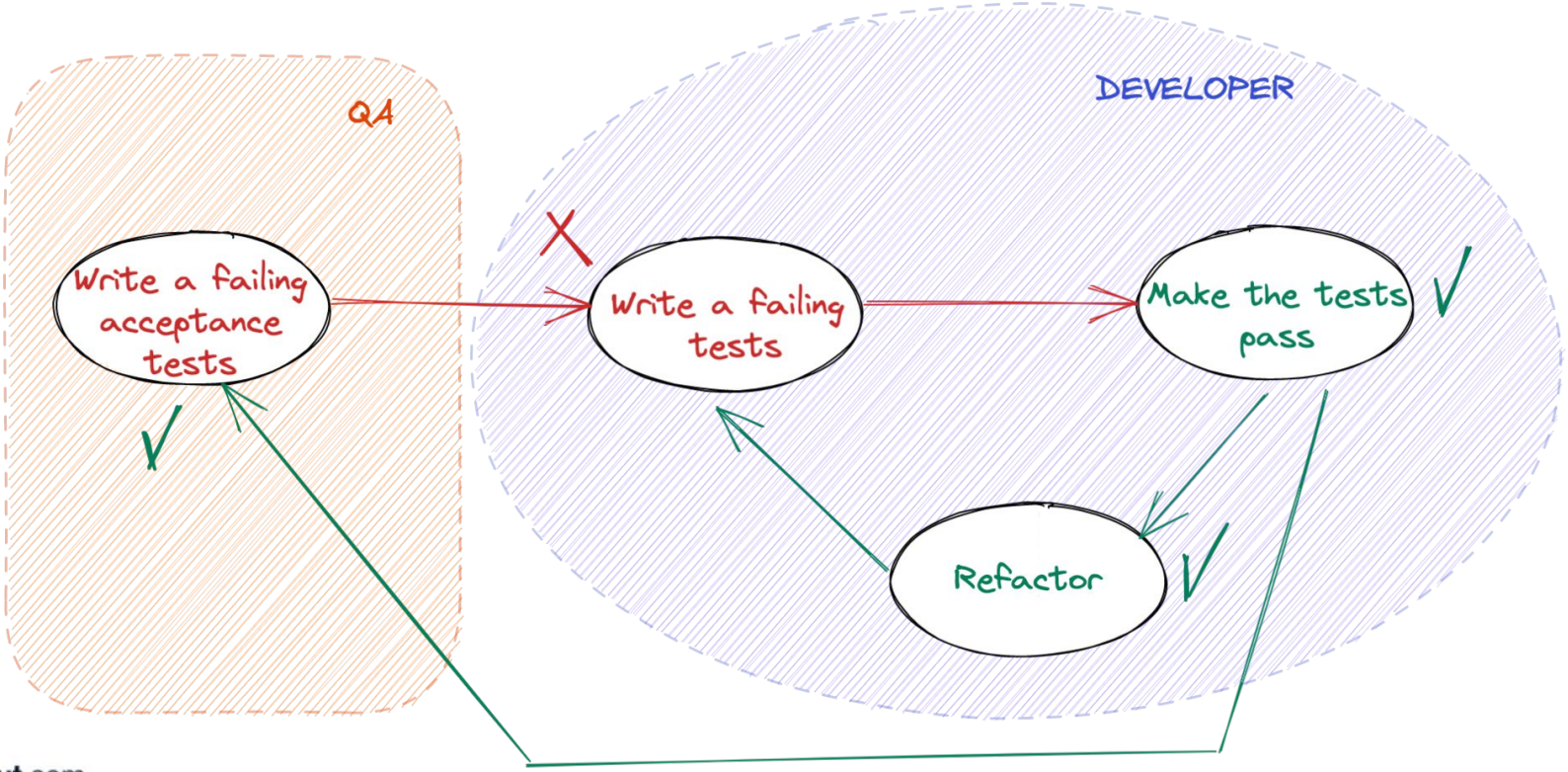
But TDD alone doesn’t ensure a shared understanding of the feature

