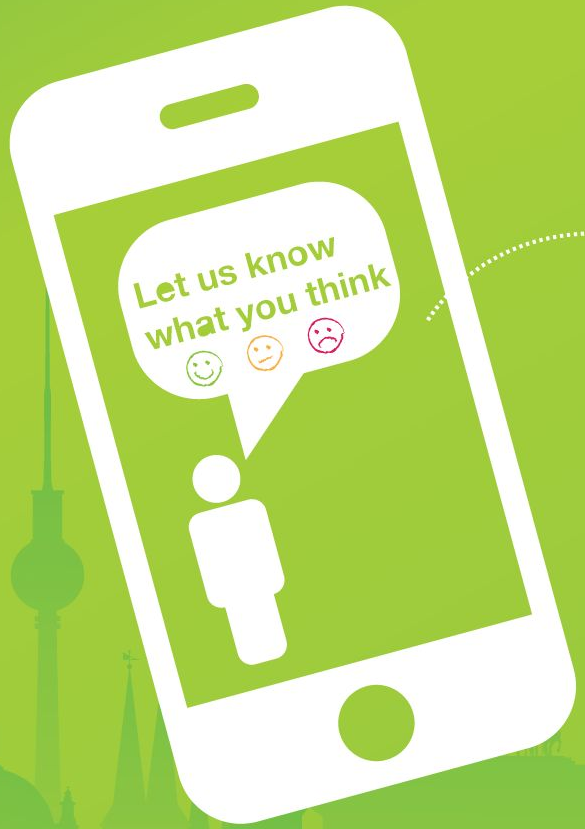
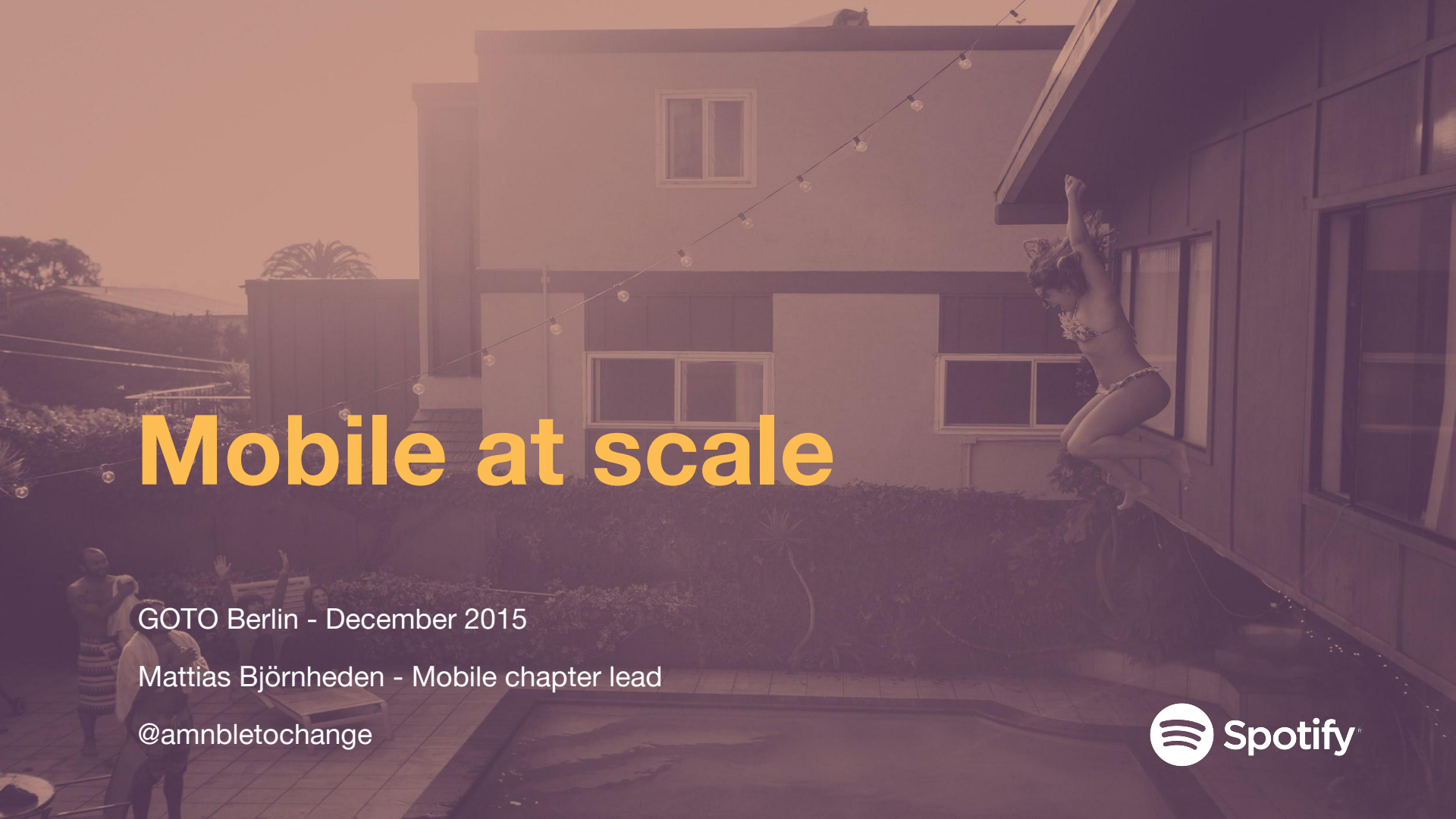
goto; conference



