Unblock!
Continuous Agile and the Continuous Enterprise

Agile New England

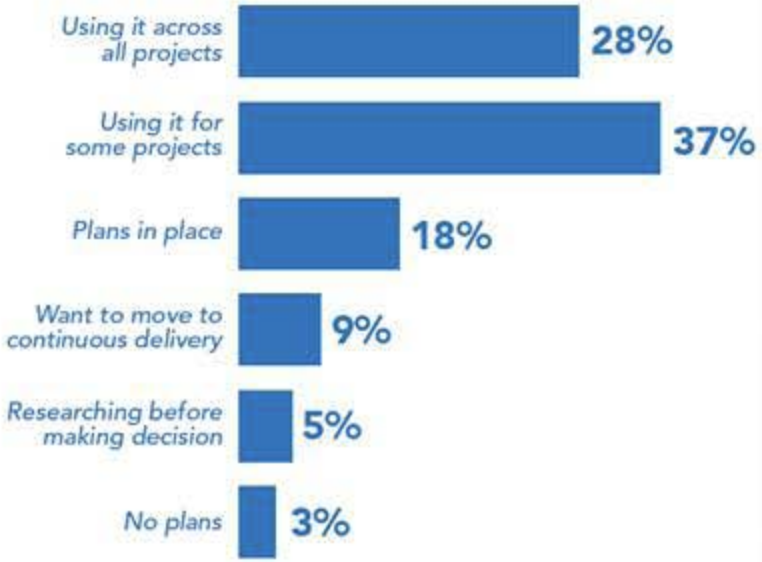
March 6, 2014

From Andy Singleton, http://andysingleton.com

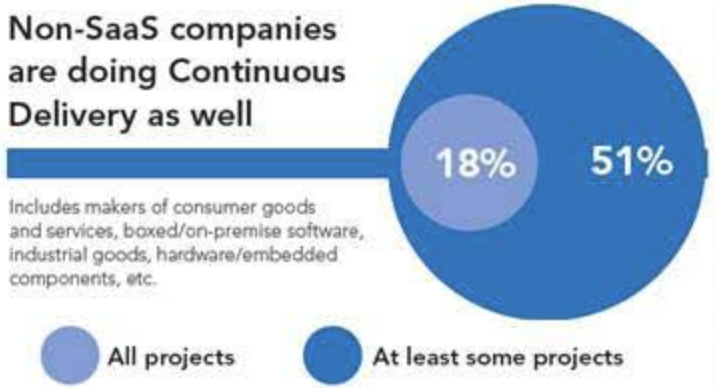
www.assembla.com

Survey on Continuous Delivery

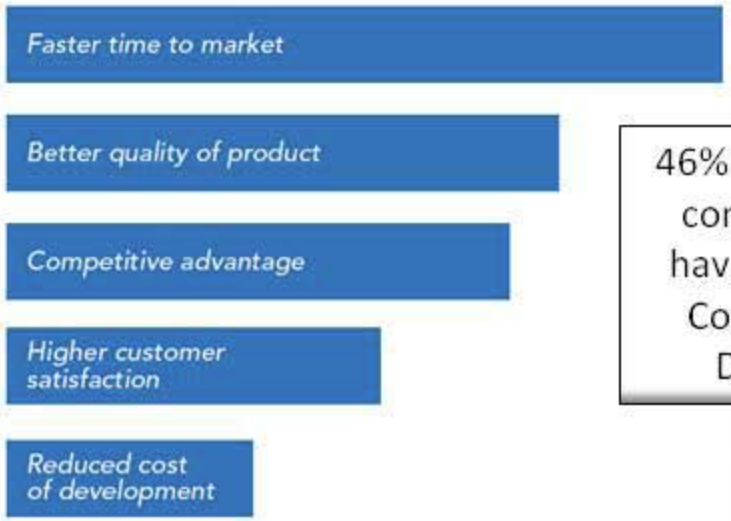
Adoption by All Company Types



Non-SaaS companies are doing Continuous Delivery as well



Ranking of Benefits



Based on ranking of top 3 benefits.

46% think their competitors have adopted Continuous Delivery

assembla

Why Continuous?

- You provide an online service. Competitive pressure will force you to continuous: Office 365 vs Google Docs.
- You provide any service with software inside
- Your release times are getting longer, or the release process is stressful
- You are developing a new product with lean startup and MVP techniques

- You have a big project with a lot of contributors

Ways to Scale

Scrum + SAFe

- Add more hierarchy
- Hold big meetings and teleconferences
- Block everyone into one cadence
- Coordinate big releases

Top Tech Companies

- Automate **management**, as well as testing and deployment.
- Communicate peer to peer
- Unblock! teams to move as fast a possible
- Release early and often. Separate release from launch

Assembla in 2011

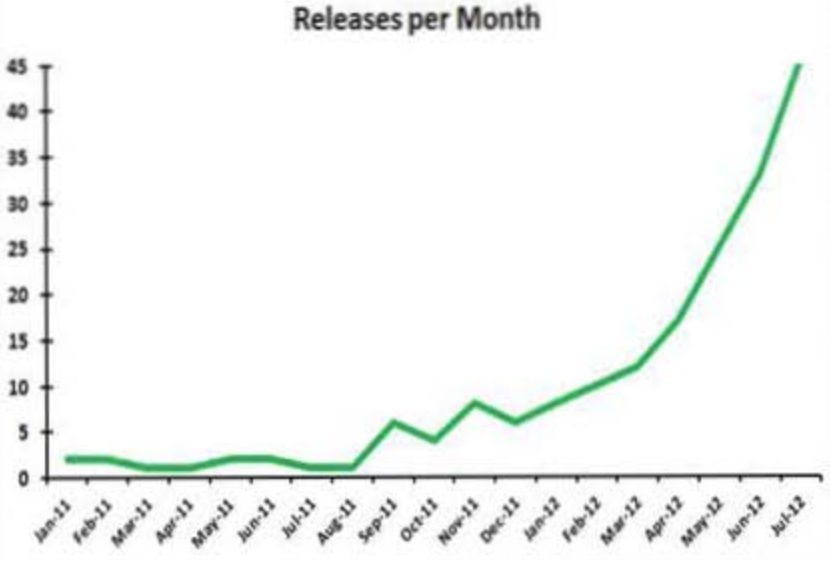
- "Scrumban" with iterative releases, but continuous planning to accommodate a distributed team.
- Releases took longer as system got bigger and there was more to test. 2 weeks -> 3 weeks
- Bugs in production. 2 days for fixes. **Stressful**
- Competitors achieved faster velocity with continuous delivery

Research

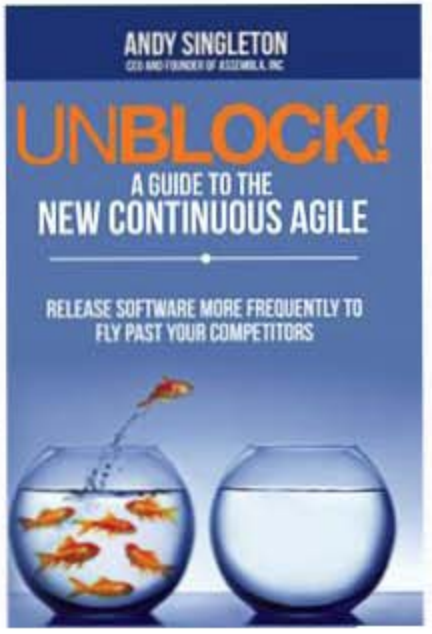
- Made a study of continuous methods with our own team, customers, and tools.
- Looked at other companies:
  - HubSpot
  - Edmunds.com
  - oDesk
  - Constant Contact
  - Google

Results

- Assembla now releasing about 250 times per month. Fewer bugs. Much less stress.



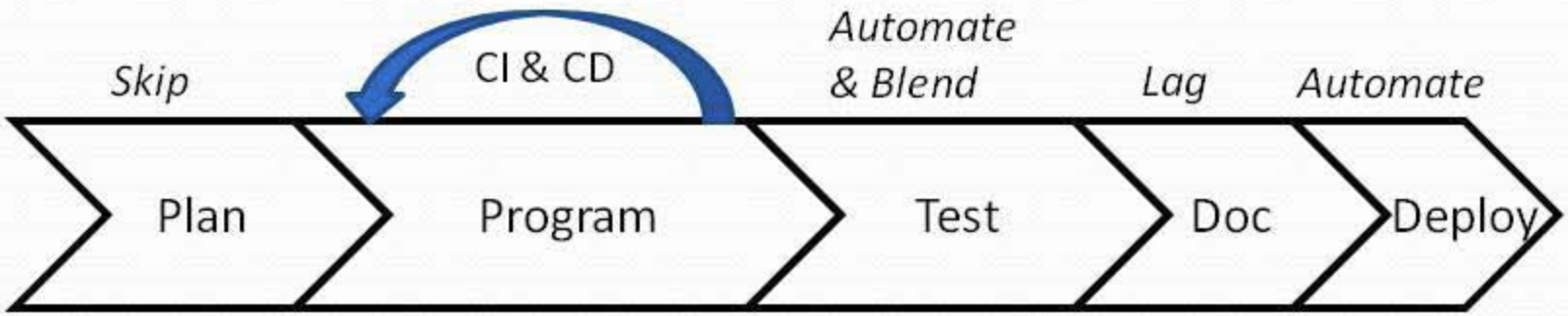
- Unblock! book (coming soon)