What is Acceptance tests

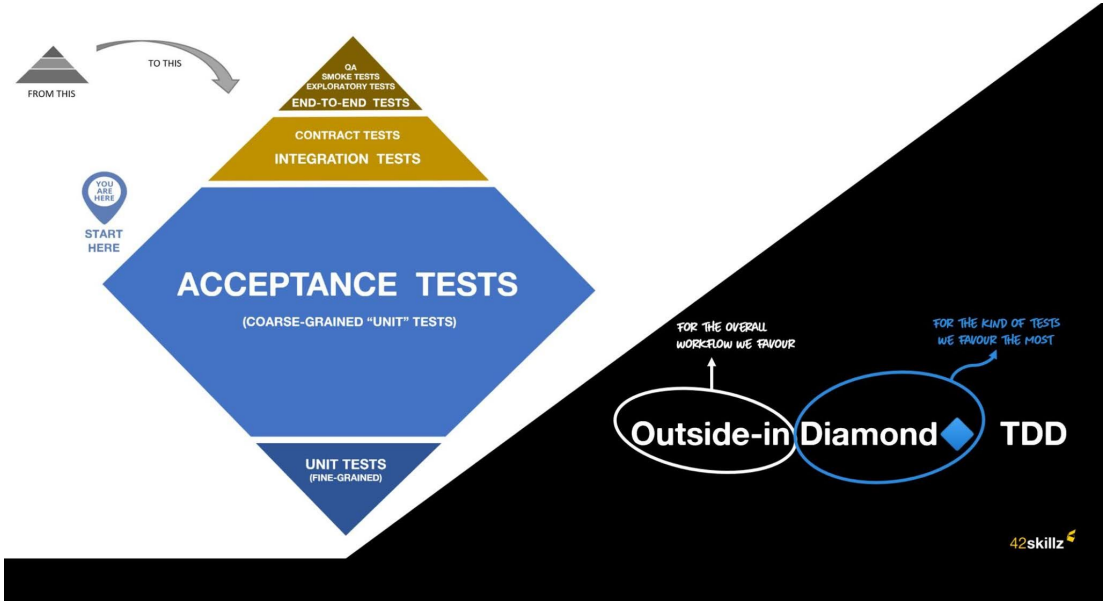


Given Pier has a 10 EUR
When Pier buy a chocolate bar for 2 EUR
Then Pier has a chocolate bar
And Pier has a 8 EUR

Acceptance TDD

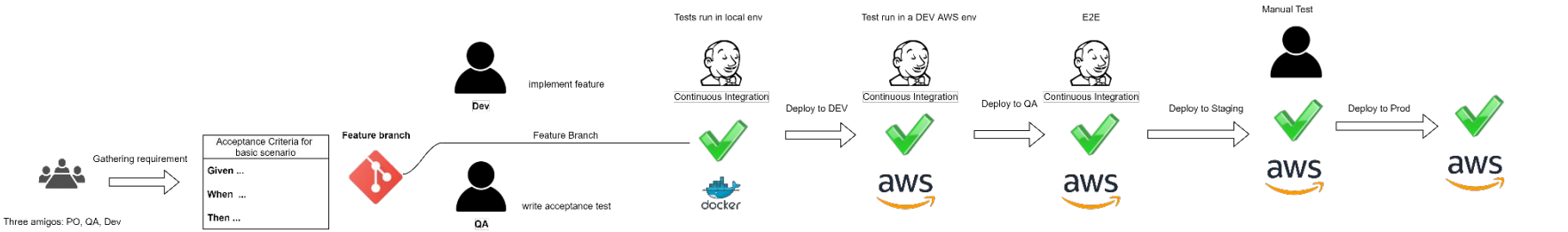


To know more about that :



