Zero to ten million daily users in four weeks: sustainable speed is king

Jodi Moran, CTO, Plumbee

Who am I?

- Mass market web entertainment for more than 8 years
- Small businesses, large businesses
- Small teams, large teams
- Variety of products
- I’m all about the big picture

About social games

- Free-to-play games on Facebook, monetized with microtransactions
- Highly interactive
- Cost per user matters
- Can grow very quickly

Case study: The Sims Social

- Released mid-August 2011
- By mid-September 2011
  - 10 million daily active users
  - 65 million monthly active users
  - 1 TB of analytics data collected daily

About Plumbee

- Social casino games
- Development started October 2011 with 3 engineers
- 5 engineers Dec 2011, 8 engineers today
- Launching first product in just a few weeks

What is sustainable speed?

- Speed measured by end-to-end time for each change
- Sustainability measured by maintaining speed over long time periods

Why sustainable speed?

- Responsiveness
  - To fickle audience
  - To changing competition
  - To changing platform
- Returns are greater
- Investments are less

Achieving sustainable speed

- Iterate and automate
- Use commodity technology
- Analyse and improve
- Build services
- Create a high-speed culture

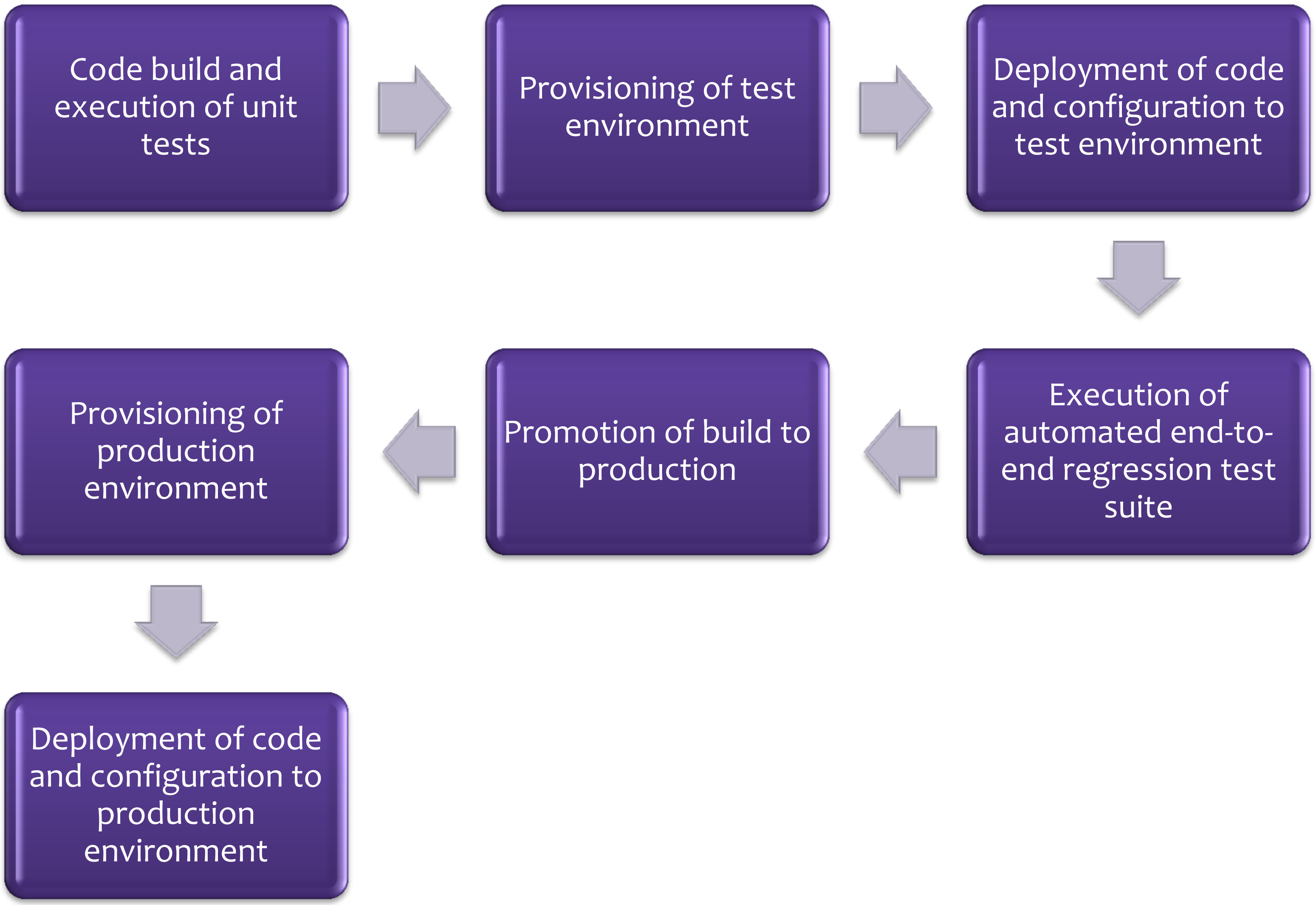
Achieving sustainable speed

- **Iterate and automate**
- Use commodity technology
- Analyse and improve
- Build services
- Create a high-speed culture

Be agile

- Framework for incremental delivery
- Incremental delivery (small batches) by definition improves end-to-end time
- Framework for reflecting on process
- Focus on principles, not practices: process is a means to an end

Automate routine work



Code build and execution of unit tests
Provisioning of test environment
Deployment of code and configuration to test environment
Execution of automated end-to-end regression test suite
Promotion of build to production
Provisioning of production environment
Deployment of code and configuration to production environment

Isolate changes

- Makes problem causes easy to identify
- Place high-risk parts of the system on different release tracks from low-risk parts
- Each release track can have a different cadence
- At Plumbee: Client, server, application configuration / content, environment configuration each separately versioned and independently releaseable

Make it minimally viable first

- Launch with minimal product
  … and minimal process
  … and minimal tech
- “If you aren’t embarrassed by your first launch, you didn’t launch early enough”

