

CS1 - The Journey to Microservices from a Startup Perspective

Group 5

Pedro Correia, No. 81002

Nuno Gonçalves, No. 81082

Sílvia Fernandes, No. 81041

Nuno Santos, No. 81703

Pedro Monteiro, No. 90850

Introduction



Susanne Kaiser, CTO at Just Software
Responsible for the software development of JUST SOCIAL

Goal: Increase and improve collaboration and communication in organizations, and support teams working together more efficiently



Susanne Kaiser is CTO at Just Software, a Hamburg, Germany based startup. She is responsible for the software development of JUST SOCIAL - providing apps for collaboration and communication in organizations.

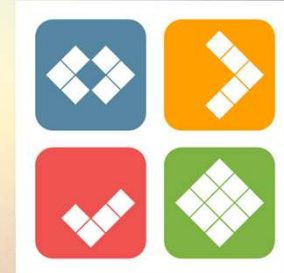
Just Social's goal is to increase and improve collaboration and communication in organizations, and to support teams working together more efficiently.

Just Social

As the project began to grow, user experience, usability and team productivity started to be affected

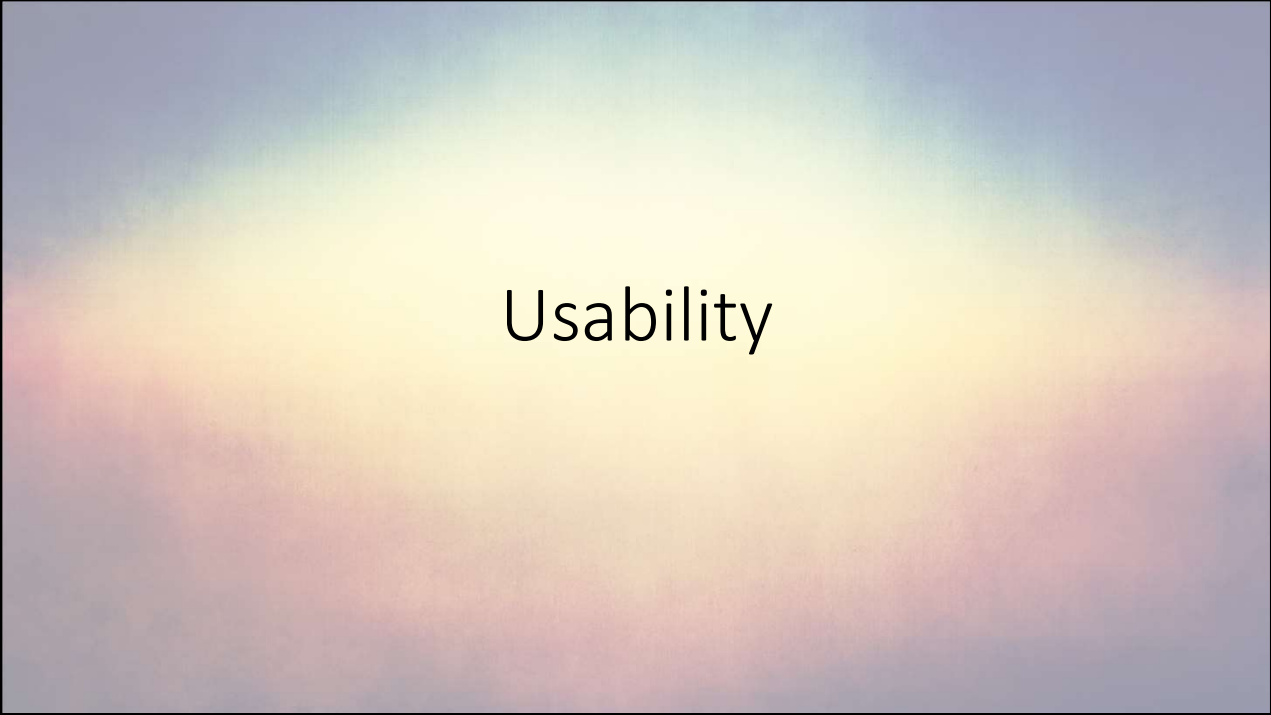
Features took longer to be developed and deployed, also affecting time to market

Decided to change the architecture by dividing the monolith into several apps



Just Social is composed of several apps, but this hasn't always been the case. They started with a monolith solution that “worked for a while until everything became clunky”. As the number of developers, users and features grew the development process took longer, due to complexity and communication issues and so users started to feel the effects in usability.

