when web services go bad

the fun begins at deployment

steve loughran
hp laboratories
slo@hpl.hp.com

October 2002

why is it so hard?

web service development =

global distribution

+global load

+expectations of “web speed” development

+http at the bottom

+integration with remote callers

+service level agreements covering QoS

=***a new set of a problems!***

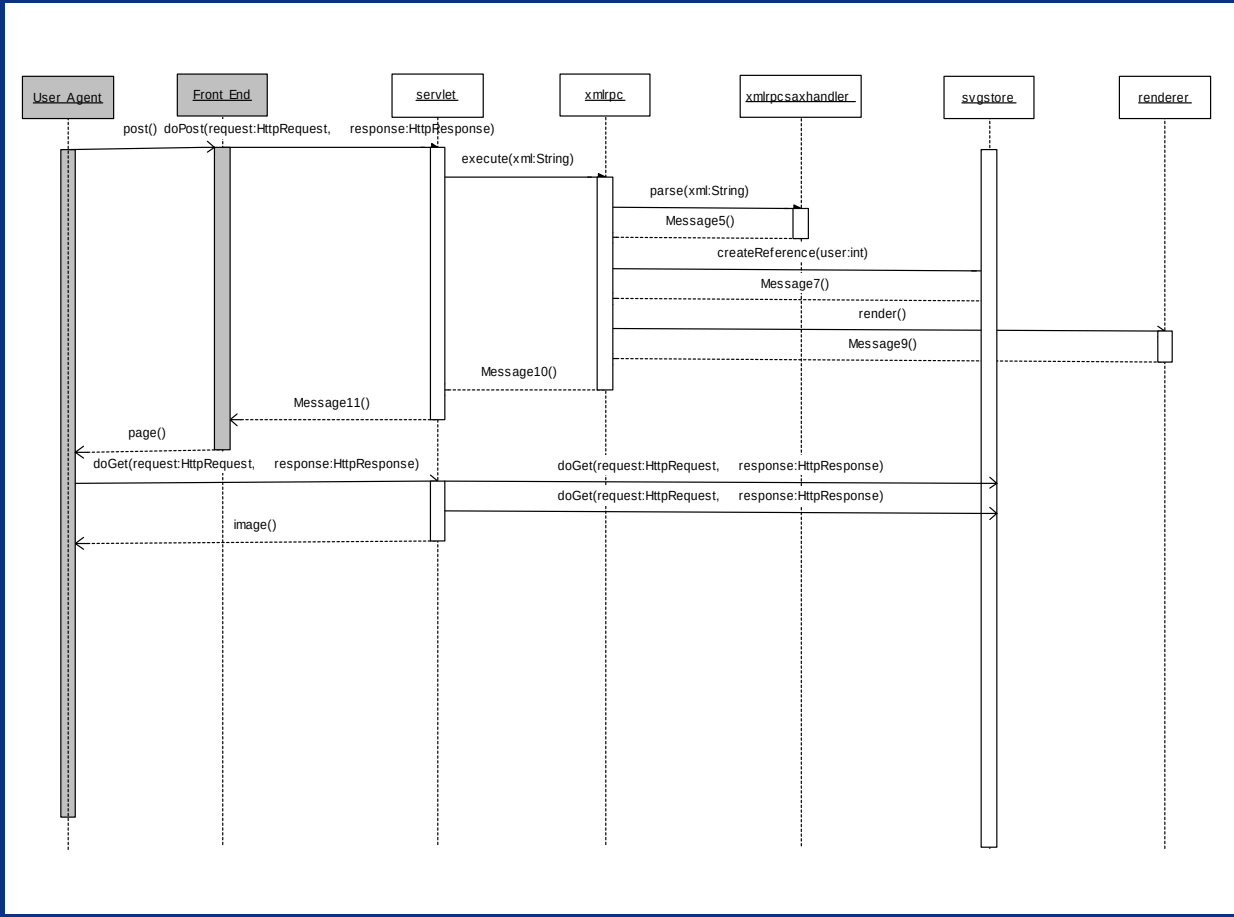
image storage and SVG rendering service

XML-RPC

1. POST SVG in payload
2. render to JPEG
3. return URL
4. user fetch

   or

   print on press



+1TB image store for customer photos

constraints

- SLA: high availability
- render times:
  2s proof render
  15s production

basic asset store done:
-SQL server
IIS/ASP

COM/MTS->Win2K/COM+
"real soon now"