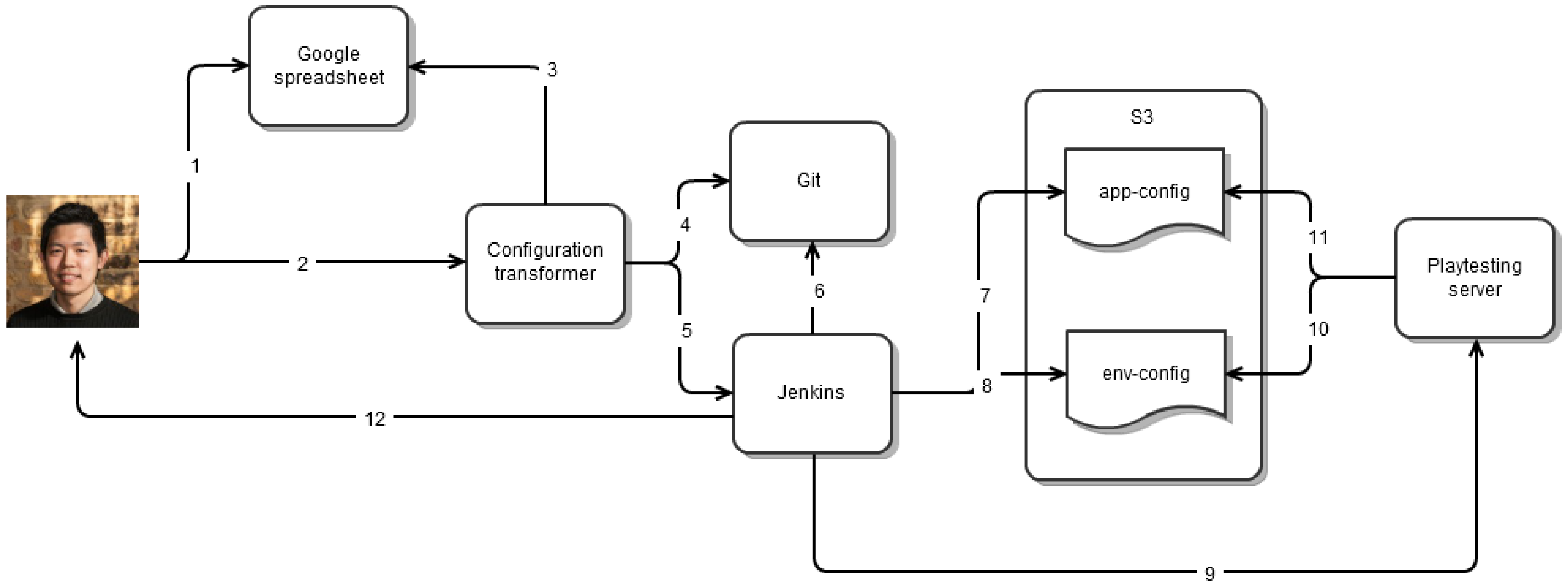
Prepare for technical debt

- Too much slows you down
- But it’s not possible to avoid
- So you will need a way to keep it under control
- Take it on intentionally when needed

Case study: content tools

- Special case of change isolation / automation
- Games have a lot of "configuration" or content that needs to be tweaked and balanced
- Content tools usually end at build stage
- At Plumbee:
  - Edit with familiar interface
  - Button click to deploy to playtesting
  - Button click to deploy to live

Content editing tools



Iterate and automate

- Small batches reduce end-to-end time
- Small batches help you find problems faster
- Automation makes things faster
- Automation reduces errors

Achieving sustainable speed

- Iterate and automate
- **Use commodity technology**
- Analyse and improve
- Build services
- Create a high-speed culture

Use commodity languages

- Large developer communities
- Many open source components
- Aids and encourages componentization and reuse
- For example: Java, Javascript, Actionscript

Use third-party services

- World-class features
- Low opportunity cost
- Maintain business focus

Internal	Company email, calendars, documents, accounting, HR, bug-tracking
Technical	Version control, build systems, monitoring, analytics, infrastructure
User-facing	Customer support, bulk and transactional email

Virtualized infrastructure with AWS

- Flexibility & agility
- Small operations team
- Infrastructure, not platform
- Advanced features
- Forces good software practice

Case study: Highly-scalable storage with commodity tech

Plumbee data access patterns

- Thick client means data is cached client side
- High ratio of writes to reads
- User primarily reads and writes their own data
- Secondarily, reads and rare writes of friends’ data

Plumbee data storage: micro view

- Data stored in key-value form, key is user id
- Multiple values stored against the user id
- Each value is a data structure serialized to binary format with Google Protocol Buffers
- {userid, valueid, value} tuples stored in single table in InnoDB/MySQL: {int, int, blob}
- Transactions managed with (modified) Spring / AspectJ

Plumbee data storage: macro view

- MySQL on multi-AZ RDS
- Read slaves handle e.g. reads of friend data
- Users are spread across many shards
- Shards are managed with custom library
- Users allocated using simple round-robin to shards, shard mapping persisted

The results

- By using commodity tech + services: MySQL/InnoDB, RDS, Java, Spring/AspectJ, GPB
- We have:
  - Fast access for use cases
  - Easy to understand and use
  - No downtime for schema changes
  - Easy monitoring and tuning
  - Horizontal scaling
  - Highly reliability
  - Automatic failover (with replica reassignment)
  - Easy snapshot backups
- All with just a few man-weeks of effort!

Commodity technology

- Easy and cheap to acquire
- Easy to hire people who know it
- Quick assembly of product from many parts
- Easy to change

Achieving sustainable speed

- Iterate and automate
- Use commodity technology
- **Analyse and improve**
- Build services
- Create a high-speed culture

