Foundations of Software Engineering

Architecture – From Styles to Hypes

Christian Kästner

Learning Goals

- Recognize architectural styles and their implications
- Reason about system structures and their tradeoffs with architectural views and styles
- Reason about tradeoffs of Microservice architectures
- Understand the key ideas of DevOps
- Understand ideas of architecture evaluation

Interlude: Teamwork Clinic

Common Issues

- Dealing with interpersonal issues
- Dealing with different expectations
- Dealing with slipping commitments

Assumptions about Relationships

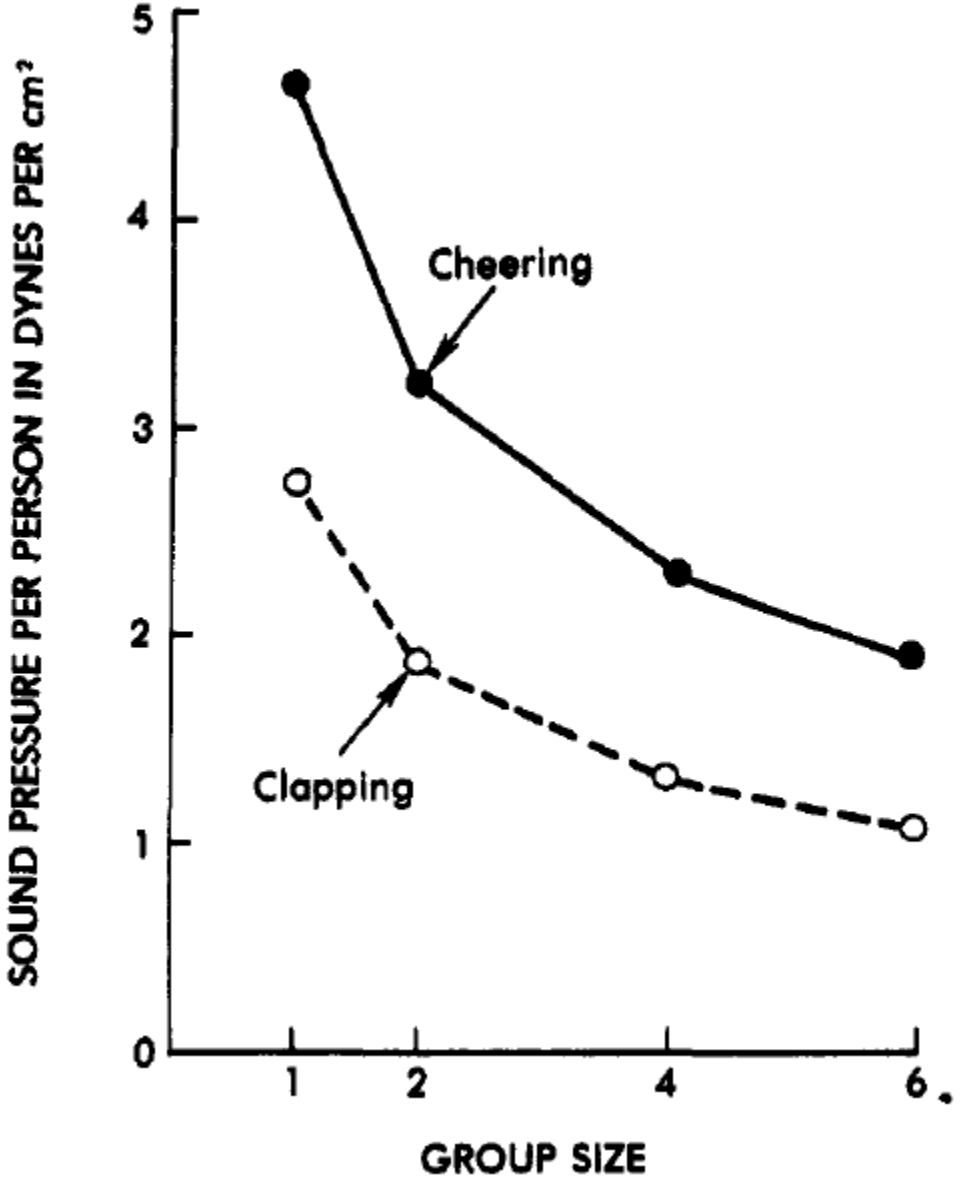
- Level -1: Exploitation, No Relationships
- Level 1: Transactional Role, Civility
- Level 2: Working Relationship, Rec. as Unique Person
- Level 3: Strong Emotions, Love and Intimacy
- Expectations differ by country, religion, ethnicity, and local cultures

[Schein 2016]

Cultural Islands

- Temporarily suspend rules to maintain face
- “Talk to the Camp Fire”
- 1 Check-In Question without interruptions
- 2 Reflection and open conversation
- External facilitator useful

[Schein 2016]



Latane, Bibb, Kipling Williams, and Stephen Harkins. "Many hands make light the work: The causes and consequences of social loafing." *Journal of personality and social psychology* 37.6 (1979): 822.

Social loafing

- People exerting less effort within a group
- Reasons
  - Diffusion of responsibility
  - Motivation
  - Dispensability of effort / missing recognition
  - Avoid pulling everybody / "sucker effect"
  - Submaximal goal setting
- “Evaluation potential, expectations of co-worker performance, task meaningfulness, and culture had especially strong influence”

Karau, Steven J., and Kipling D. Williams. "Social loafing: A meta-analytic review and theoretical integration." *Journal of personality and social psychology* 65.4 (1993): 681.

Social Loafing: Mitigation Strategies

- Involve all team members, colocation
- Assign specific tasks with individual responsibility
  - Increase identifiability
  - Team contracts, measurement
- Provide choices in selecting tasks
- Promote involvement, challenge developers
- Reviews and feedback
- Team cohesion, team forming exercises
- Small teams

Mitigating Social Loafing: Responsibilities & Buy-In

- Involve team members in decision making
- Assign responsibilities (ideally goals not tasks)
- Record decisions and commitments; make record available

Common Issues

- Dealing with interpersonal issues
- Dealing with different expectations
- Dealing with slipping commitments

More on Architectural Reasoning

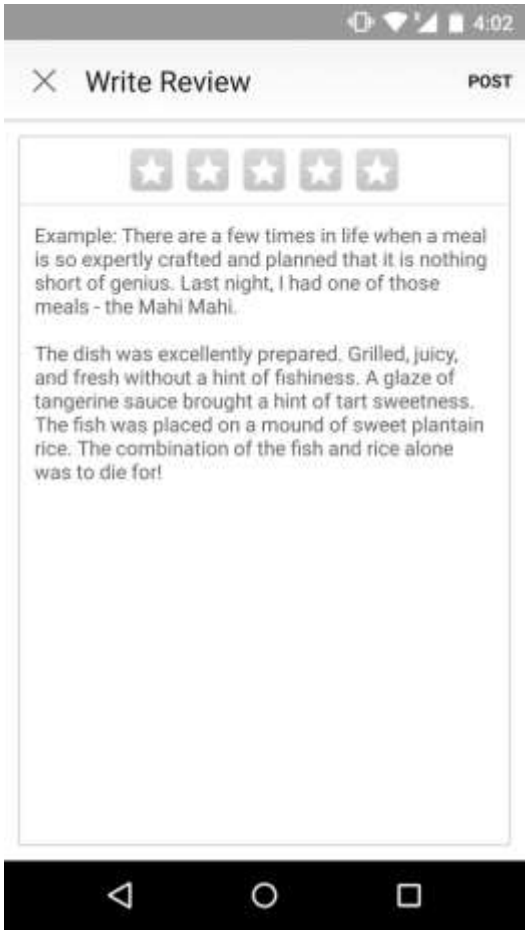
Client

Server

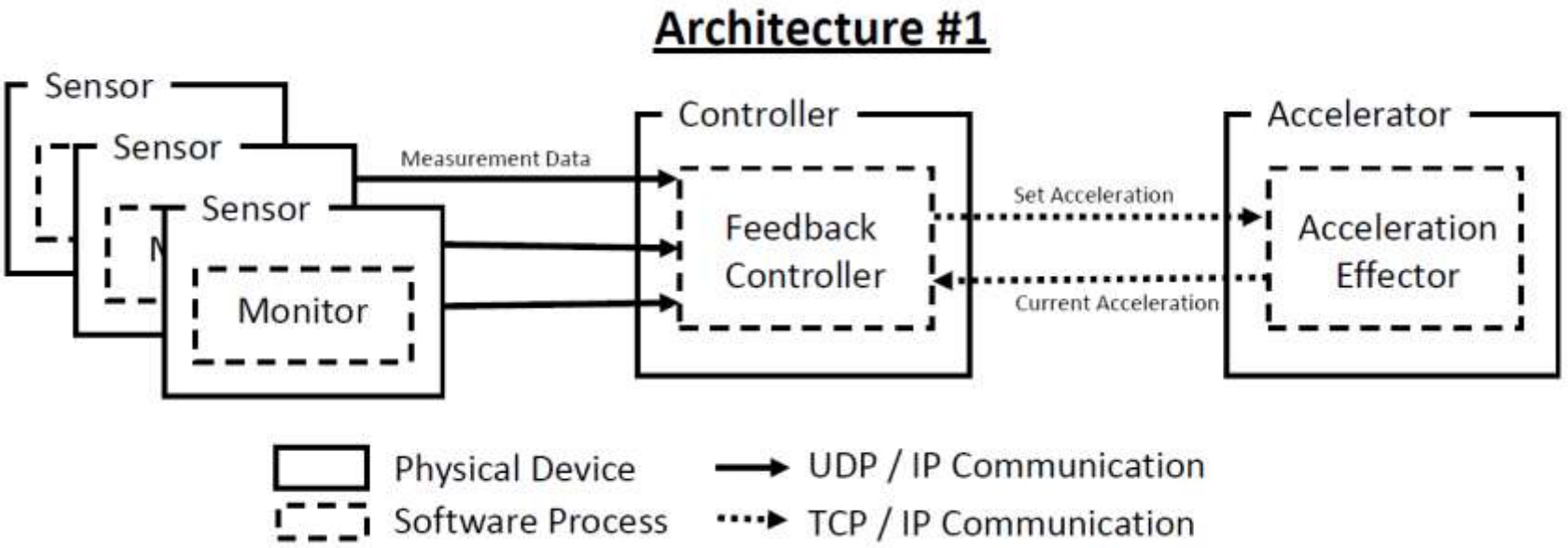
Database

Where to validate user input?