Click ‘engage’
to rate sessions
and ask questions

Follow us on Twitter @GOTOber

www.gotober.com



Mobile at scale

GOTO Berlin - December 2015

Mattias Björnheden - Mobile chapter lead

@amnbletochange

Spotify

Spotify

Founded 2006

75M+ Users

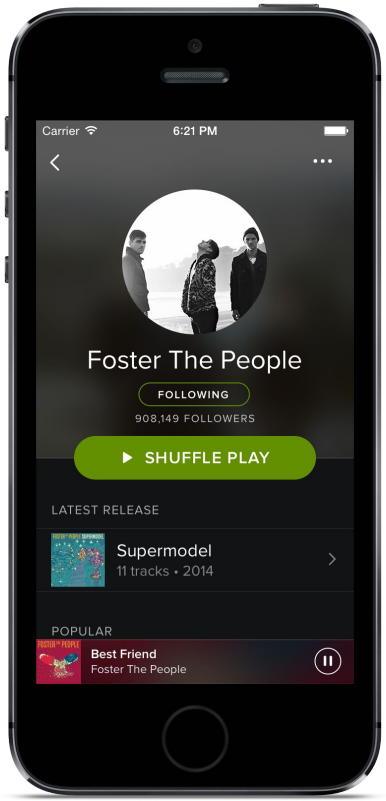
2+ Billion user playlists

58 Countries

1500+ employees

Spotify Mobile apps

- Majority of our users
- Offline capable
- 30+ developers on each platform
- 1000+ changes in each release



Building a small app

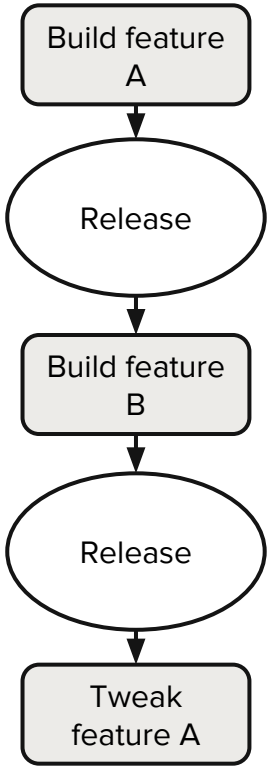
One team

“The mobile team”

Linear development

The team has capacity for one or two features at a time.

Release when ready.



Constant pressure

Just need to get feature x out and all will be great...

