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]
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”

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
Where to validate user input?

Example: Yelp App
PeopleCars 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
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
• "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

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
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...

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

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

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

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

Customer Experience
Core Experience
Custom Experiences
Channels

Application Platform Services
Login Identity
Catalog
Search
List
Pricing
Offer
ADs
Messages
Cart
Coupons
Payment
Shipping
CS

Technology Platform
App Stack
Data Access Layer
Dev Tools
Presentation
Messaging
SOA
Cloud

Operations Infrastructure Layer
Power
Data Center
Hardware
Network
Database
Tools
Operations
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

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
monolith - single database

microservices - application databases

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
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 Logo](image-url)
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 xzf 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

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

Case Study: Facebook

Python File "create_job.cinc"

```python
import_thrift ("job.thrift", "*")

def create_job (name, priority=5):
    if name == "security":
        priority = 20
    return Job (name = name, sched = Schedule (priority=priority))

import_python ("create_job.cinc", "*")
cache_cfg = create_job (name = "cache")
export_if_last (cache_cfg)
```

Thrift Schema File "job.thrift"

```python
struct Job {
    1: string name;
    2: Schedule sched;
}

struct Schedule {
    1: i32 priority;
}
```

Python File "job.thrift-cvalidator"

```python
def my_job_validator (job):
    ensure_not_empty (job.name)
    ensure_positive (job.sched.priority)
    ensure_less_than (21, job.sched.priority)

add_validator(Job, my_job_validator)
```

Compiler-generated JSON File "cache_job.materialized_JSON"

```json
{"job": {
    "name": "cache",
    "sched": {"priority": 5
}}}
```
Case Study: Facebook
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
Utility tree

Performance
- Data Latency
  - (M,L) Minimize storage latency on customer DB to 200 ms.
  - (H,M) Deliver video in real time
- Transaction Throughput

Modifiability
- New product categories
  - (L,H) Add CORBA middleware in < 20 person-months
- Change COTS
  - (H,L) Change web user interface in < 4 person weeks

Utility

Availability
- H/W failure
  - (M,M) Restart after disk failure in < 5 mins
- COTS S/W failures
  - (H,M) Network failure is detected and recovered in < 1.5 mins

Security
- Data confidentiality
- Data integrity
  - (L,H) Credit card transactions are secure 99.999% of time
  - (L,H) Customer database authorization works 99.999% of time

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