http://tpierrain.blogspot.com/2021/03/outside-in-diamond-tdd-1-style-made.html

Lifecycle of a feature



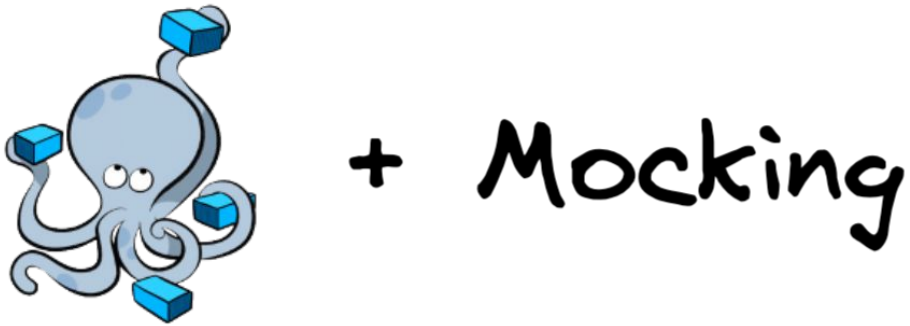
Devops

Goal: Ability to run the platform on local environment

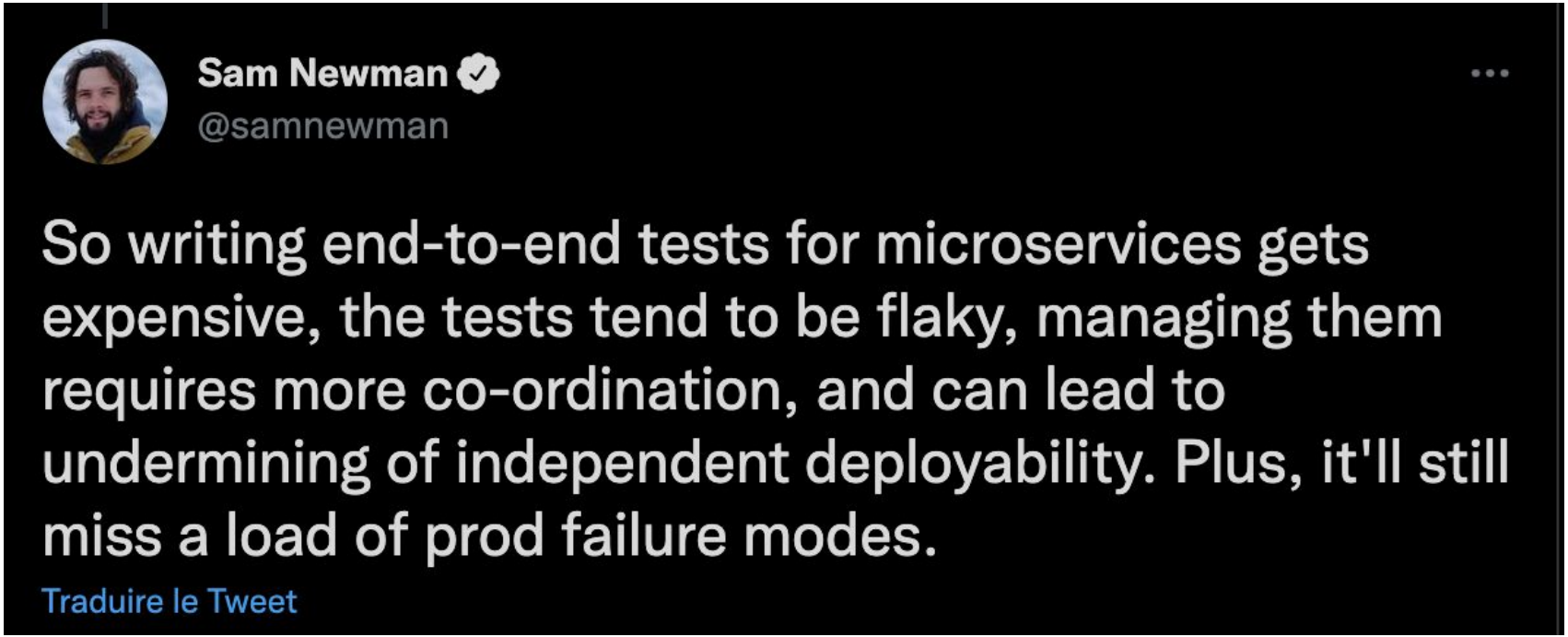
Potential problems:

1. Different OS used in the team
2. Heavy application / a lot of services

Solution:

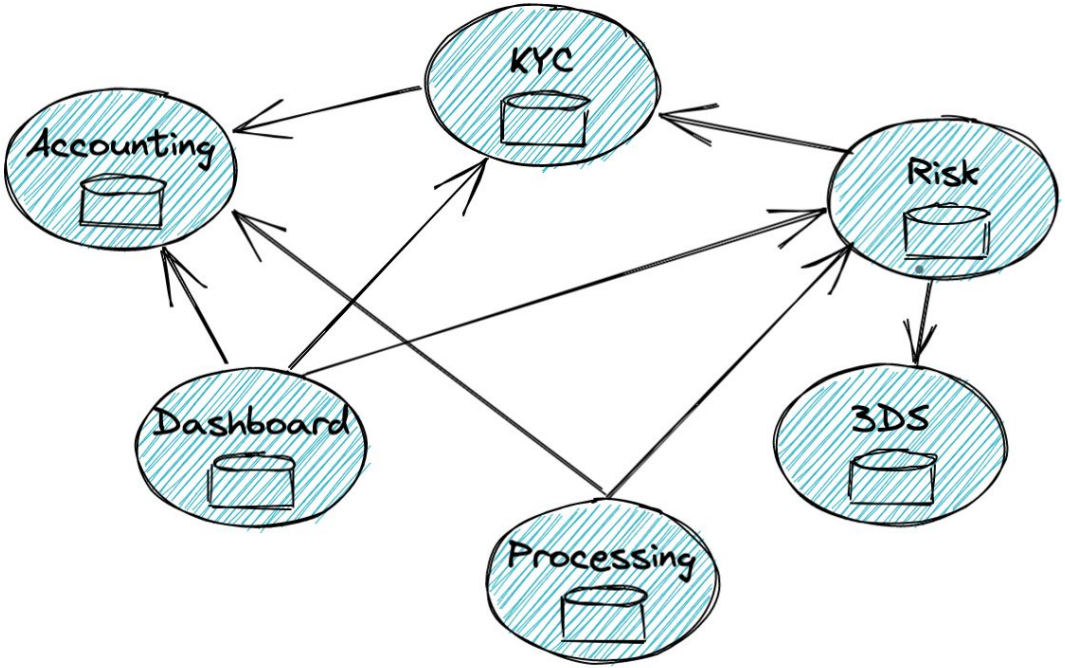


End to end testing and Microservices

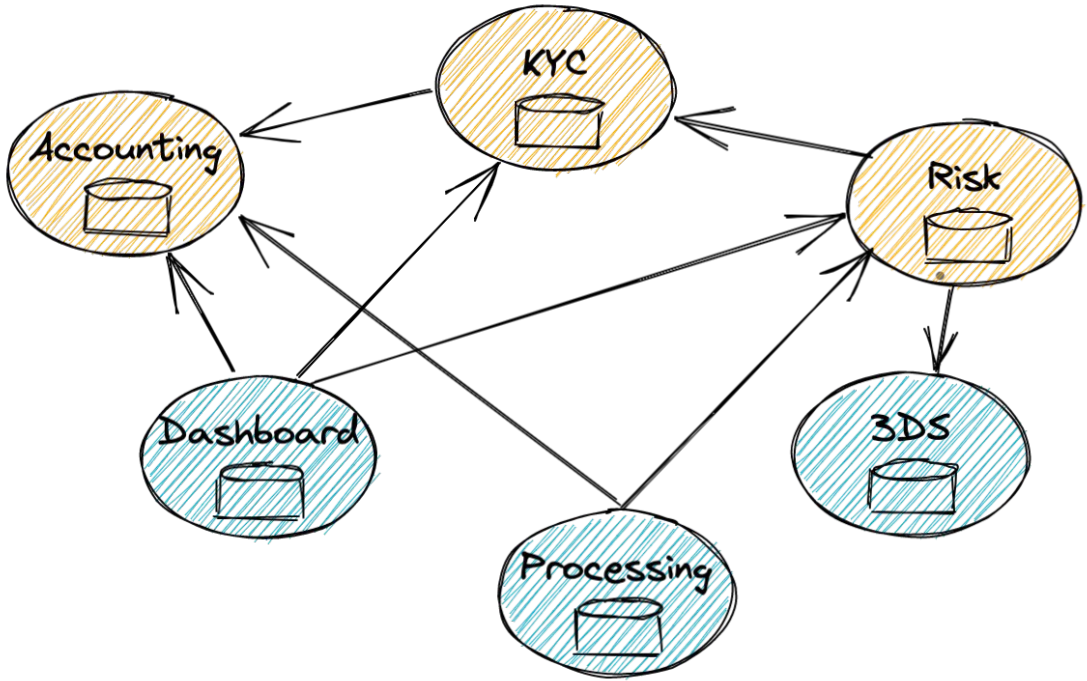


https://www.twitter.com/samnewman/status/1357642975271124992

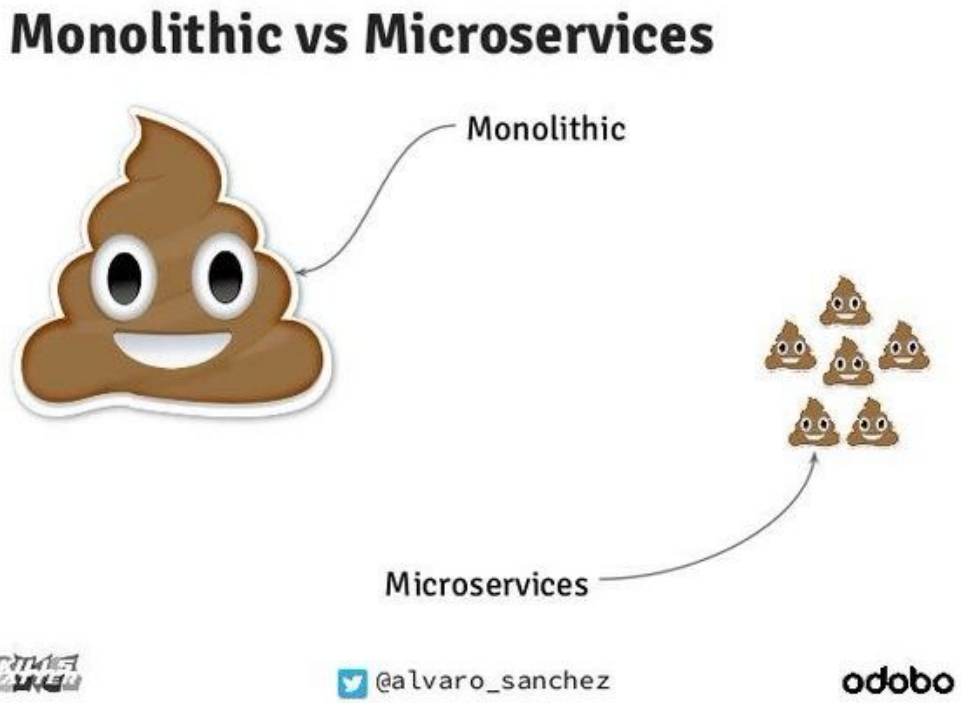
Microservices pros/cons



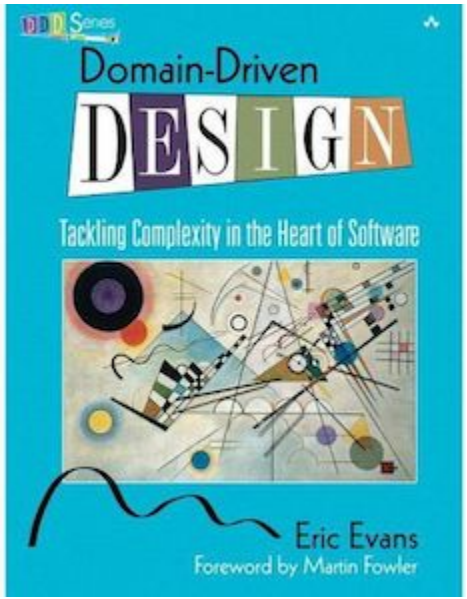
Microservices pros/cons



What is a good amount of microservices?