Focused on usability and user experience, they decided to switch from a monolith architecture to a new microservices solution. This new architecture would have to respect and improve a few significant requirements.



Usability

Usability

- Stakeholders : Users
- Needed features are missing
- New features and bug fixes take too long to be released
- Microservices approach and Conway's law
- Usability and Modifiability go in hand

Why is the system less usable? Needed features are missing and take too long to be implemented due to the system needing to be entirely redeployed.

How do they make it more usable? Microservices and the structure they bring: have independent teams with well defined responsibilities focused on one or two apps which results in quick development and deployment.

Usability Scenario and Tactic

- Source of Stimulus: End Users
- Stimulus: Users want to use the system efficiently
- Environment: Runtime
- Artifacts: System (user interface)
- Response: The system should provide the users with the features needed
- Response Measure: High User satisfaction
- Tactic : Support System Initiative > Maintain task model

Response: The UI the system provides should be bug-free, simple to use and have well organized features, which allows the anticipation of the users needs.

Response Measure: (It englobes time between new features, bug fixing and user interface complexity)

Tactic : Since the interface is now divided in several apps with a well defined scope, it is easier for users to find what they want, and for the app to determine what task the user is trying to accomplish.



Modifiability

Modifiability

- Stakeholders : Developers
- Why? Productivity, usability and user experience suffered
- How? Split the change in small steps
- Motivation? Develop and deploy independently to release changes quickly, reduce time to market
- In the case study they modify the system in small steps. Ex: Extract Web App, Business Logic and Data Storage

Why do we need change ? Productivity suffered, new features released slowly , usability and UX suffered.

How to change ? Separate collaboration apps, small autonomous teams with well defined responsibilities.

Challenges of change ? Different skills and tools required, core functionality is hard to untangle , you still have to take care of your existing system, transformation take longer than anticipated.

Motivation to change ? Product and organizational/culture driven, enabling autonomous teams with well defined responsibilities, develop and deploy independently to release changes quickly, reducing time to market.

Modifiability Scenario and Tactic

- Source of Stimulus: Developers
- Stimulus: Extract Web App from monolith
- Environment: Design Time
- Artifacts: Monolith, Web App, REST interface
- Response: The modification is made, tested and deployed
- Response Measures: Amount of time for the change to be made
- Tactic: Reduce Coupling -> Encapsulate the Web App

Reduce Coupling (decouple the Web App from the monolith)
Encapsulate (Create an interface to the monolith)



Testability

Testability

- Stakeholders : Developers
- Confirm that changes aren't breaking API
- Use Consumer driven contract and test to confirm
- Developer sandbox

Create dev sandbox for more efficient development (full access to the system, not needed to set up all the services, easier to test)

Consumer driven contract

- Define which parts of that provider contract currently support the business value realized by the system, and which do not
- Make sure that compatibility with former versions is maintained
- Introduce unit tests for the various expectations the clients have from the contract
- More info:
<https://martinfowler.com/articles/consumerDrivenContracts.html>