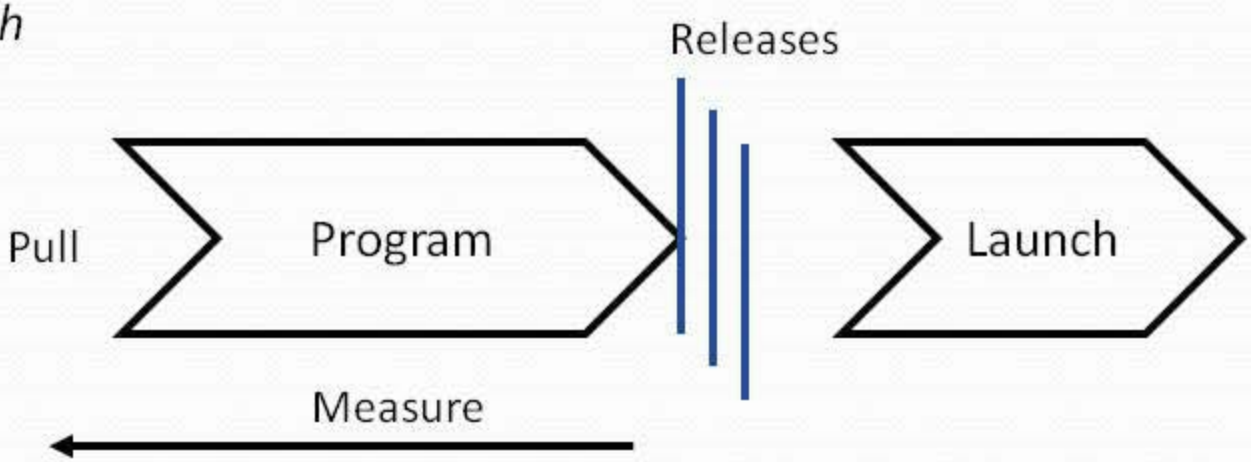
assembla

Continuous Agile

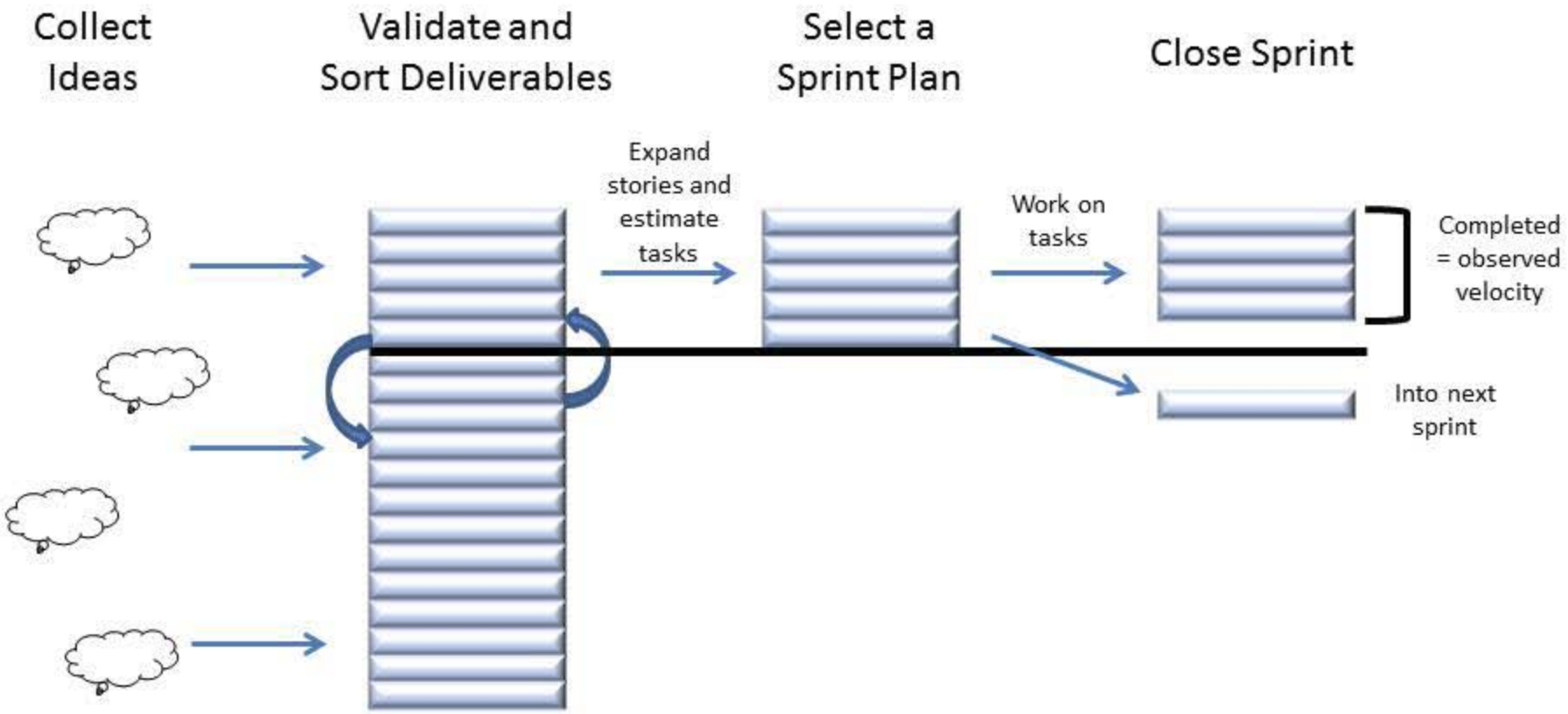
Waterfall/Sprint to Continuous



End up with



Scrum Sprint



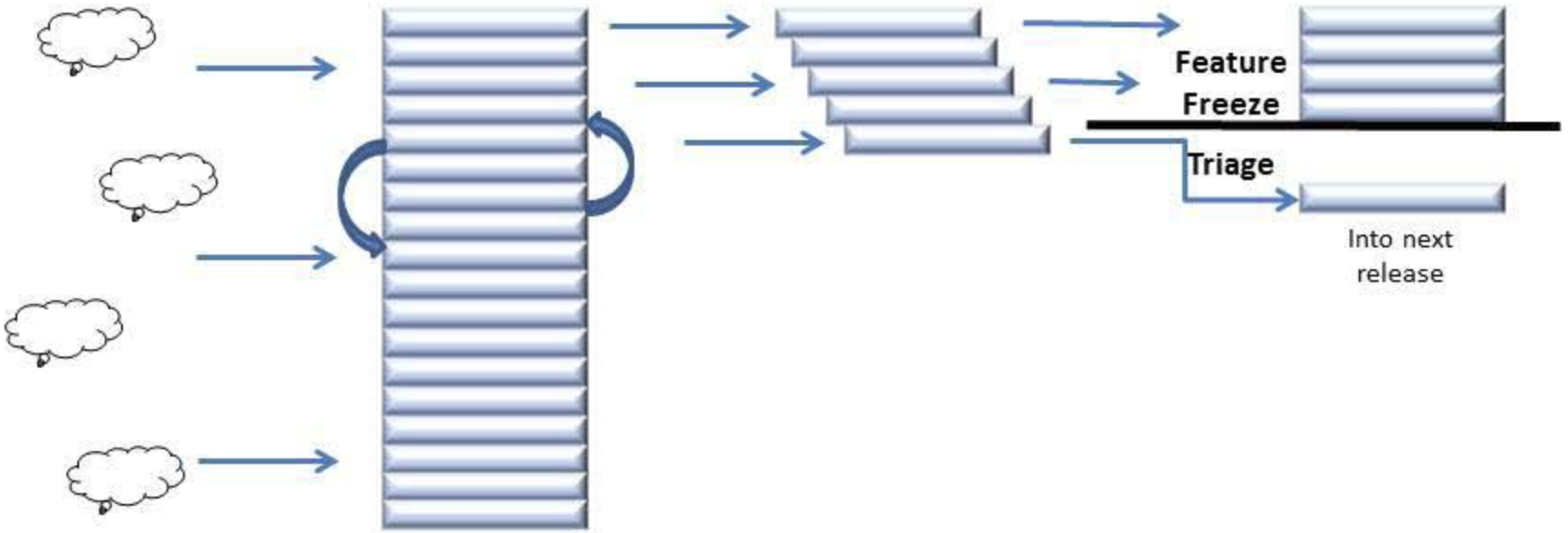
Scrumban Iteration

Collect Ideas

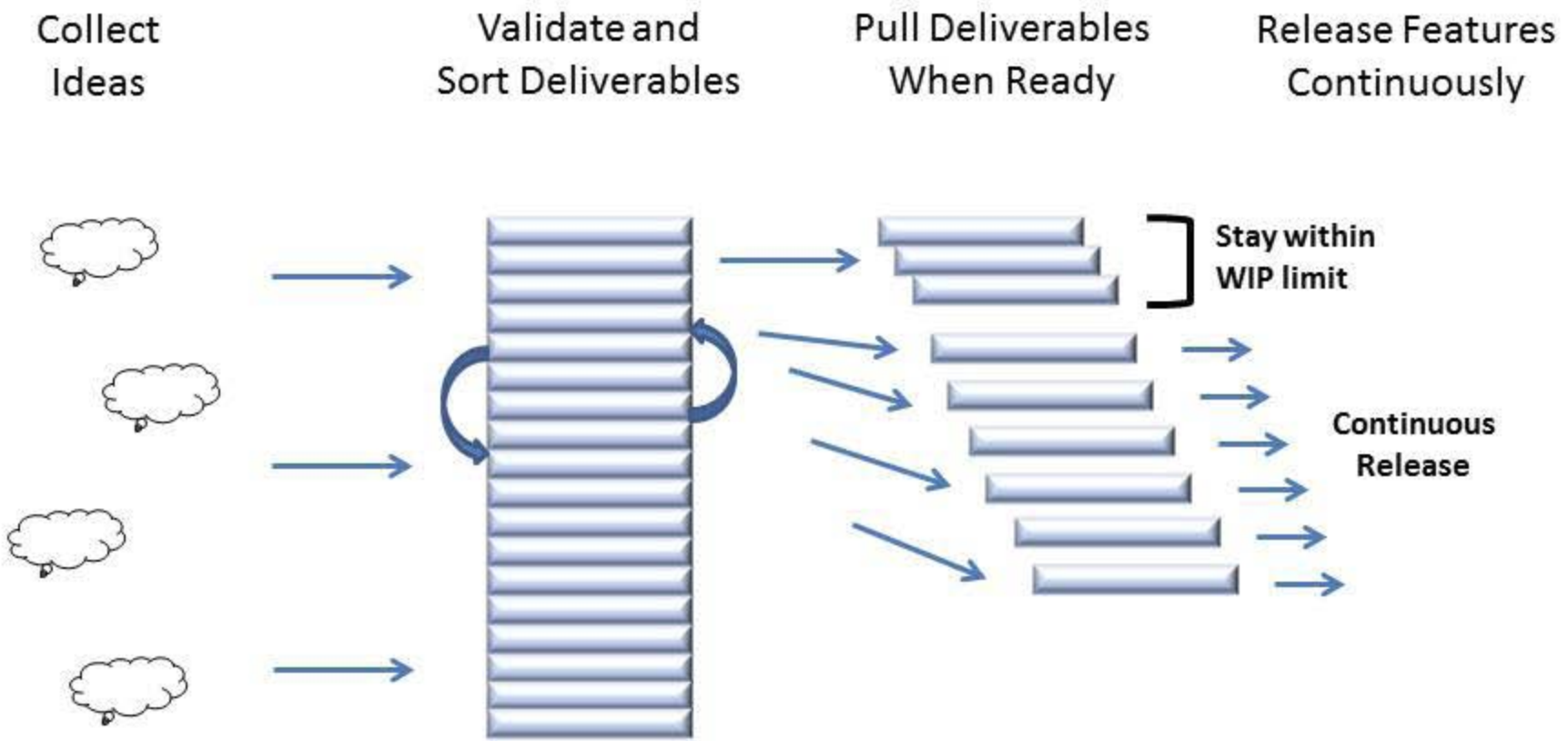
Validate and Sort Deliverables

Pull Deliverables When Ready

Stabilize Release



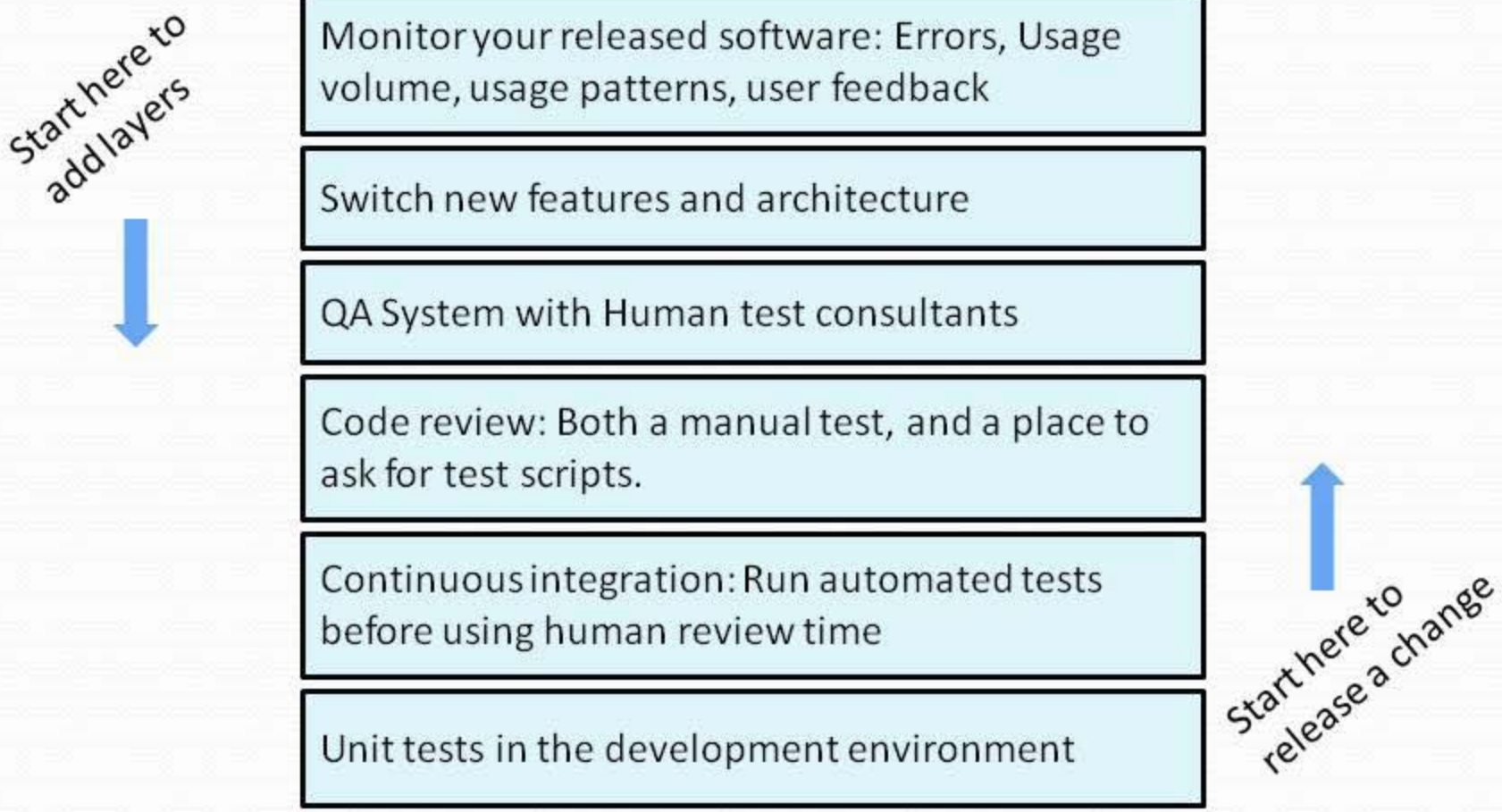
Kanban / Continuous



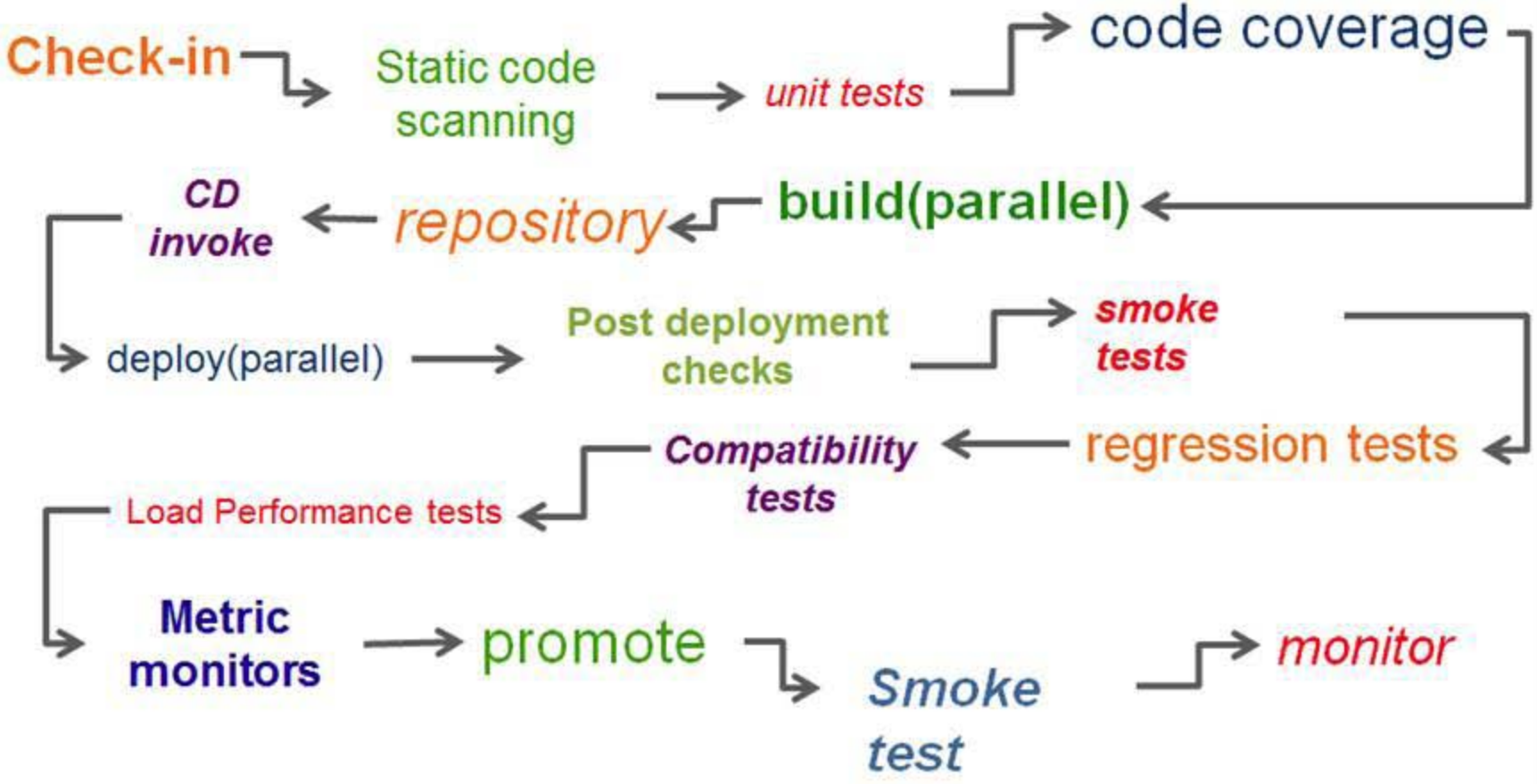
The big question: How to test?

- We release software in batches so that we can test it. That is the whole reason for doing it. We test software "release candidates" to make sure everything works together.
