

---

# Best Practices in Web Performance Monitoring

Alistair A. Croll

VP Products and Chief Strategy Officer  
Coradiant, inc.

**CORADIANT**<sup>TM</sup>



---

So you want to  
monitor things.

---

But there are too  
many toys out there...

# A top-down approach to web performance monitoring

---

**Business goals**

**Operating processes**

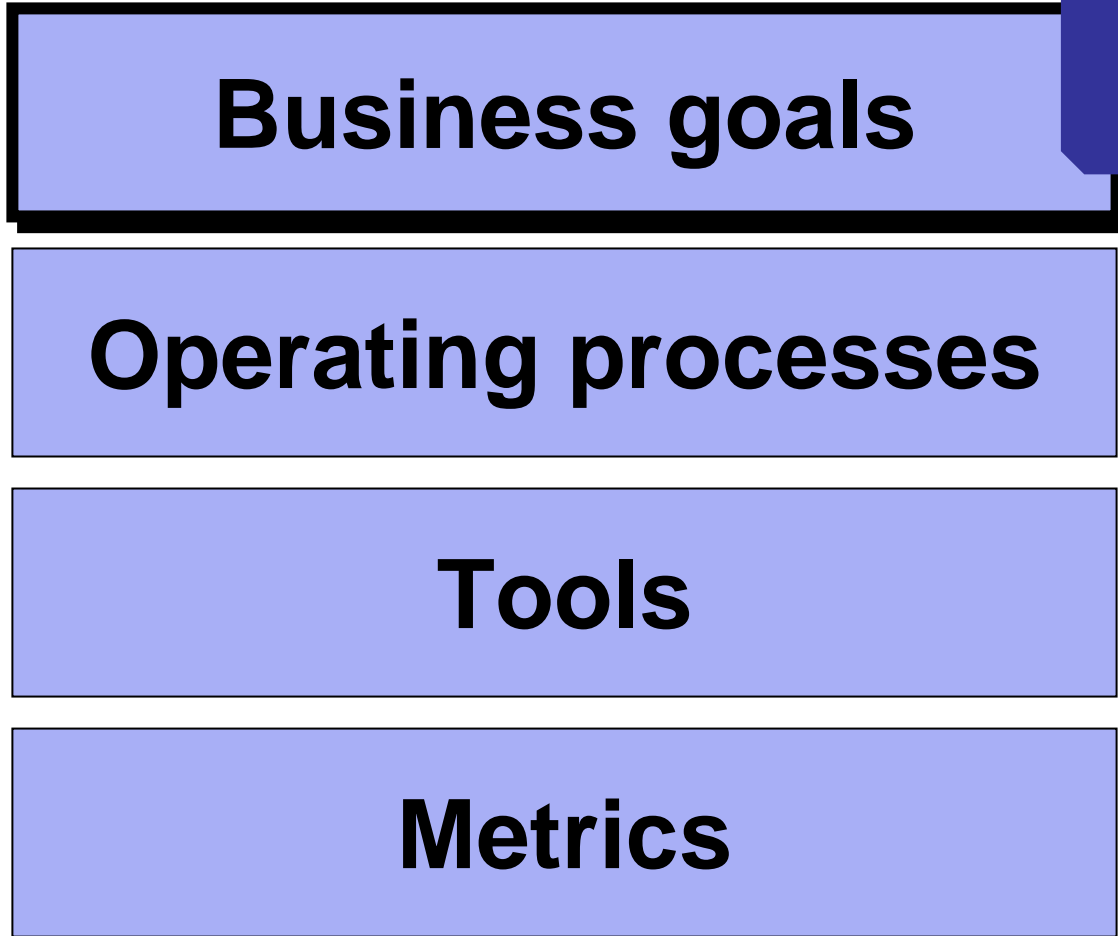
**Tools**

**Metrics**

# A top-down approach to web performance monitoring

---

↑  
Simplify & interpret



Start  
here!

---

What goals?  
(in plain English)

# Goals

---

- Make the application available
  - *I can use it*
- Ensure user satisfaction
  - *It's fast & meets or exceeds my expectations*
- Balance capacity with demand
  - *It handles the peak loads*
  - *It doesn't cost too much*
- Minimize MTTR
  - *When it breaks, I can fix it efficiently*
- Align operations tasks with business priorities
  - *I work on what matters first*

---

They can use it

# Make the application available

---

- The most basic goal
- App should be reachable, responsive, and functionally correct
- 3 completely different issues
  - Can I communicate with the service?
  - Can I get end-to-end responses in a timely manner?
  - Is the application behaving properly?

---

They're happy &  
productive

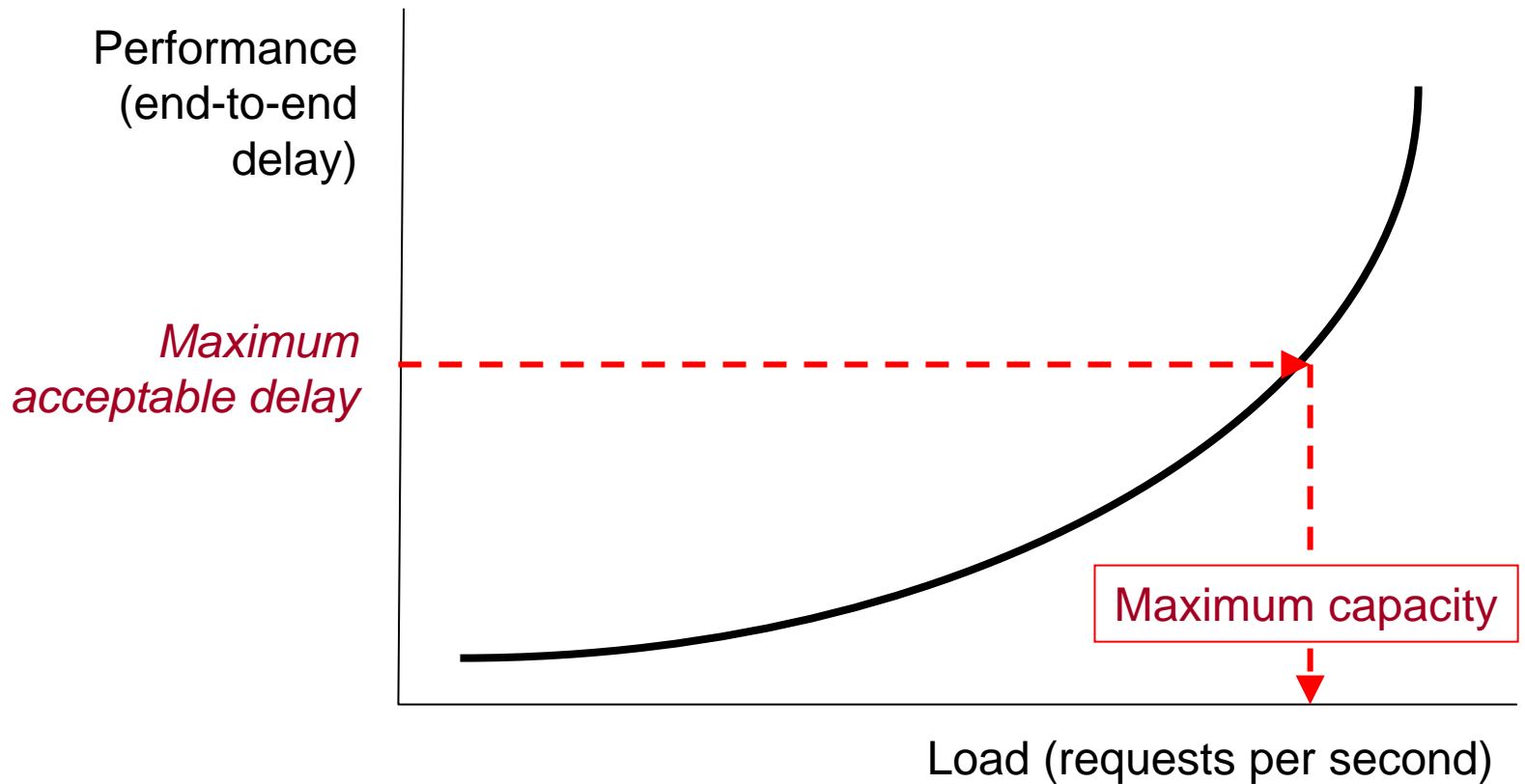
# Ensure user satisfaction

---

- How fast is fast enough?
- Depends on the task
  - Login versus reports
- Depends on user expectations
  - ATMs versus banking systems
- Depends on the user's state of mind
  - Deeply engaged versus browsing