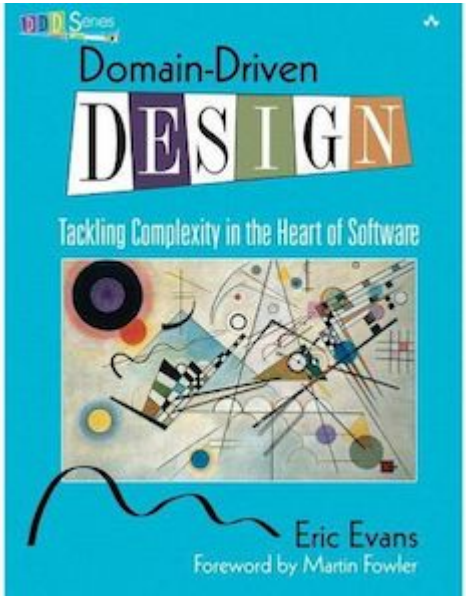
Domain Driven Design to the rescue



Bounded context

Eric Evans définition:

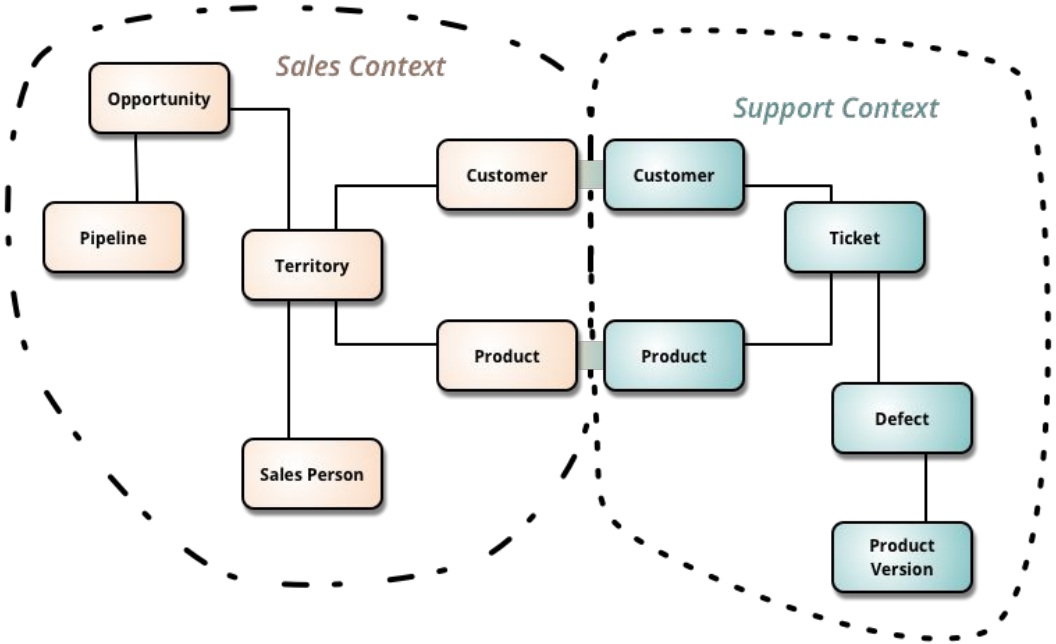
A BOUNDED CONTEXT delimits the applicability of a particular model so that team members have a clear and shared understanding of what has to be consistent and how it relates to other CONTEXTS. Within that CONTEXT, work to keep the model logically unified, but do not worry about applicability outside those bounds. In other CONTEXTS, other models apply, with differences in terminology, in concepts and rules, and in dialects of the UBIQUITOUS LANGUAGE. By drawing an explicit boundary, you can keep the model pure, and therefore potent, where it is applicable. At the same time, you avoid confusion when shifting your attention to other CONTEXTS. Integration across the boundaries necessarily will involve some translation, which you can analyze explicitly.