- In continuous delivery, we might get as little at 10 minutes to test a release candidate

Test Layering



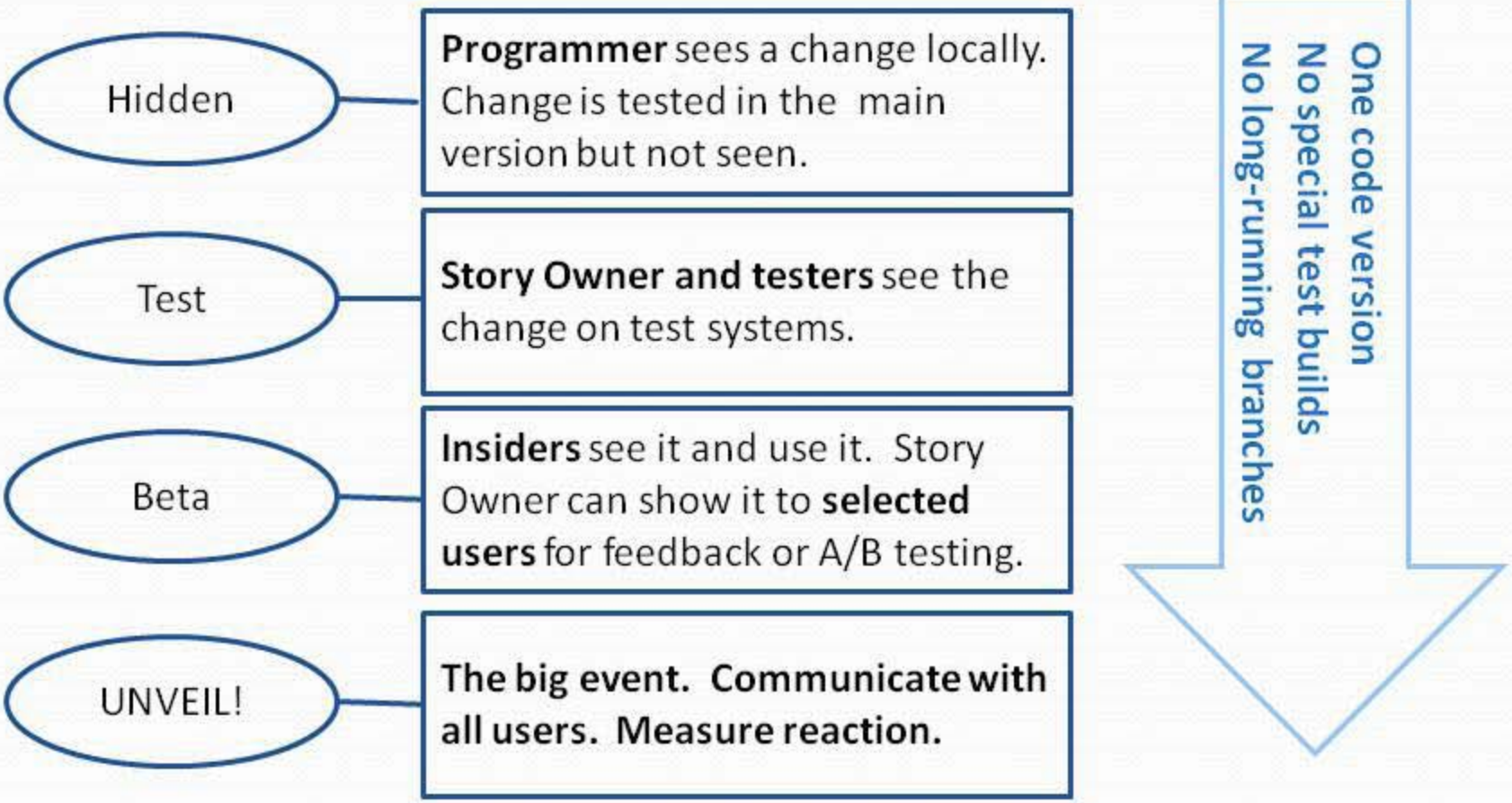
9 (sparse) Layers at Edmunds



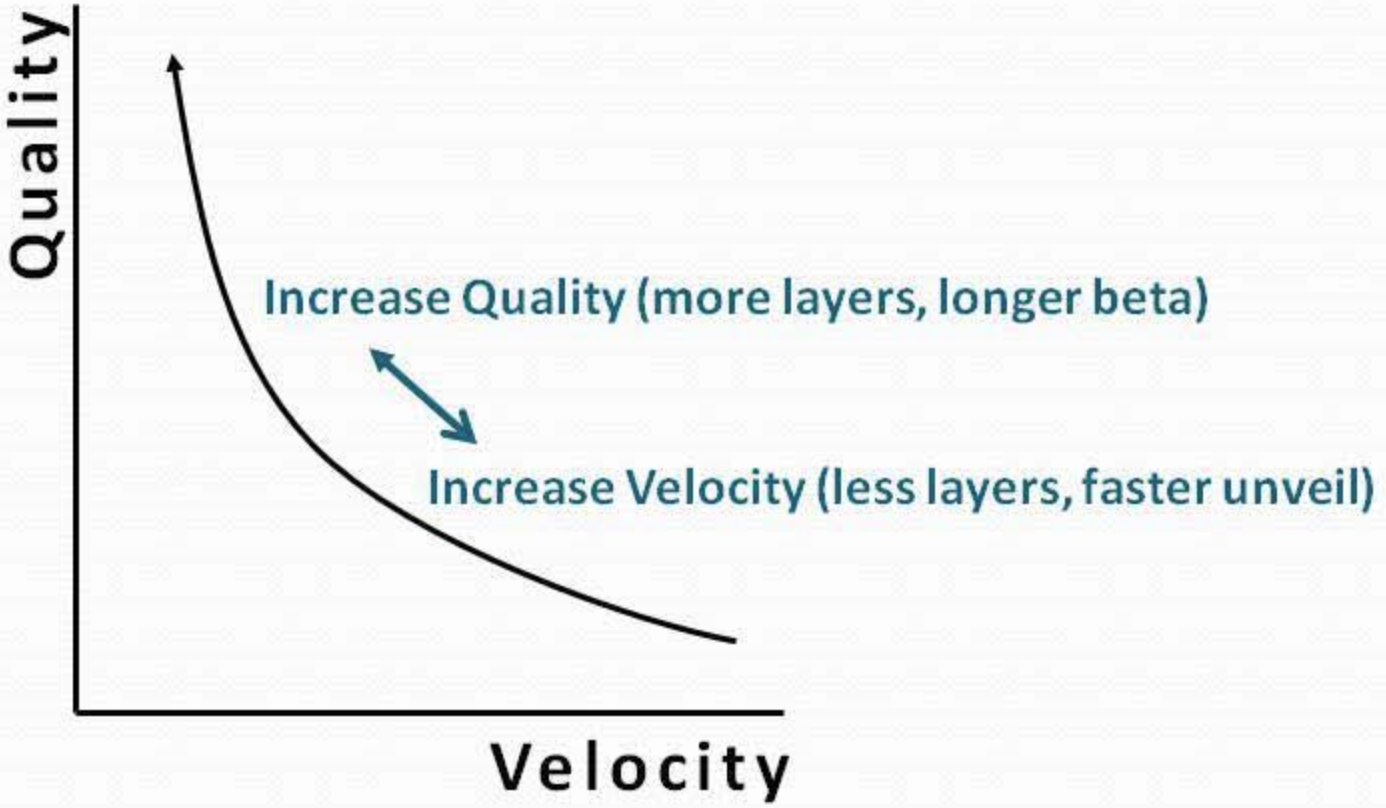
Feature switches

- Constant Contact showed a new UI for Agile New England (at nonprofit rates). Showed the old UI for Assembla (at for-profit rates). They had installed a feature switch that showed the new UI to specific users.
- I have seen similar behavior in ATM's and cars. Switched areas on chips.
- Separate release from launch. Developers run ahead of marketing, and we learn before launch. Unblock! PM's control what gets unveiled.