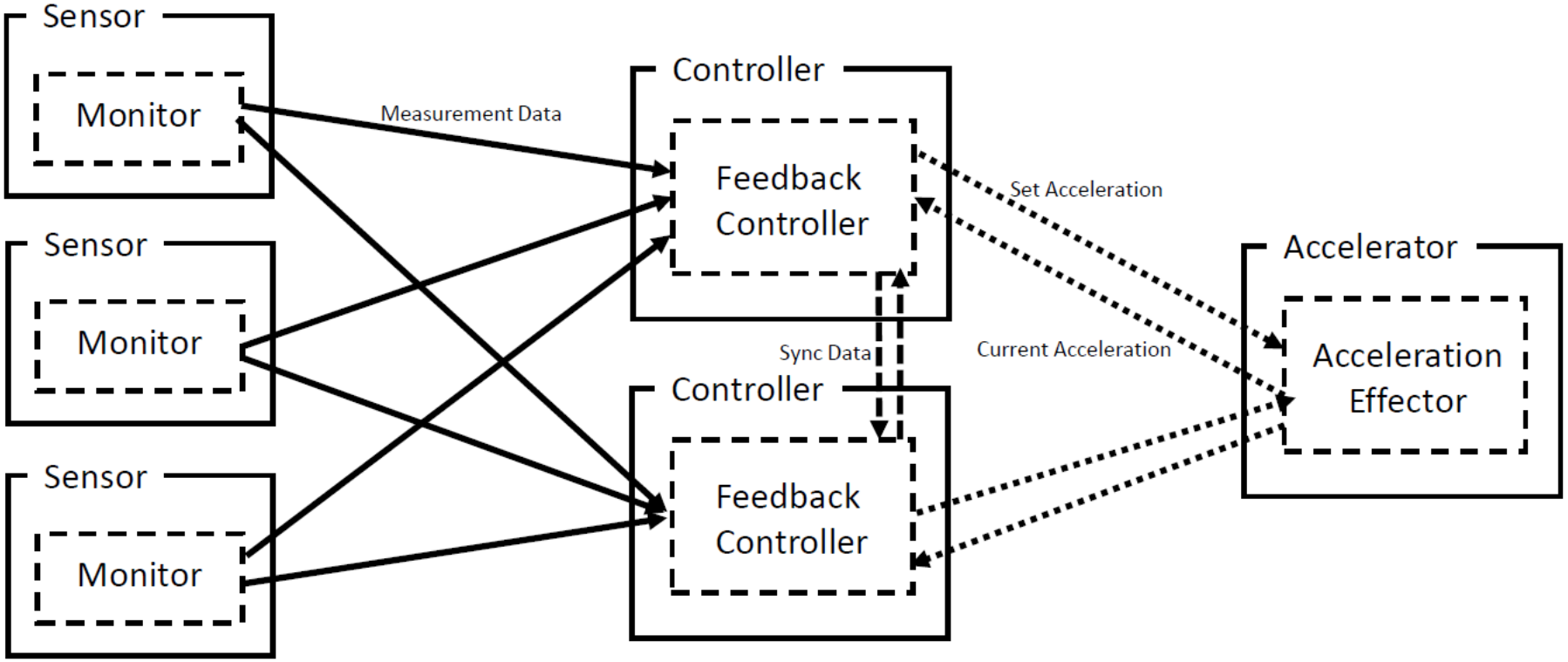
Example: Yelp App



FeopleCars Scenario (Final Exam 2015)



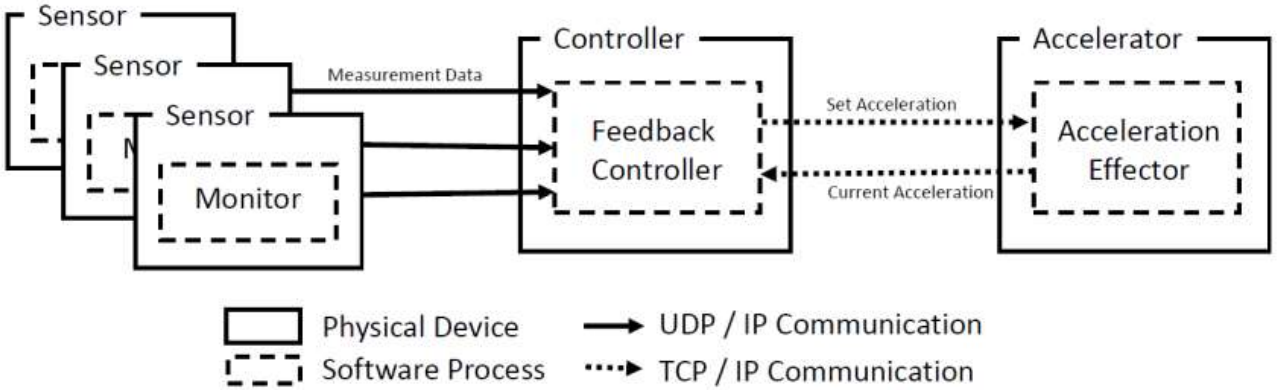
Architecture #2



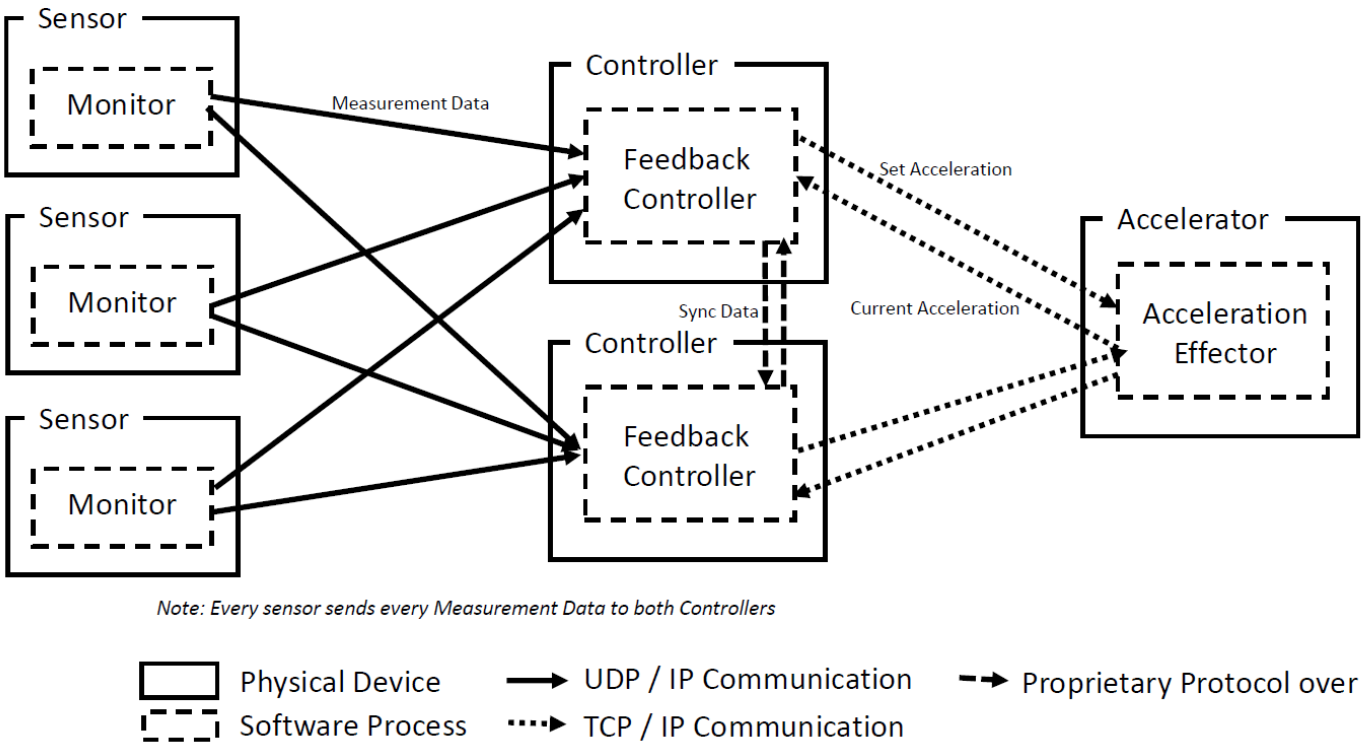
Note: Every sensor sends every Measurement Data to both Controllers

Physical Device
Software Process
UDP / IP Communication
TCP / IP Communication
Proprietary Protocol over IP