- XML-RPC agreed on RPC mechanism
- wildly optimistic timescales

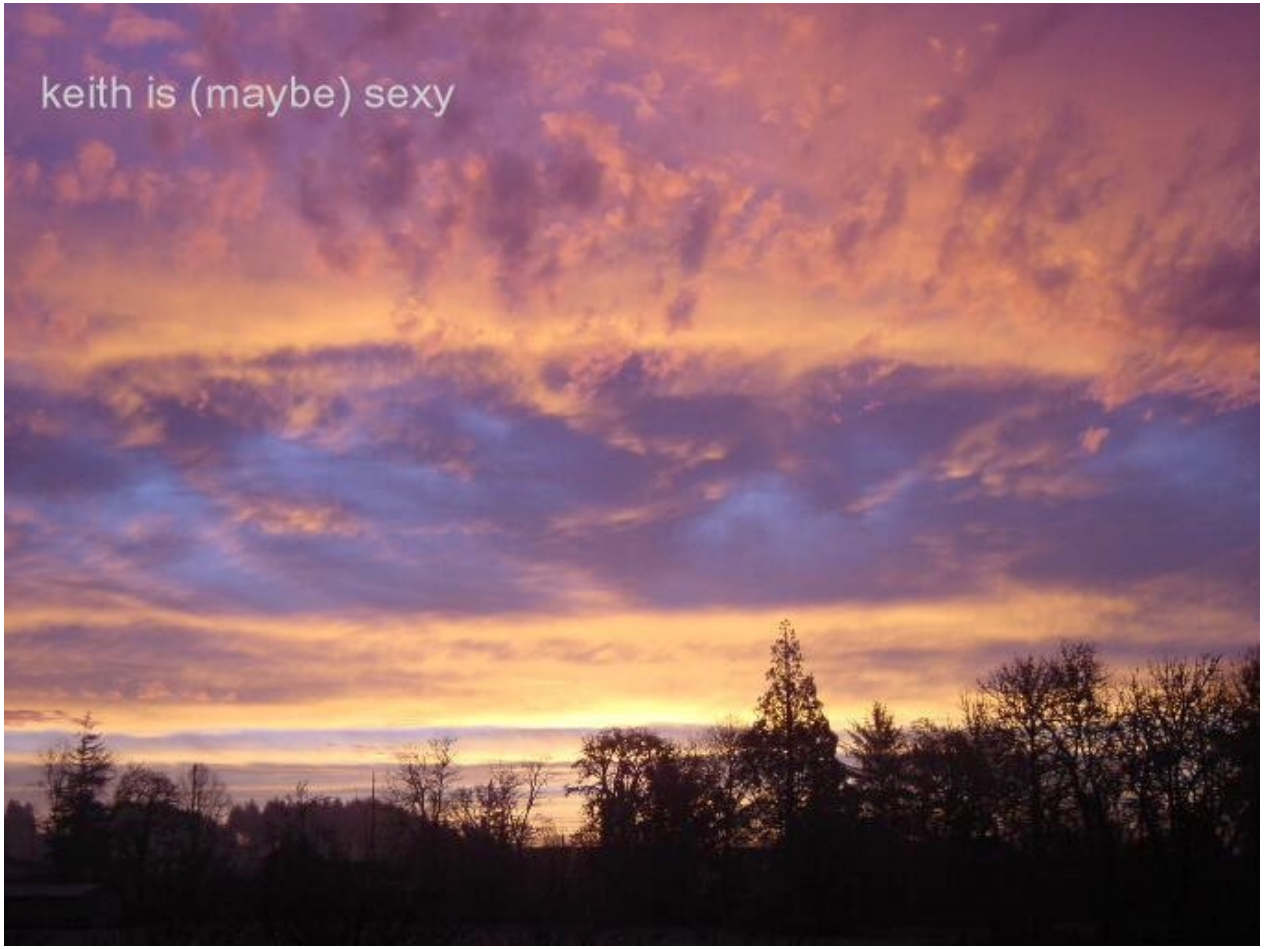
SVG into XML-RPC

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
 "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">
<svg width="1024pt" height="768pt">
 <g id="picture" ><image width="1024pt" height="768pt"
   xlink:href="http://stelvio:8080/sunset.jpg"/></g>
<text x="29pt" y="85pt"
 style="font-family:Helvetica;font-size:36pt;fill:#d0d0d0;">
Keith is (maybe) sexy </text>
</svg>
```

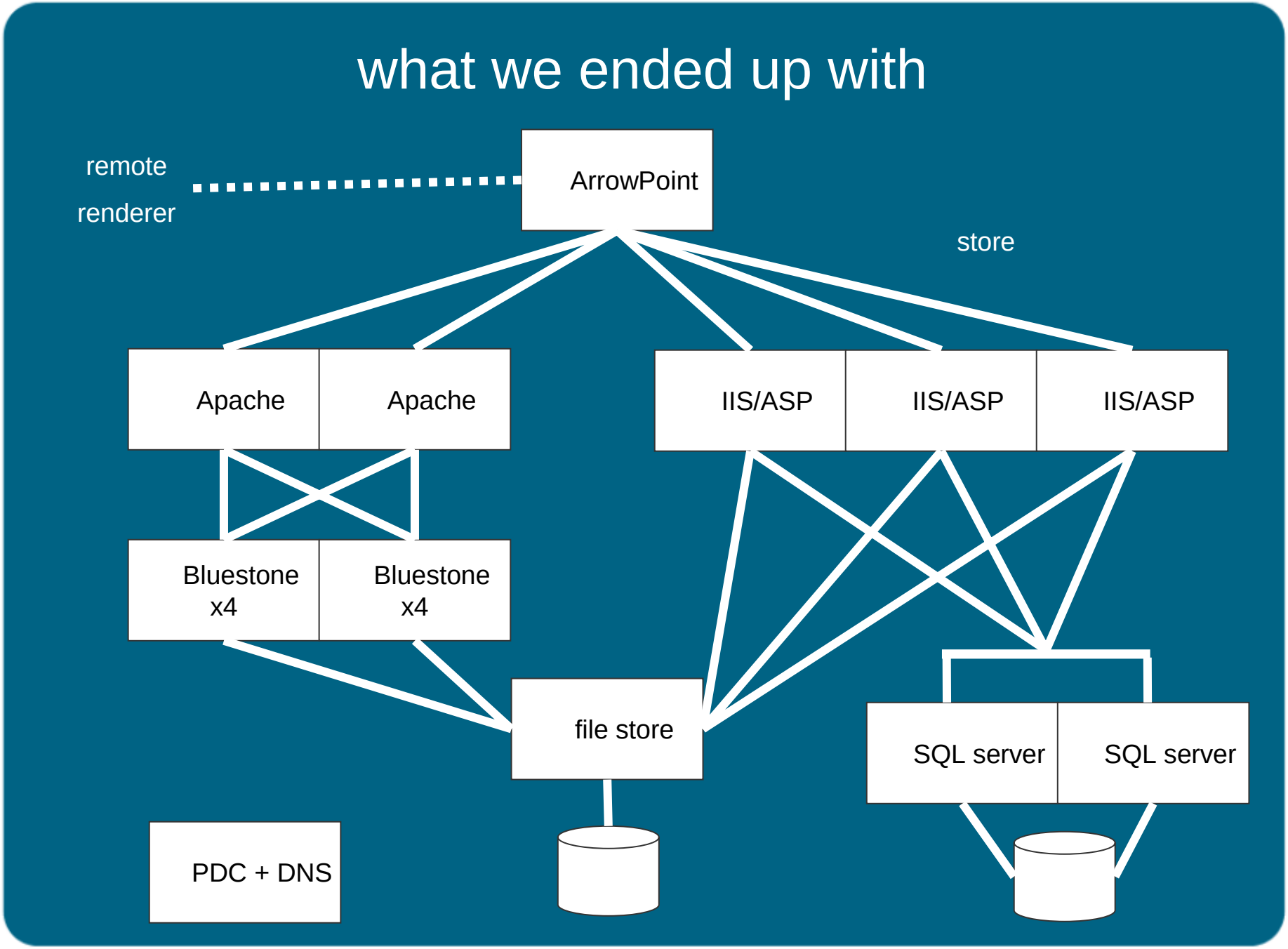
```
<?xml version="1.0"?><methodCall version="1.0">
<methodName>render_svg</methodName>
<params>
<param><value><string>110</string></value></param>
<param><value>
<string>
PD94bWwgdmVyc2lvbj0iMS4wIiBzdGFuZGFsb25lPSJubyIgPz4NCjwhRE9DVFlQRSBzdmcgUFVC
TElDICItLy9XM0MvL0RURCBTVkcgMjAwMDExMDIvL0VOIg0KICJodHRwOi8vd3d3LnczLm9yZy9U
Ui8yMDAwL0NSLVNWRy0yMDAwMTEwMi9EVEQvc3ZnLTIwMDAxMTAyLmR0ZCI+DQo8c3ZnIHdpZHRo
PSIxMDI0cHQiIGhlaWdodD0iMTAyNHB0Ij4NCiA8ZyBpZD0icGljdHVyZSIgPjxpbWFnZSB3aWR0
aD0iMTAyNHB0IiBoZWlnaHQ9Ijc2OHB0Ig0KICAgeGxpbms6aHJlZj0iaHR0cDovL3N0ZWx2aW86
ODA4MC9zdW5zZXQuanBnIi8+PC9nPg0KPHRleHQgeD0iMjlwdCIgeT0iODVwdCINCiBzdHlsZT0i