Feature Switch and Unveil

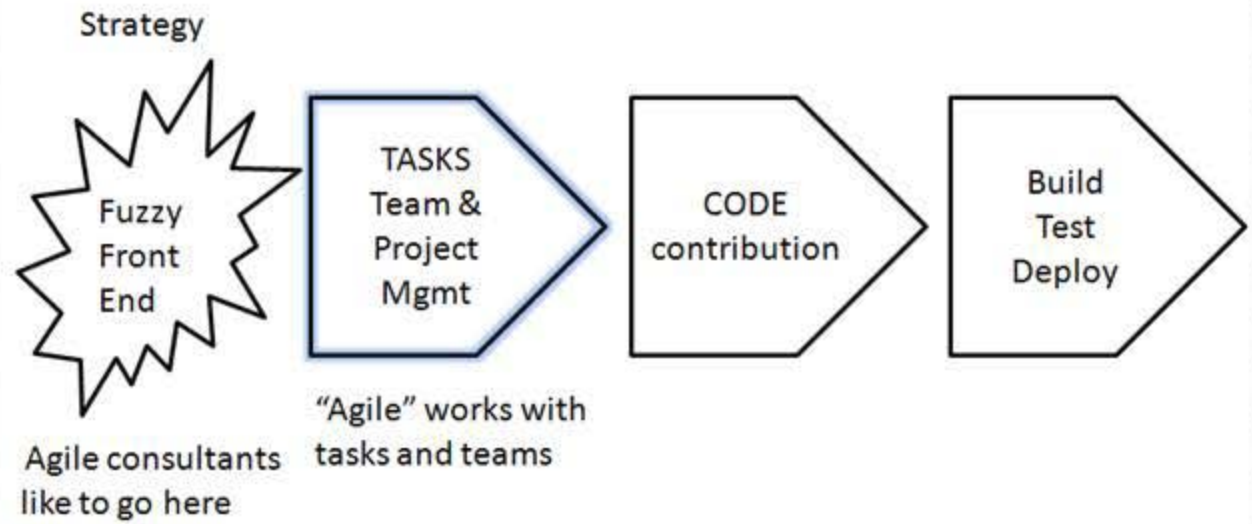


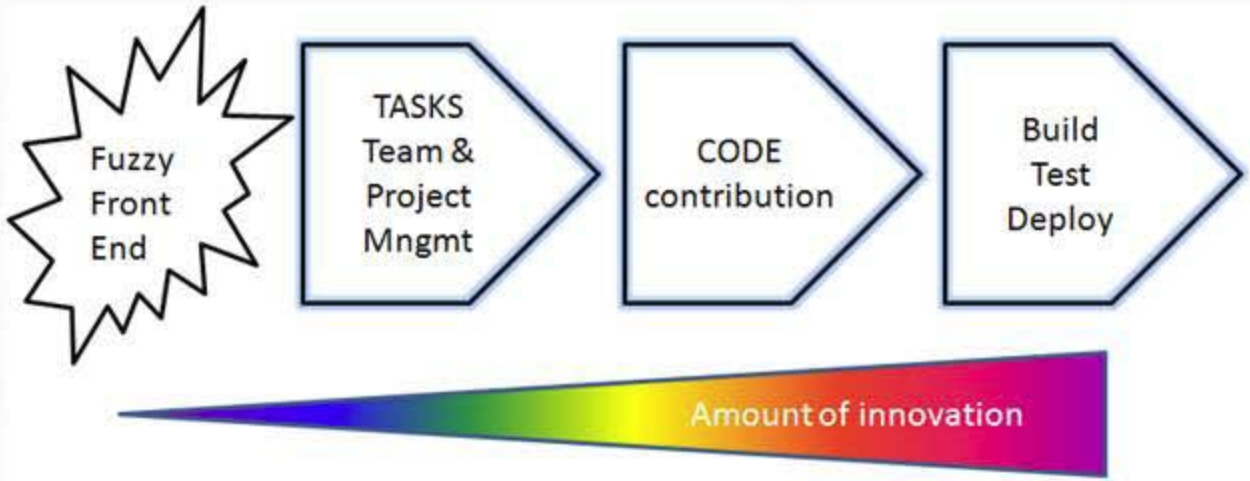
Go Both Ways



assembla

Coding and Continuous Delivery

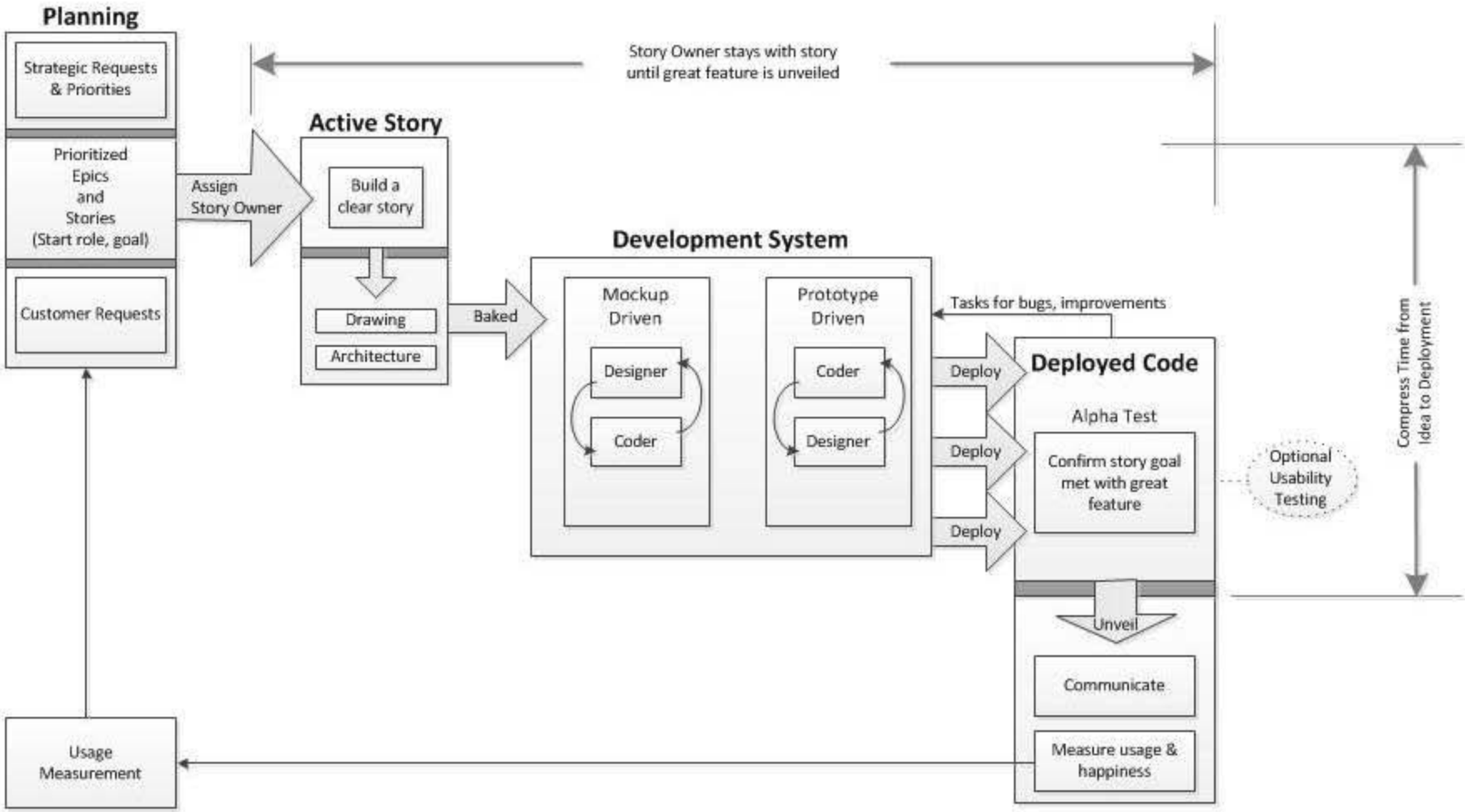




The new agile brings in a surge of innovations around coding and testing workflows

assembla

Product Owner -> Story Owner



assembla

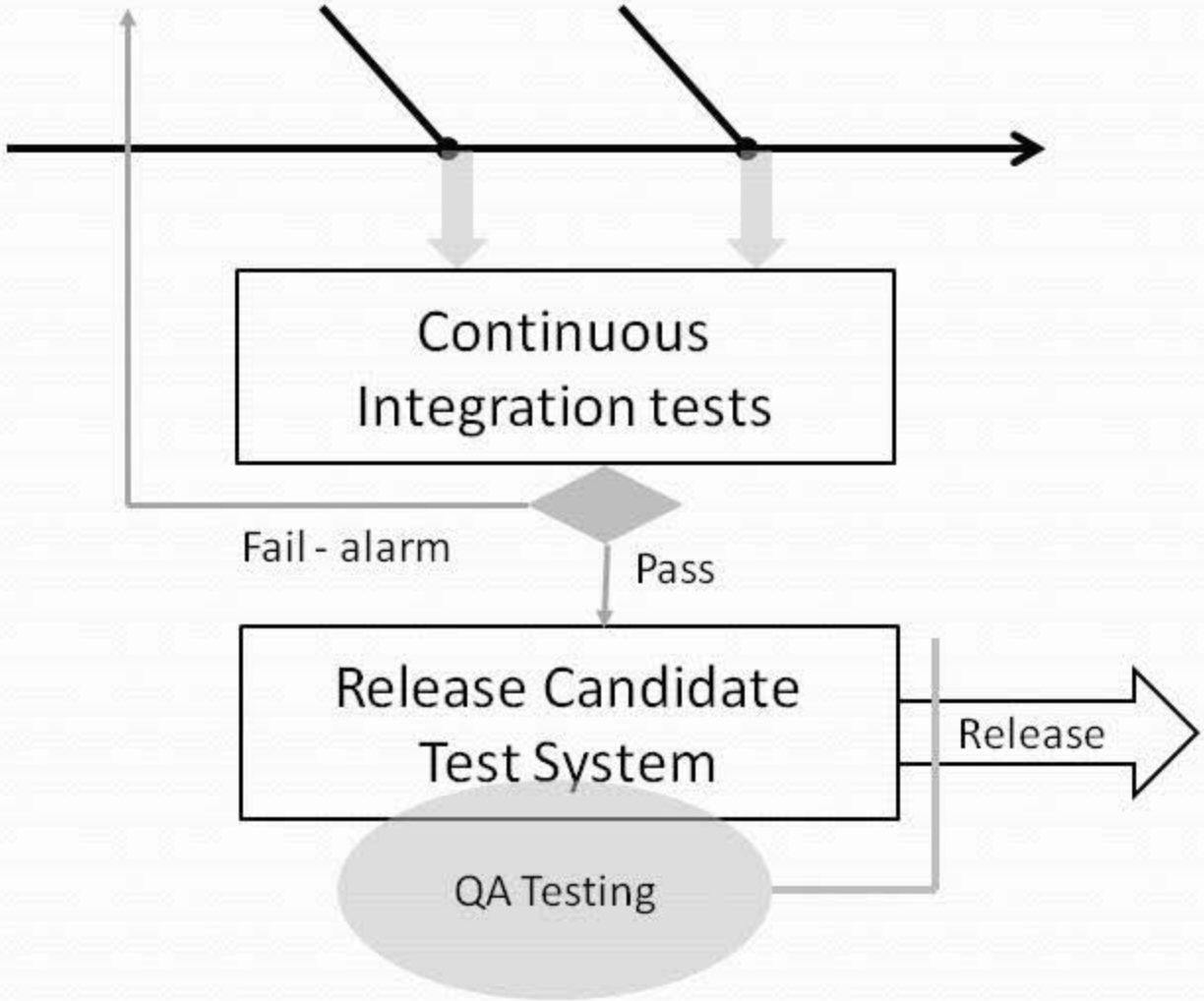
Code Contribution Patterns

Manage code if possible. People are hard to **manage** and can't be automated. They want to **contribute**.