Architecture #1



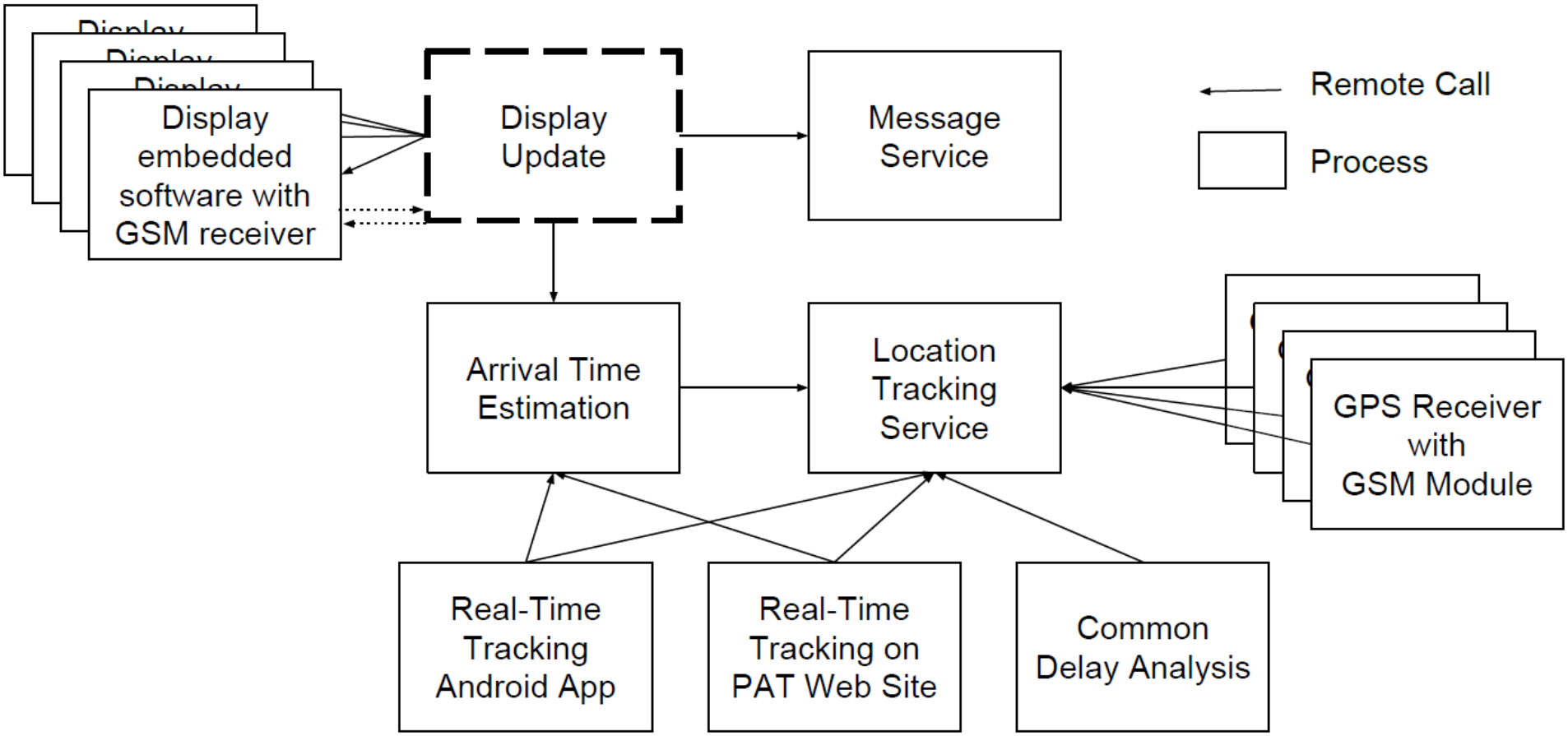
Architecture #2



Note: Every sensor sends every Measurement Data to both Controllers

What qualities can you reason and not reason about? Tradeoffs? For which quality is Arch 1 better? For which Arch 2?

Real-time Bus Tracking (Midterm 2016)



Architectural Styles

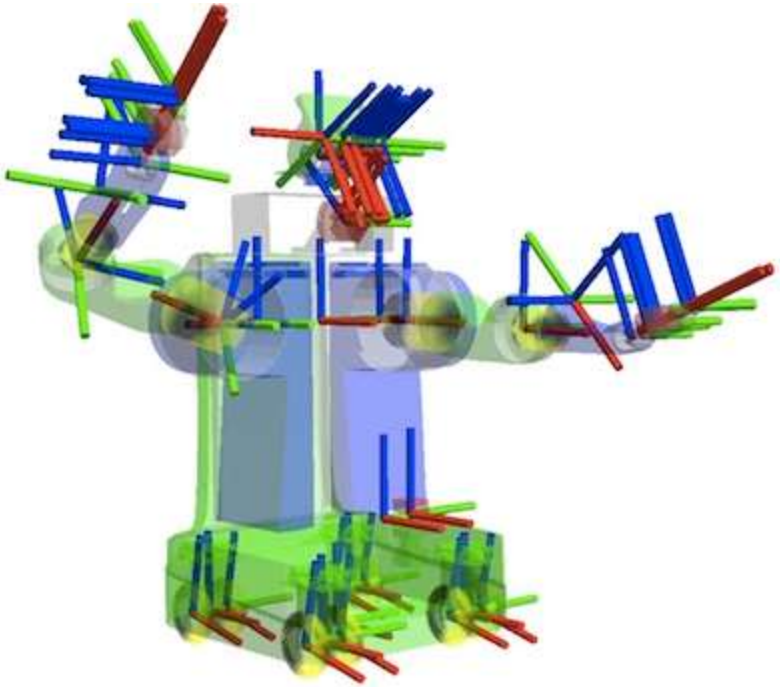
- Pipes and Filters
- Object-Oriented Organization, Services
- Event-Based, Implicit Invocation
- Layered System
- Repositories
- …

Give one new example of a system with each architectural style and discuss why it is (or is not) appropriate.

Case Study: ROS

ROS

- "Robot Operating System", open source
- The philosophical goals of ROS can be summarized as:
  - Peer-to-peer
  - Tools-based
  - Multi-lingual
  - Thin
  - Free and Open-Source



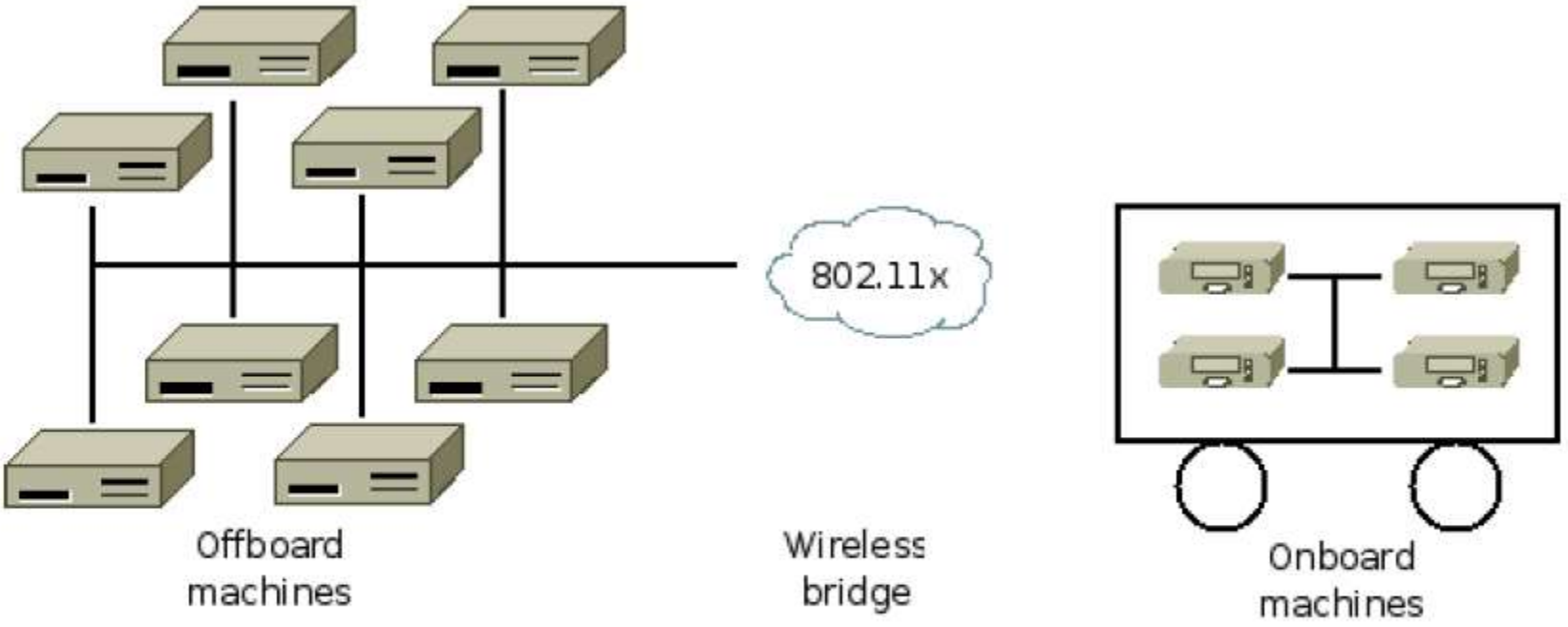


Fig. 1. A typical ROS network configuration

Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." *ICRA workshop on open source software*. Vol. 3. No. 3.2. 2009.

Quality Goals?

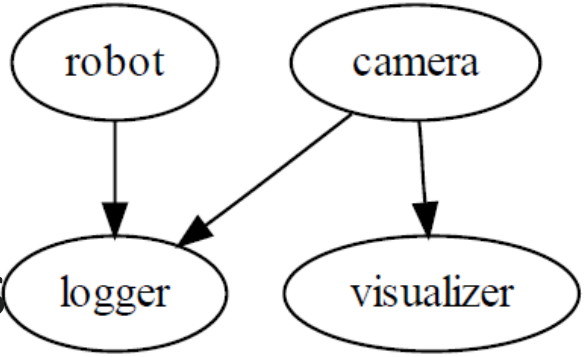
" A Distributed, Modular Design"

- users can use as much or as little of ROS as they desire
- modularity of ROS allows you to pick and choose which parts are useful for you and which parts you'd rather implement yourself
- large community of user-contributed packages (3000 packages)

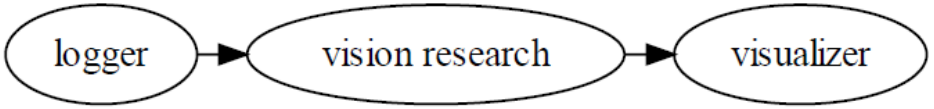
Architectural Style?