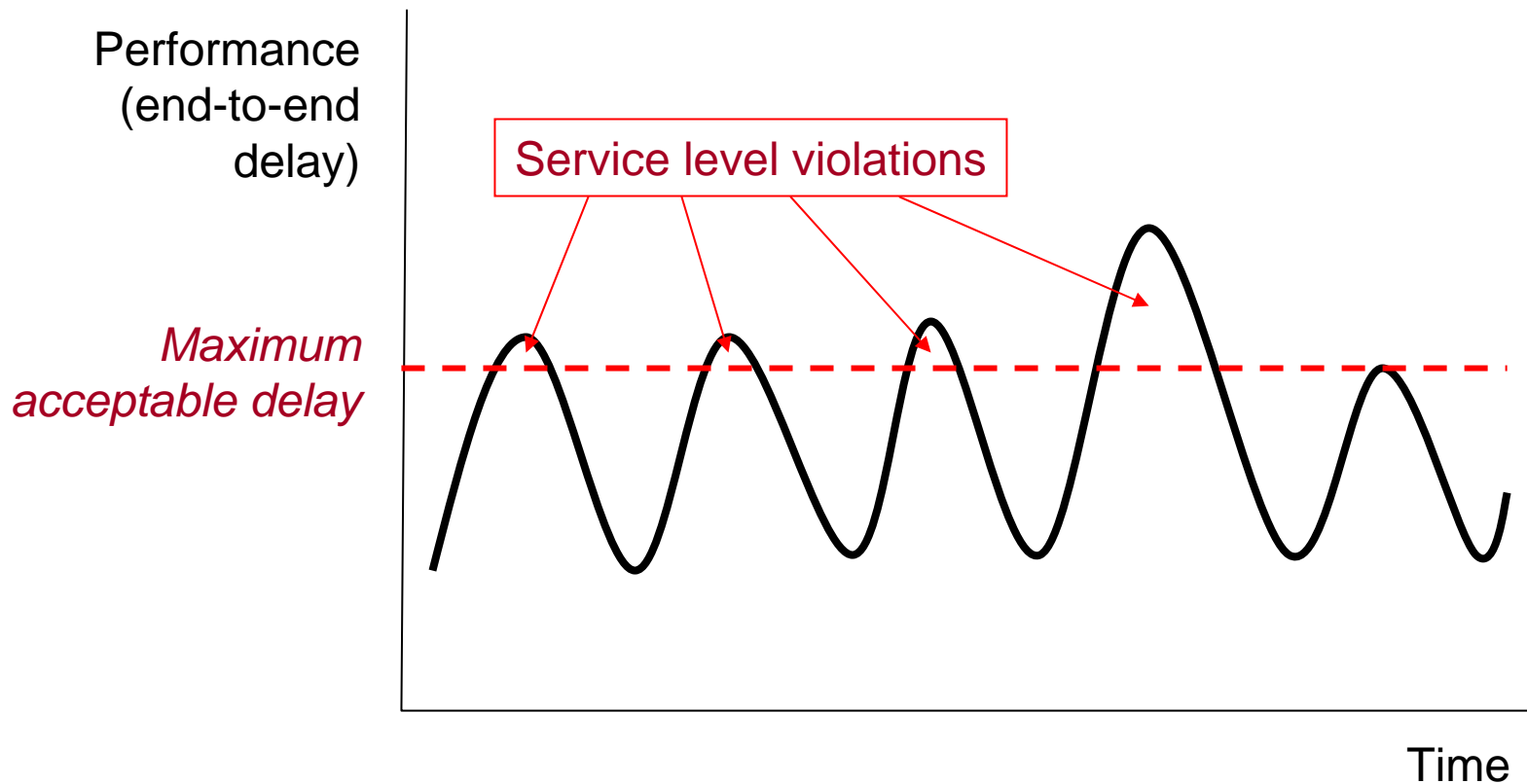
# Balance capacity with demand

- Performance degrades with demand



# Balance capacity with demand

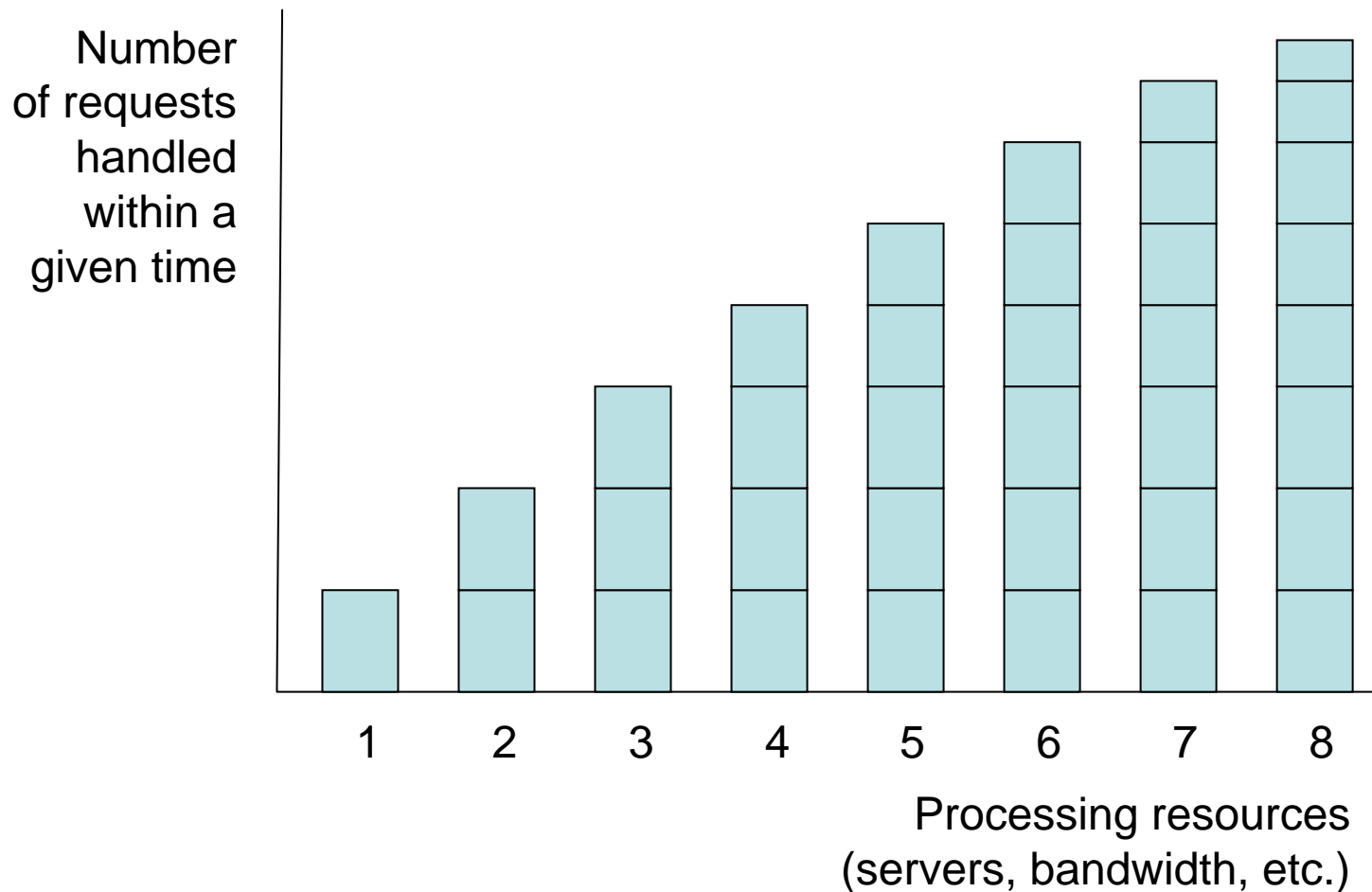
- And varies throughout the day



# Balance capacity with demand

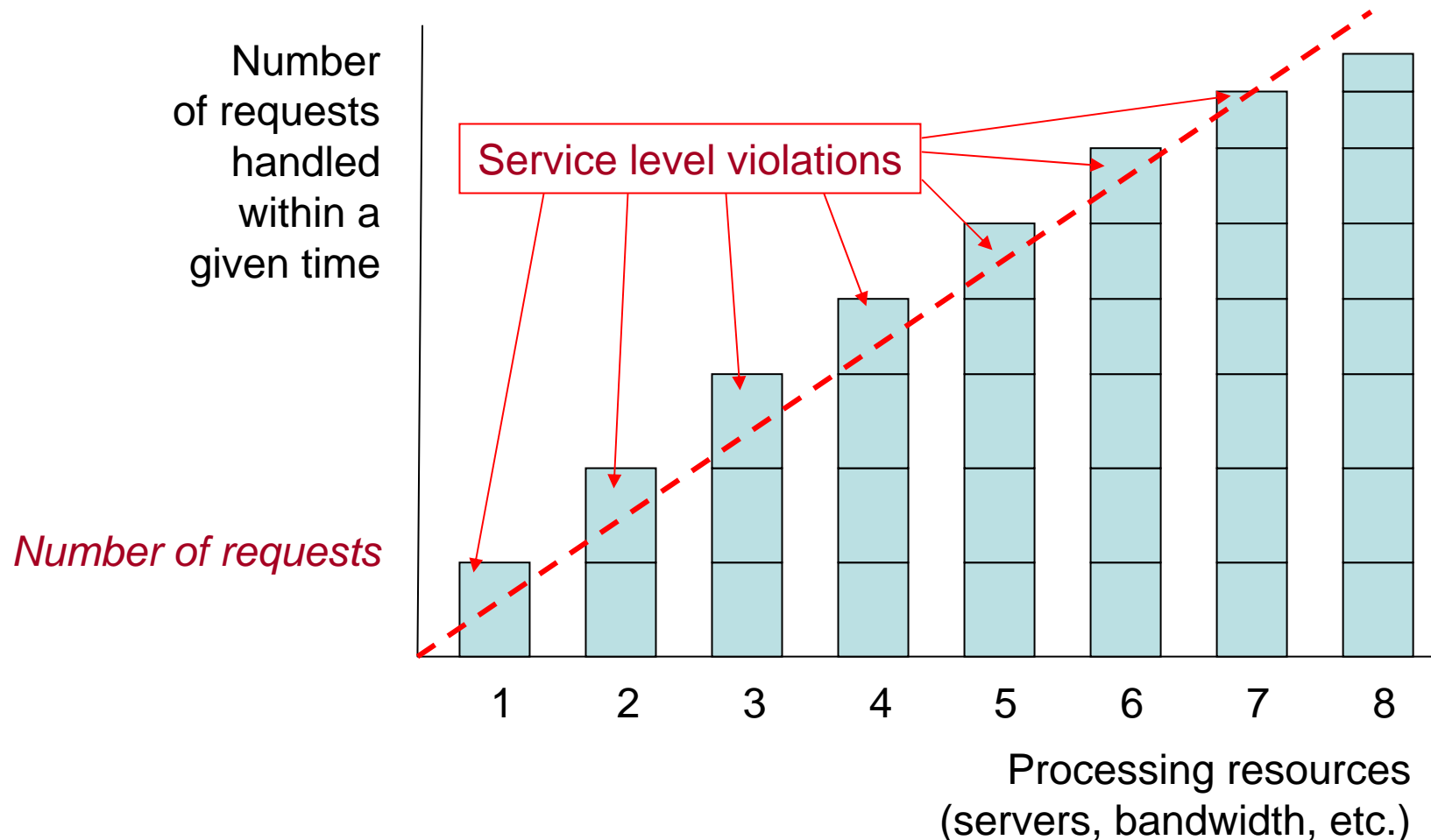
---

- But capacity gets added in chunks



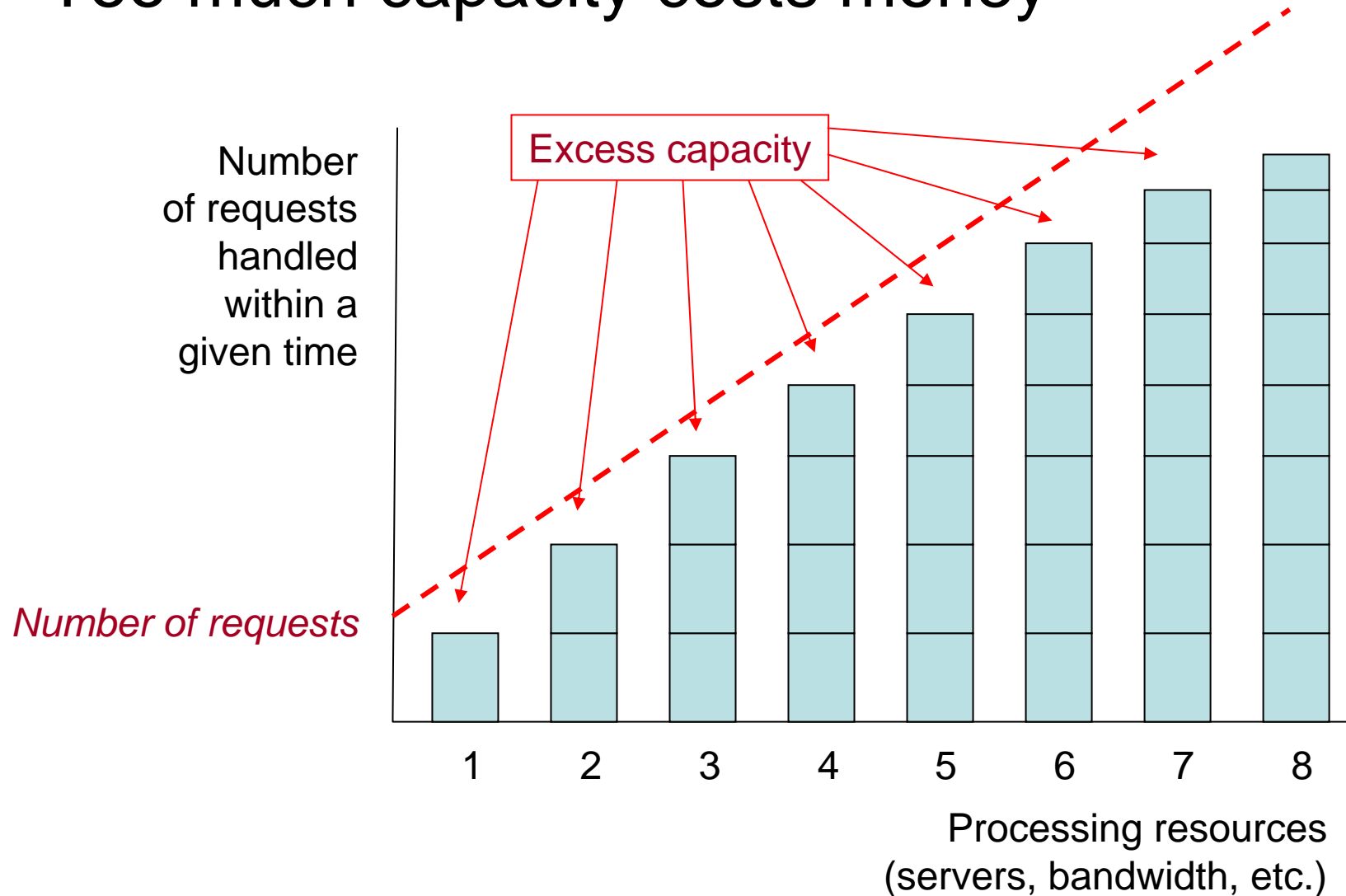
# Balance capacity with demand

- And users load a system gradually



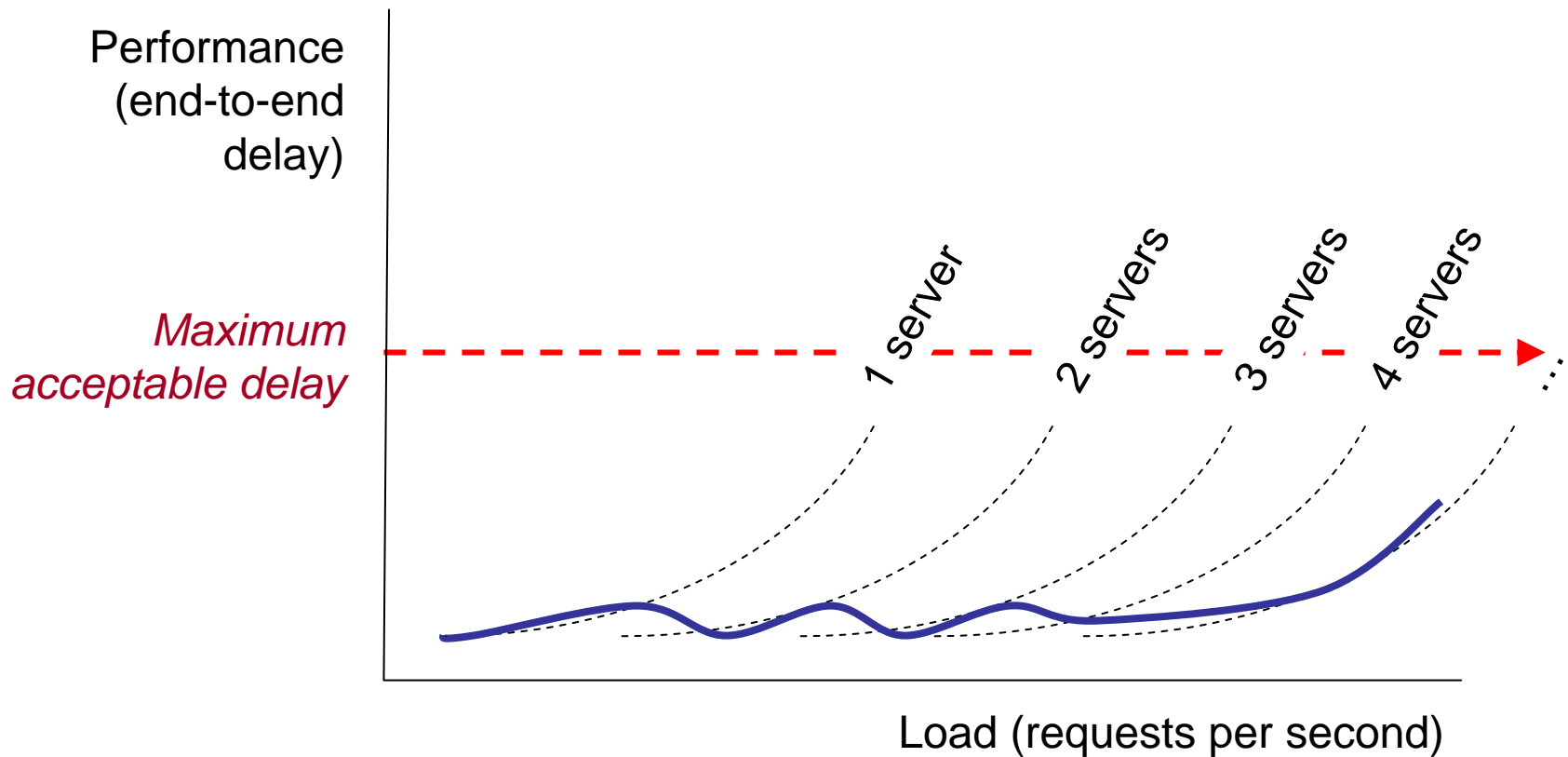
# Balance capacity with demand

- Too much capacity costs money



# Balance capacity with demand

- So planning means “fitting the curve”



---

I can fix it fast



# Minimize MTTR

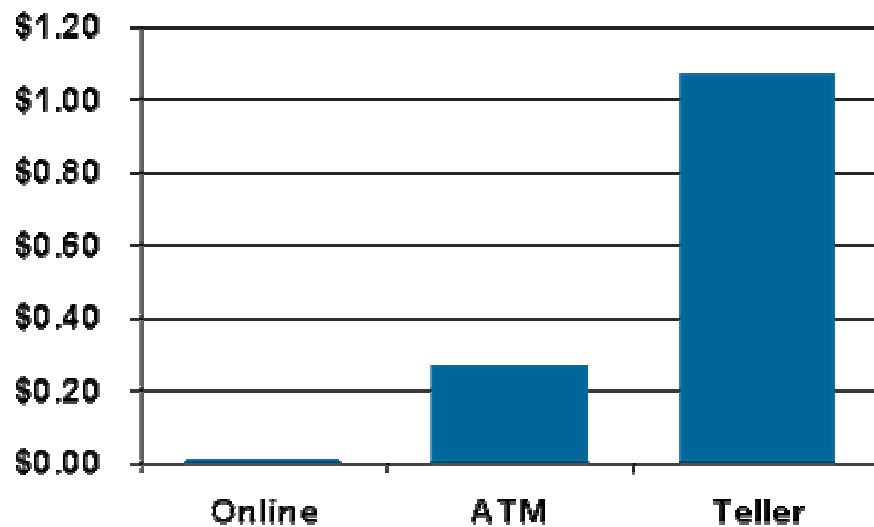
---

- Fix it efficiently
- Know the costs of downtime
- Application- and business-dependent
  - Direct (operational) costs
  - Penalties
  - Opportunity costs
  - Abandonment costs