Zm9udC1mYW1pbHk6SGVsdmV0aWNhO2ZvbnQtc2l6ZTozNnB0O2ZpbGw6I2QwZDBkMDsiDQo+a2Vp
dGggaXMgKG1heWJlKSBzZXh5PC90ZXh0Pg0KPC9zdmc+DQoNCg0K</string></value>
</param>
<param><value><int>72</int></value></param>
</params></methodCall>
```

Response

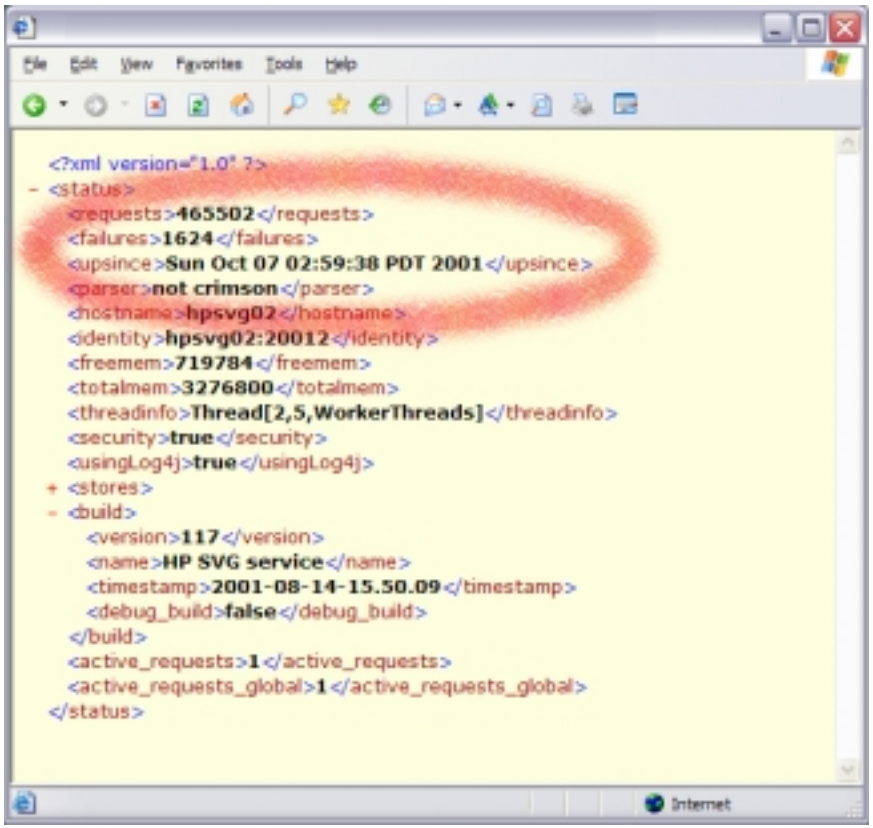
```
<?xml version="1.0"?>
<methodResponse>
<params>
<param><value><string>2453</string></value></param>
<param><value><string>
http://stelvio:8080/hpsvgservlet/fetch?ref=_2_ec5132765b_4553.jpg-110
</string></value>
</param>
</params>
</methodResponse>
```



what we ended up with



what worked?



Web Service Model!

✓XML based RPC

✓HTTP for transport

✓SVG for request

✓Ant build & deploy

✓JUnit for unit tests

✓incremental development

(nearly) no more WORKSFORME

all server-side problems could be replicated

- log all failures server side for easier post mortem
- SEH catching of all win32 renderer errors
- phase II: SEH errors ⇒ email to support alias

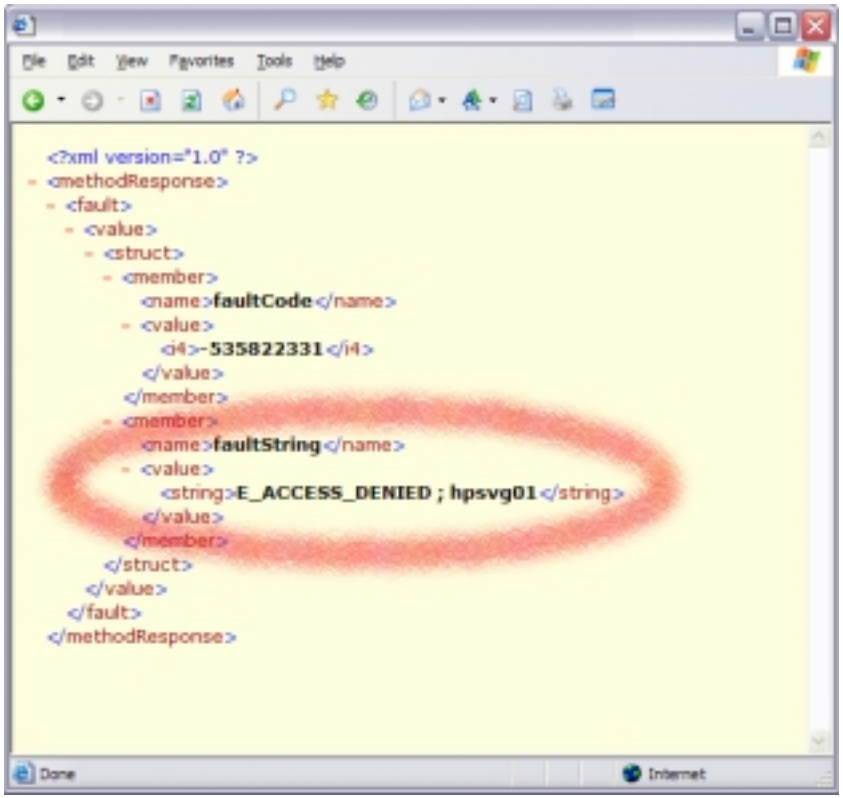
just integration…

networking, client code issues (proxies, wrong URL, authentication), *interop*

Ant + JUnit

- **fully automated build process**
- **integrated JUnit testing**
- **automated deployment to local server stacks**

security