- Pipes and Filters
- Object-Oriented Organization, Services
- Event-Based, Implicit Invocation
- Layered System
- Repositories
- …

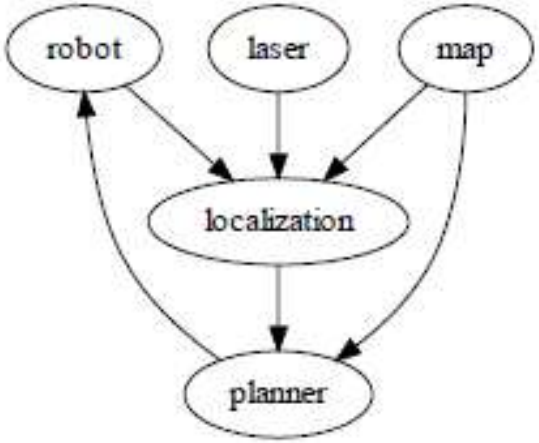
ROS Communication Infrastructure

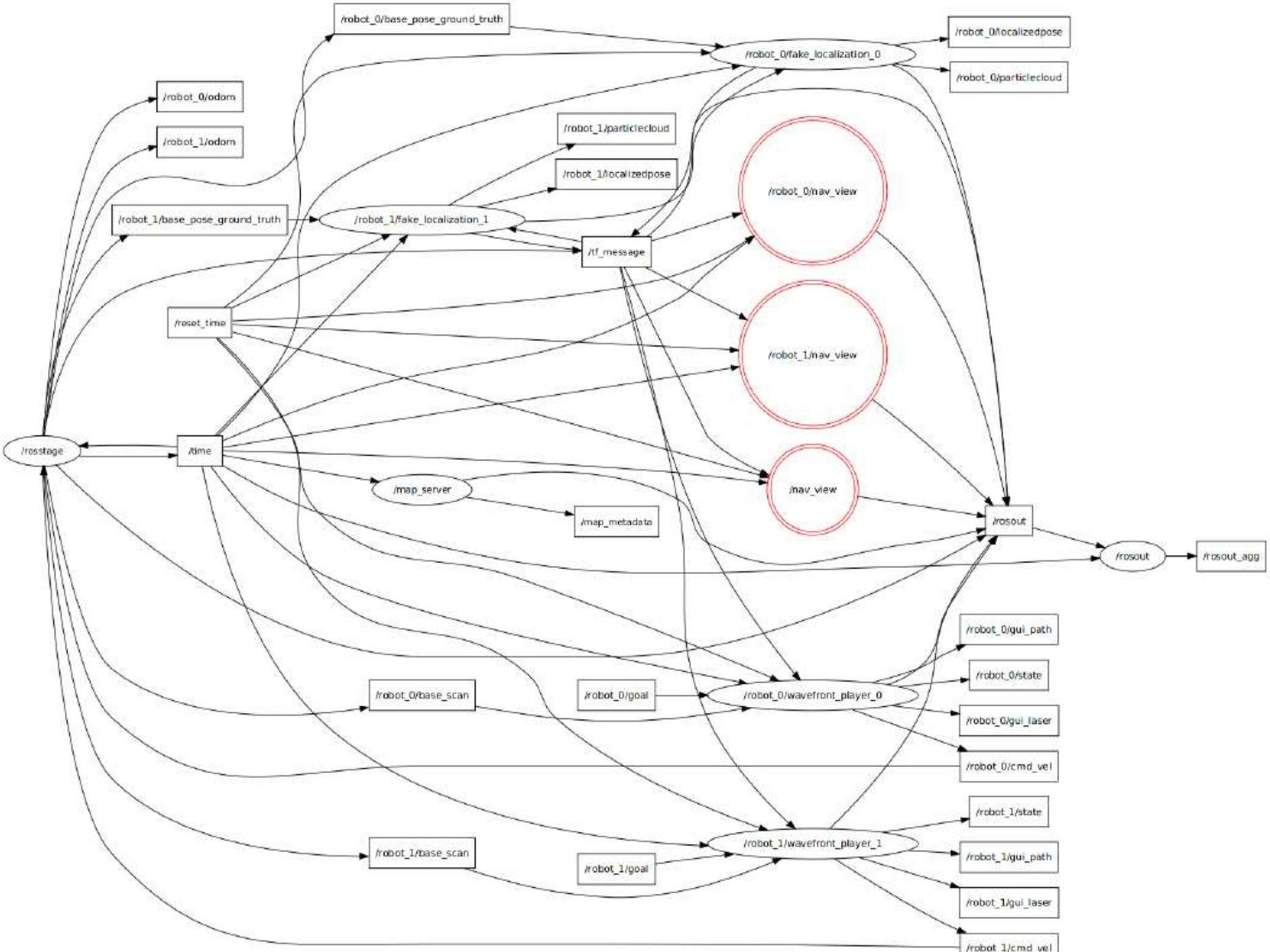


- Message Passing
  - Publish/subscribe for channels
  - Messages interfaces through IDL (cross-language)



- Recording and Playback of Messages
- Remote procedure calls
- Share configuration through global key-value store




/robot_0/base_pose_ground_truth
/robot_0/fake_localization_0
/robot_0/localizedpose
/robot_0/particlecloud
/robot_0/odom
/robot_1/odom
/robot_1/particlecloud
/robot_1/localizedpose
/robot_0/nav_view
/robot_1/base_pose_ground_truth
/robot_1/fake_localization_1
/tf_message
/reset_time
/robot_1/nav_view
/rosstage
/time
/map_server
/nav_view
/map_metadata
/rosout
/rosout
/rosout_agg
/robot_0/gui_path
/robot_0/state
/robot_0/base_scan
/robot_0/goal
/robot_0/wavefront_player_0
/robot_0/gui_laser
/robot_0/cmd_vel
/robot_1/state
/robot_1/base_scan
/robot_1/wavefront_player_1
/robot_1/goal
/robot_1/gui_path
/robot_1/gui_laser
/robot_1/cmd_vel

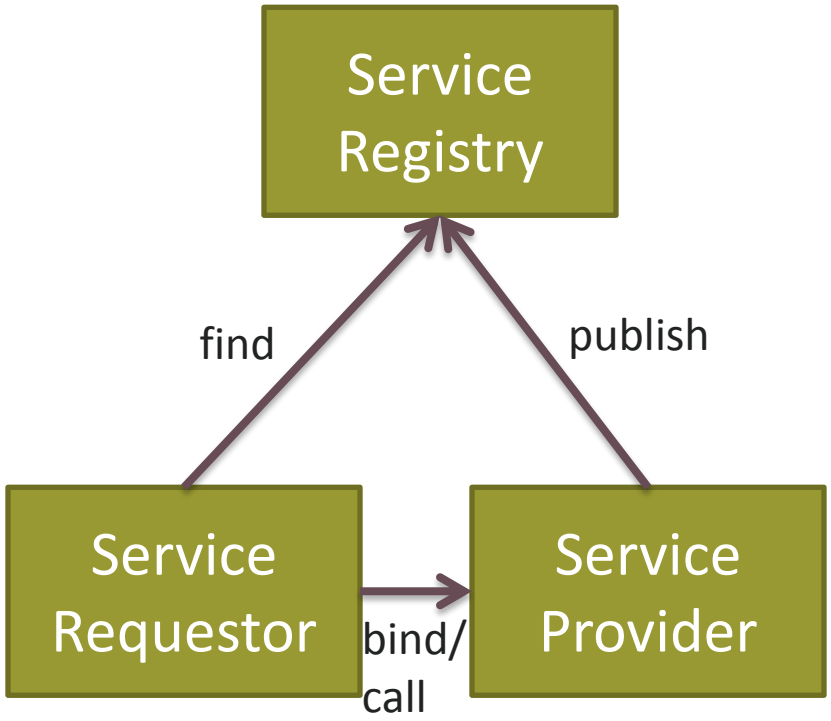
Tradeoff discussion

- Decoupling
- Reuse, Extensibility
- Reliability
- Understandability
- Performance
- Community contributions

A current architectural hype: Microservices

Service Oriented Architectures (SOA)

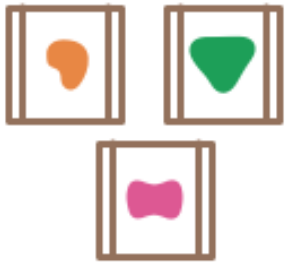
- Service: self-contained functionality
- Remote invocation, language-independent interface
- Dynamic lookup possible
- Often used to wrap legacy systems



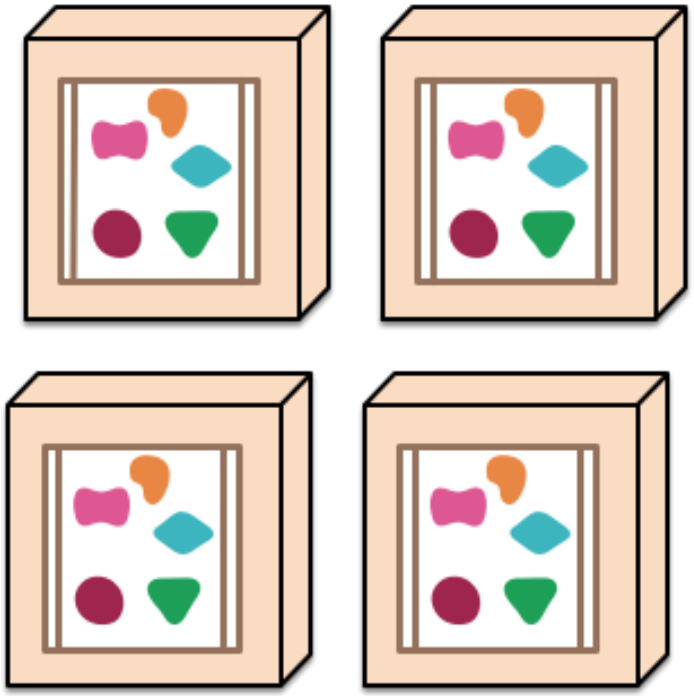
Microservices

A monolithic application puts all its functionality into a single process...