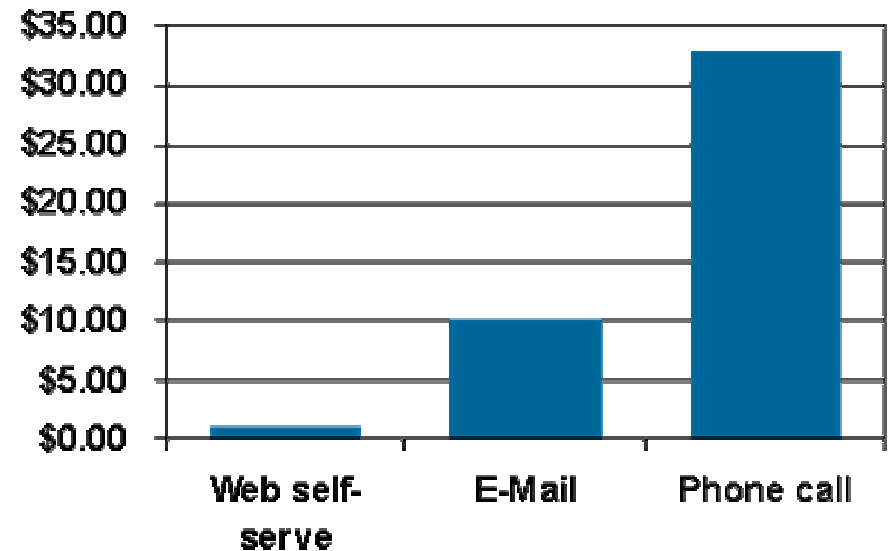
# Minimize MTTR

- Don't just think about lost revenue

**Relative cost of banking transaction**



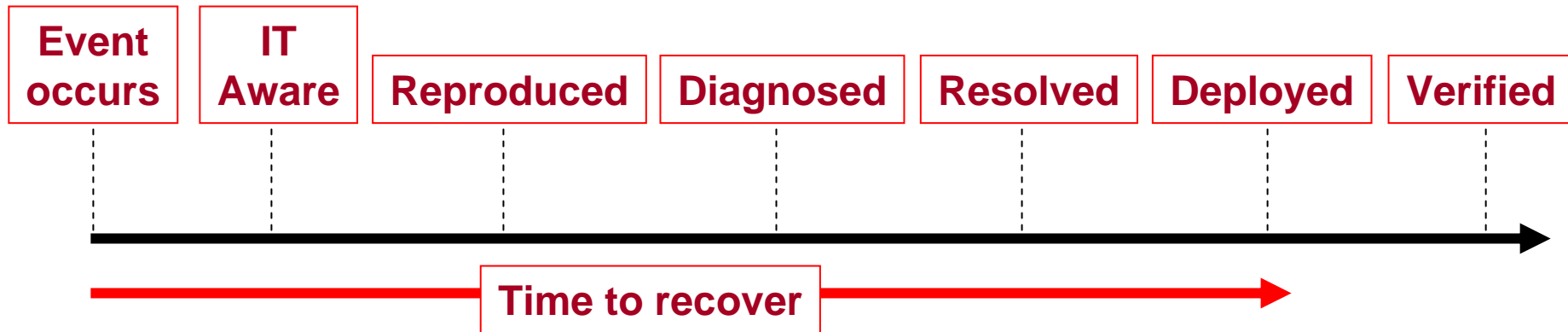
**Relative cost of customer support**



# Minimize MTTR

---

- And consider the whole resolution cycle



---

I worry about what  
matters

# Align operations tasks with business priorities

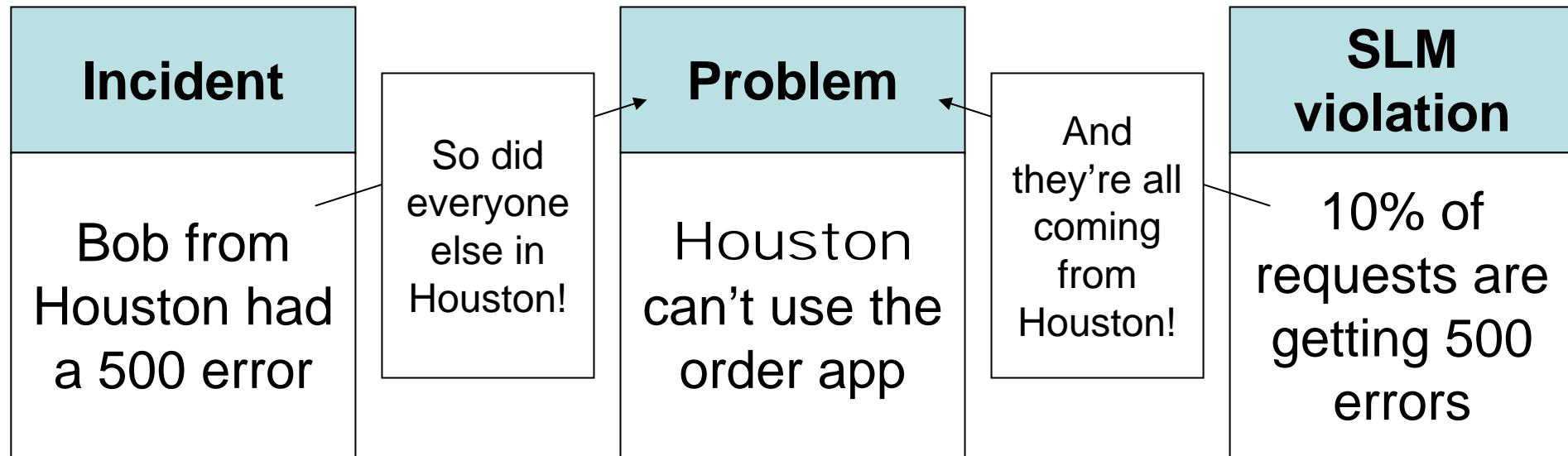
---

- Know what the business goals are
- Fix problems, not incidents
- Know the *real* impact of an issue

# Align operations tasks with business priorities

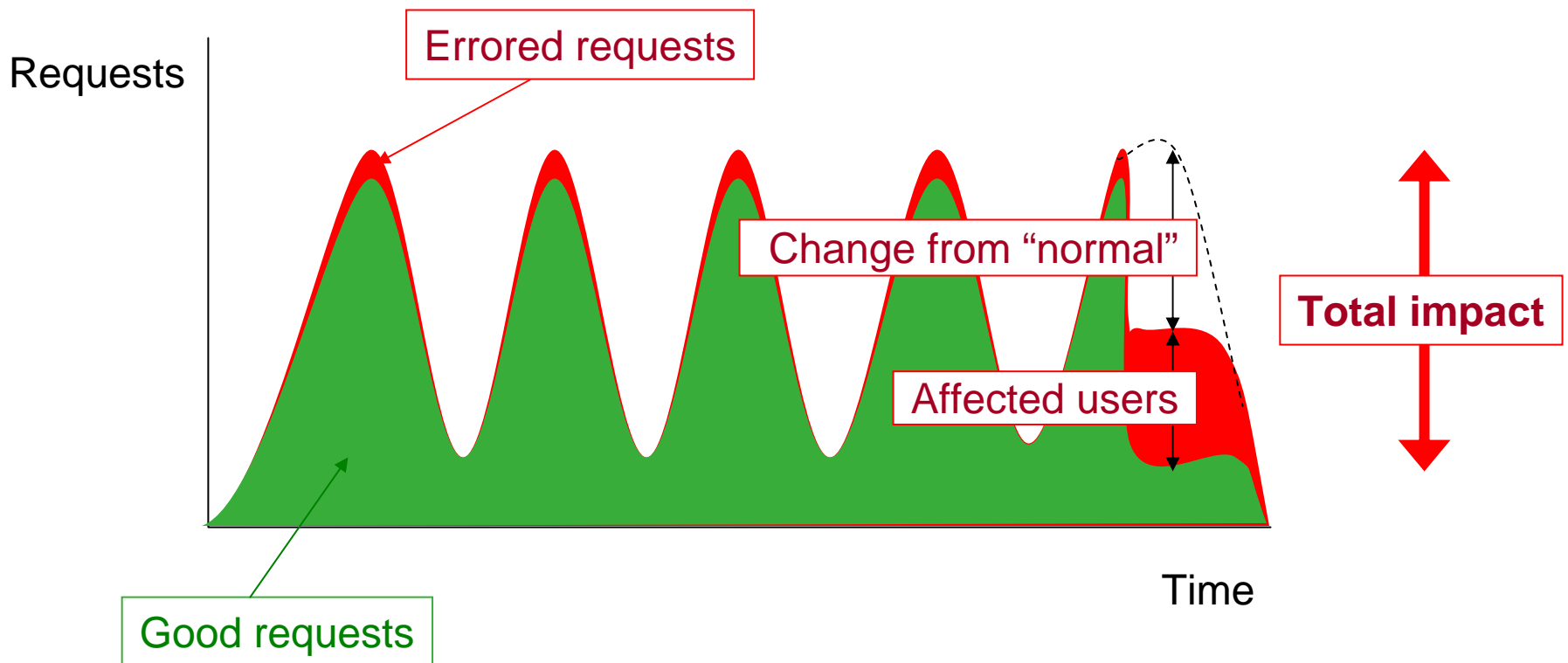
---

- Tackle problems, not incidents



# Align operations tasks with business priorities

- Know the *real* impact of issues



# So I have these goals...

---

- Make the application available
- Ensure user satisfaction
- Balance capacity with demand
- Minimize MTTR
- Align operations tasks with business priorities
  
- *How do I make sure I meet them repeatably and predictably?*

---

Okay, got the goals

---

But how do I make  
this real?

# A top-down approach to web performance monitoring

---

**Business goals**

**Operating processes**

Goals drive processes

**Tools**

**Metrics**

# Processes

---

- Reporting & overcommunication
- Capacity planning
- SLA definition
- Problem detection
- Problem localization & resolution

---

# Keep people informed



# Reporting & overcommunication:

## *Know the audience*

---

### **Network operations**

Network latency, throughput, retransmissions, service outages

### **Marketing**

Abandonment, conversion, demographics

### **Server operations**

Host latency, server errors, session concurrency

### **Security**

Anomalies, fraudulent activity

### **Finance**

Capacity planning, time out of SLA, IT repair costs

---

### **Different stakeholders**

The same data sources

---

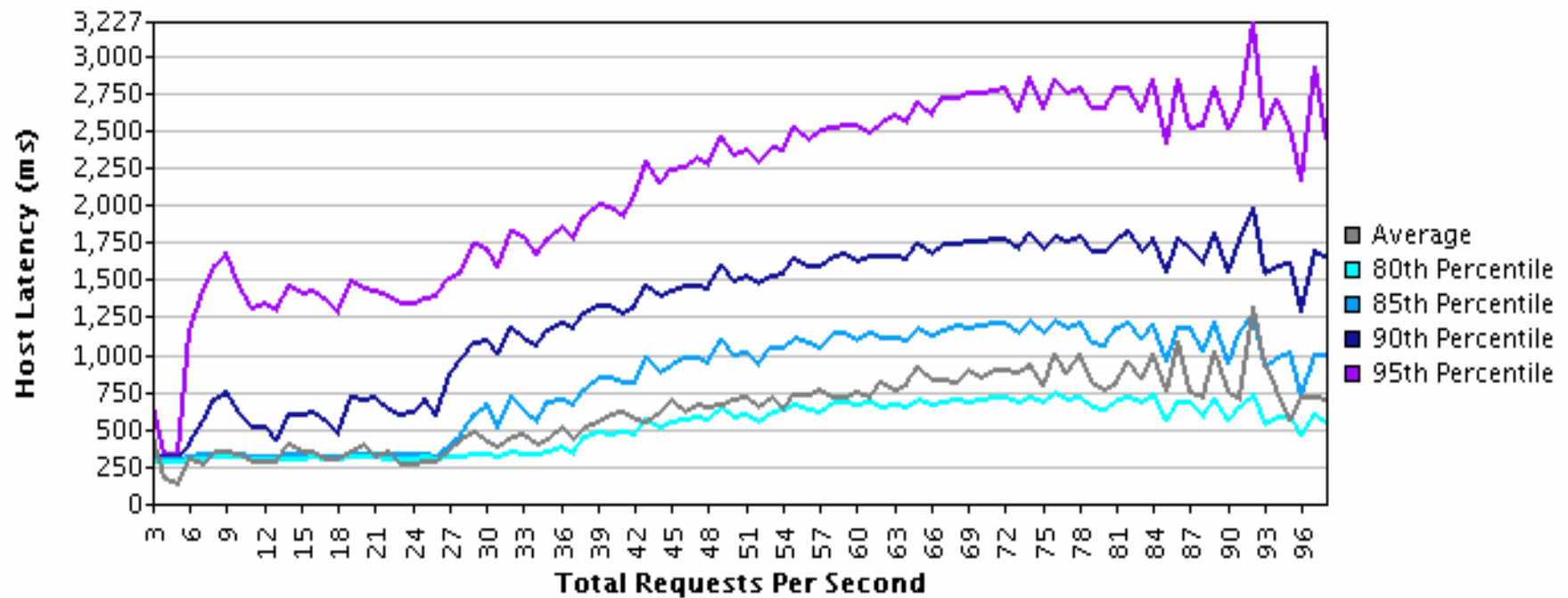
I have enough juice

# Capacity planning

---

- Define peak load
- Define acceptable performance & availability
- Select margin of error
  - Cost of being wrong
  - Variance and confidence in the data
- Build capacity & monitor
  - Performance versus load

# Capacity planning



---

We all agree on what's  
"good enough"

# SLA definition

---

- Select a metric
- Select an SLA target
  - That you control
  - That can be reliably measured
- Define how many transactions can exceed this target before being in violation
- Monitor
  - Metric, percentile

# SLA definition

---

- 95% of all searches by zipcode by all HR personnel will take under 2 seconds for the network to deliver

95%

**Percentiles, not averages**

All searches by zipcode

**Application function, not port**

All HR personnel

**User-centric, actual requests**

Under 2 seconds

**Performance metric**

For the network to deliver

**A specific element of delay**

---

I know where  
problems are...

# Problem detection

---

- Detect incidents as soon as they affect even one user
- Is the incident part of a bigger problem?
- Prioritize problems by business impact
  - Number of users affected
  - Dollar value lost
  - Severity of the issue

---

...and I can figure out  
what's behind them

# Problem localization & resolution

---

- Reproduction of the error
  - Capture a sample incident
- Deductive reasoning
  - Check tests to see what else is failing
  - Do incidents share a common element?
  - Do incidents happen at a certain load?
  - Do incidents recur around a certain time?