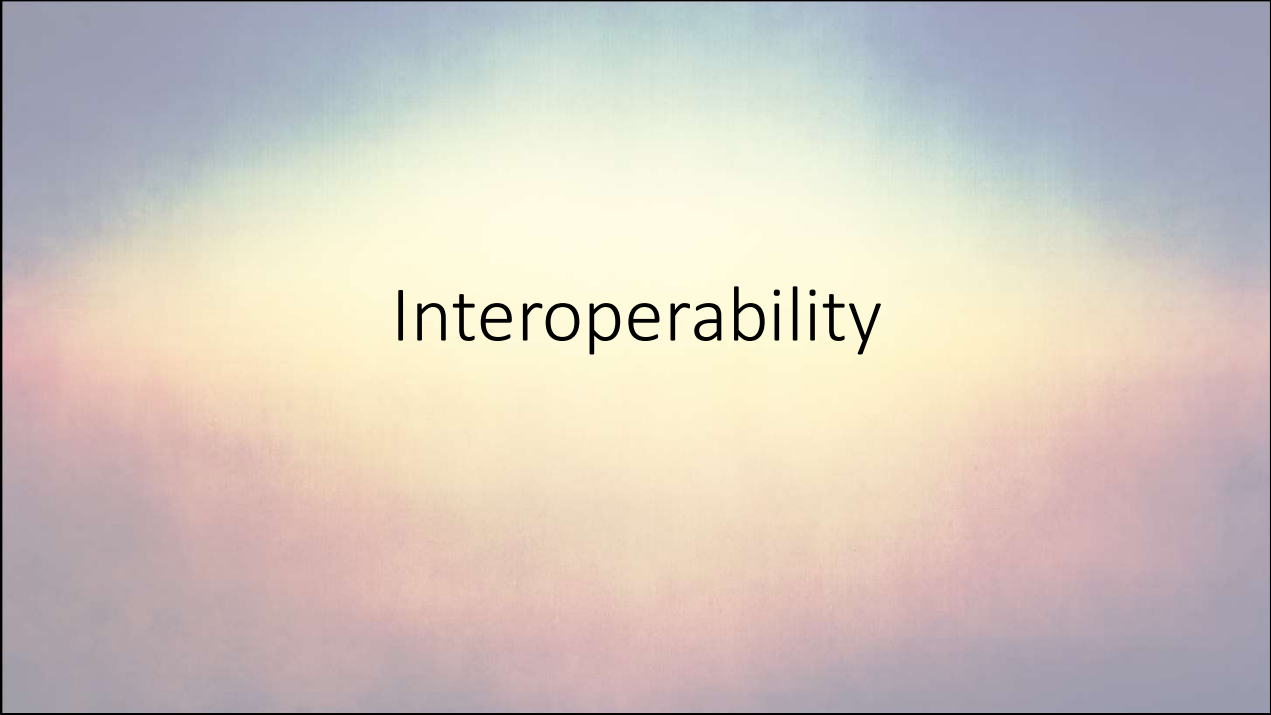
Testability Scenario and Tactic

- Source: Developers
- Stimulus: Unit tests to verify that the consumer contracts are still being fulfilled after a change to the API offered to the clients
- Environment: Development time
- Artifacts: System API
- Response: Execute test suite and capture results
- Response Measure: Time to perform tests
- Tactic : Developer Sandbox

Source could be developers or testers, not clear

Response and measure are not mentioned by the speaker

Developer sandbox : mentioned explicitly by speaker



Interoperability

Interoperability

- Stakeholders : Developers
- Communication between microservice and monolith during separation.
- Implemented API for communication using REST
- Centralized authorization system for the microservices

Communication involves both the webapp and the business interface

Auth system: lower coupling between services but creates a single point of failure

Interoperability Scenario and Tactic

- Source of Stimulus: Monolith requests updated information from the split microservice using the API
- Stimulus: Request sent to the microservice
- Artifacts: Monolith and microservice
- Environment: Both systems are known prior to runtime
- Response: The updated information is sent back to the monolith
- Response measure: Percentage of information correctly exchanged
- Tactic: Tailor Interface

The response and measure are not mentioned by the speaker. Assuming that these are used.

Tailor interface: because during the separation process new API operations are added for handling monolith requests