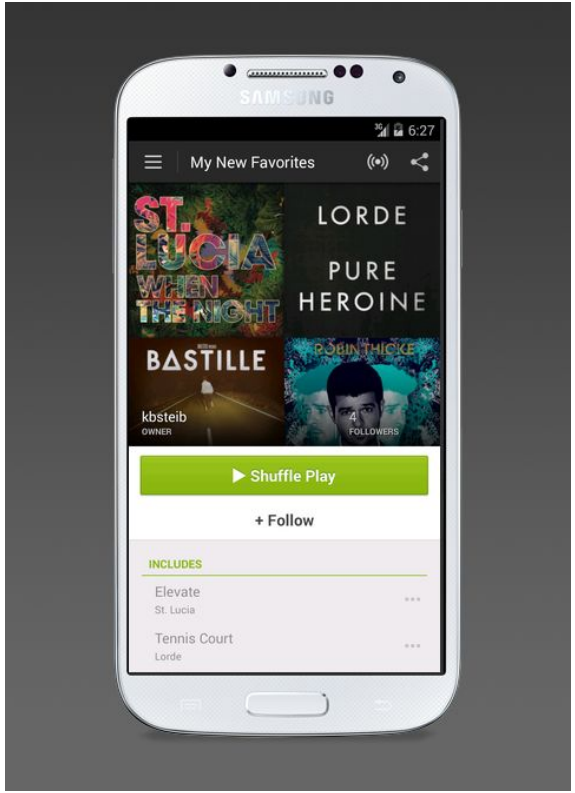
Shifting priorities

“We know x is almost done but right now we really need to work on Y”

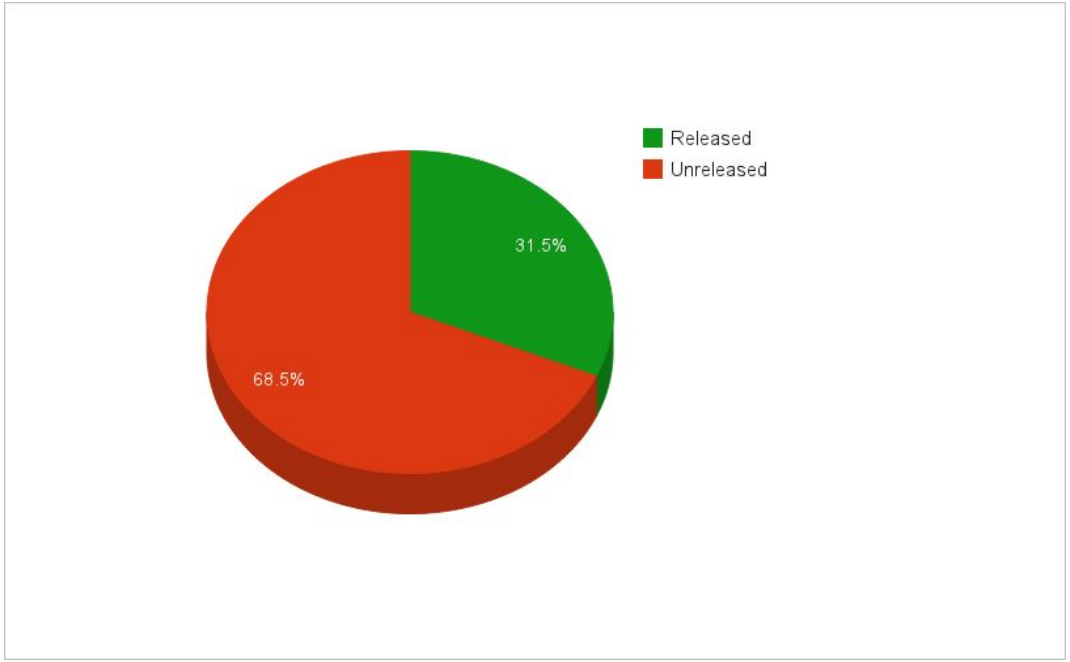
What about quality?