# Problem localization & resolution

---

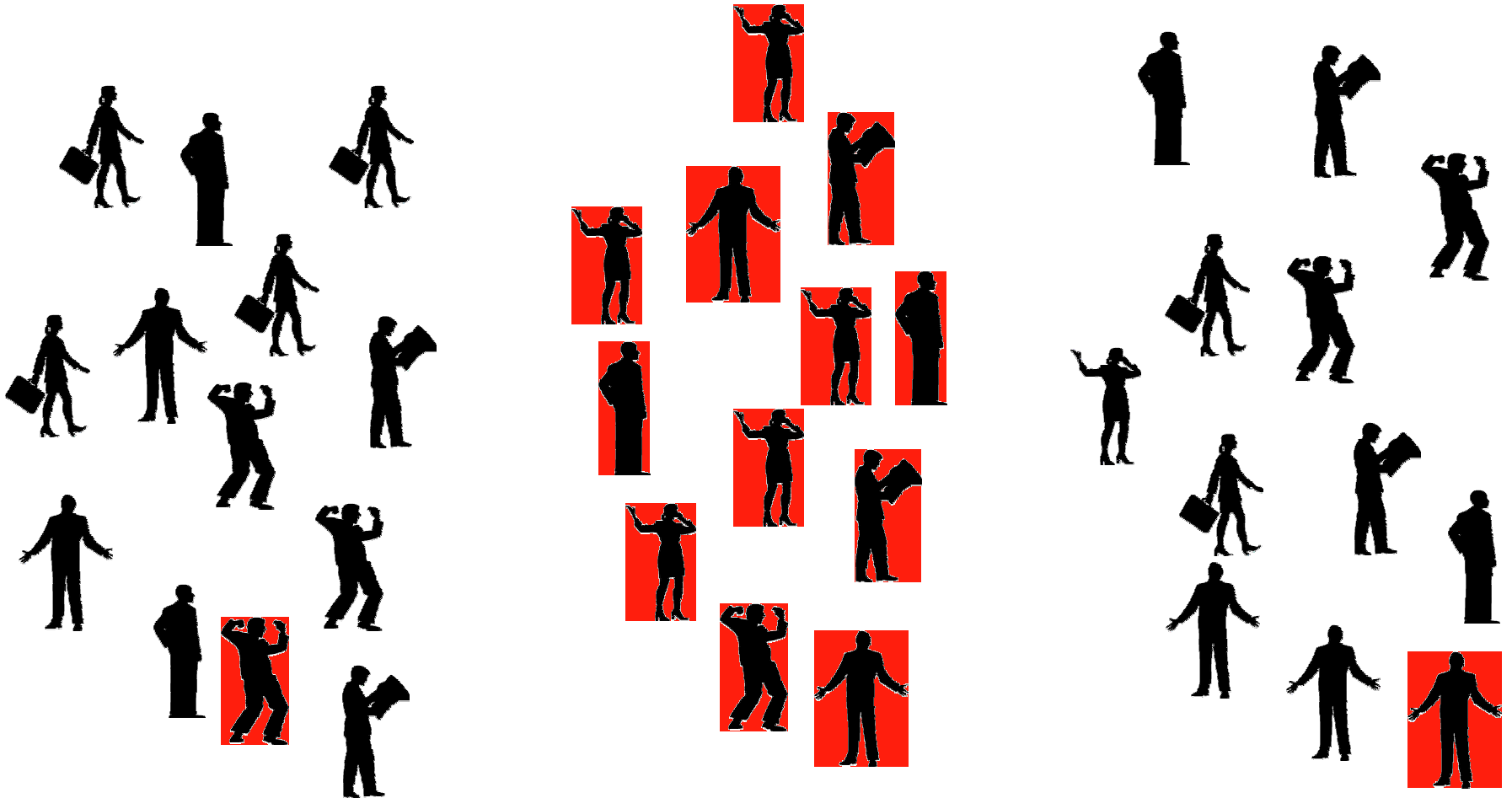


# Problem localization & resolution

- What do they have in common?

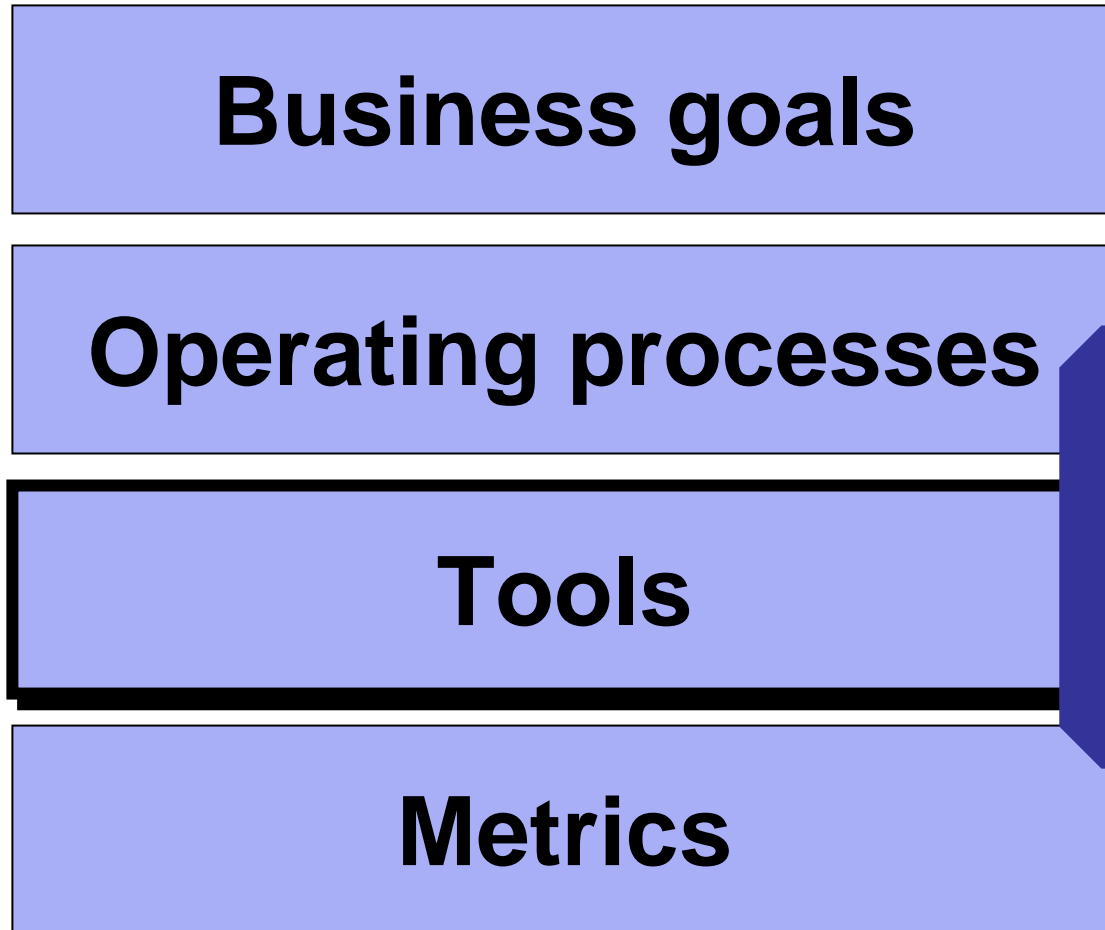


# Problem localization & resolution



# A top-down approach to web performance monitoring

---



Select tools that make processes work best

---

...and I can figure out  
what's behind them

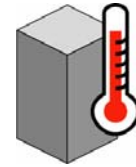
# Tools:

## *The three-legged stool*

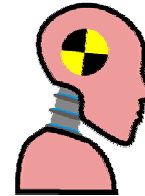
---



Device



Synthetic



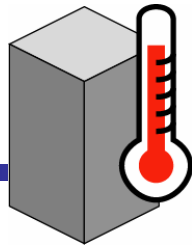
Real User



# Device monitoring:

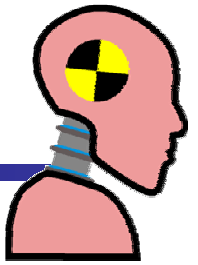
## *Watching the infrastructure*

---



- Less relation to application availability
- Vital for troubleshooting and localization
- Will show “hard down” errors
  - But good sites are redundant anyway
- Correlation between a metric (CPU, RAM) and performance degradation shows where to add capacity

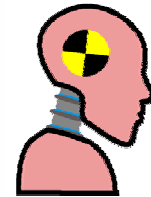
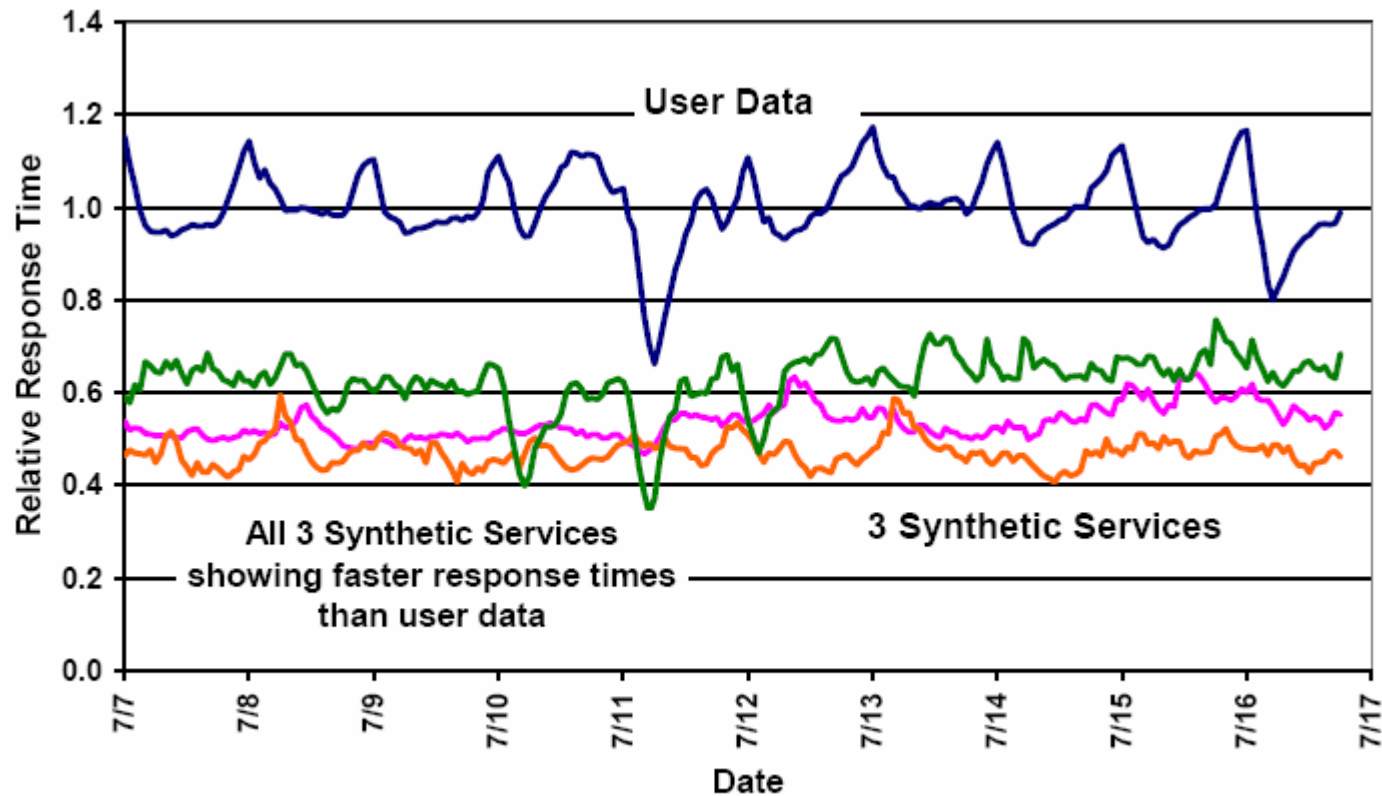
# Synthetic testing: *Checking it yourself*



- Local or outside
- Same test each time
- Excellent for network baselining when you can't control end-user's connection
- Use to check if a region or function is down for everyone
- Limited usefulness for problem re-creation



# Synthetic testing: *Checking it yourself*



# Real User Monitoring:

## *2 main uses*

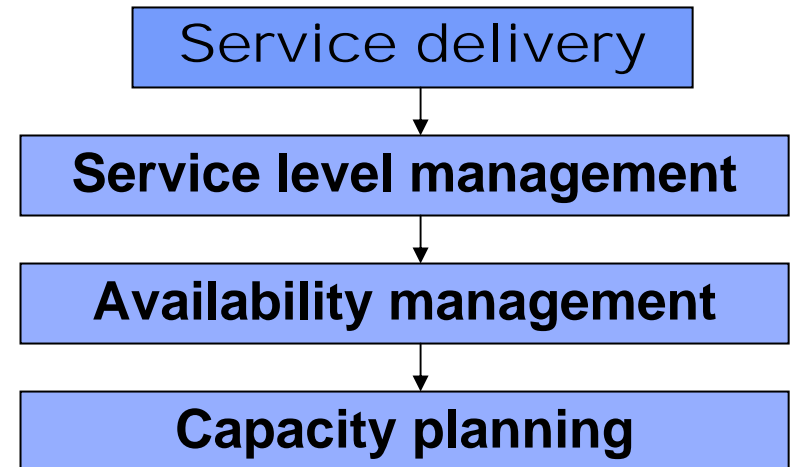
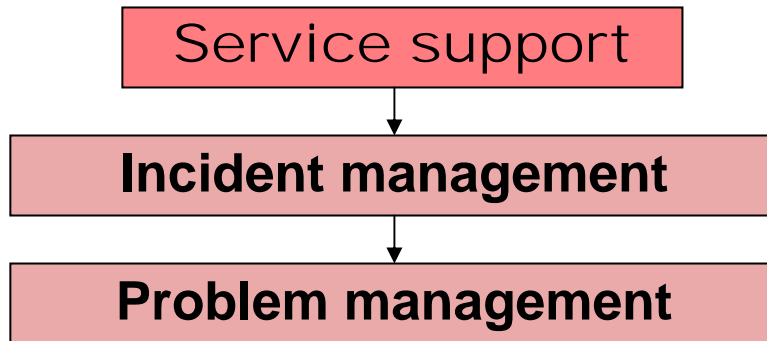


- Tactical
  - Detect an incident as soon as 1 user gets it
  - Capture session forensics
- Long-term
  - Actual user service delivery
  - Performance/load relations
  - Capacity planning

# Real user monitoring: *2 main uses*



- Outlined in ITIL



---

OK, I've got the tools.  
What do I look at?

