

# Advanced Tools for Operators at Amazon.com

Peter Bodik, Armando Fox, Michael Jordan, David Patterson  
RAD Lab, UC Berkeley

Ajit Banerjee, Ramesh Jagannathan, Tina Su,  
Shivaraj Tenginakai, Ben Turner, Jon Ingalls  
Amazon.com

- Before: separate development/testing/operations teams, few operators
- Now: one team responsible for develop, test, deploy, operate
- Consequences for problem resolution
  - more dynamic, frequent updates
  - 100s of operators/resolvers
  - “every” developer is a resolver → high turnover in resolvers staff
- AC needs to concentrate more on operators
  - need to keep humans in the loop
    - explain the actions of the tool
  - build tools that understand how operators work
    - learn from the operators

- two-pizza teams
  - most of the software developed in-house
  - ~50 software teams
    - each responsible for a few services (e.g. shopping cart, recommendations, ...)
    - design, develop, test, deploy, and operate their service
  - networking, hardware, monitoring, operators
  - each team has a **primary-resolver** on-call 24x7
- dynamic environment
  - continuously adding new features
  - e.g. over 100 code pushes/month in Monitoring team
  - hundreds of documentation changes per month
- high turnover rate of developers/resolvers within Amazon

- Monitoring team
  - provide infrastructure for monitoring of SW/HW at Amazon
  - easy for anybody to instrument/monitor their SW/HW
  - provides visualization, analysis tools (web-based), alarms
  - provides API for accessing the data
    - other teams build their own visualization tools
- ~10 operators
  - monitor whole web, site don't fix problems, page resolvers
- ~1000 resolvers: all are SW developers
  - fix problems, expected to respond in 15 minutes

- both very important, but for different reasons
  - different challenges
- sev1
  - affect lot of customers, components
  - operators detect the problem (using a business metric)
  - resolvers from multiple teams involved
- sev2
  - affects just one component, handled within one team
    - easier to diagnose, fix
  - 100x more frequent
  - could turn into a sev1

- problems that affect customers
  - affect multiple components/services
  - not very frequent
  - often detected as decrease of traffic to certain URLs
- resolving sev1 problems
  1. operators notice the problem, initial troubleshooting
    - they don't try to fix the problem
  2. engage primary resolvers in multiple teams
    - expected to respond, join a con-call in 15 minutes
  3. later assign the problem to one team
    - sometimes misdiagnosed and bounced to a different team
    - could be escalated to higher management

## Why sev1 problems are hard

- uniqueness: problems don't recur
  - limited opportunity to learn from repeated incidents
- dependencies and propagation of failures
  - failure in one component affects many others
  - many components appear broken, but only one is
- situation awareness
  - too many dependencies; hard to remember
  - operators/resolvers don't see the “big picture”
- too much information
  - thousands of metrics for each component, many active alarms
  - hard to analyze so much data
    - which of the metrics are actually useful?

- interactive visualization
  - components, their health, dependencies (logical & hardware)
  - zoom in to see datacenters, racks, machines, load balancers
- dependencies
  - most detected automatically
  - let resolvers add/remove dependencies
- health of components & dashboards
  - dashboards built like a wiki
  - resolvers from the same team collaborate
  - anybody can add/remove metrics, alarms, notes, links to documentation, ...



Search for: [All](#) [Hosts](#) [Hostclasses](#) [Environments](#) [Services \(beta\)](#)

Search

Dependency Graph Beta:

[FAQ](#) - [Feedback](#)

[My Monitoring](#)