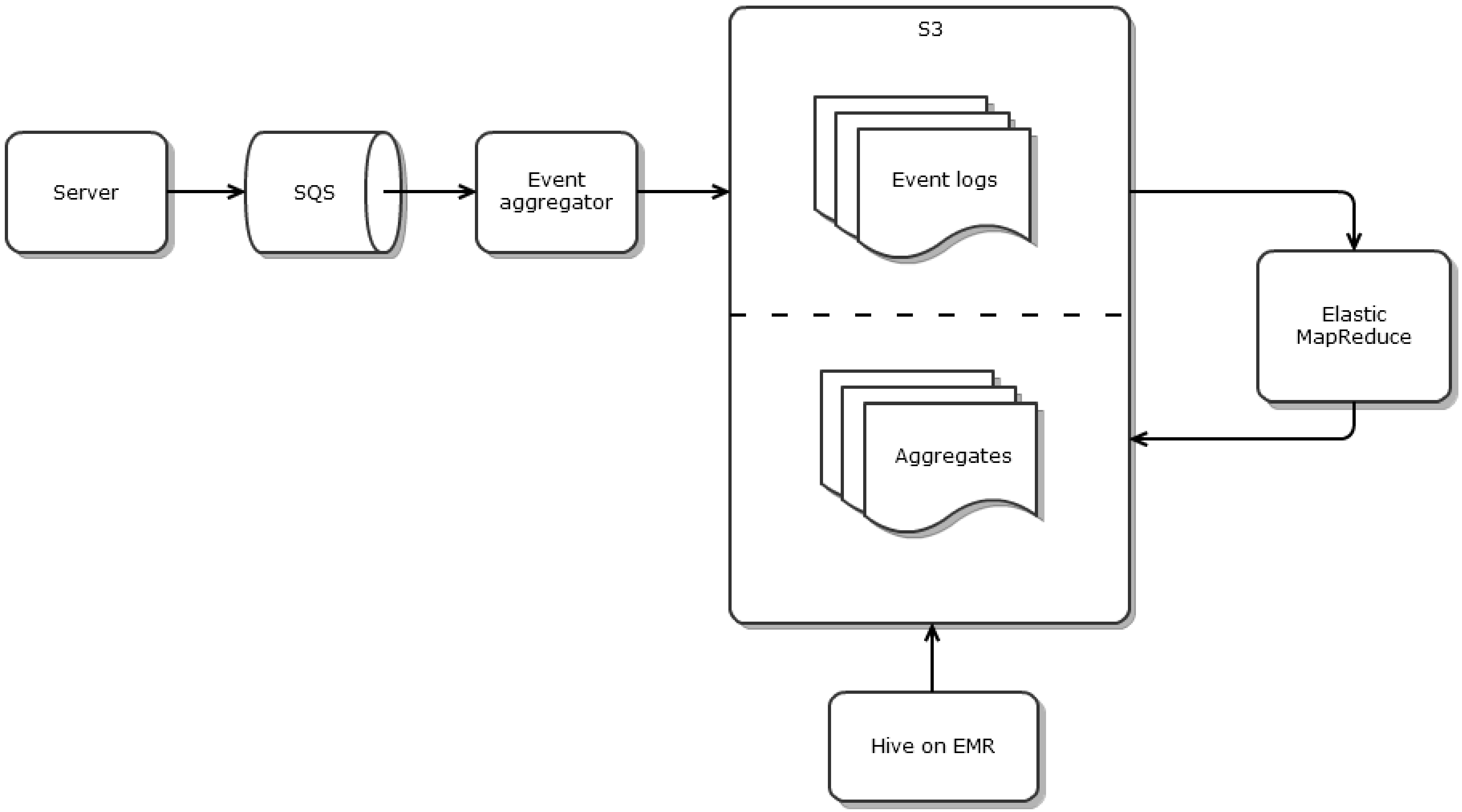
Collect user data

- Never too much data: collect everything and store it forever
- Collect data through events
- At Plumbee: we collect the entire content of every request and every database write

Collect system data

- Just another kind of analytics
- Instead of “what is user doing”, “what is system doing”
- Collect and use system data alongside user data
- Report on and monitor both user and system metrics

Analytics with commodity tech



What can you do with data?

- Reporting
- Monitoring
- Data mining
- Predictive analytics
- Personalization
- Split-testing

Split testing

- Run controlled experiments to determine how changes affect users
- To do this: assign users randomly to one of several product versions, called “variants”
- Tag all collected events with variant
- Calculate metrics are separately for each variant
- Perform statistical tests to determine whether the difference in metric is significant

Simple random sampling

```
variants = empty;
foreach (test in currently running tests) {
   selectedVariant = test.getStoredVariantForUser(user);
   if (selectedVariant == null) {
      if (shard in test) {
         selectedVariant = test.chooseRandomWeightedVariant();
         user.storeVariantForTest(test, selectedVariant);
      }
   }
   variants.addTestAndVariantPair(test, selectedVariant);
}
serverGroup = getServerGroupForVariants(variants);
serverGroup.forwardRequest(request, user, shard, variants);
```

Simple significance testing

- Conditions
  - Metric to be improved is a proportion: e.g. percent of users converting to spender.
  - Proportions are not too close to 0 or 1
  - Independent samples
  - Random sampling
- Result: super-simple test for confidence (z-test) that runs in linear time wrt size of test

Analyse and improve

- Analysing your system tells you how to improve it
- The more accessible and timely your data, the quicker your decision-making
- And the greater your responsiveness to changes
- Good analysis and split-testing means you do less work!