We used to be here

- Unknown, varying quality
- Manual tests
- Unpredictable releases
- A lot of abandoned work in progress



Half a year of work



There are solutions

- Agile
- Clear prioritization
- Continuous integration
- Feature flags
- Focus on testing and test automation

We implemented some of these

and also started thinking about....

Building a big app

Multiple teams

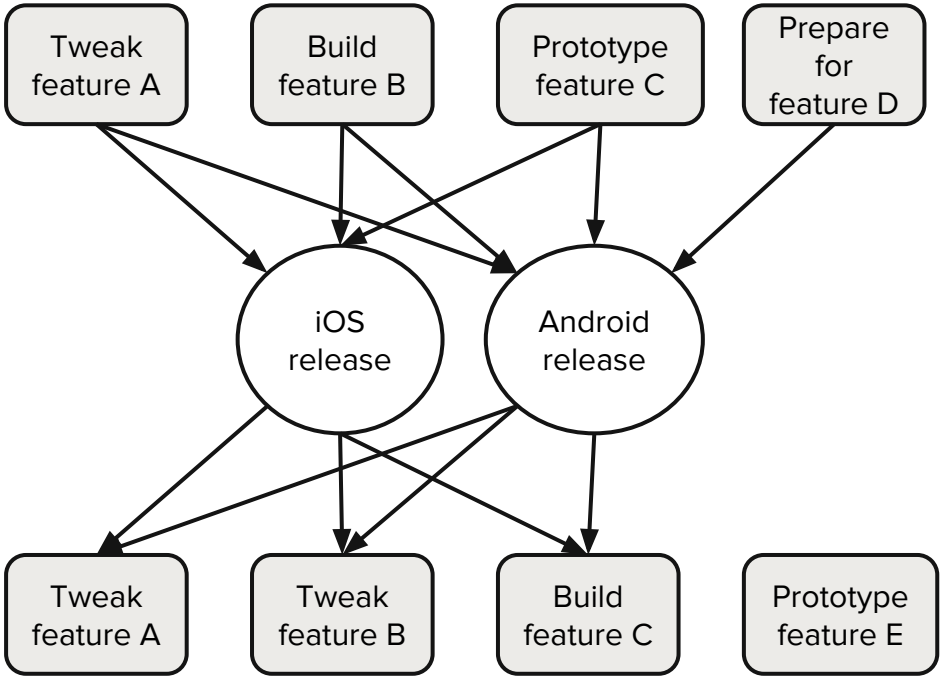
How do we organize them?

Parallel development

Teams have capacity for multiple features.

Synchronization.

Division of work



Dependencies

With multiple teams and division of work they start to depend on and block each other

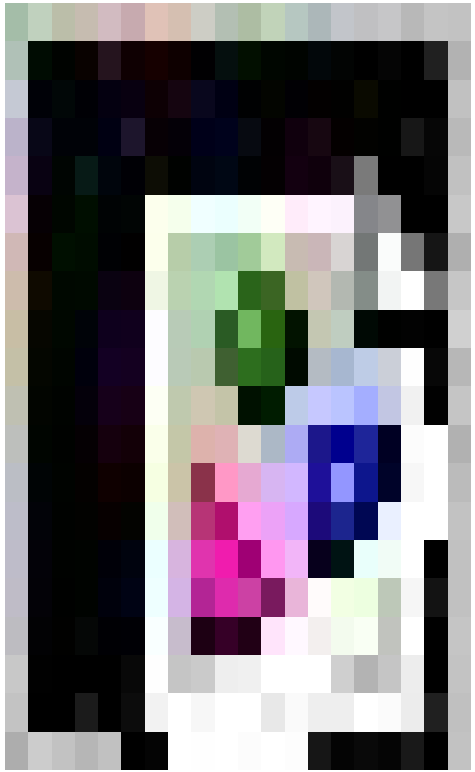
System design

“Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations” - Conway’s law

What about quality?