- Centralized continuous delivery
  - No branches, finds and fixes problems as early as possible
- Distributed continuous delivery
  - Release every change with its own branch and test
- Temporary branches
  - Combines benefits of centralized and distributed
- MAXOS
  - Use centralized continuous integration to manage a massively scalable IT system

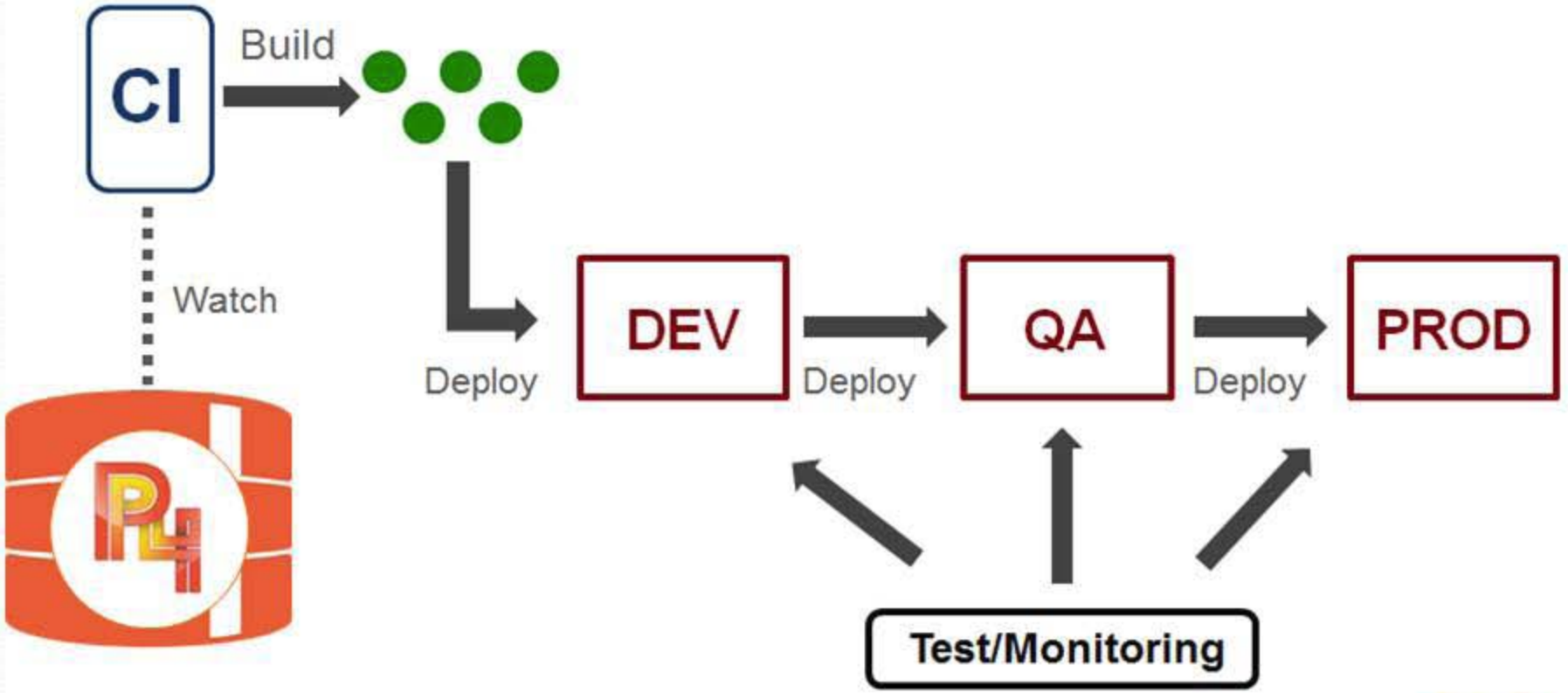
Centralized CI/CD

Contributor Commits – "as early as possible" to find problems



assembla

Continuous delivery at Edmunds.com

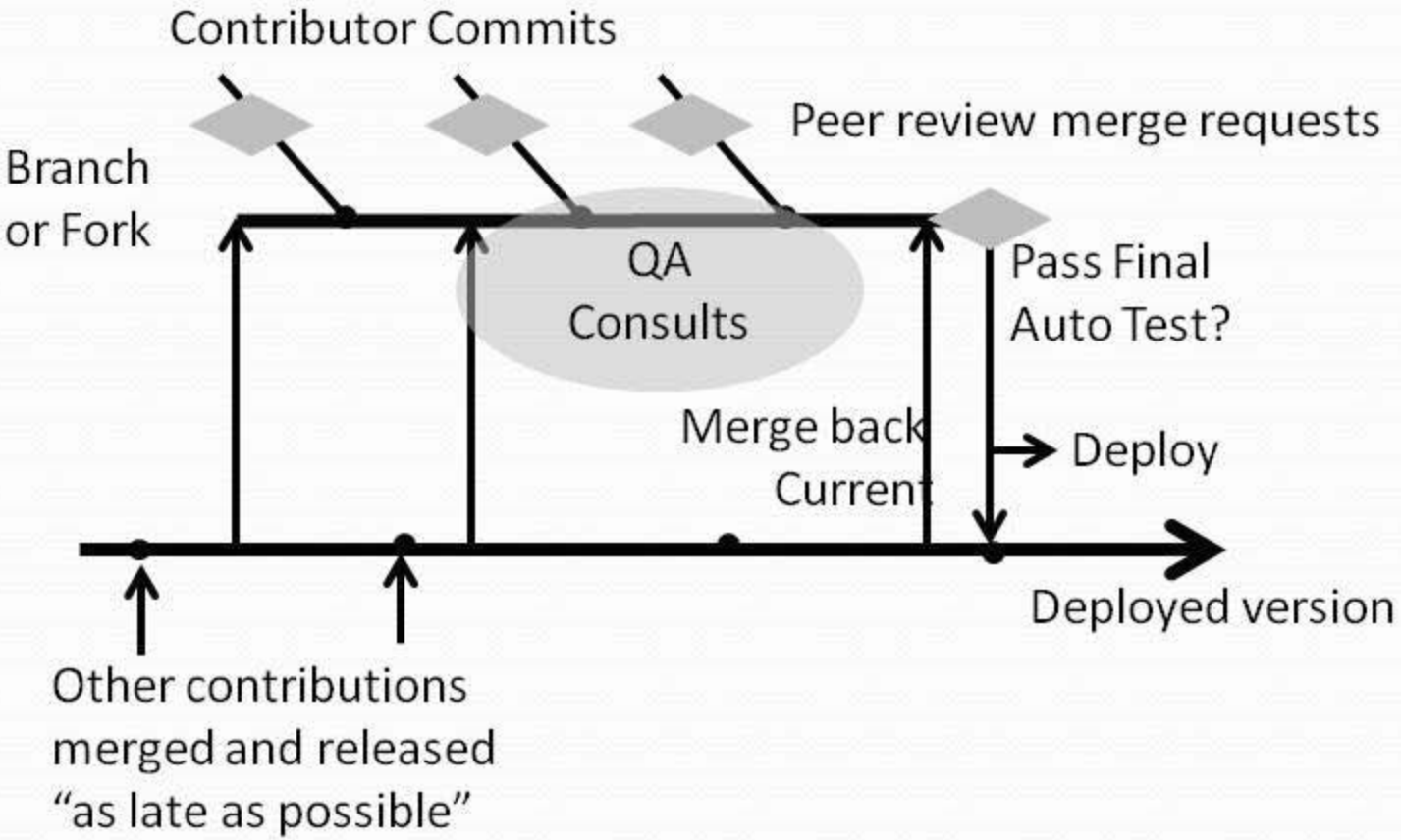


Outside view: "it's automation"
Inside view: "it's confidence"

edmunds.com

assembla

Distributed Continuous Delivery



assembla

Changing Roles

Role: Developer

- Developers have more power and responsibility.
- Developers have more responsibility for testing.
- Developers (not QA or PM) decide when to release. **This is a strong finding.**
- Incentives are correct. Developer might have to come back from Friday night beers to fix a problem. This provides a motivation to make good decisions and automate testing.
- Features can be released but hidden. Product Managers and Marketers will unveil when they are ready. Unblock!

Role: QA

- QA is a consultant when asked, not a required gate
- QA gets more respect. Developers have to ASK for service.
- Developers do more of the testing work. They should organize reviews and automated tests so that bugs don't go through into the manual test process.
- QA has more time to investigate usability
- QA monitors productivity and quality metrics

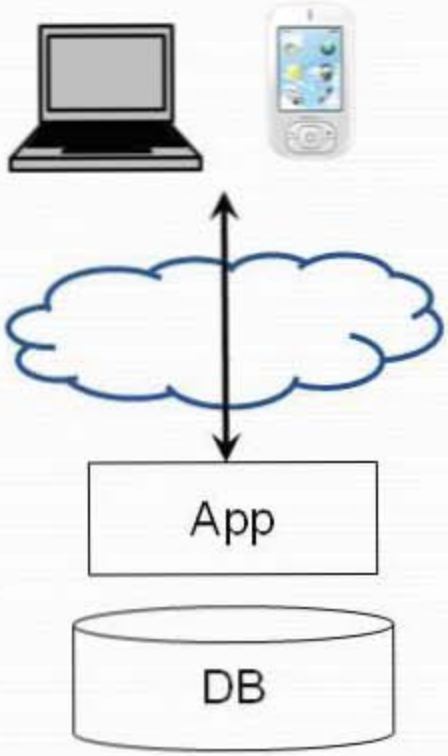
Role: Product Manager/Owner

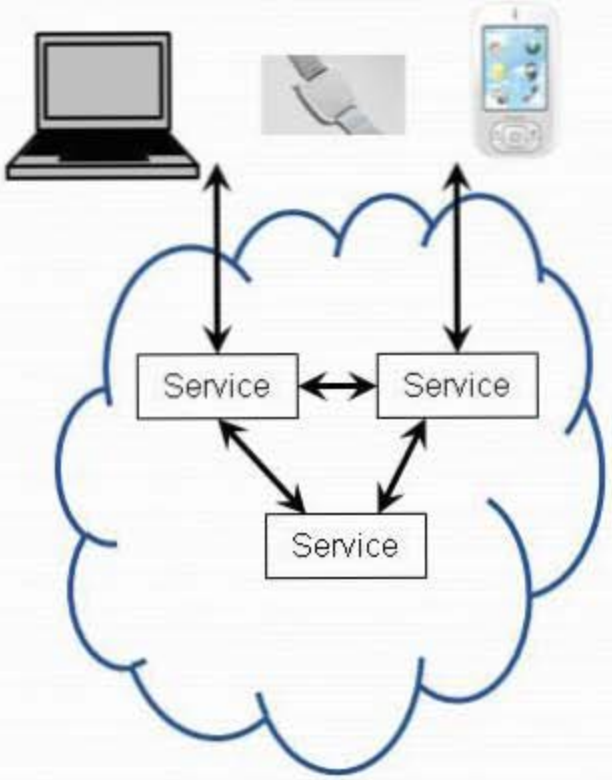
- Batch -> Continuous
- Requirements -> User Experience
- Strategy -> Measurement
  - Usage measurements are so important, so underutilized
  - Double your productivity