Achieving sustainable speed

- Iterate and automate
- Use commodity technology
- Analyse and improve
- **Build services**
- Create a high-speed culture

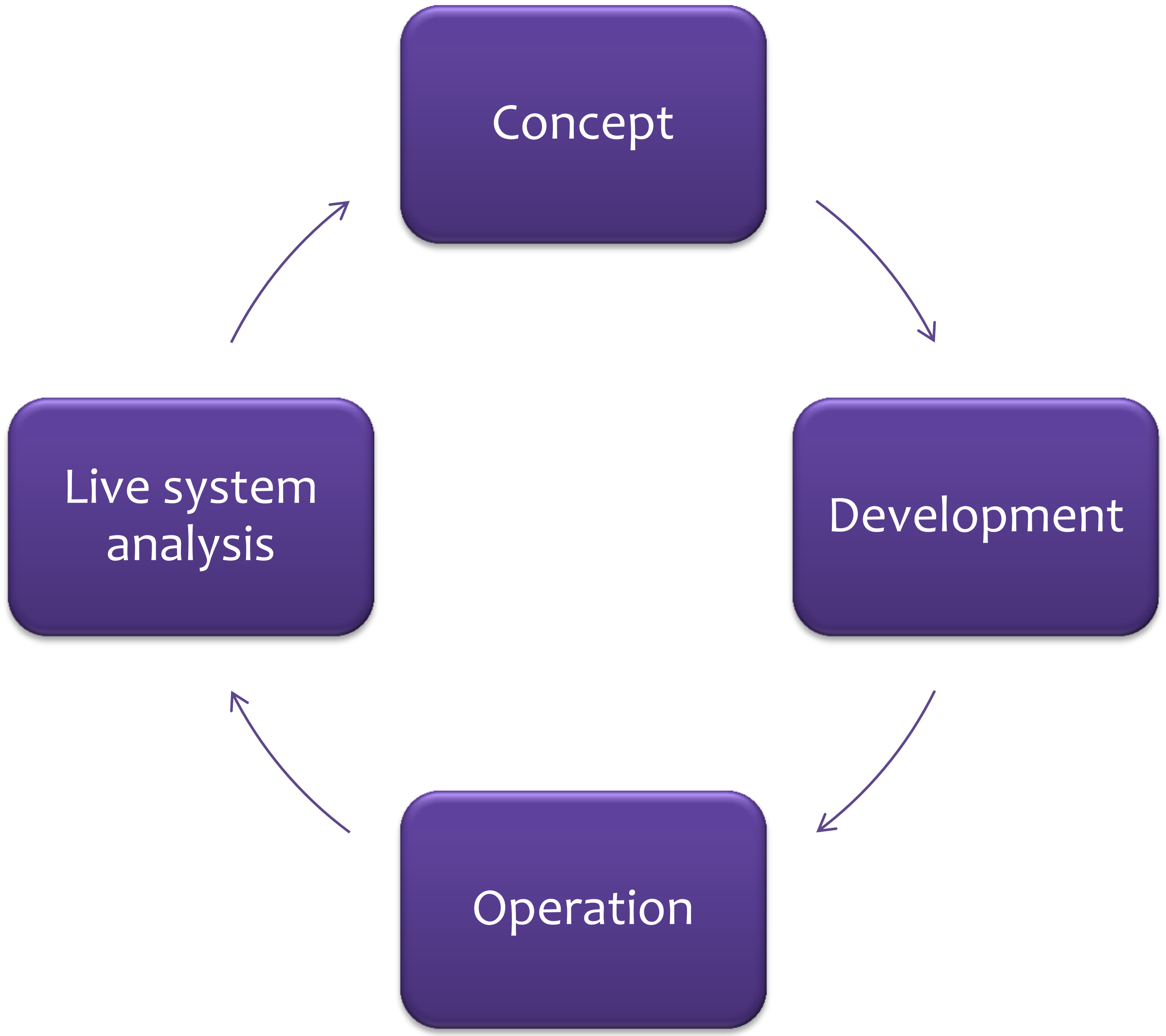
What are services?