We have spent some time here

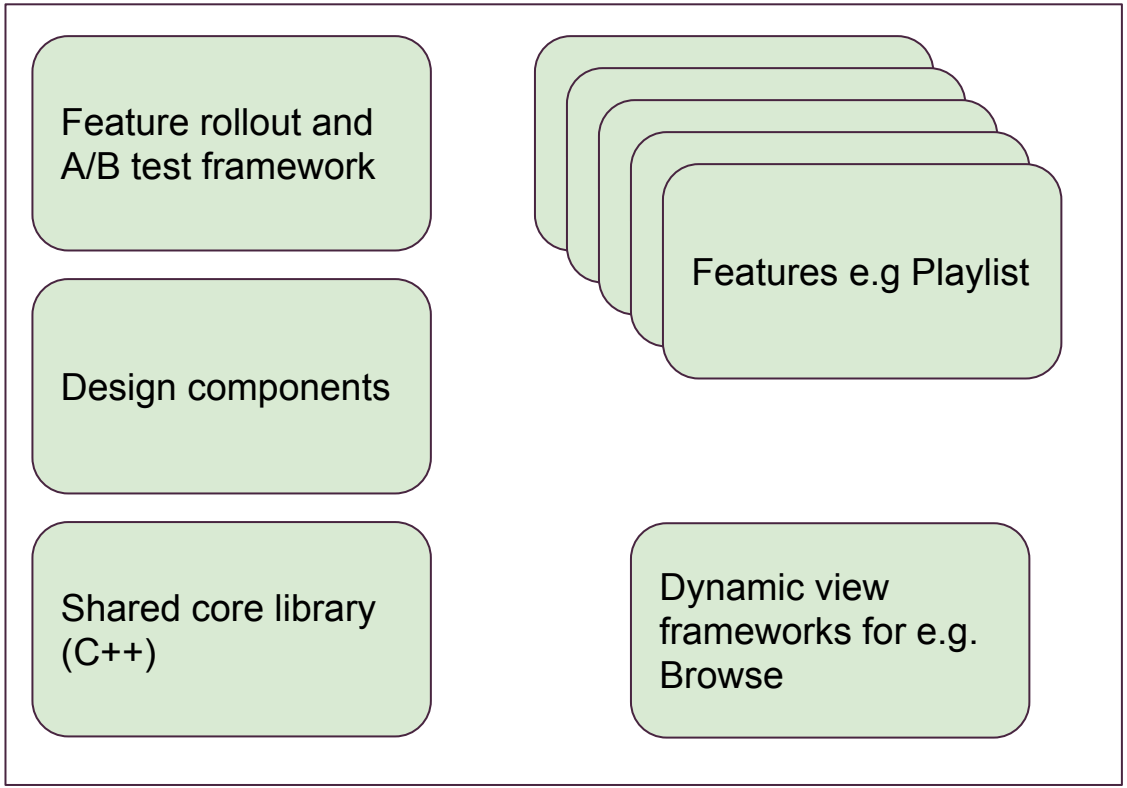
- Duplication of work
- Regression
- Teams blocking on each other
- Bloat
- Navigation items named after our teams