Bounded context quickly

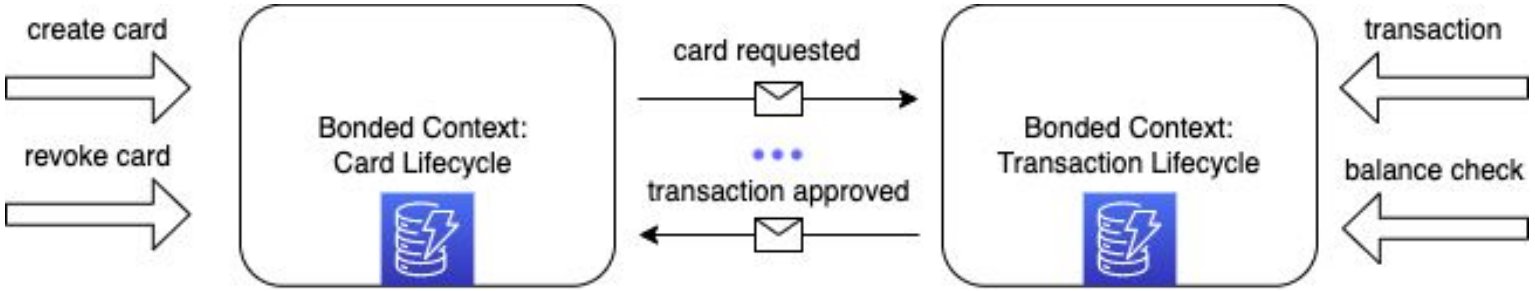
- Prefer to deal with multiple models rather than one huge model
- Each bounded context has his own **ubiquitous language**
- In general, bounded context match with **sub domains**
- Should be driven by the communication structure of the organization (cf **Conway's law**)
- The boundary are implemented with a set of explicit public **contracts**
- **This boundary is not necessarily a network boundary. The boundary can be inside a monolith**

Bounded context



https://martinfowler.com/bliki/BoundedContext.html

Discovering your domain boundaries



Acceptance test Example

Given single use card is created
When transaction is approved
Then card is revoked

To conclude about microservices

- Microservices can be interesting for socio-technical purpose
- Boundaries / contracts of your microservices / bounded context should be clearly defined
- Prefer rely on Acceptance Test at Bounded Context level rather than E2E testing

Conclusion

- This practices (AT and Bounded Context) help us to have to reduce the feedback between dev and qa
- Boundaries between QA and Developers are more or more blurry
- In our situation, QA validation isn’t anymore the main limitation to release very frequently
- Beware to split too early your system into many microservices
- Feature validation at very early stage improve our ability to release more frequently
- Feedback loop - good KPI to measure

Q/A