# A top-down approach to web performance monitoring

---

**Business goals**

**Operating processes**

**Tools**

**Metrics**

Use the right  
metrics for  
the audience  
& question

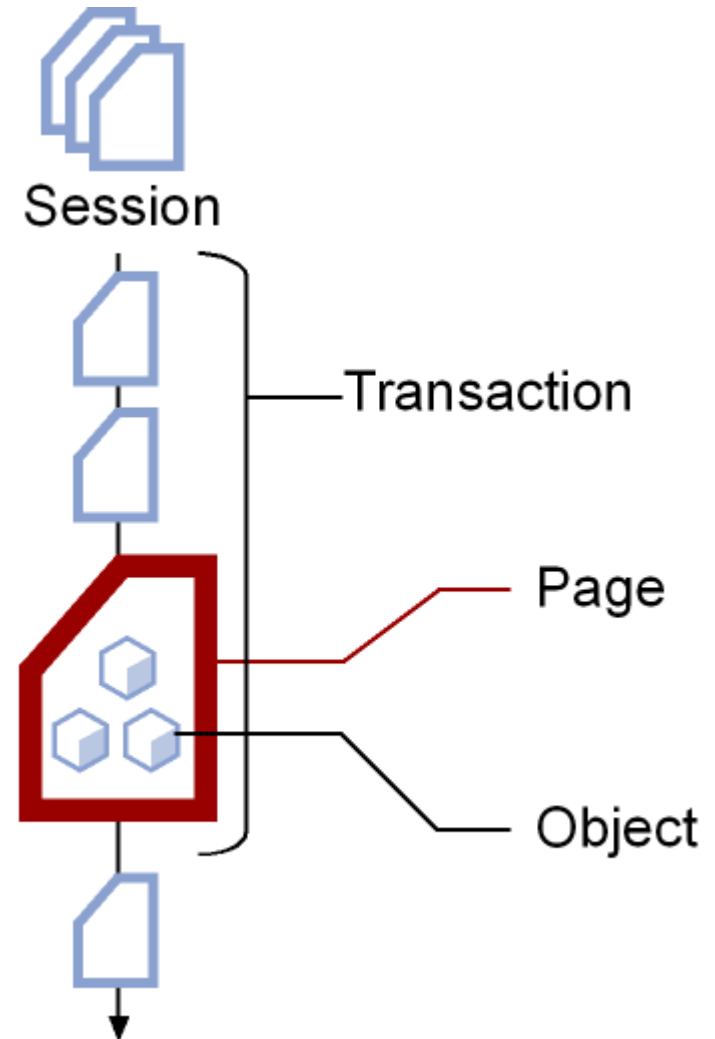
# Metrics

---

- Measure everything
  - A full performance model
- Availability
  - Can I use it?
- User satisfaction
  - What's the impact of bad performance?
- Use percentiles
  - Averages lie

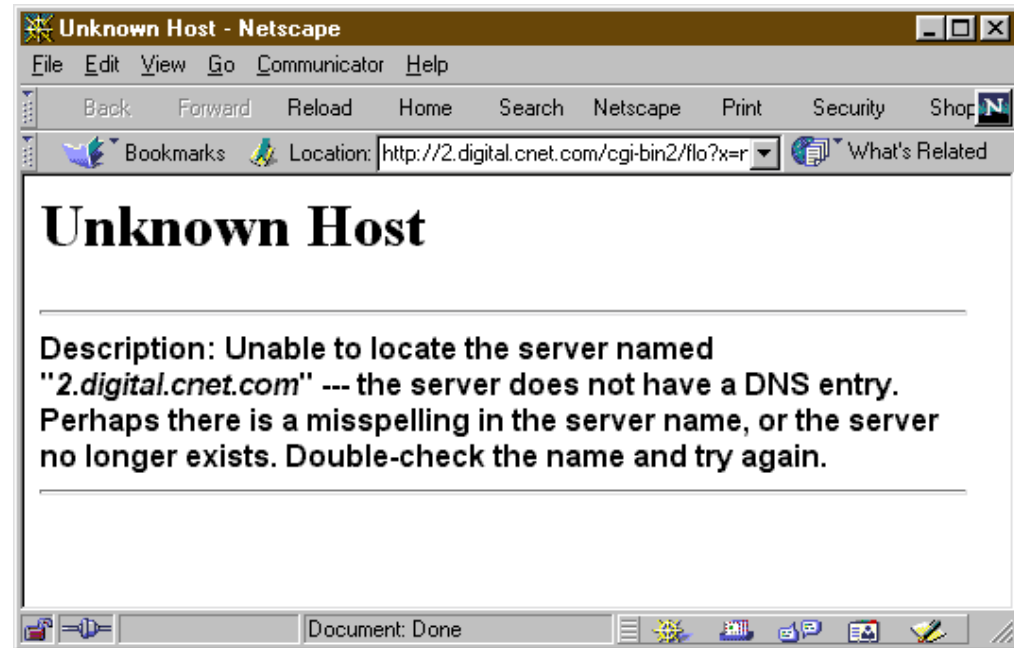
# A full performance model

- The HTTP data model
  - Redirects
  - Containers
  - Components
  - User sessions
- HTTP-specific latency
  - SSL
  - Redirect time
  - Host latency
  - Network latency
  - Idle time
  - Think time