- Essential quality: data & functions on that data combined into one component
- Data only accessible through remote API
- Each service is developed, deployed, and operated independently of other services
- “Service-oriented architecture” is an extrapolation of object-oriented programming to distributed systems

Technical benefits of SOA

- Scalability improvements: data is partitioned
- Performance improvements: data storage optimized for specific use cases
- System availability improvements: system can fail in parts
- But there's other benefits too…

Consider the round-trip



Apply distributed systems design

- Minimize communication – especially long-distance communication
- Make local progress
- Optimize the 90% case

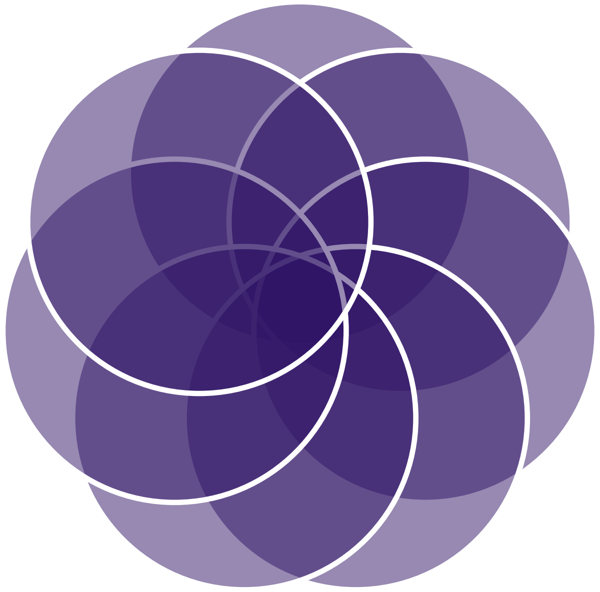
⬇

- Place people who need to communicate the most in the same team

Match architecture and organization

- Create little startups
  - small cross-functional teams
  - completely accountable for a specific service
  - act independently of other teams
- “Two-pizza teams”
  - Small teams can act fast
  - Scope is restricted
  - Everyone can understand team purpose
  - Greater sense of shared responsibility

A social game team



Business, product management

Art, design

Marketing, community

Engineering

Automated infrastructure

Data analysis

Production, process

Build services

- Small, independent, cross-functional teams can iterate quickly
- Forming little startups creates organizational scalability
- Services act as internal “commodity tech” that can be easily reused

Create a high-speed culture

- Iterate and automate
- Use commodity technology
- Analyse and improve
- Build services
- **Create a high-speed culture**

Post-modern programming

- Glue code is beautiful
- “Functionality is an asset, code is a liability”
- Embrace heterogeneity, don’t try to standardize
- There is no big picture

Test is dead

- Alberto Savoia's keynote at GTAC 2011
- What are tests for?
  - Is the functionality what the developer intended?
  - Is the functionality what the product owner intended?
  - Is the functionality what the user wants?
- Testing less is less risky

Corollary: load test is dead

- (Good) load tests are very difficult
- To test ability to add capacity?
  - Design for horizontal scalability
- To predict capacity needed for more users?
  - If you have IaaS, good design, good monitoring, and speed, you can scale just-in-time
- To predict capacity needed for new features?
  - Invest in ability to do dark launches and gradual rollouts

And also: operations is dead

- All engineers need mission-critical mentality
- When all engineers operate the system, the constraints and requirements of the operational environment will always be taken into account
- At Plumbee: all engineers have (audited) access to production, and all engineers take turns on call

High-speed culture

- Hire the best people
- Get them passionate about the product
- Provide easy access to information
- Give them freedom & responsibility
- Trust them to make decisions
- Encourage them to take risks

PLUMBEE

Sustainable speed

From this…

- Iterate & automate
- Use commodity technology
- Analyse & improve
- Build services
- Create a high-speed culture

Get this!

- Get ahead of your customers, competition, and platform
- Achieve greater returns
- Do less work!

PLUMBEE

Come and help us!

www.plumbeegames.com