Example Products
GoToMeeting
GoToConnect
GoToWebinar

Characteristics
SaaS products (+ installed parts)
World wide usage
24/7 usage
Frequent usage:
Millions of interactions per day
Complex
Our today’s software systems as a whole are complex

Complexity indicators

– No single human does understand the system as a whole
– Grew over a period of 15+ years
– Worldwide distributed infrastructure and development
– Interacts with millions of individual users per day (i.e. local networks, local machines, …)
– “😲 This is the way some people use/operate our product...”
Conclusion #1

Adopt **lean principles** to evolve complex systems
Implement **lean principles** to **learn fast**

- Build, measure, learn
- Frequent small incremental steps and early frequent feedback
Break evolution down to small increments to learn from

Per increment
- Construct / Design with reduced complexity, small time invest
- Build
- Measure
- Learn
Conclusion #2

Take small architecture and project risks and learn from actual system behavior. But make sure to learn.

Learn from feedback. There cannot be full knowledge upfront construction.
How to Learn?
Continuously & Early!
Continuous feedback in working context

Research cooperation project between KIT, Robert Heinrich, Maximilian Wagner and LogMeIn
Continuous feedback in working context

Research cooperation project between KIT, Robert Heinrich, Maximilian Wagner and LogMeIn
Approach Overview

Increment incl. telemetry implemented

(guided) implementation

deploy

JIRA Epic:
Increment incl. metrics definition

JIRA accessible dashboard:
Results

Production infrastructure:
Capture telemetry

Result:
appraise
We want at least 1,300 daily active users by August 31, 2019 for this feature.

Each user request is processed at the server at latest after 300ms; median: 173 ms.
Example Query on Product Usage

Source my_source
sourceType: Splunk
sourceID: citrix_ma_splunkcloud
timeParamID: None
search: "search index=launcher_summary source=summarysearch-differentjoins"
;

Requirement my_requirement
"An example comment to illustrate how comments are used."
response_time in my_source <= 300

from 2019-10-01 until 2019-10-03
updateInterval: 6 hours
updateWidth: 24 hours
;
Drivers

Little overhead upfront – eliminating follow-up work
Lead design and implementation through hypotheses
Immediate feedback loop
Conclusion #3

Establish a **tight feedback cycle** to learn from **actual execution** of complex software systems.
“So we plan, execute and observe. We execute! What about risk mitigation?”
Making Risks Calculatable

**Built-In**
- Small changes
- Fast feedback (i.e. short duration of feedback cycles)

**Rollout strategy**
- Limit blast radius
  - Phased rollout
  - Canary releases
  - A/B testing

**User feedback**
- Customer Effort Score (CES)
- Transactional Net Promoter Score (tNPS)
Conclusion #4

Combine with risk mitigation matching continuous delivery.

Listen to customers and observe customer systems interacting with own systems.
Adopt Continuous Delivery

- Implement continuous delivery as a competitive differentiator

- **Continuous**: at the cadence required by business, achieved through agility in all areas

- **Value**: small incremental changes to reduce complexity and risk vs. Big Bang

- **High quality**: based on rock solid test automation including automated regression, integration, load tests as well as tests in production

- **Goal**: increase velocity of the feedback loop between customers and companies that serve them
Most today’s software systems are complex.

- Learn from actual system behavior. **Automate.**
- Learn from customer feedback. **Validate.**
- Take small risks. **Do change.**
Example Query on Product Usage

Source my_source
sourceType: Splunk
sourceID: citrix_ma_splunkcloud
timeParamID: None
search: "search index=launcher_summary source=summarysearch-differentjoins"

Comparison my_comp
"This is a default comment."
count in my_source for (g2m, g2w, g2p_viewer, g2t) in product

aggregateType: Sum
from 2019-03-01 until 2019-03-07
updateInterval: 6 hours
updateWidth: 24 hours

;