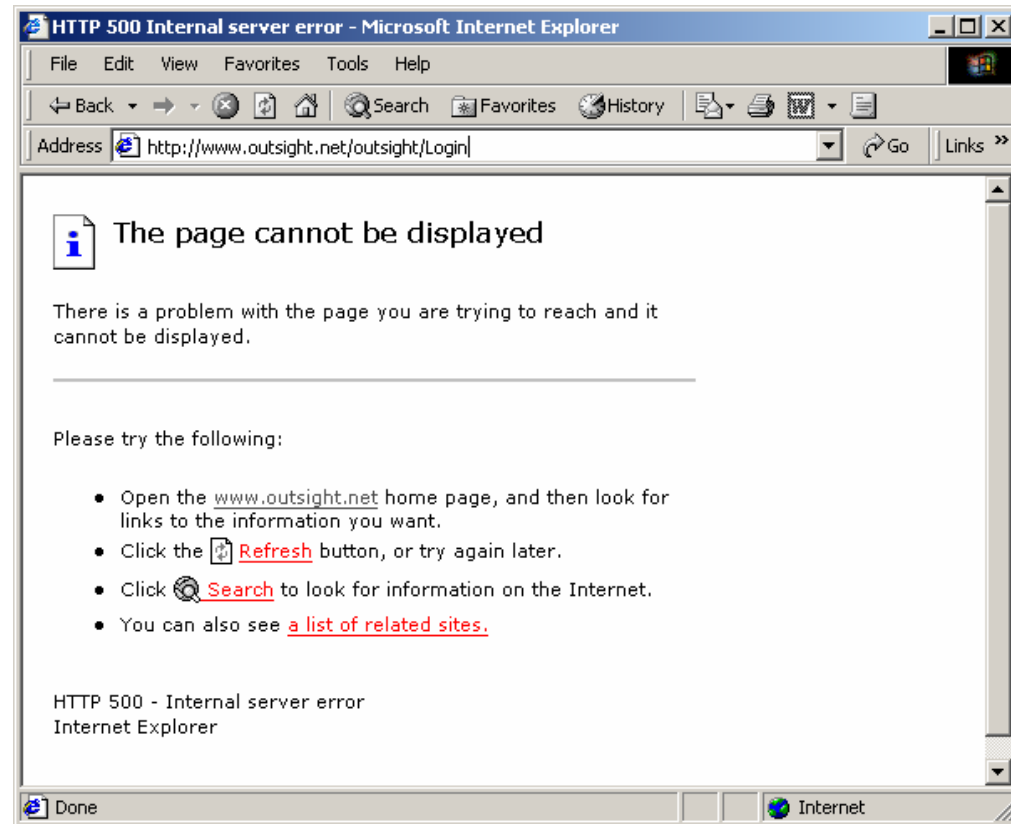
# Availability

- Network errors
  - High retransmissions, DNS resolution failure



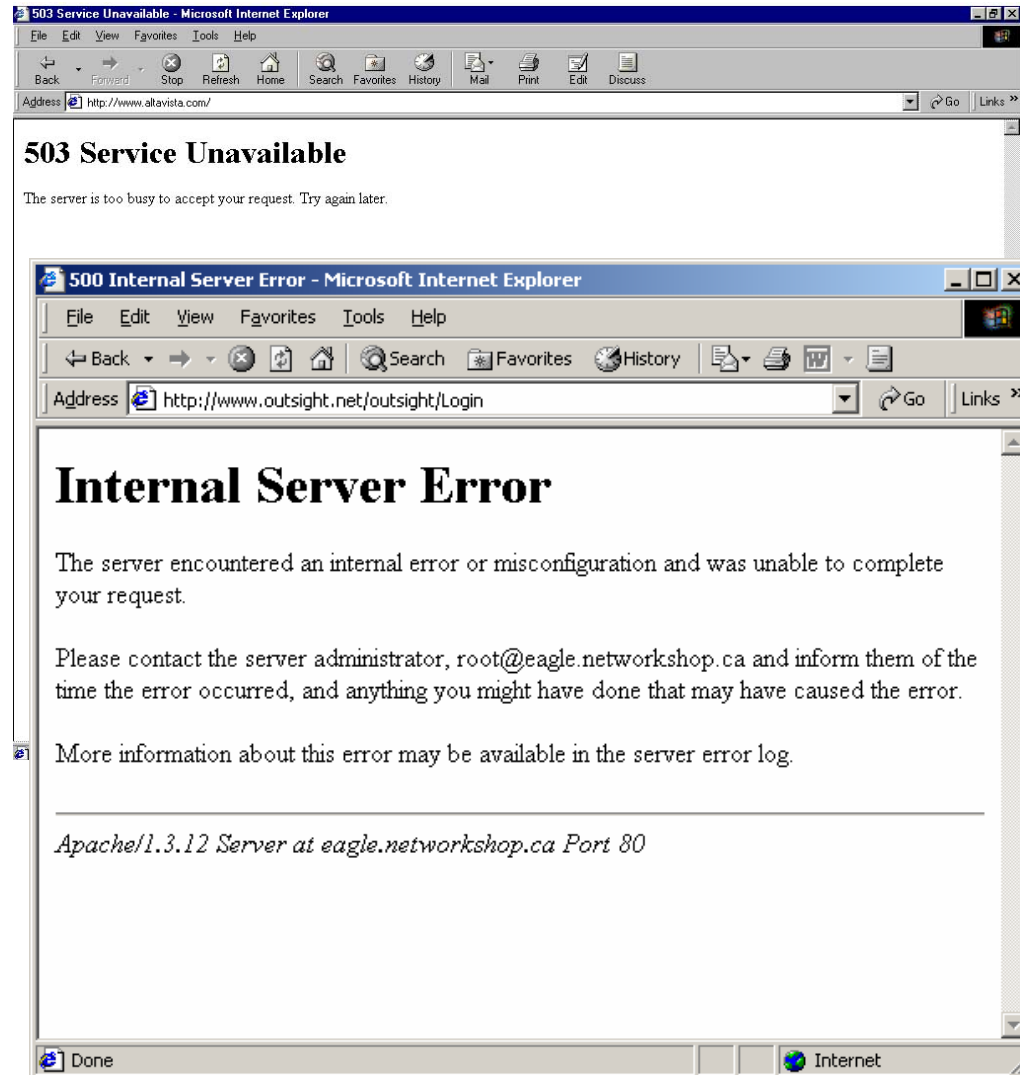
# Availability

- Client errors
  - 404 not found



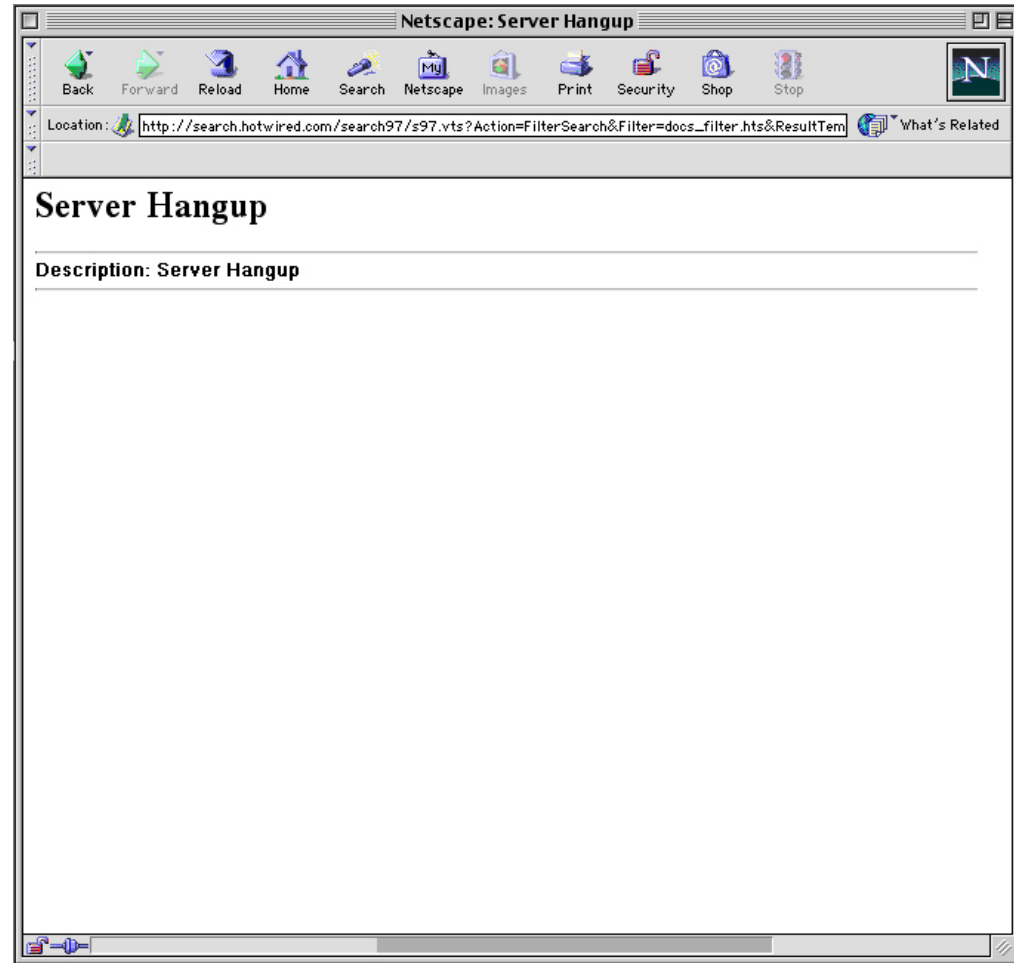
# Availability

- Application errors
  - HTTP 500



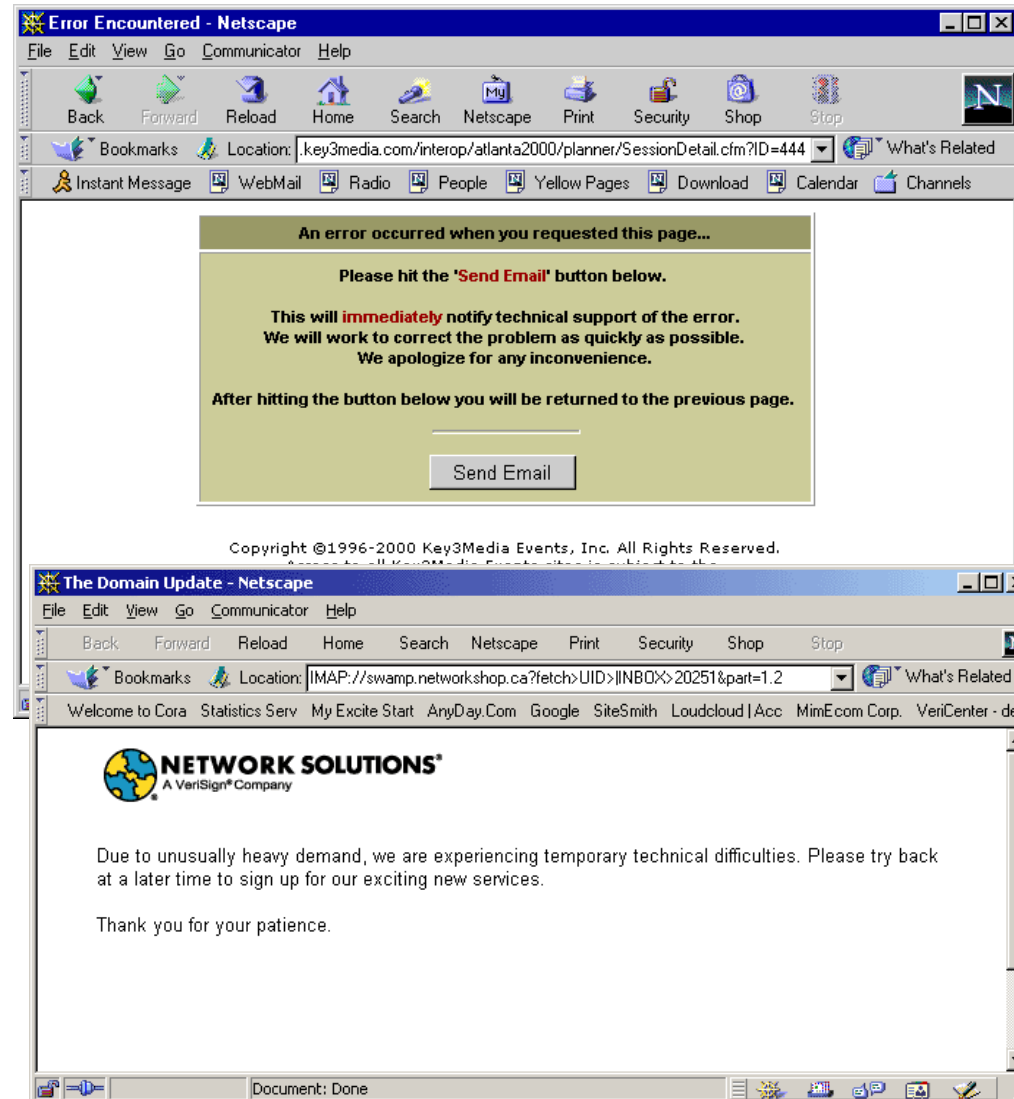
# Availability

- Service errors



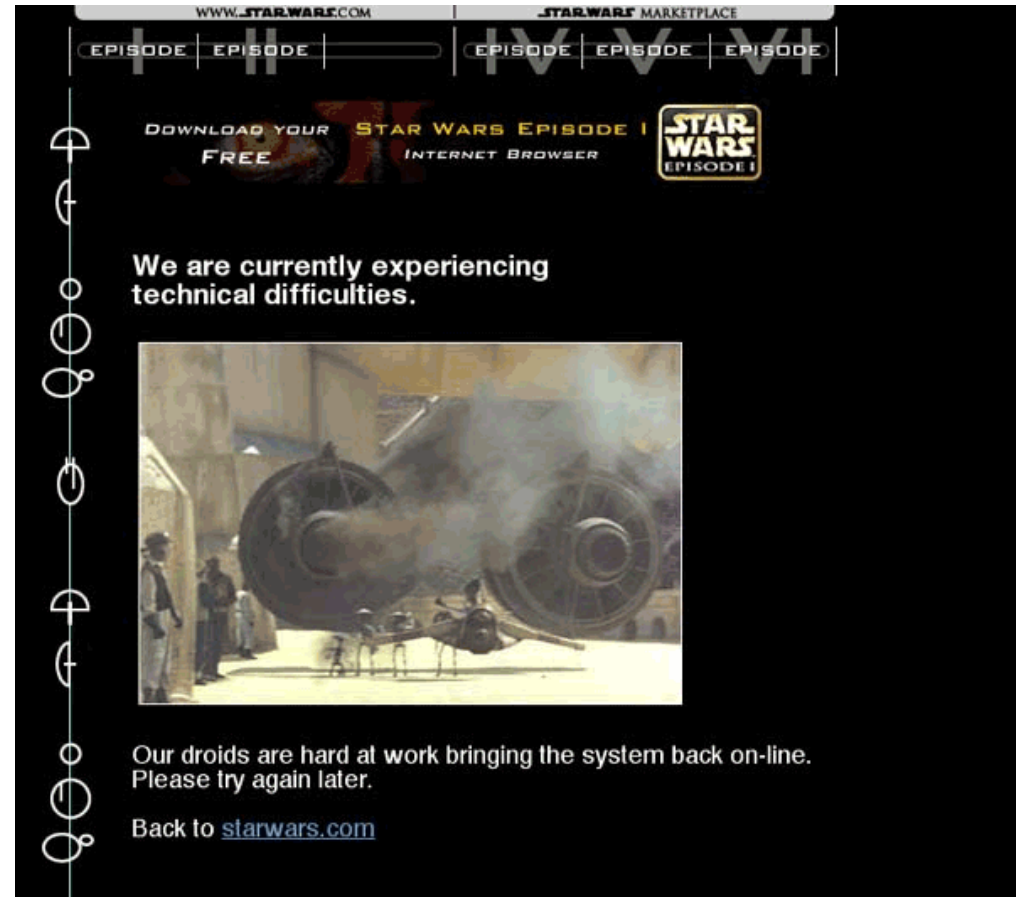
# Availability

- Content & back-end errors
  - “ODBC Error #1234”



# Availability

- Custom errors
  - Specific to your business

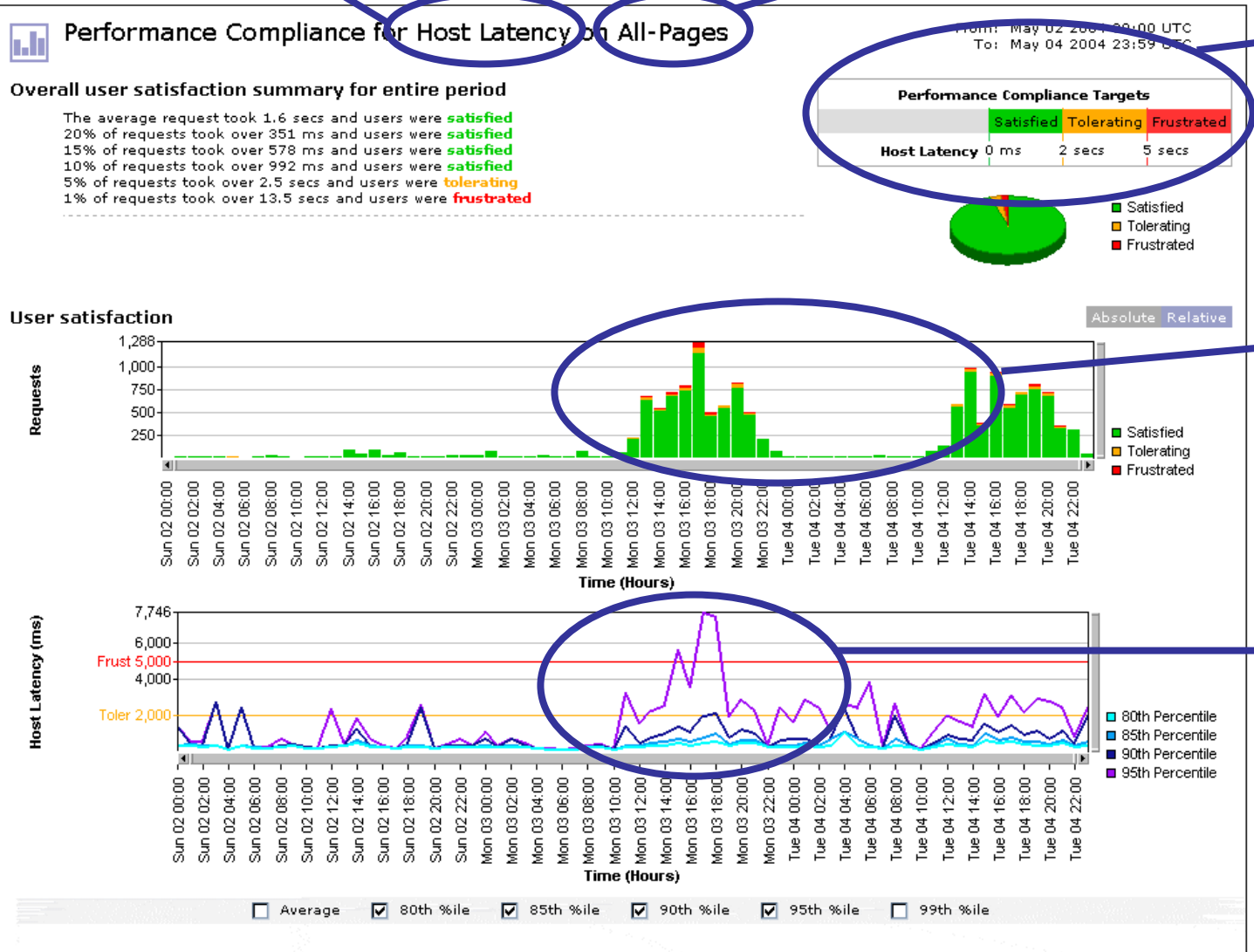


The image shows a screenshot of a Star Wars website error page. The page has a black background with white text and a central image. At the top, there are navigation links for "WWW.STARWARS.COM" and "STAR WARS MARKETPLACE". Below these are several "EPISODE" links. The main content area features a promotional banner for "STAR WARS EPISODE I" with the text "DOWNLOAD YOUR FREE STAR WARS EPISODE I INTERNET BROWSER" and the "STAR WARS EPISODE I" logo. The central message reads: "We are currently experiencing technical difficulties." Below this is a screenshot of a scene from Star Wars Episode I, showing a large, cylindrical object being moved by droids in a workshop. At the bottom, the text says: "Our droids are hard at work bringing the system back on-line. Please try again later." and a link "Back to [starwars.com](http://starwars.com)". On the left side of the page, there is a vertical navigation bar with circular icons.

# User satisfaction: *Satisfied, tolerating, frustrated*

What metric?

What function?