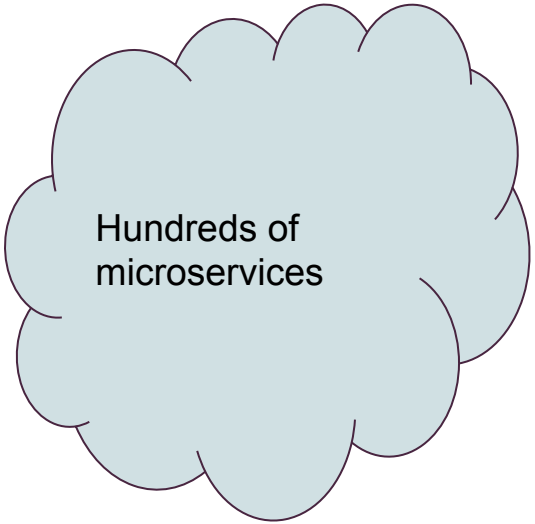


There are solutions

Building the Spotify app

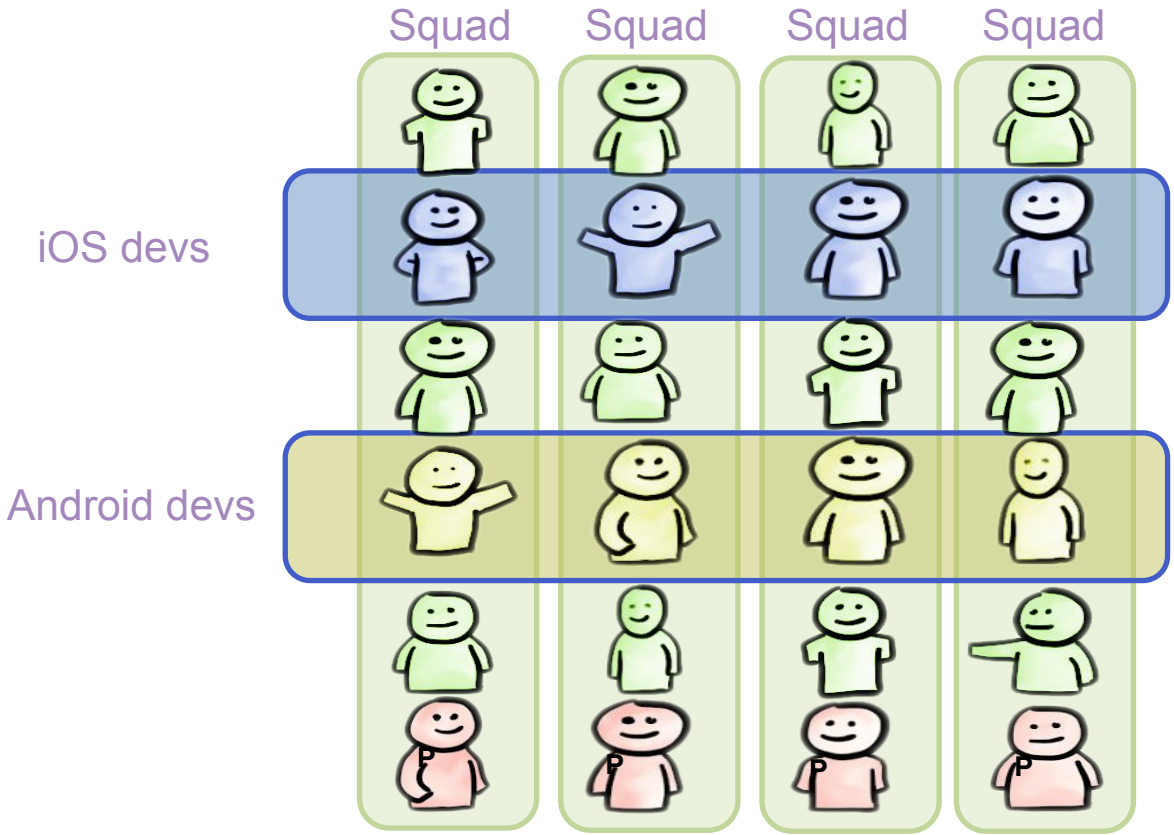
What it is


Feature rollout and A/B test framework
Features e.g Playlist
Design components
Shared core library (C++)
Dynamic view frameworks for e.g. Browse