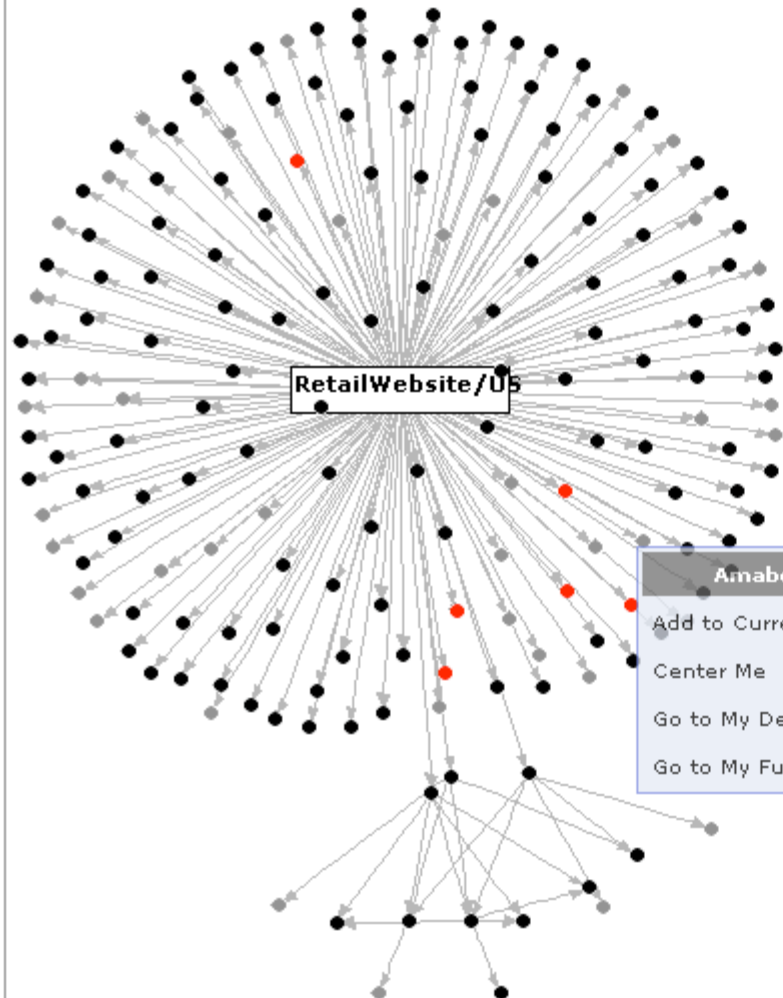
### RetailWebsite/US Dependency Map

View: **Logical** | Physical

[Show Graph Options](#)

Find:

Page Type: ALL



**AnabotServer**

- Add to Current Dashboard
- Center Me
- Go to My Deployments
- Go to My Full Dashboard

Color Key: Black: healthy - Red: unhealthy - Gray: unknown - Orange border: highlighted

### Current Dashboard

#### RetailWebsite/US

##### Health

+ [Order rate:](#)



[Average page latency:](#)



[Page views:](#)



[Change Health Metrics](#)

[More Metrics](#)

[Host Alarms](#)

[Info](#)



Cron Daemon

Cron <perfuser@integ-node-03482> \$ENVIRONMENT\_...  
Could not create directory  
/apollo/env/PerformanceMonitor/PerformanceHome/perfus...

- sev2 problems
  - don't directly affect the customers
    - can turn into sev1s
  - affect a single component
    - handled by resolvers (operators not involved)
  - 100x more frequent than sev1 problems
- lifecycle of sev2 problems
  - develop new features, test, deploy → bugs cause sev2 problems
  - **short-term solution:** restart process, reboot machine, ...
  - **long-term solution:** fix the bug
  - recur frequently
    - set up an automatic alarm (later detected automatically)
- “notes for the primary”
  - how to diagnose and fix the most common problems

- “notes for the primary”: obsolete very quickly, need to be updated
  - new bugs cause new types of problems
- not everything is in the docs
  - primary resolver sometimes can't fix the problem
  - but another resolver might have experienced it before
  - no systematic way to share, store such information
- new resolvers
  - shadow the primary resolver for a few weeks
  - need to ask other colleagues, search through emails

- database of past problems
  - for a new problem, suggest actions that would help
  - populate database by monitoring operators
- detect similar problems
  - Ira Cohen, *et. al.*, Capturing, indexing, clustering and retrieving system history, SOSP 2005
- monitor actions of resolvers
  - who worked on the problem
  - when they worked on the problem
    - resolvers multitask
  - what actions they performed
    - web-based tools: access logs
    - command-line: sudo logs, history

- new way of developing and running Internet services
  - same team builds and operates the web site
- different categories of problems
  - different consequences to site
  - pose different types of challenges
  - require different types of tool support
- tools that understand operators
  - let operators add content
    - dependencies, important metrics, links to documentation
  - monitor operators, learn from their actions

1. humans will always be part of managing complex systems
2. need to learn from operators
  - spend time with them
  - learn from them
3. new way of building and operating Internet services
  - one team responsible for build, test, deploy, and operate
  - rapid development of Internet applications