Matrix of Services
Breaking the scale barrier

The Services Megatrend

Desktop → Web App → Cloud Services



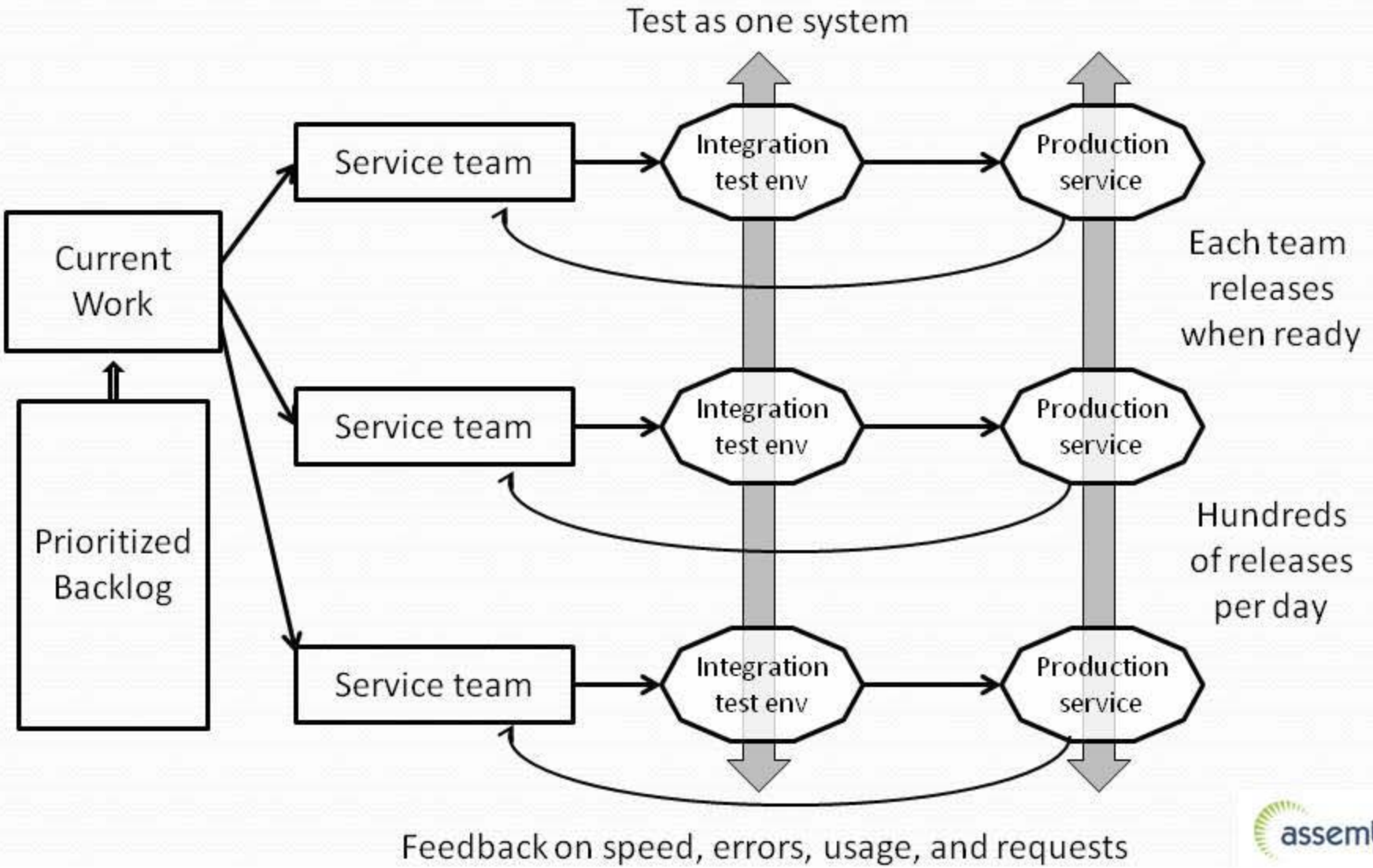


assembla

Scale it like Google

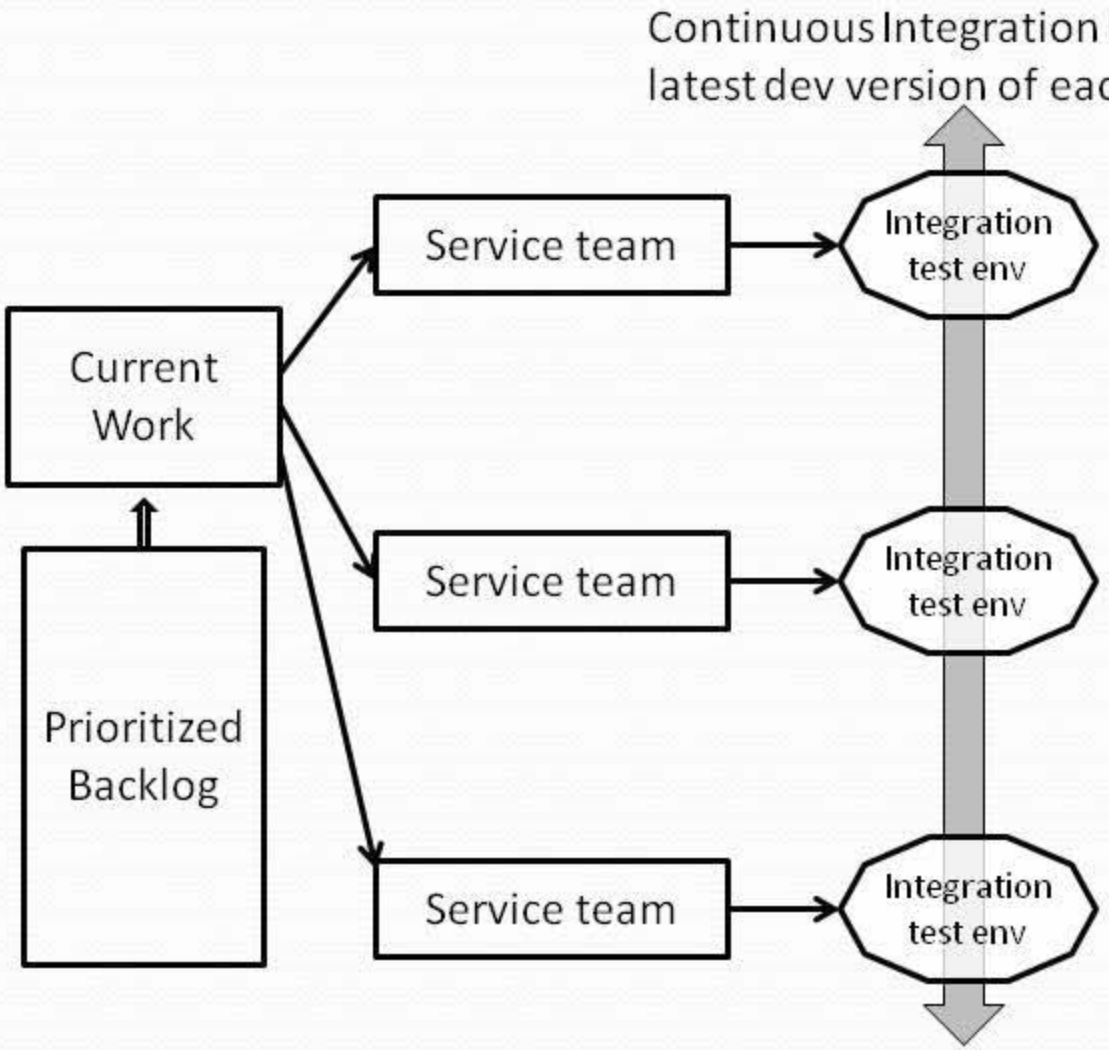
- 15,000 developers, 5,000 projects, one current version of the code (2013). They can go from an idea to a release in 48 hours
- Vast Internet system divided into thousands of "services"
- Most programming done by teams of 3-4
- Centralized process with single version of the test system – run 100 million test cases daily
- Before developers release a change, they test it with the most recent version of all the other services. If a test script finds conflicts, it tells developers who to contact to resolve them