- cookie authentication in common domain
- encrypt user, session in cookie
- restricted ports on beta site
- admin pages by IP address
- sanitize incoming SVG XML:

  -catch file: references

  -downgrade http: access

  -test XML &includes;

configuration

- ASP: config in source: brittle, scaling issues
- Java: per cluster config files in the WAR
  +kept under SCM; cleaner
  -delays to change, scaling

1. Use a database ?
2. Use a directory service?

what didn’t work:

turning a web site into a web service

what went wrong

- cluster race conditions
- boot race conditions
- MP thread safety
- resource leakage
- URL of death

- COM+ authentication
- DNS
- JVM ‘lockup’
- time differences between servers

- FedEx
- cabling
- raid controller
- unreliable switch
- router config

- forgotten passwords
- IP address issues
- accidental deletion of 8GB test data (!)

operations paranoia

1. R&D not allowed near production boxes
2. response to any security issue is “no live”

- when things don’t work, then they call us
- escalated minor security niggles into blocking issues

effect #1: threat to weekends

effect #2: threat to schedule

effect #3: we stopped telling them of “issues”

error messages

java.io.NoRoute ToHostException

if you don’t have Java/C# engineers on the ops team, you get called in for every message

fix #1: “defect” tracking of operations issues, from early days of the app

fix #2: always identify the errant system in fault codes

HTTP

- Content-length is all you get
- how do you propagate a transient failure

E_TRANSIENT_FAILURE

...retry with exponential backoff

-how you really test this?

-what about integrity?

firefighting

- it takes 10 minutes to deploy
- it takes 30s to report a defect

therefore, it must take 11 minutes to fix a bug and deploy the update

…stay in control by never updating public servers more frequently than nightly

availability $A(X) \stackrel{def}{=} P(X)$ is working

$0 \leq A(X) \leq 1$

service S depends on services $s_{1..n}$

$$A(S) = A(node)*A(s_1)* A(s_2)\ldots A(s_n)$$
$$= \mathbf{A(node)*\Pi_{1..n} A(s)}$$

latency:

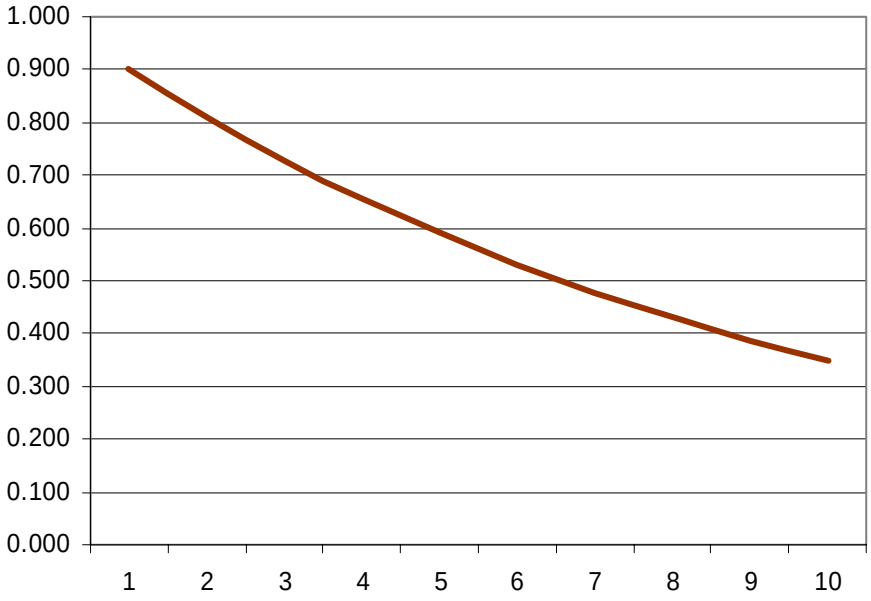
$$L(S) = L(node)+L(s_1)+L(s_2)\ldots L(s_n)$$
$$= \mathbf{L(node)+\Sigma_{1..n} L(s)}$$

redundancy:

$$A(n_1+n_2)=A(n_1)+A(n_2)-A(n_1)A(n_2)$$

cost of redundancy depends on scalability of service: O(1), O(n), $O(n^2)$, $O(2^n)$ …

complexity is the enemy



availability is the primary casualty of complexity