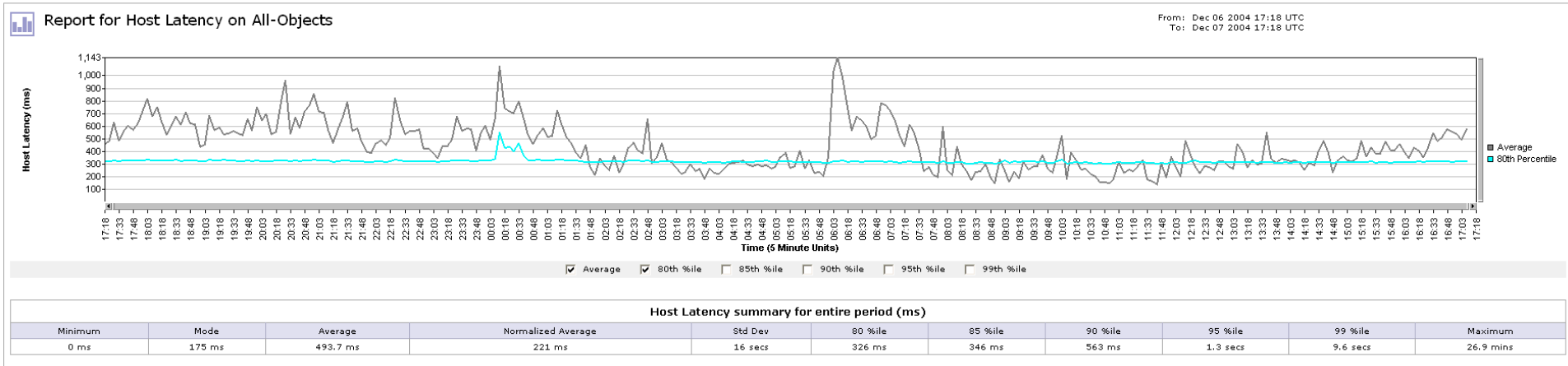
Target performance

Impact on users

Percentile data

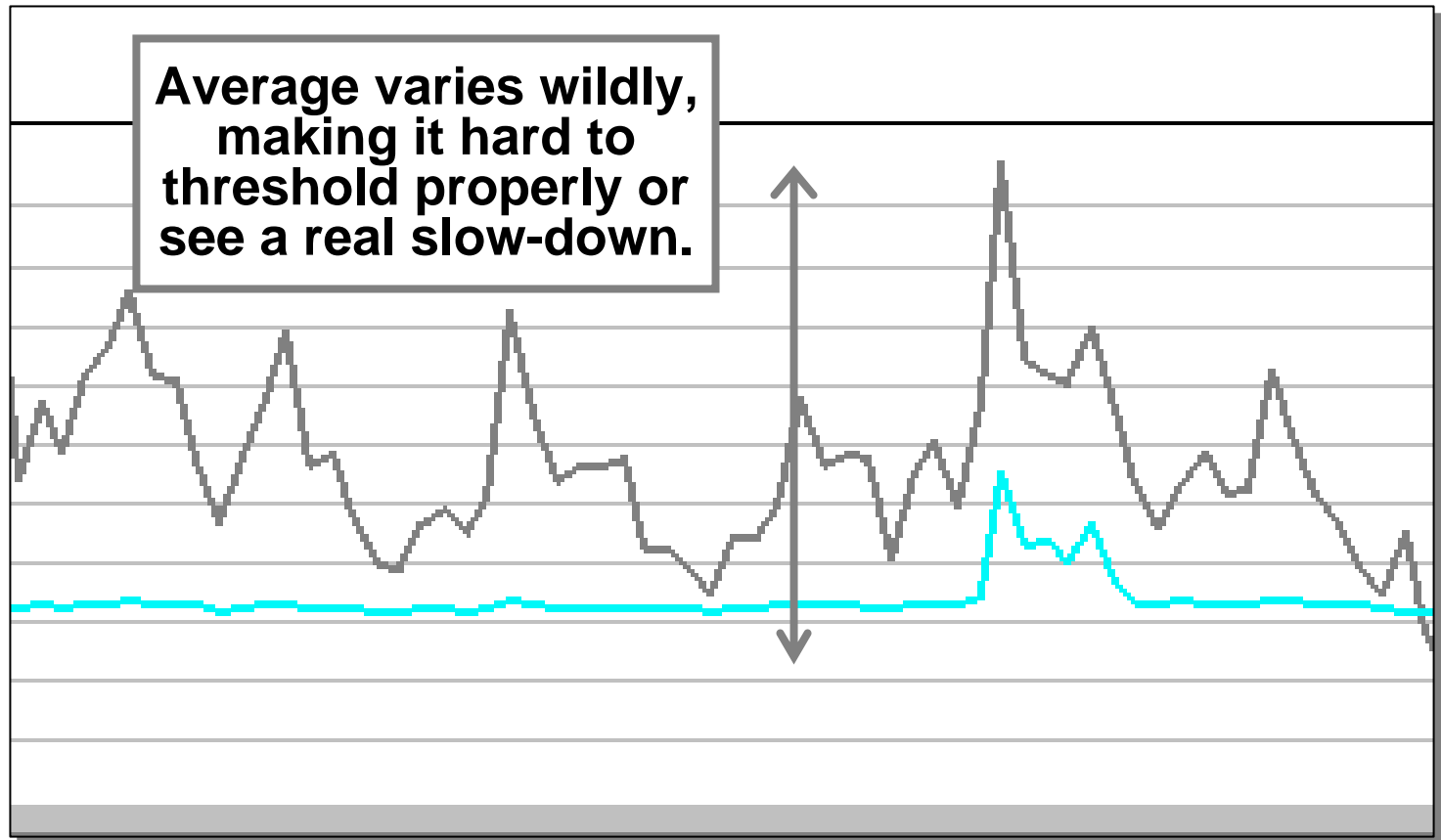
# Averages lie: *Use percentiles*

[Float as image](#) [Float ?](#)



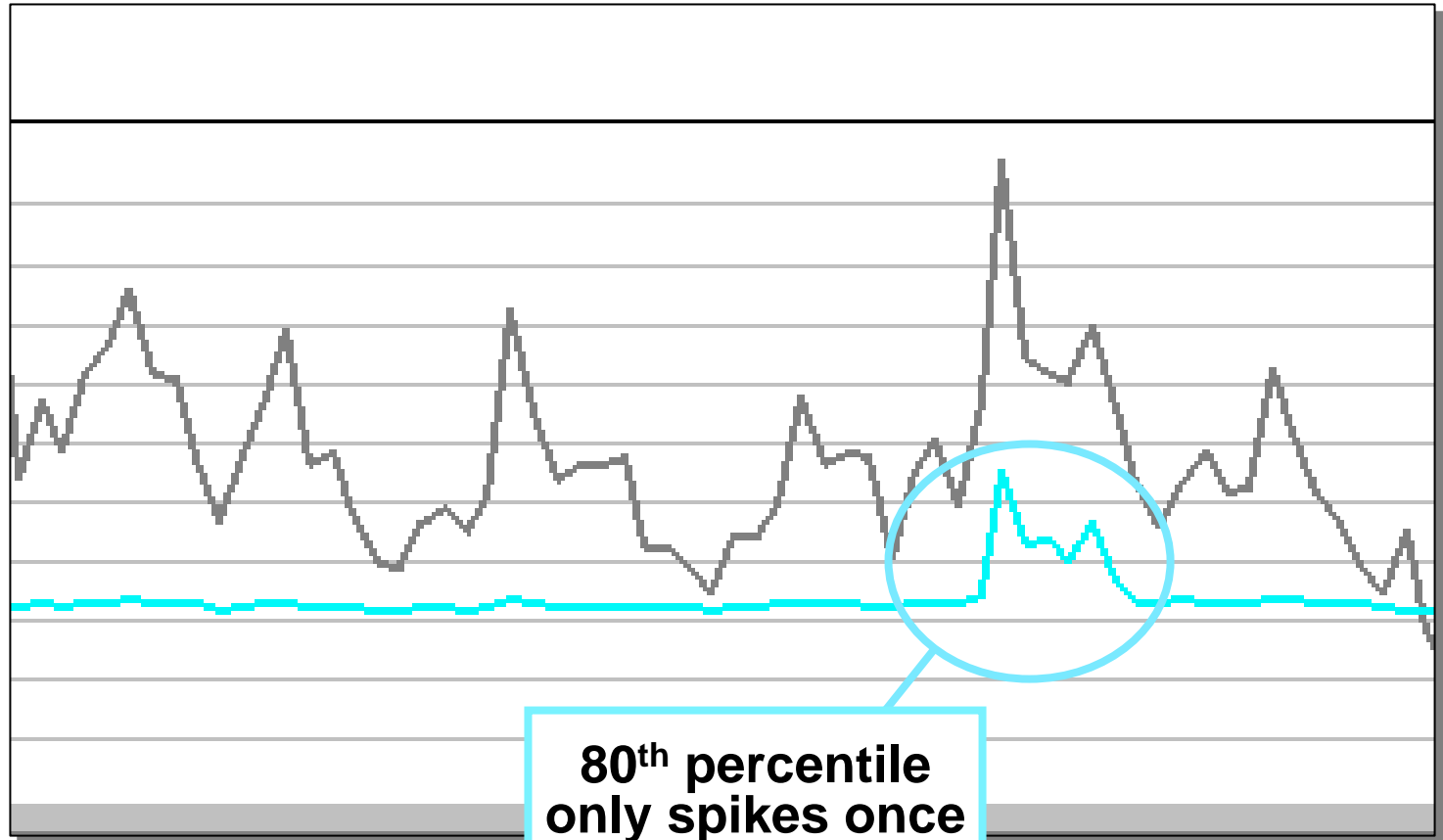
# Averages lie: *Use percentiles*

---



# Averages lie: *Use percentiles*

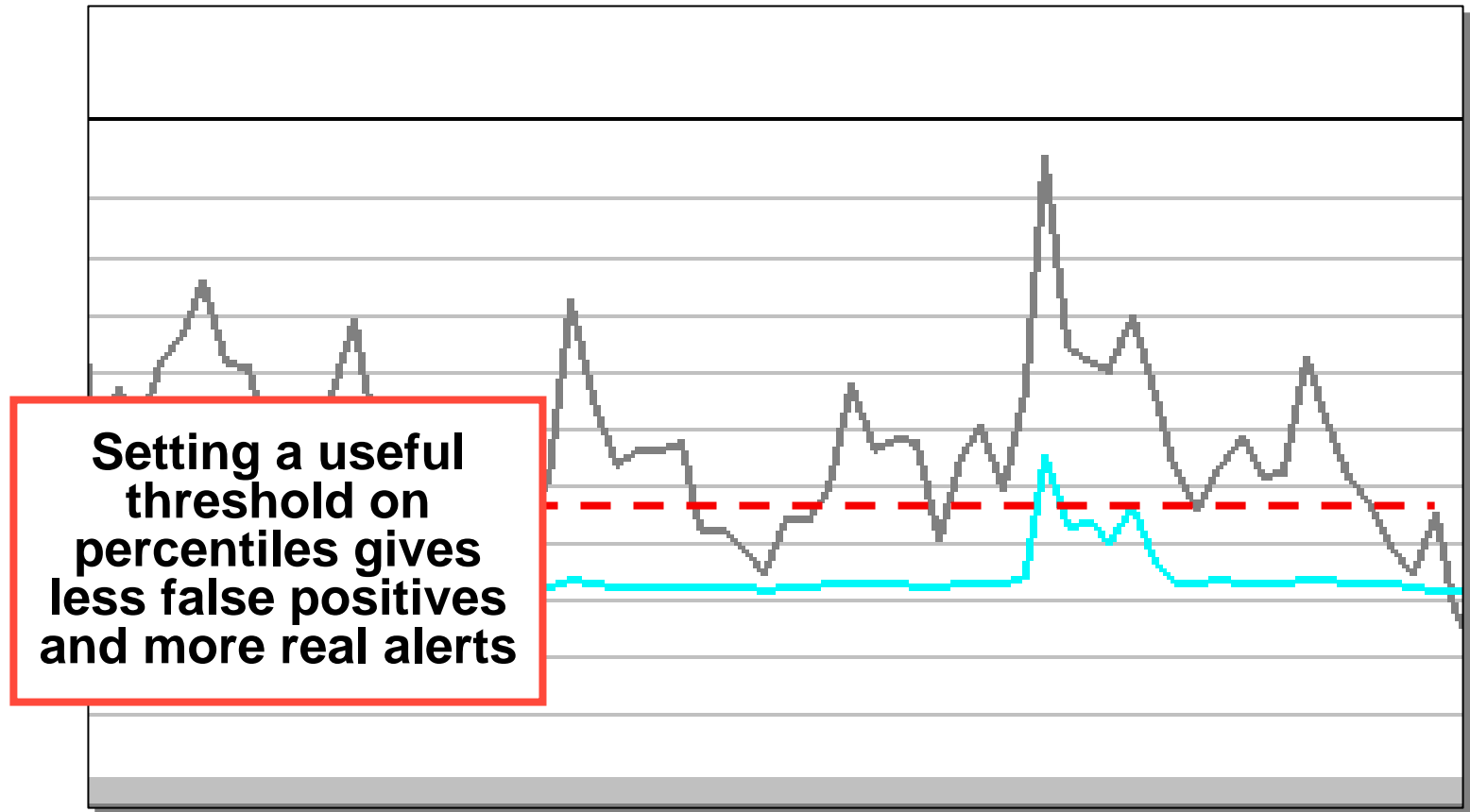
---



**80<sup>th</sup> percentile  
only spikes once  
for a legitimate  
slow-down (20%  
of users affected)**

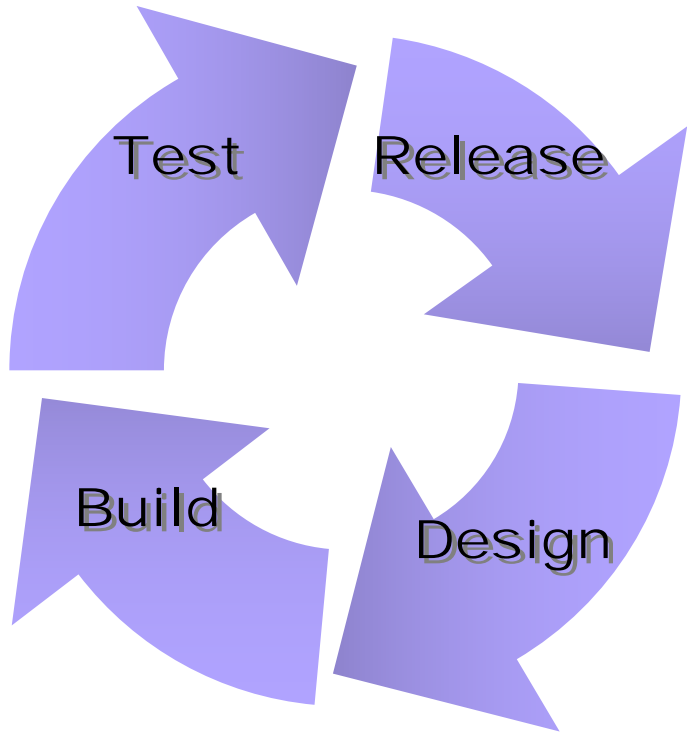
# Averages lie: *Use percentiles*

---



# Best practice workflow

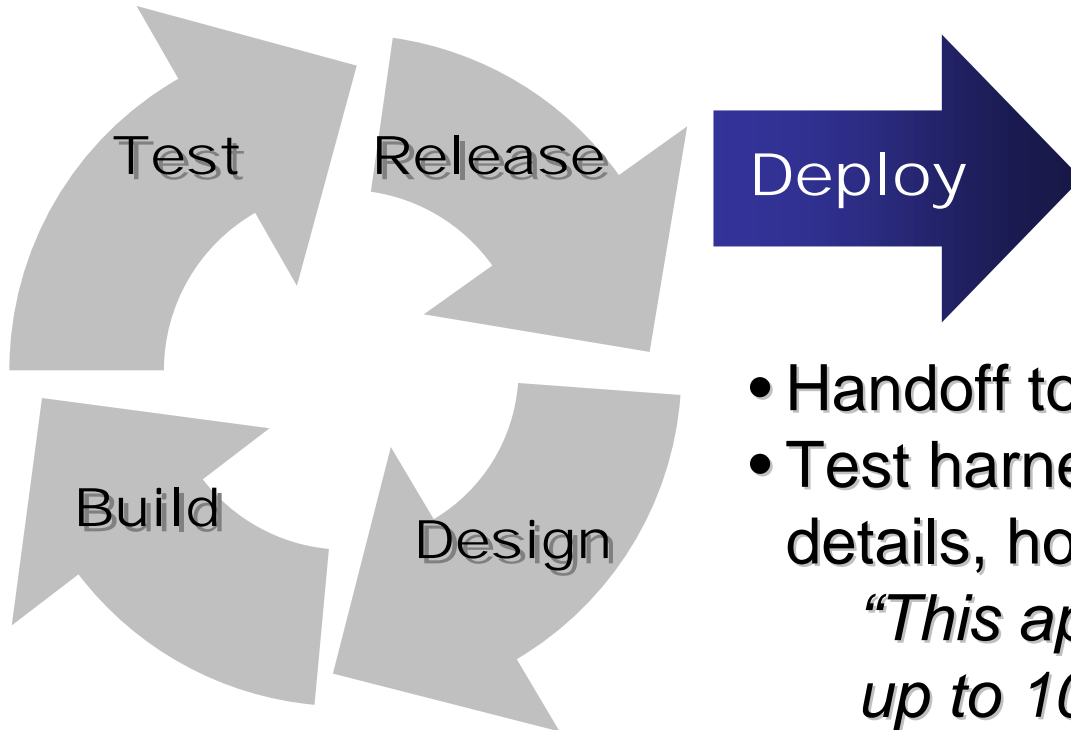
---



- The web development cycle
  - Specify the application
  - Build and test it
  - Release it