Matrix of Services - MAXOS



assembla

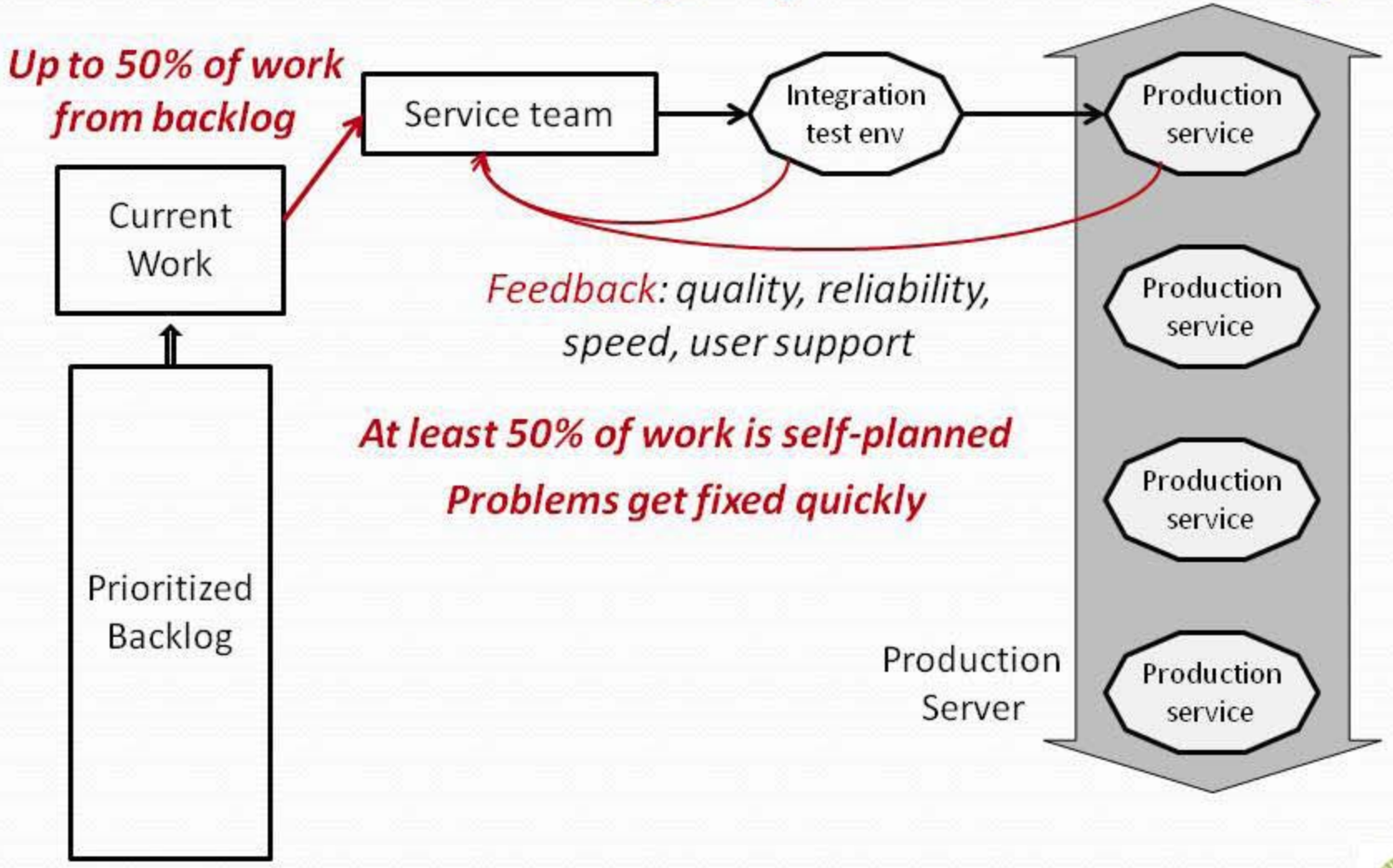
Coordinate without big meetings



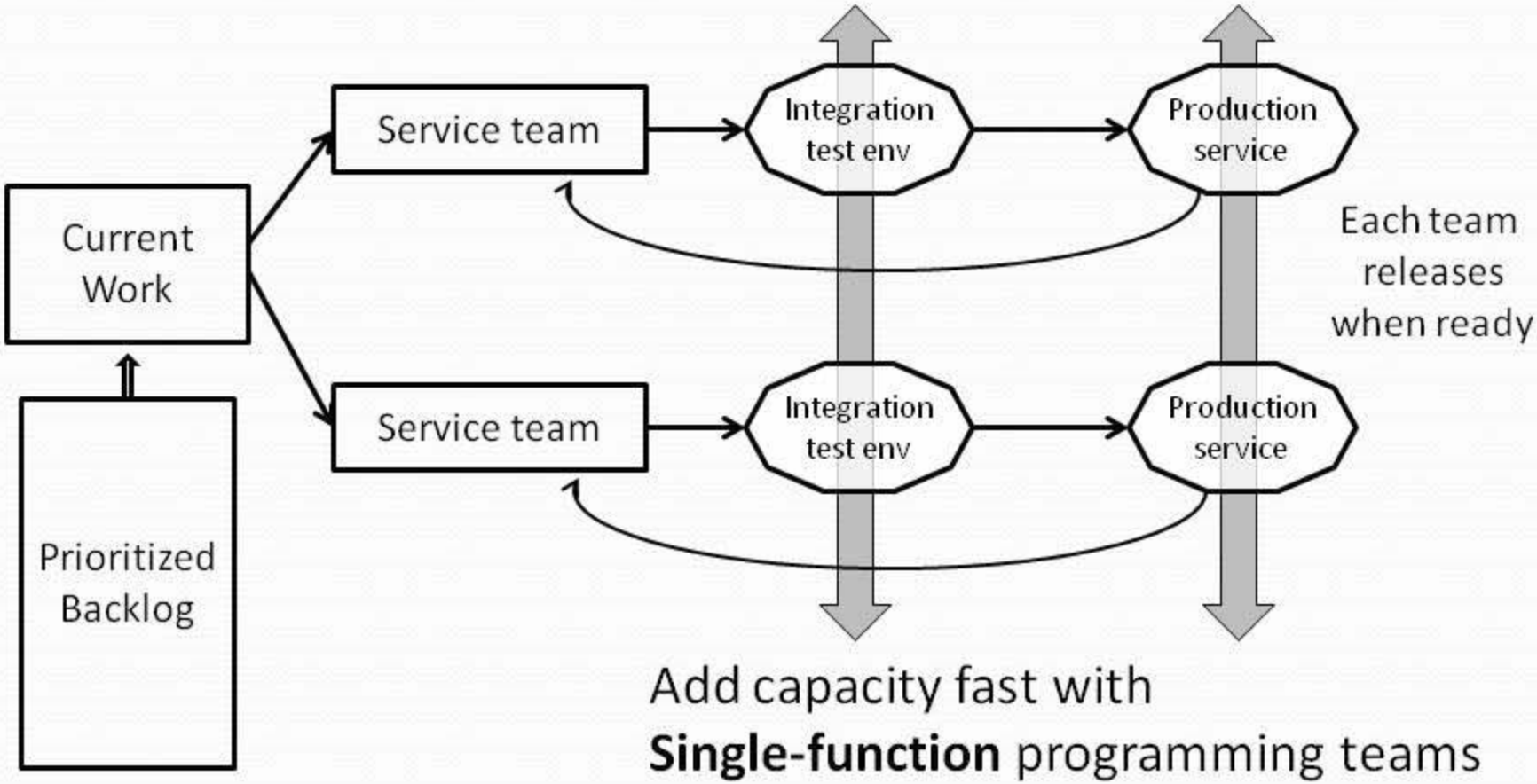
- Continuous integration helps teams coordinate.
- See dependencies between "producers" and "consumers"
- Errors and conflicts show related team contact info
- Meetings and changes negotiated between two teams, not many

assembla

Teams are largely self-managing



Scaling

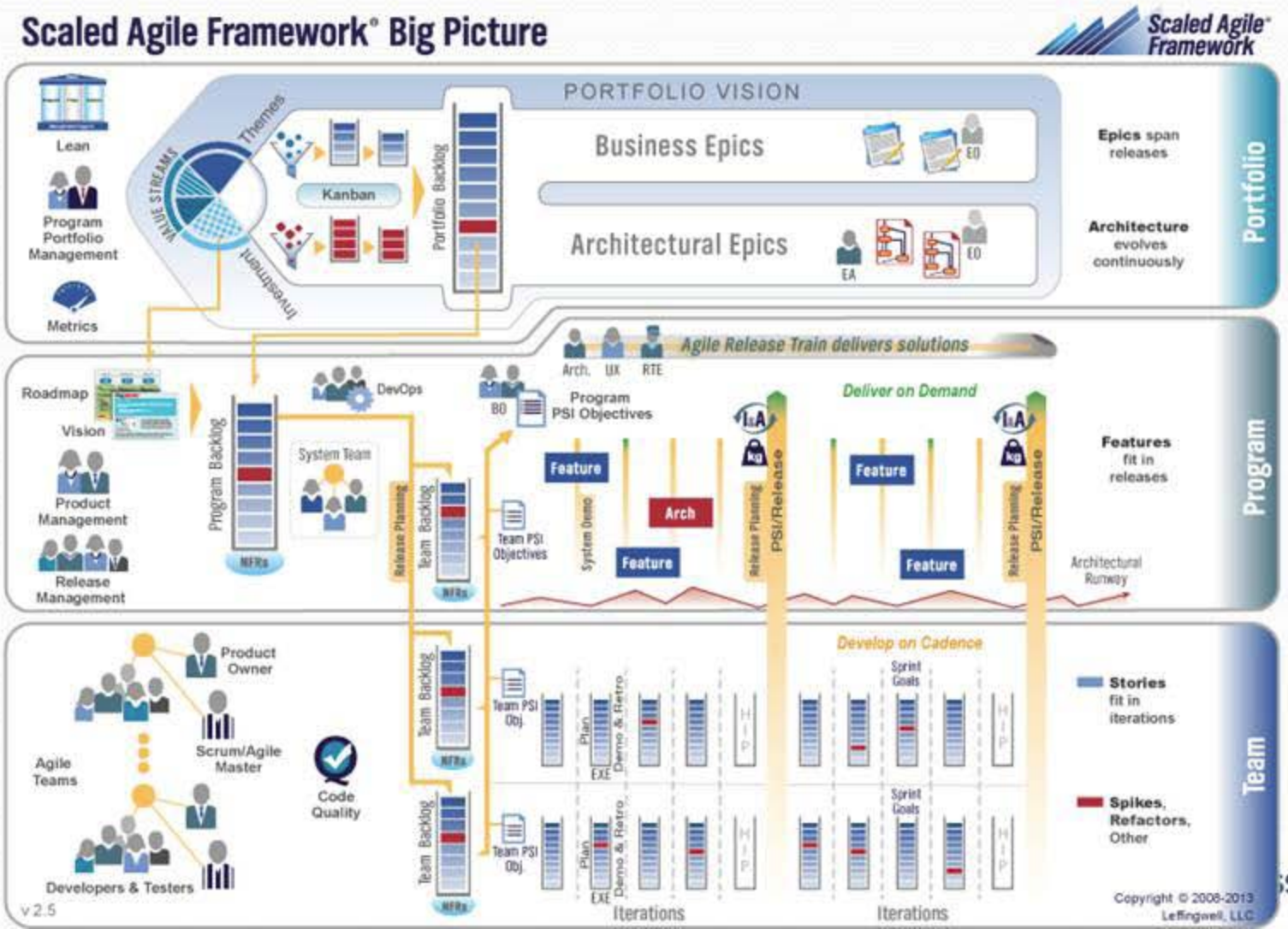


Add capacity fast with
Single-function programming teams

Hubspot – Great at Mid-scale

- Transformed a monolithic app to 200 services over one year
- 3-person programming teams. Each of 20 teams is responsible for about 10 services
- Dev teams responsible for design, programming, testing, release, monitoring, and responding to production problems. No full-time QA. Shared PM and UX.
- Lot's of tooling and dashboards to help teams deploy, manage, and monitor their services
- Feedback from customer support also grouped by team

SAFe (Copyright Dean Leffingwell)



Ways to Scale

Scrum + SAFe

- Add more hierarchy
- Complex multifunction teams
- Hold big meetings and teleconferences
- Block everyone into one cadence
- Coordinate big releases

Top Tech Companies

- Automate **management**, as well as testing and deployment.
- Dev-lead teams
- Communicate peer to peer
- Unblock! teams to move as fast a possible
- Release more frequently

Competing with MAXOS

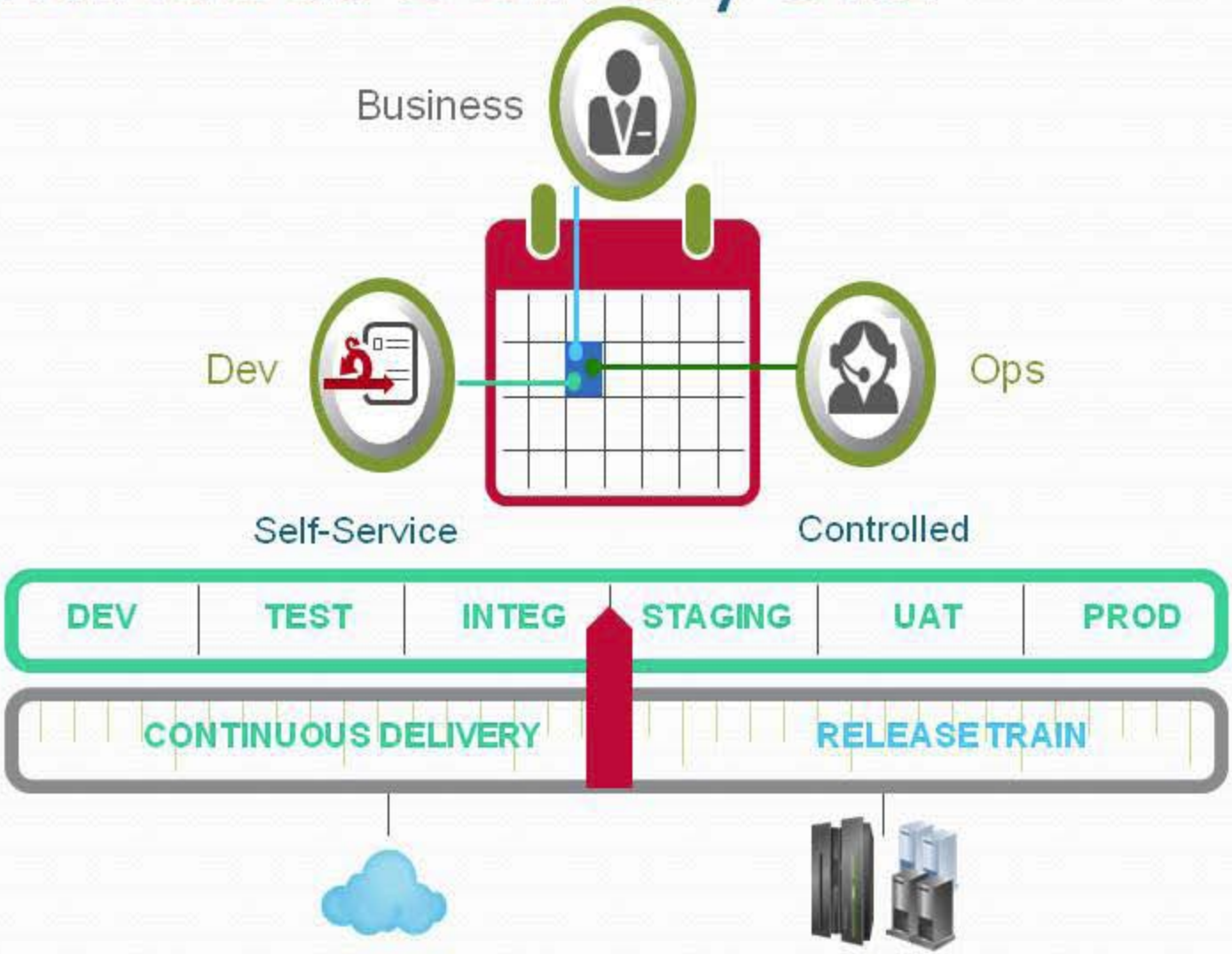
The secret weapon that Silicon Valley is using to disrupt and destroy competitors