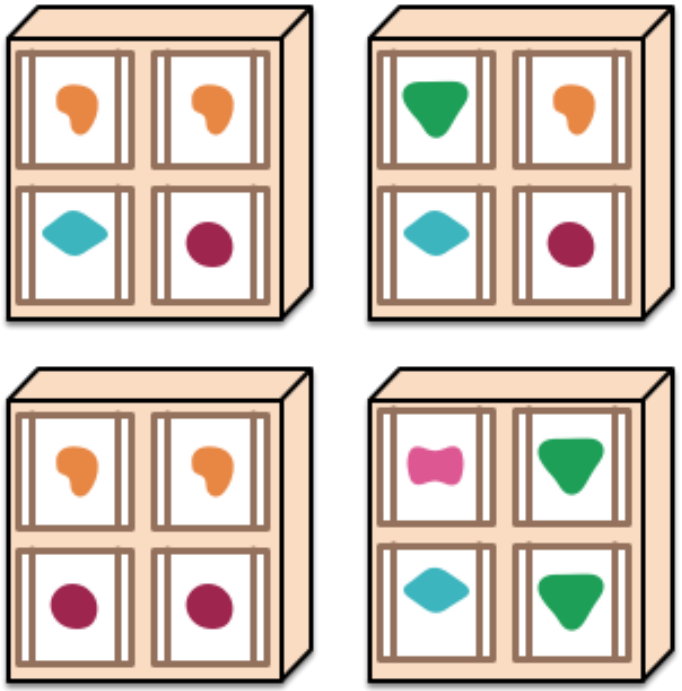
A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers

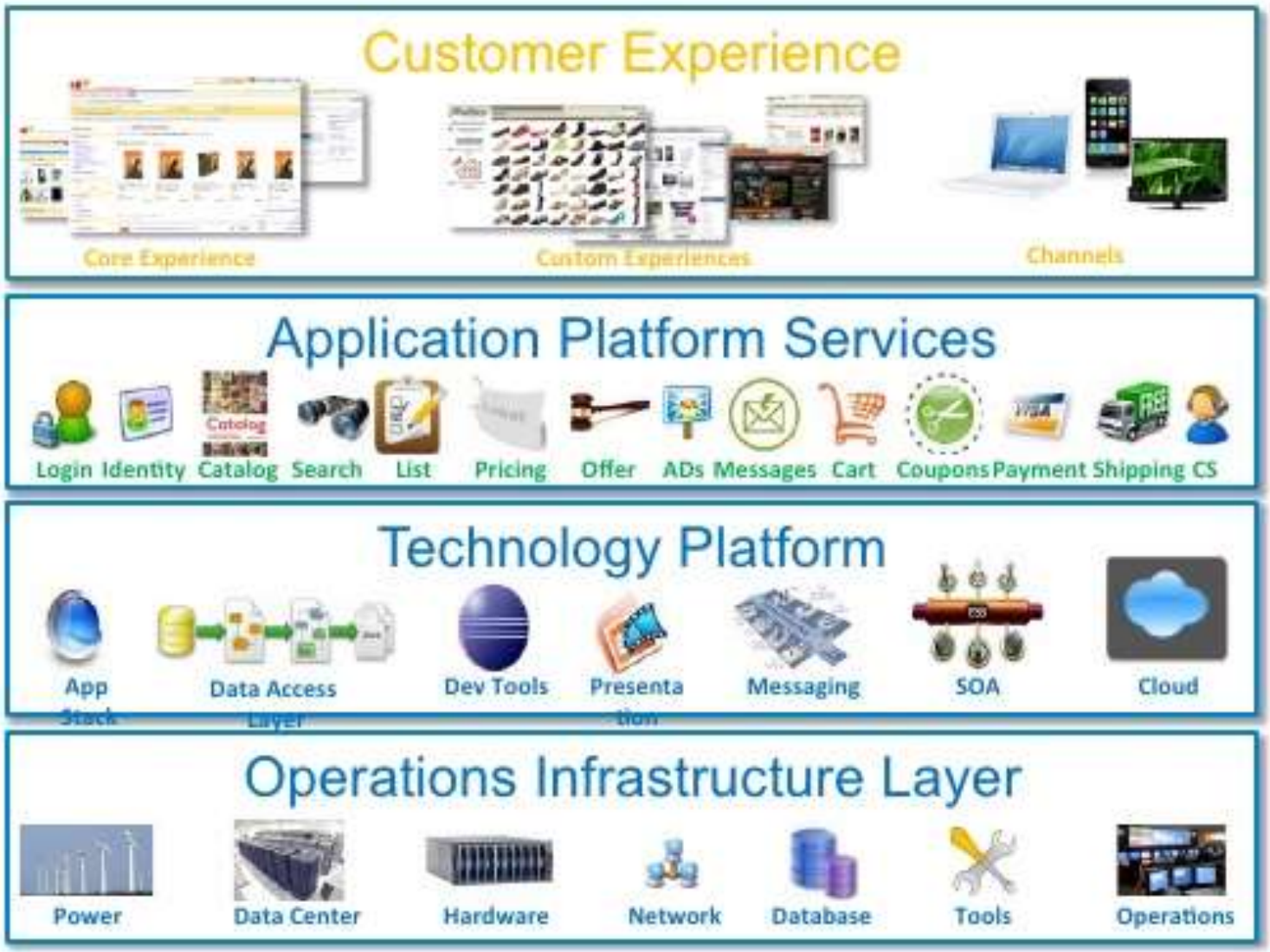


... and scales by distributing these services across servers, replicating as needed.



source: http://martinfowler.com/articles/microservices.html

Architecture Vision

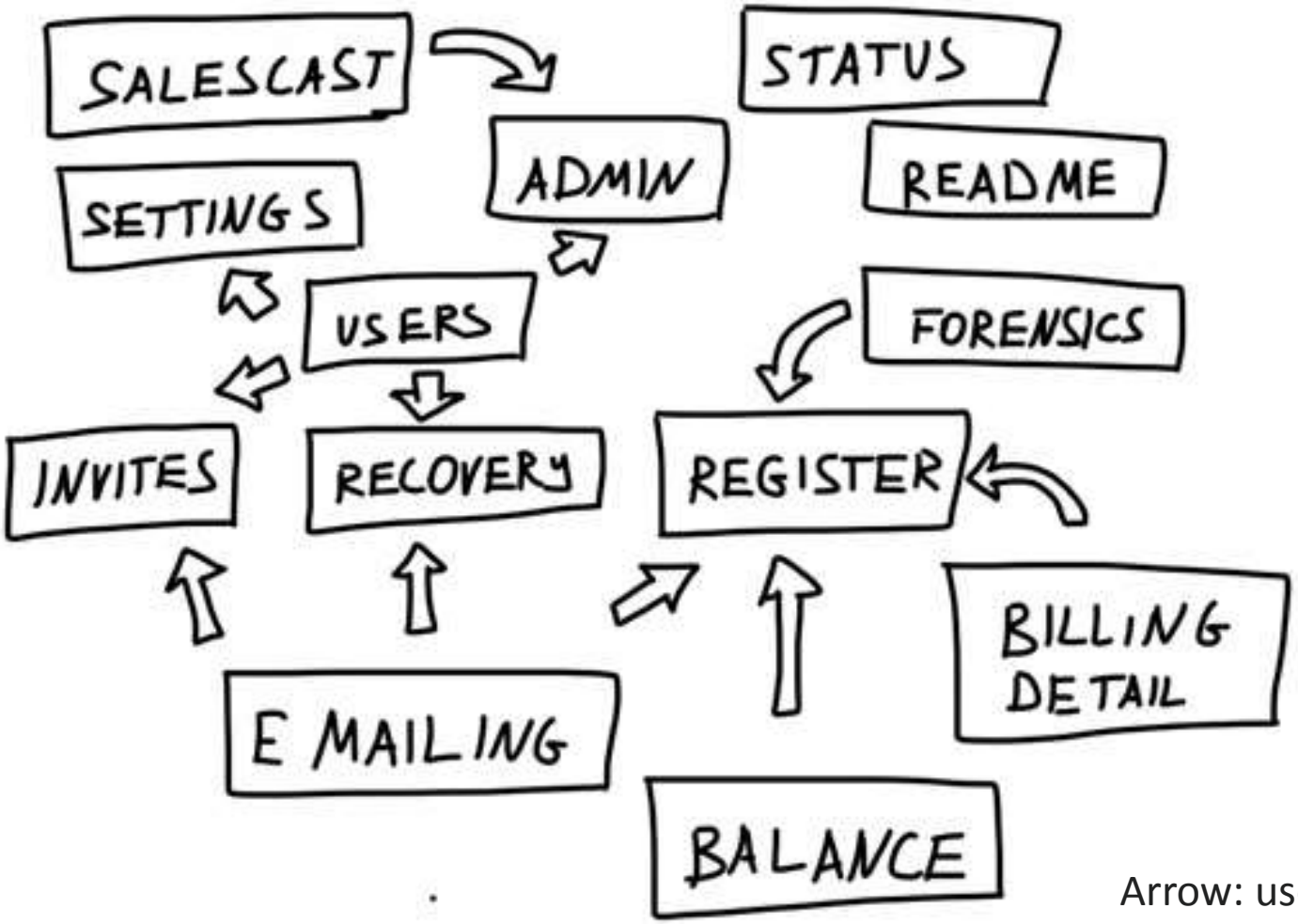


Microservices

- Building applications as suite of small and easy to replace services
  - fine grained, one functionality per service (sometimes 3-5 classes)
  - composable
  - easy to develop, test, and understand
  - fast (re)start, fault isolation
- Interplay of different systems and languages, no commitment to technology stack
- Easily deployable and replicable
- Embrace automation, embrace faults

Example Services

- Send text message / email / letter
- Credit card transaction
- Get product/profile image
- User management, settings
- Subscriptions
- Recommendations, ads