Hundreds of microservices

Built by autonomous feature teams

Aligned through design, product and quality guidelines


Squad
Squad
Squad
Squad
iOS devs
Android devs

Successes

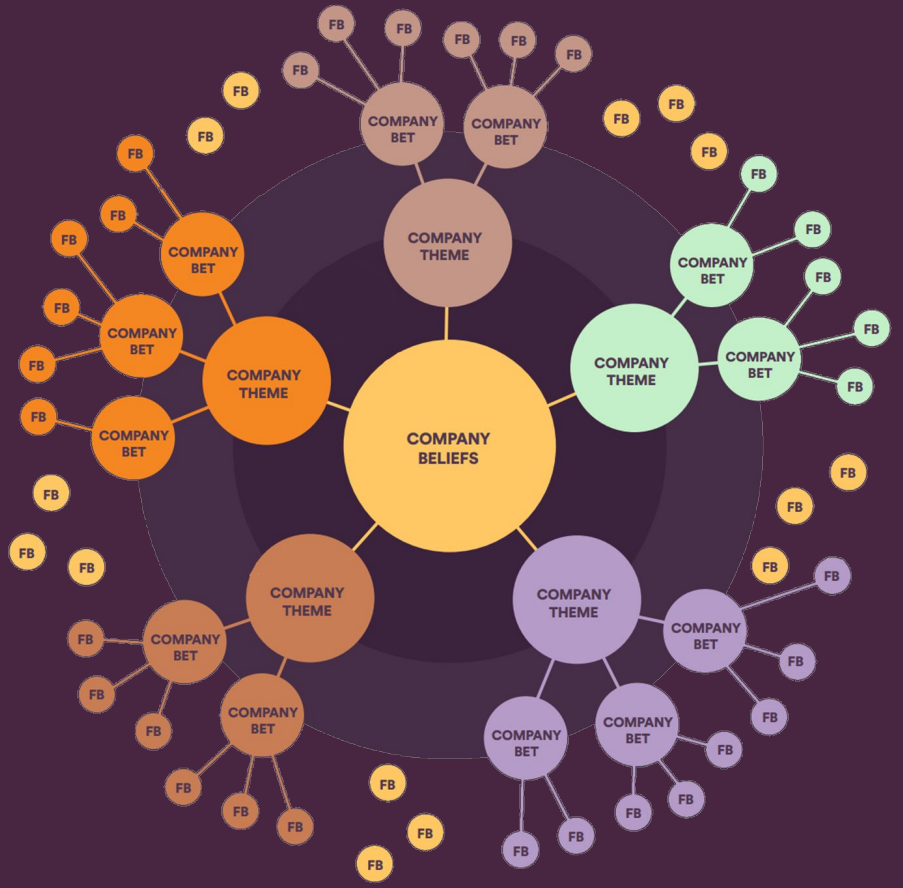
- Solves a lot of synchronization
- Fast - teams seldom block on each other
- Feature parity across platforms
- Autonomy -> happy developers

Failures we have learned from

- Hard to execute on big projects
- Suboptimization
- Inconsistent design
- One squad -> one view -> one navigation item
- Rewrite surprise
- Duplication

Alignment

On priorities



On design


Search
Sara Chin
Browse
Radio
Follow
Top Lists
Messages
Play Queue
Devices
App Finder
YOUR MUSIC
Songs
Albums
Artists
Local Files
Browse
OVERVIEW
TOP LISTS
GENRES & MOODS
NEW RELEASES
NEWS
DISCOVER
Good Morning, Monday!
Morning Commute
Your Favorite Coffee
Ready for the Day
Feelin Good
On Wheels To Work
Spotify
SPOTLIGHT
Foster The People
PAUSE
FOLLOWING
OVERVIEW
RELATED ARTISTS
BIOGRAPHY
POPULAR
Pumped up Kicks
Helena Beat
Houdini
Don't Stop (Color on the Walls)
Best Friend
SHOW 5 MORE
ALBUMS
RELATED ARTISTS
Coldplay
The Killers
Mark Foster
Elite Gymnastics
RELATED MUSIC
Best Friend
Sex
Sleepyhead
The Mother We ...
Contact High
The Struggle
Miss You
09:41
Foster The People
FOLLOWING
1 079 908 FOLLOWERS
SHUFFLE PLAY
POPULAR
Pumped up Kicks
Helena Beat
Best Friend
PLAYING FROM ARTIST
Calvin Harris
CALVIN
HARRIS
BLAME
FEAT. JOHN NEWMAN
Blame
Calvin Harris

On quality

Through strict release rules

For a year we spent about a third of our mobile capacity building continuous integration tools and infrastructure.