a modest proposal:

deployment-centric software processes

operations use cases/ XP stories

- update live server
- add new fonts
- bind to new database
- change account/passwords
- backup/restore system
- partition cluster
- multi-home the server
- diagnose intermittent failure

these need support in software and/or process

deployment and operations test cases

- probe for needed exes, COM objects
- validate remote sites visible
- check configuration

treat all deployment issues as defects to track and to have test cases

run the test cases at install time, and from the load balancing router

operations issues are defects

treat all deployment issues as defects to track

don’t just fix it once, at it will crop up again.

you *need* regression tests

you *need* a repository of defects for easy searching.

or they will phone you at 3am

deploy early
deploy often

- involve operations in system design
- give them regular builds to deploy during early development
- use a tool like CruiseControl for continual integration and local deployment
- provide 5x6 support at this stage
- maintain slow bug fix times

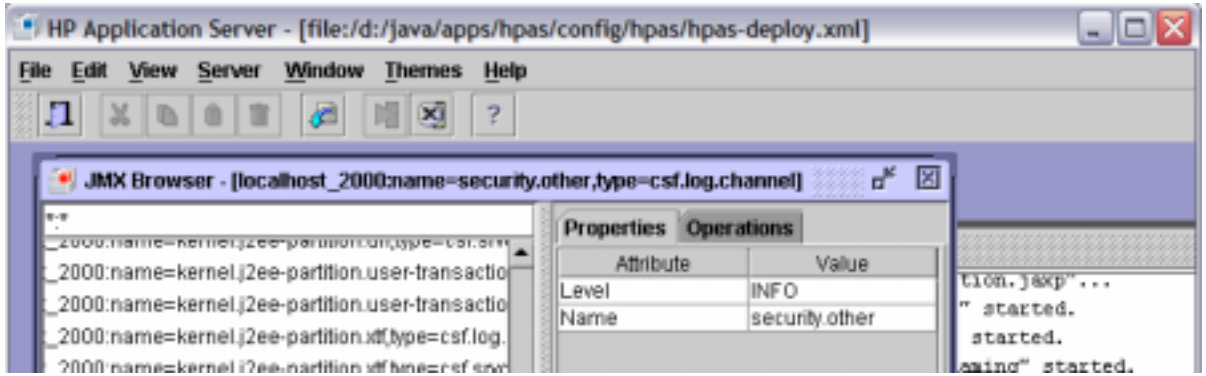
instrument with JMX (Java) WMI (.NET)