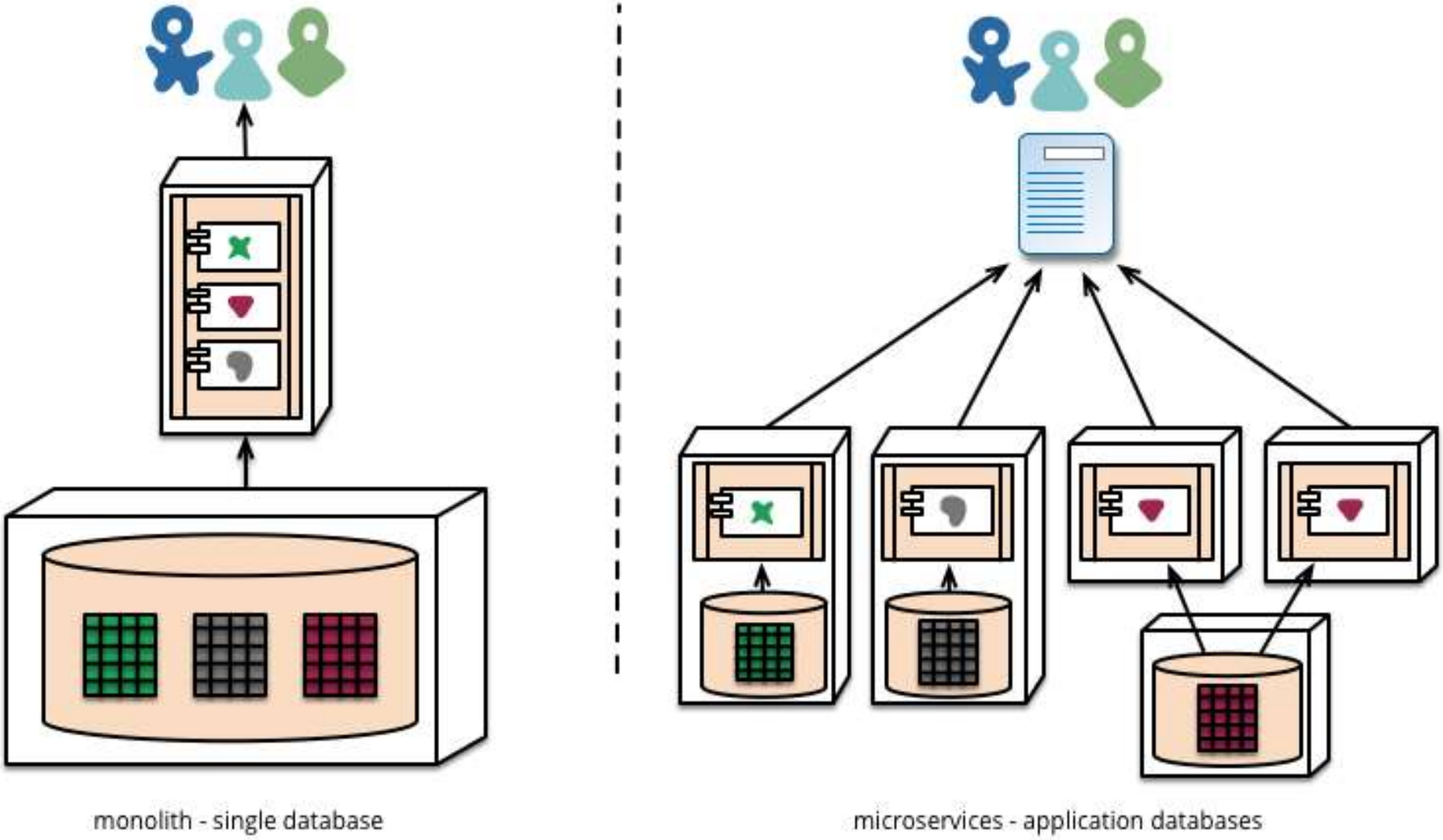


Arrow: use data from

source: https://abdullin.com/post/how-micro-services-approach-worked-out-in-production/

Technical Considerations

- HTTP/REST/JSON communication
- Independent development and deployment
- Self-contained services (e.g., each with own database)
  - multiple instances behind load-balancer
- Streamline deployment

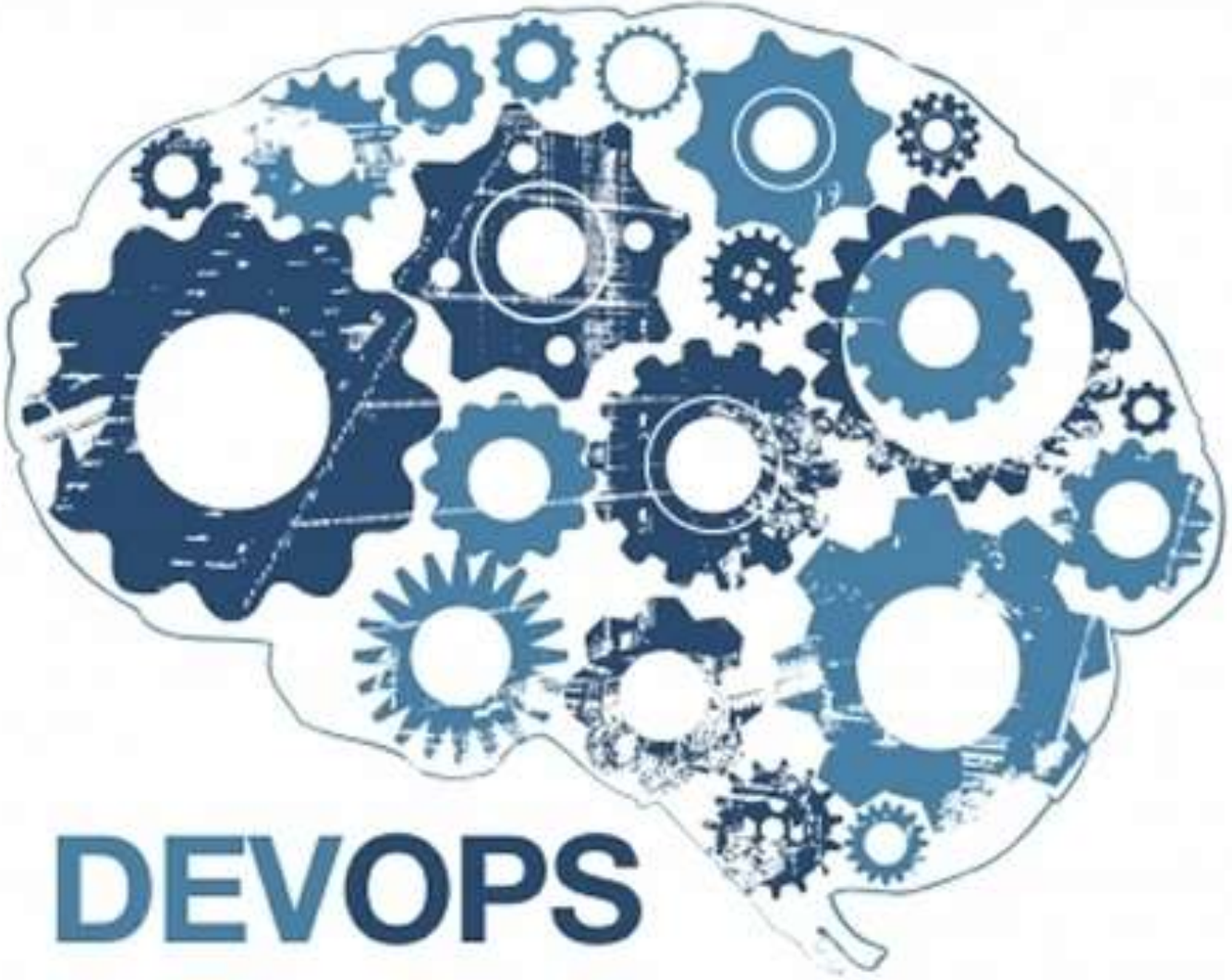


source: http://martinfowler.com/articles/microservices.html

Drawbacks (excerpt)

- Complexities of distributed systems
  - network latency, faults, inconsistencies
  - testing challenges
- Resource overhead, RPCs
- Shifting complexities to the network
- Operational complexity
- Adoption frequently by breaking down monolithic application

DevOps


DEVOPS

Automating Deployment

- Release several times per day
- Incremental rollout, quick rollback

Quality Goals

- Rapid releases and feedback (despite large code base)
- Quick onboarding

Infrastructure/Configuration as Code

- Manage configuration files in version control system
- Consistent infrastructure setup for testing, development, and deployment
- Configuration includes ports, target servers and routing, …

- Lightweight virtualization
- Sub-second boot time
- Sharable virtual images with full setup incl. configuration settings
- Used in development and deployment
- Separate docker images for separate services (web server, business logic, database, …)

Docker example

```
FROM ckaestne/typechef-kconfig

RUN apt-get -y update && apt-get install -y git-core gcc make

ADD https://github.com/.../master.tar.gz linuxa2.tar.gz
RUN tar xfz linuxa2.tar.gz; rm linuxa2.tar.gz;
        mv TypeChef-LinuxAnalysis2-master LinuxAnalysis2;
        cd LinuxAnalysis2; sbt mkrun

ADD config.txt LinuxAnalysis2/config.txt

CMD cd LinuxAnalysis2; ./run.sh
```

https://github.com/ckaestne/TypeChef-docker

Configuration Automation

- Chef, Puppet, Kubernetes, Mesos, Ansible
- Managing large-scale deployment (different containers on different machines)
- Matching containers to resources, scaling as needed based on metrics
- Automated restarts and upgrades
- Declarative high-level configuration generates specific configuration files
- Automated rollouts and rollbacks
- Dependency resolution on and setups

Case Study: Facebook

- Challenges
  - Configuration sprawl across many systems
  - Many tuning decisions during runtime
  - Configuration errors were common cause of downtime