# Best practice workflow

---

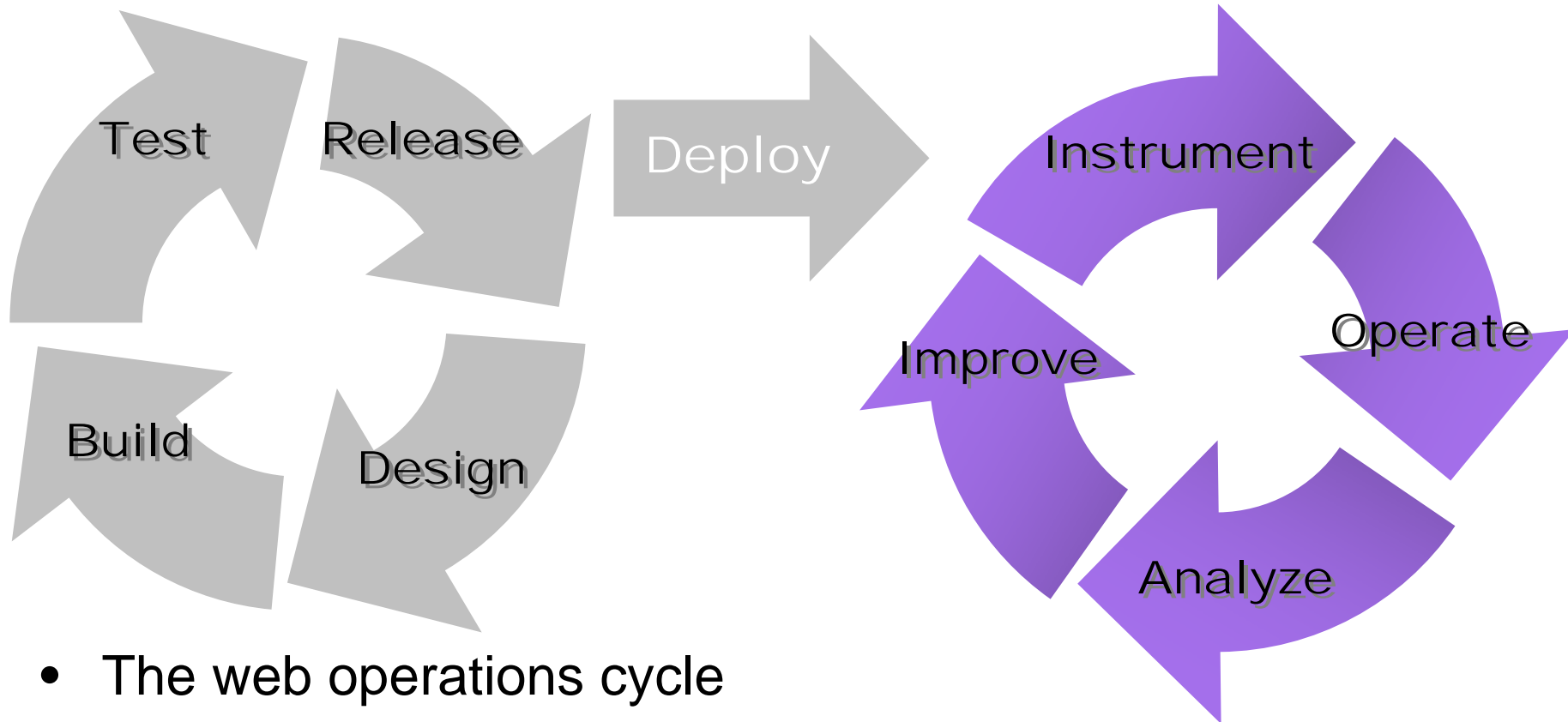


- Handoff to operations
- Test harness, load & capacity details, hooks

*“This application will respond to up to 10 requests per second with no more than 1 second of host-side delay per request. It will be usable across a 128Kbps link.”*

# Best practice workflow

---

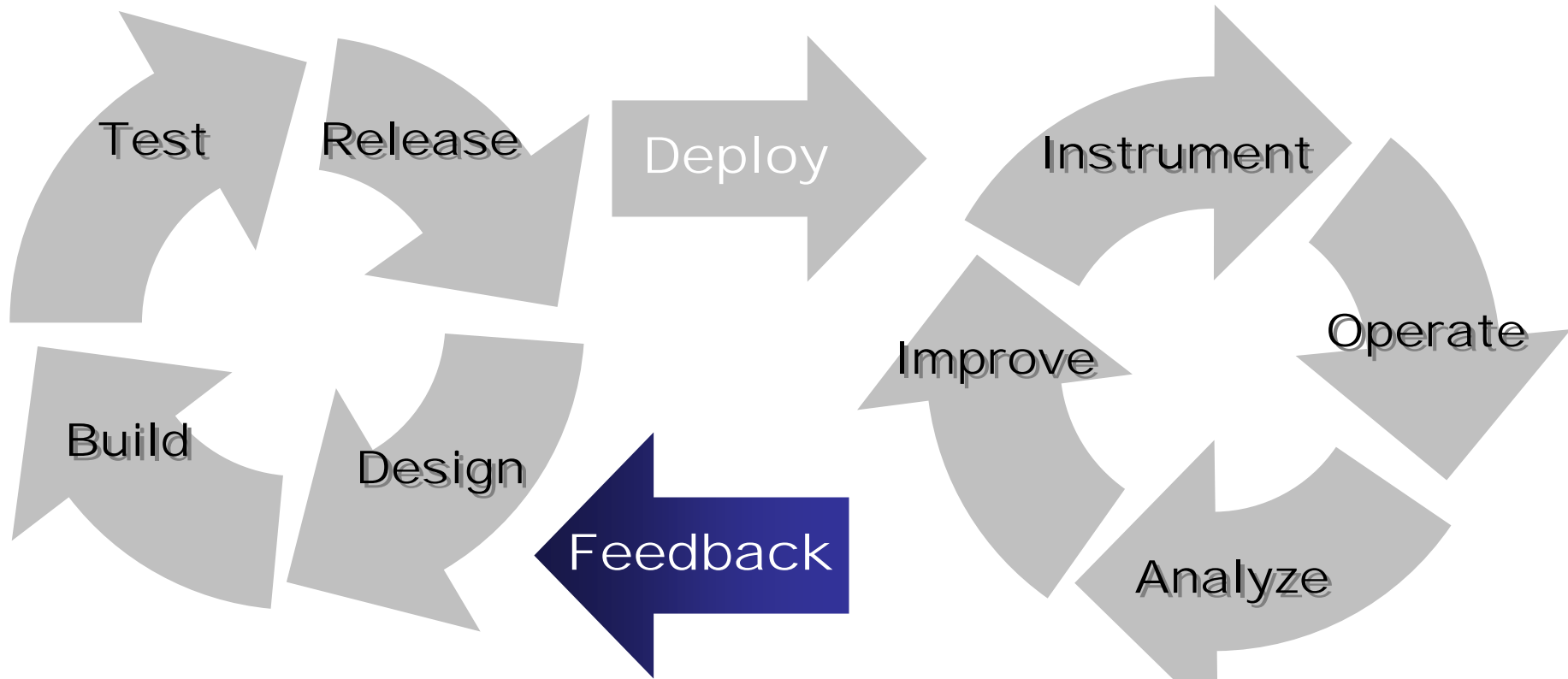


- The web operations cycle

- Instrumentation using the three approaches
- Incident management and service-level management
- Analysis of results for capacity planning
- Continuous improvement of baselines and thresholds

# Best practice workflow

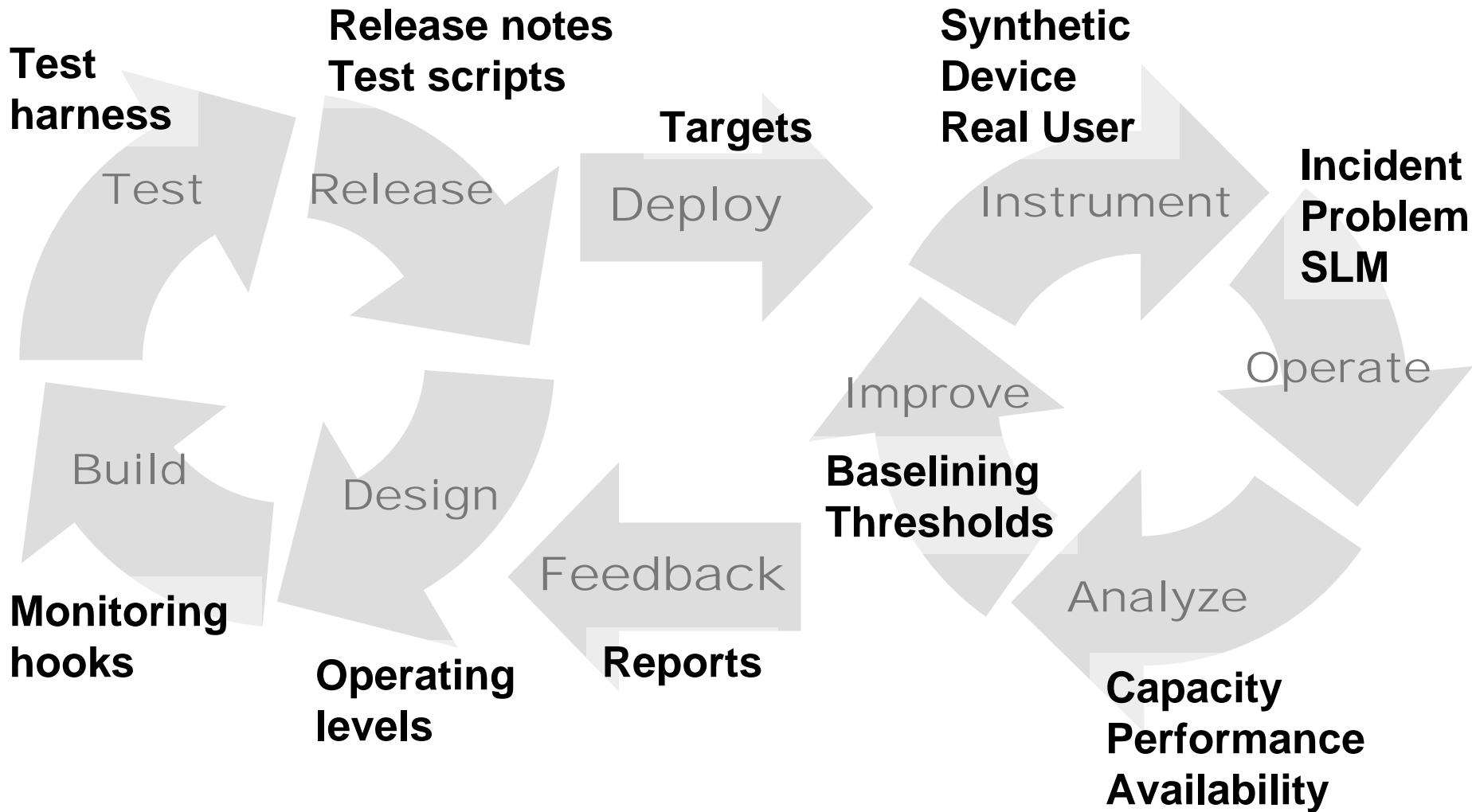
---



- IT feedback to development teams
  - *Reporting according to stakeholder*
  - *Common source data*
  - *Performance and availability*
  - *Load-to-performance relationship*

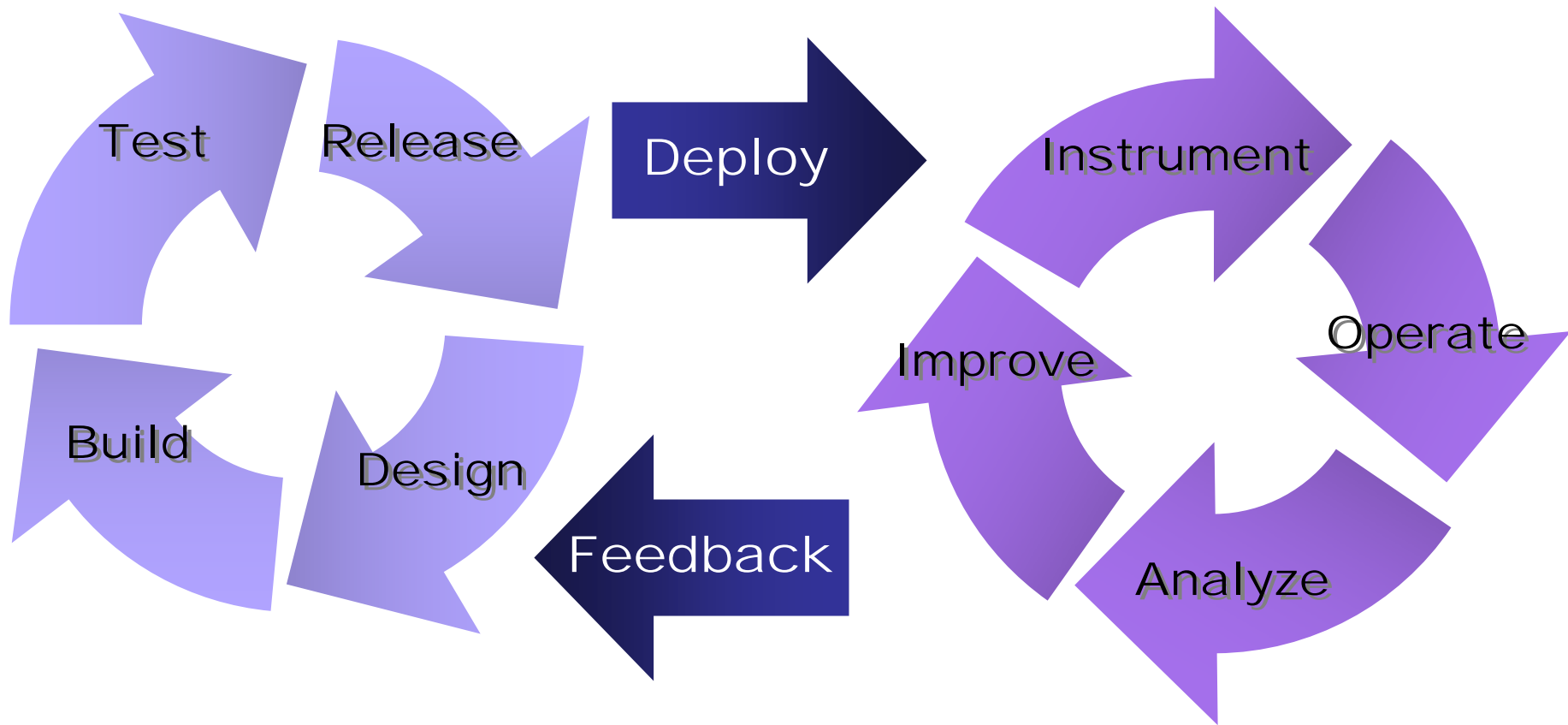
# This affects all of IT

---



# Best practice workflow: The big picture

---



- Business goals
- Operating processes
- Tools
- Metrics

---

# Questions?

acroll<at>coradiant.com

(514) 944-2765