```
public class ServiceManager
             implements ServiceManagerMBean {

    protected Service _owner;
    protected int _serviceTransactionCount;
    protected int _serviceUseCount;
    protected double _sellThroughDollars;

    public synchronized void bookSales(int count,
                                       double dollars)
    {
        _sellThroughDollars+=dollars;
        _serviceUseCount+=count;
        _serviceTransactionCount++;
    }

    public Double getSellThroughDollars()
    {
        return new Double(_sellThroughDollars);
    }
```



Current project

web service printing
Schema-based content

- JMX instrumentation
- operations use cases, tests
- *continual deployment* with CruiseControl to local server
- interop tests

-regression against Axis is a continual pain

-interop is trouble

vision of the future:

attributes of server-side components

- instrumented for remote management
- unified configuration mechanism
- come with unit tests
- common logging API

vision #2:

Web Services as components

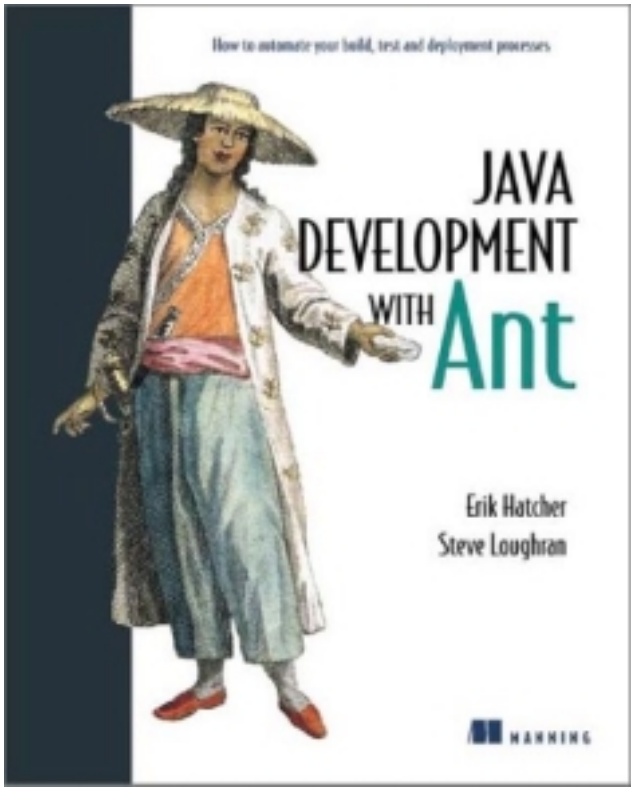
What would it *really* take?

summary

- web services are more complex than web sites or intranet applications to deploy
- developers: deployment is (nearly) everything; integration & interop is the rest
- operations: developers are not your enemy
- management: recognize the new problems; address them

strive for simplicity

www.hpl.hp.com



manning.com/antbook

chapters 12,15,16,18