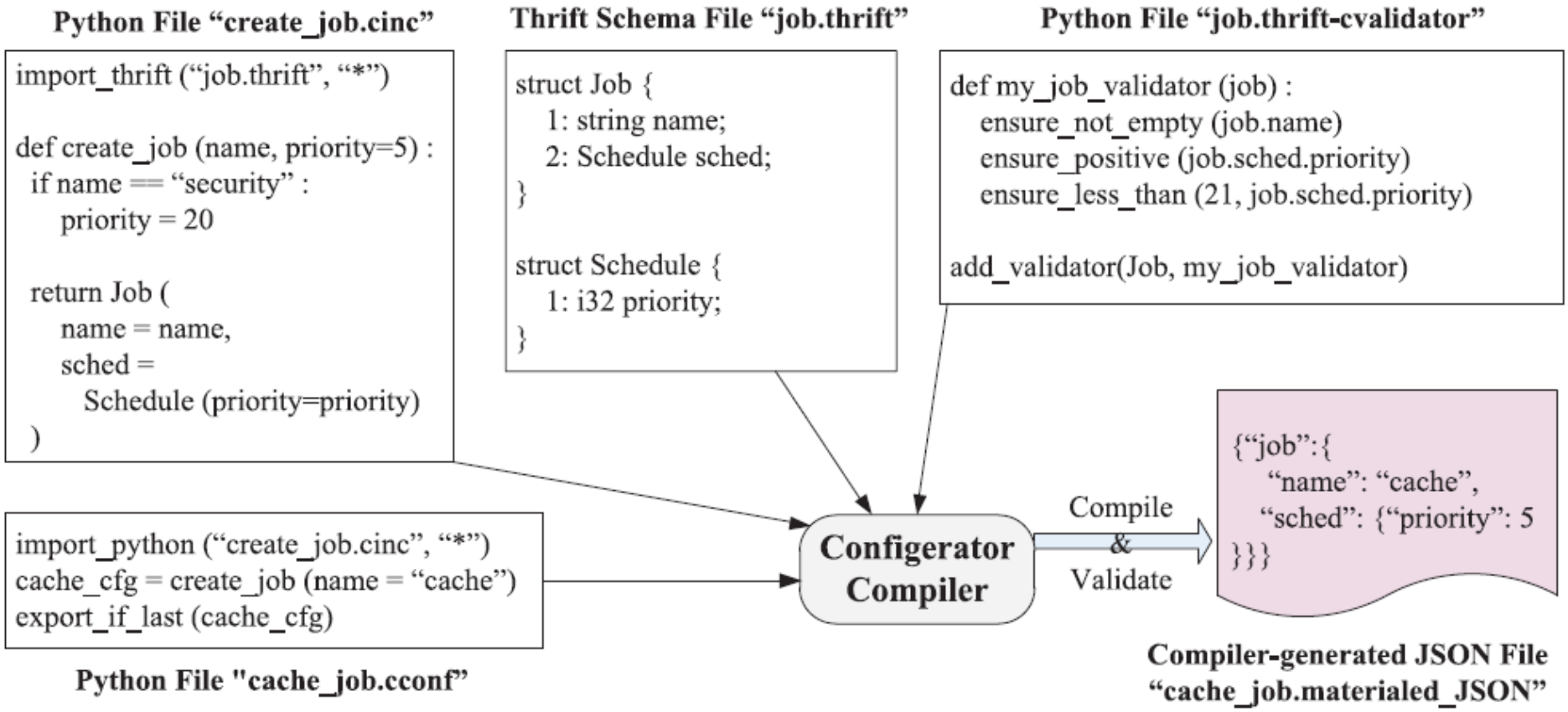
Tang, Chunqiang, et al. "Holistic configuration management at Facebook." Proceedings of the 25th Symposium on Operating Systems Principles. ACM, 2015.

Case Study: Facebook

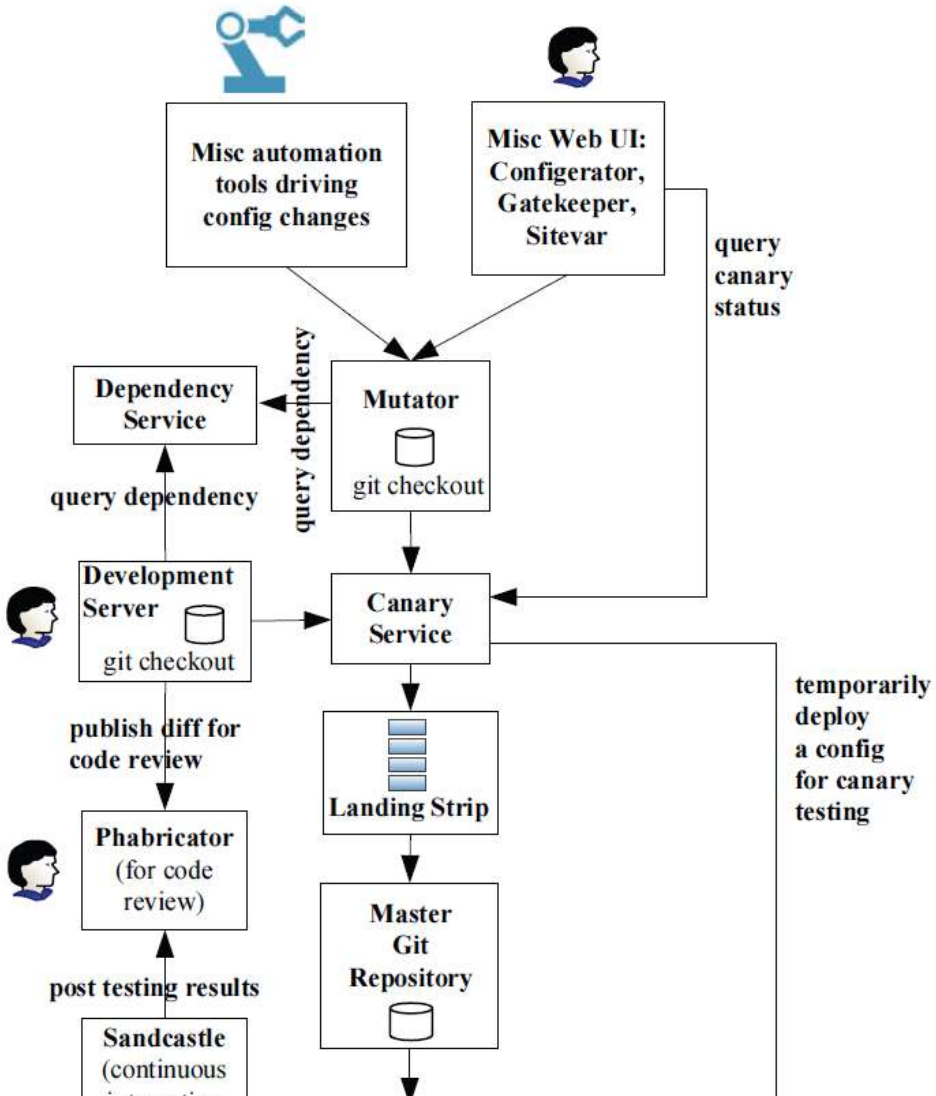
- Goals
  - Gating new product features, frequent and early releases (e.g. for 1% of users)
  - Conducting experiments
  - Traffic control and load balancing
  - Monitoring, alters, remediation

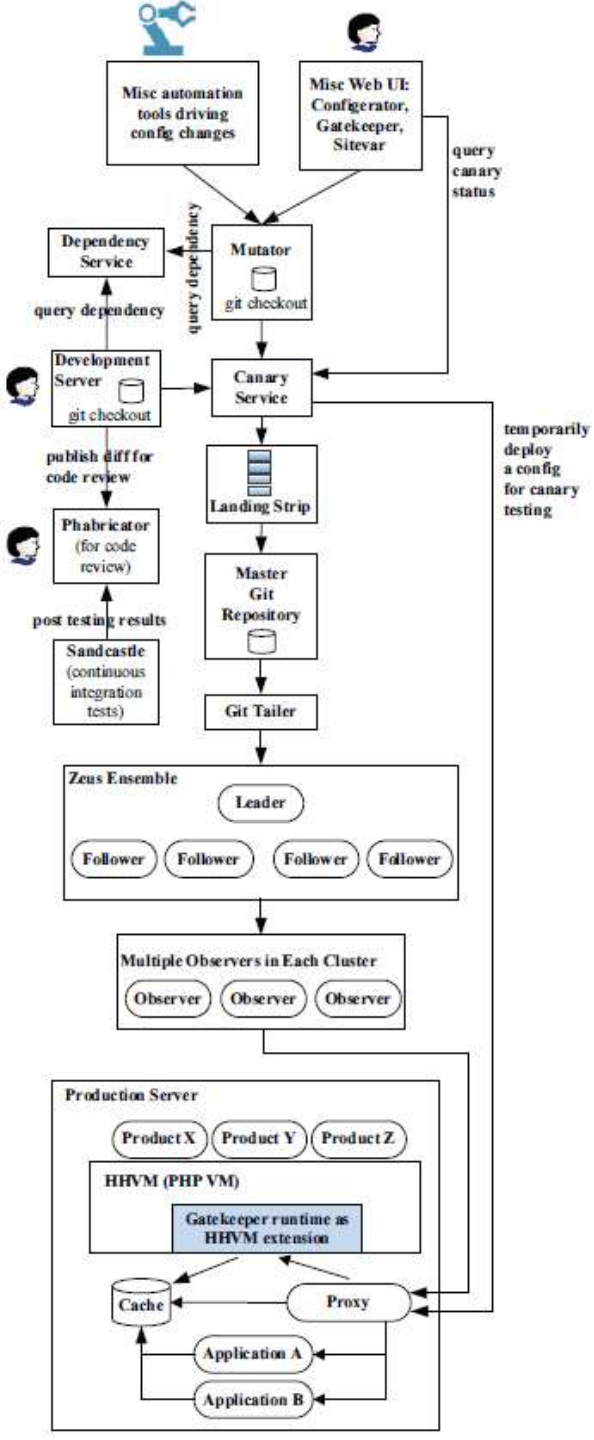
Tang, Chunqiang, et al. "Holistic configuration management at Facebook." Proceedings of the 25th Symposium on Operating Systems Principles. ACM, 2015.

