Yegge on SOA
Tom Anderson and Doug Woos

Google (other large software companies) should use SOA as a software architecture and engineering discipline.

SOA at Amazon: Bezos's rules

All teams must expose data/functionality through service interfaces
Teams communicate through these interfaces
No other communication (e.g. direct linking, shared FS, etc.) allowed—only calls over network
Service interfaces must be externalizable—designed to be exposed to the outside world

SOA at Amazon: Implementation

Decompose website into 1000s of primitive services
Each team runs its service as a standalone product
- Including ops!
Each service provides a service level agreement to its clients (i.e. other teams’ services)

Service level agreements

Guarantee provided to clients re: service response time and availability
- ex: Availability = 5 9s (99.999% uptime)
- ex: Response time = 3ms @ 90th percentile
- SLA is also a guarantee from the client, e.g., won’t send more than X reqs/s

Meanwhile, at Google*

* then! maybe!

Fewer services
Culture encourages reuse via linking
- Monolithic codebase
- Libraries carefully maintained
Operations separate from development
Capacity centrally planned
- clients assumed to be well-behaved
Why SOA?

Internal reasons
- Resilient to buggy components
- Forces excellent monitoring
- Can scale services independently

Big external reason
- Companies need to build platforms
- Platforms require good external APIs
- Separate external/internal interfaces = bad APIs
- Need to eat your own dog food!

SOA lessons

Pager escalation
The core problem might not be the responsibility of the team whose on-call members get woken up in the middle of the night!

Need automated service registry
Every client is potential source of DoS
Including amplification attacks!

Only way to tell if a service is functioning is to use it
Testing = monitoring
Cross-service debugging—need universal sandbox

Why Yegge was worried

In order to be usable (accessible), applications need to be platforms

Must design for SOA from scratch
- Can’t bolt it on

What about upgrades?

SOA makes it harder to make backwards-incompatible changes
- Both an advantage and a disadvantage!
- At Google: monolithic codebase, change everyone’s API usage

Formalize API versioning, deprecation
- Some teams will upgrade early, others late

Discussion

In your experience, is SOA helpful?
Are there challenges in implementing SOA that Yegge didn’t address?

Next few papers

Facebook Memcache
Three real-world systems from Google
GFS: storage for bulk data
BigTable: storage for structured data
Chubby: coordination service
All four highly influential
GFS -> HDFS
BigTable -> HBase, Cassandra, other NoSQL stores
Chubby -> Zookeeper, etcd