We (aim to) ship every two weeks with strict quality rules. If a feature is not release ready it is disabled.

Rules are not top down

We fail, we discuss, we decide on new rules to follow.

It is not strong managers who come up with and enforce rules. It is strong squads and guilds who agree on best practices.

We accept some duplication

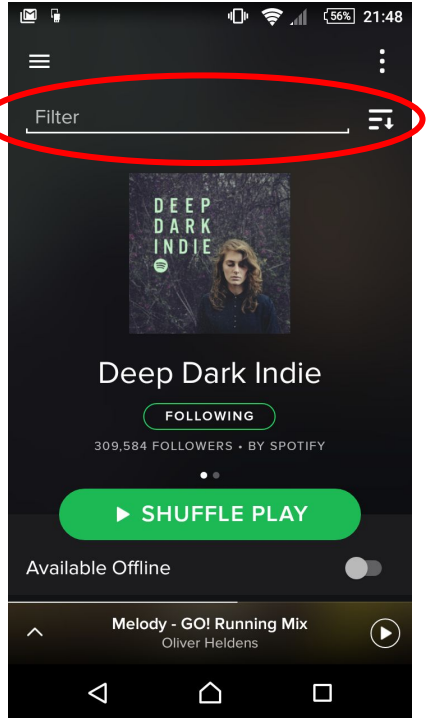
- We believe autonomy and simplicity is more important than trying to synchronize all efforts.
- We treat duplication similar to optimization. Fix it when we need to.
- It is often easier merge two or three working solutions into a great one than trying to build a generic one from the start.


In practice
Spotify

Feature example

Playlist filter and sorting

Assume we did not have this feature, what would implementation look like from start to finish.



Mission

“Help users find things in big playlists”

Squad planning meeting

- **Where does filter logic go?**
  - UI layer, C++ layer, backend
- **What is the user experience?**
  - Input from product & design
- **How do we test?**
  - AB versions
  - Lead platform
- **Who will implement?**
- **Who do we depend on?**

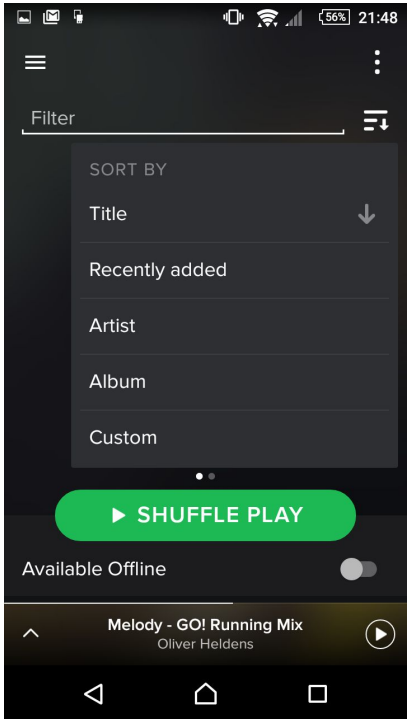
The squad should have the people and skills to own all these points.

Implementation

- Agile process - specifics decided by squad with help from agile coach.
- Sync through stand ups and daily collaboration.
- Designer and product owner heavily involved.

Development

- Start by creating flags for AB-testing and rollout.
- Build feature behind flags.
- Continuous integration.



Quality

- All code is reviewed
- Unit tests for all code, run pre-merge
- Automated tests run pre- and post merge.
- Manual QA in squad for all steps.
- Employee testing before rollout
- Gradual rollout (both clients and features)
- Feature and client metrics monitored constantly

Deployment

- Client release branches cut every 2 weeks.
- Release branches stabilizing 0-10 days.
- Incremental rollout starts as soon as there are no blockers on release branch.
- Nightly builds from master to employees.


TLDR
Spotify

goto; conference

Please

Remember to rate session

Thank you!

Follow us on Twitter @GOTOber

www.gotober.com