- Leading retailer deploys changes to their monolithic online ordering app once every six weeks. Ops holds for three weeks to make sure the complete system is stable.
- Amazon has thousands of services and more than 1000 service teams. They release something about once every 11.6 seconds. In the time that Retailer X takes to try one new release, Amazon has made 300,000 changes.
- Amazon hosting competitor: “It’s an emergency”.

assembla

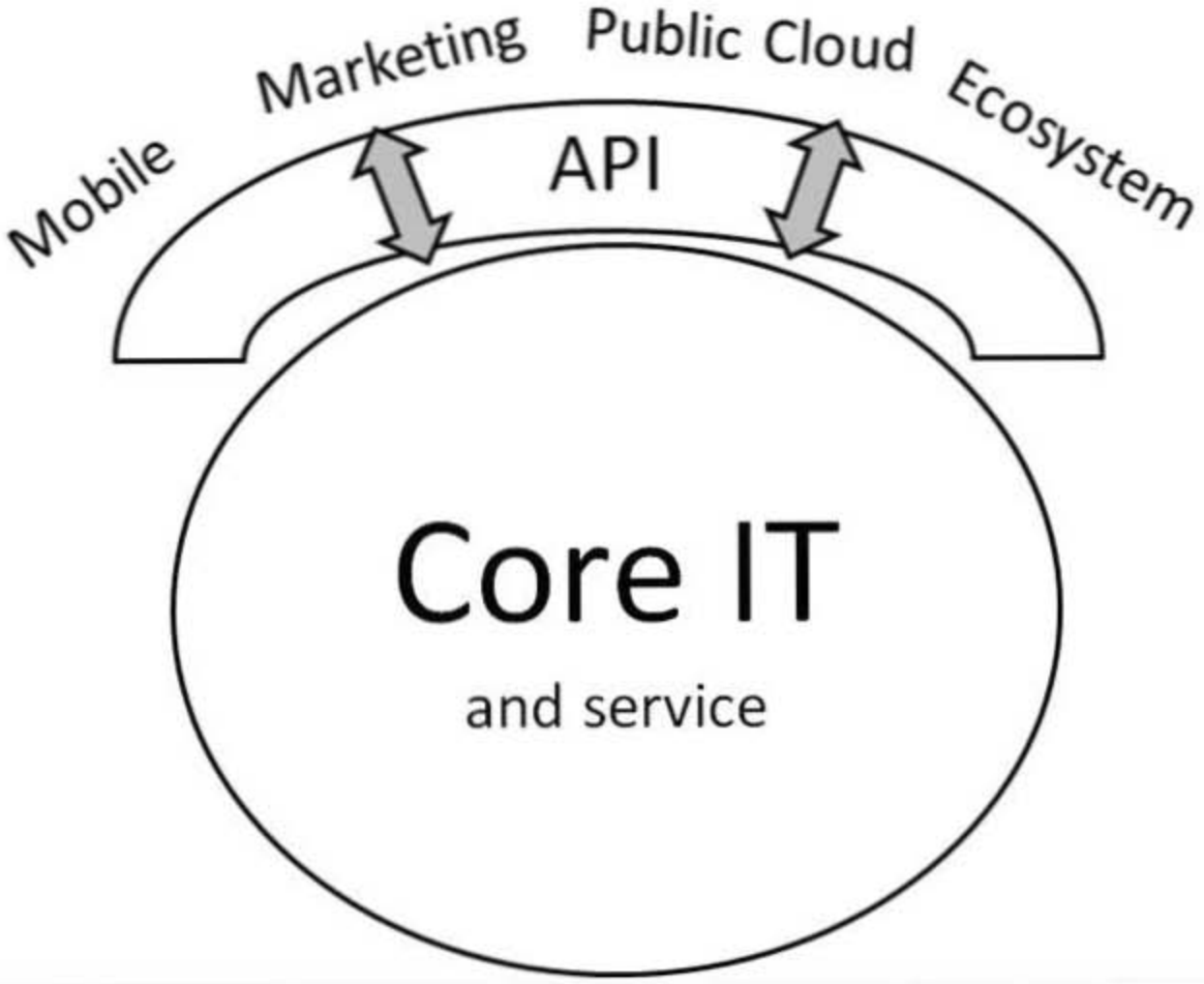
Adoption, Simplified

Continuous Delivery Dial



From Steve Brodie and Rohit Jainendran

Core IT and Fast IT



assembla

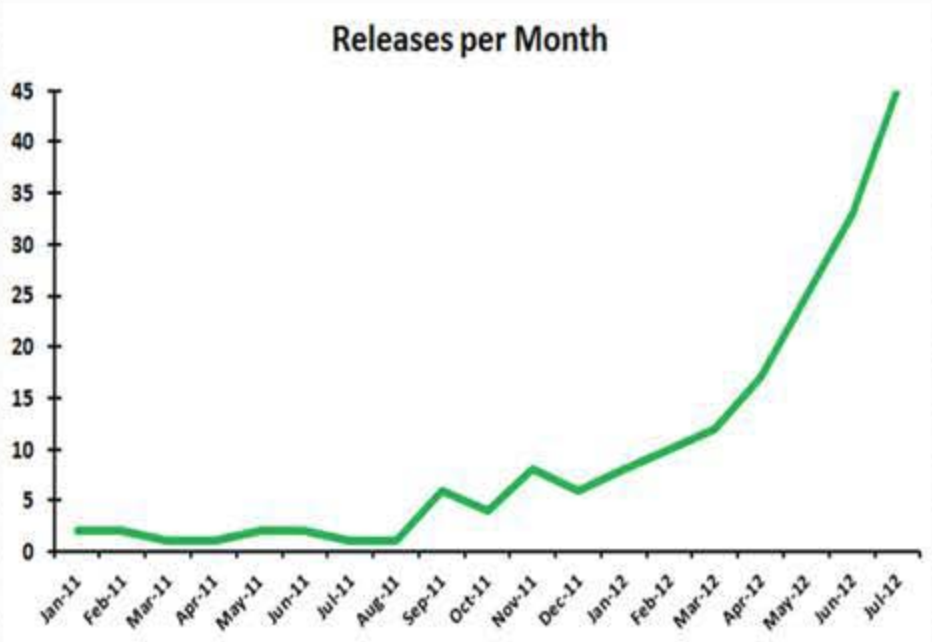
Incentives for Continuous Flow

You don't need culture change. You only need to release more frequently

1. If you do **code review**, you can get automated. No browbeating and cajoling is required. Developers will ask for tests when they review.
2. If developers decide to release, they will take more responsibility for testing and automated **tests**.
3. If you **release** more frequently, developers will quickly learn not to break the build
4. If PM's take **unveil and measurement** responsibility, they will make better features

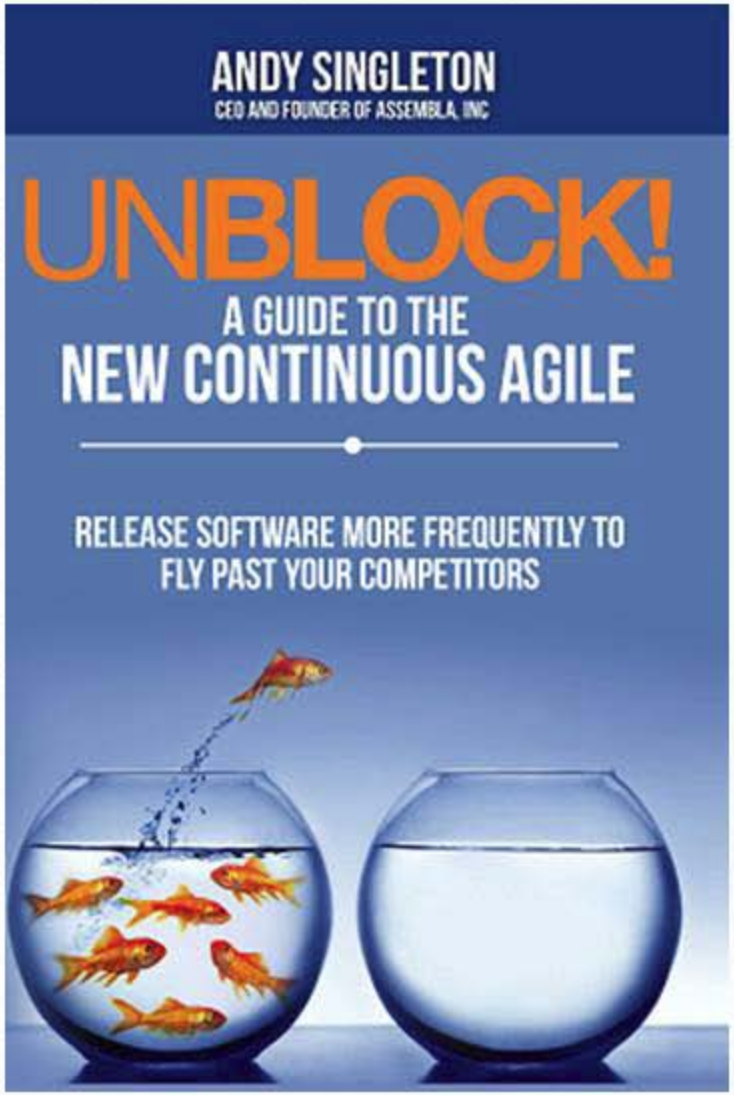
Our Master Plan

1. Release more frequently
2. Improve



assembla

www.continuousagile.com/unblock



Some terms

- Continuous Integration – run automated tests on every code change
- Continuous Delivery – Update releasable version at least once per day, and release when ready
- Continuous Release – Release every change (typically for SaaS)
- Continuous Agile – Kanban task management, continuous delivery code management, and continuous product management (metrics + story owners)
- MAXOS - All of the above with many unblocked services

Single-functional Scales Faster

- Building multi-functional teams is complicated, requires donations by multiple departments, and takes time and coaching and "culture"
- MAXOS service teams are typically run by developers, who pull in other experts as needed. Often three people. Can start up instantly with only one tech lead.
- You need system operations training and capacity for any new components
- Maxos service teams take responsibility for operating their services

In this session

- Continuous Agile management
- Code contribution and continuous delivery
- Changing roles – Dev, QA, PM/PO
- Scaling - MAXOS (Matrix of Services)
- Adoption, Simplified

assembla