Case Study: Facebook



Case Study: Facebook



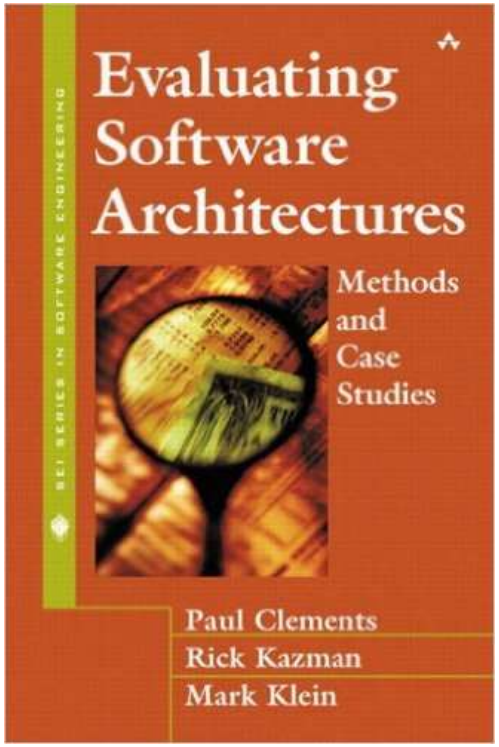


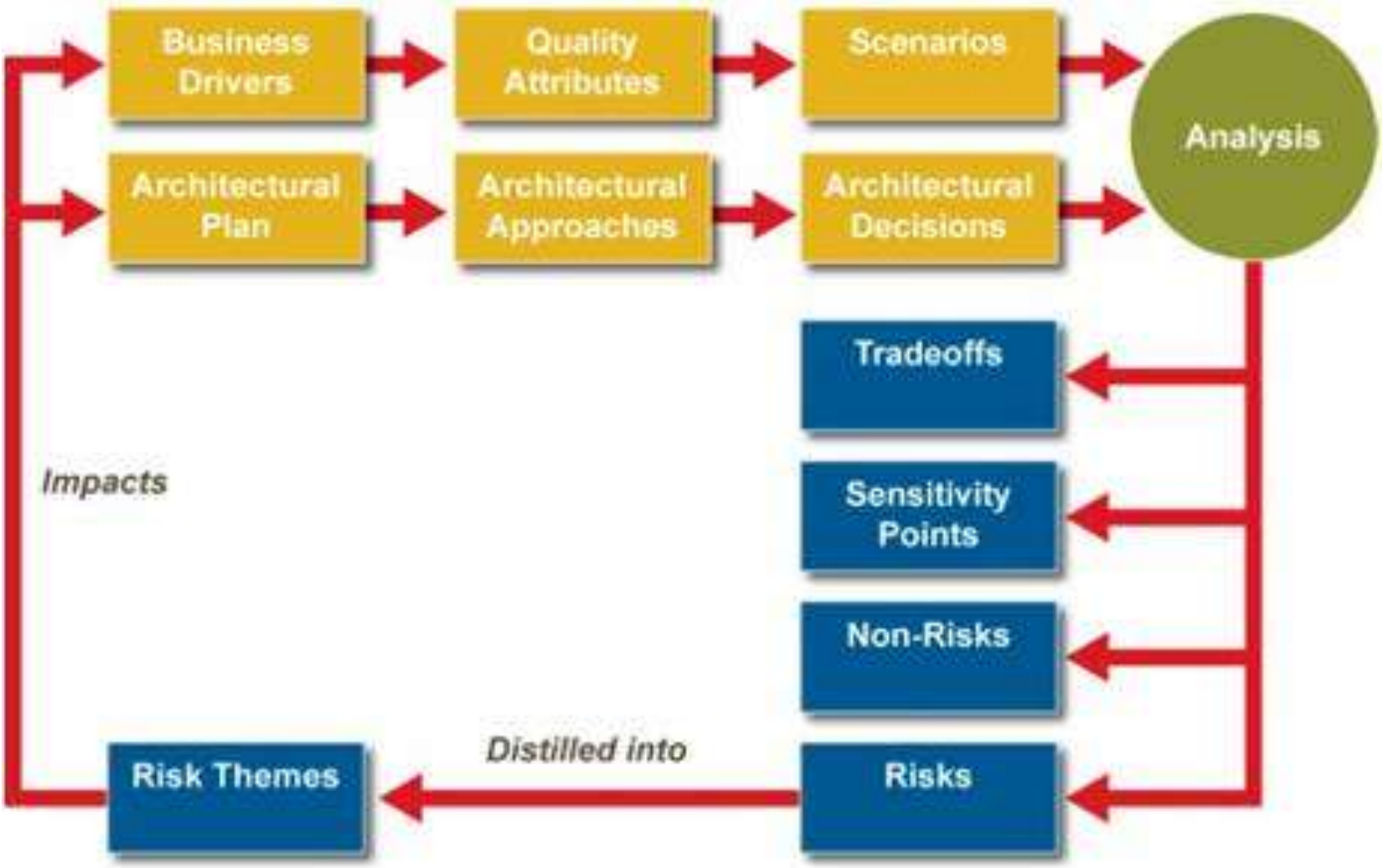
ring

Architecture Evaluation

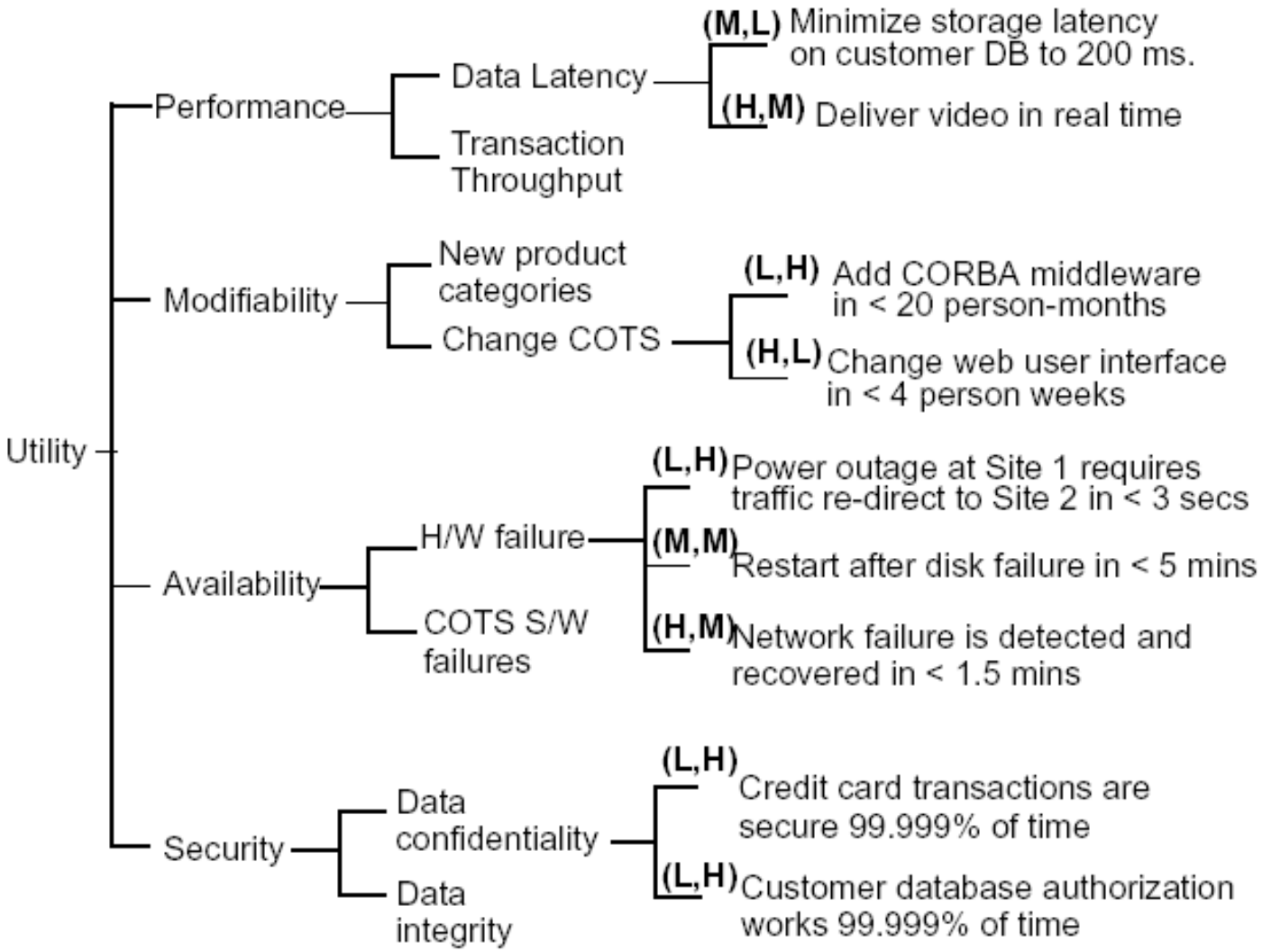
Architecture evaluation

- Goal: does the architecture satisfy requirements?
- ATAM – Architecture Tradeoff Analysis Method
  - Present requirements
  - Present architecture
  - Analyze architecture
  - Present results – risks and non-risks





Source:sei.cmu.edu



Source:arnon.me

Summary

- Address team issues early and explicitly
- Architecture helps with reasoning about qualities
- Architecture styles help with reasoning about tradeoffs and implications
- Microservices, DevOps and their advantages and problems
- Architecture evaluation is a thing