SearchSoftwareQuality.com
Michael Kelly, www.MichaelDKelly.com

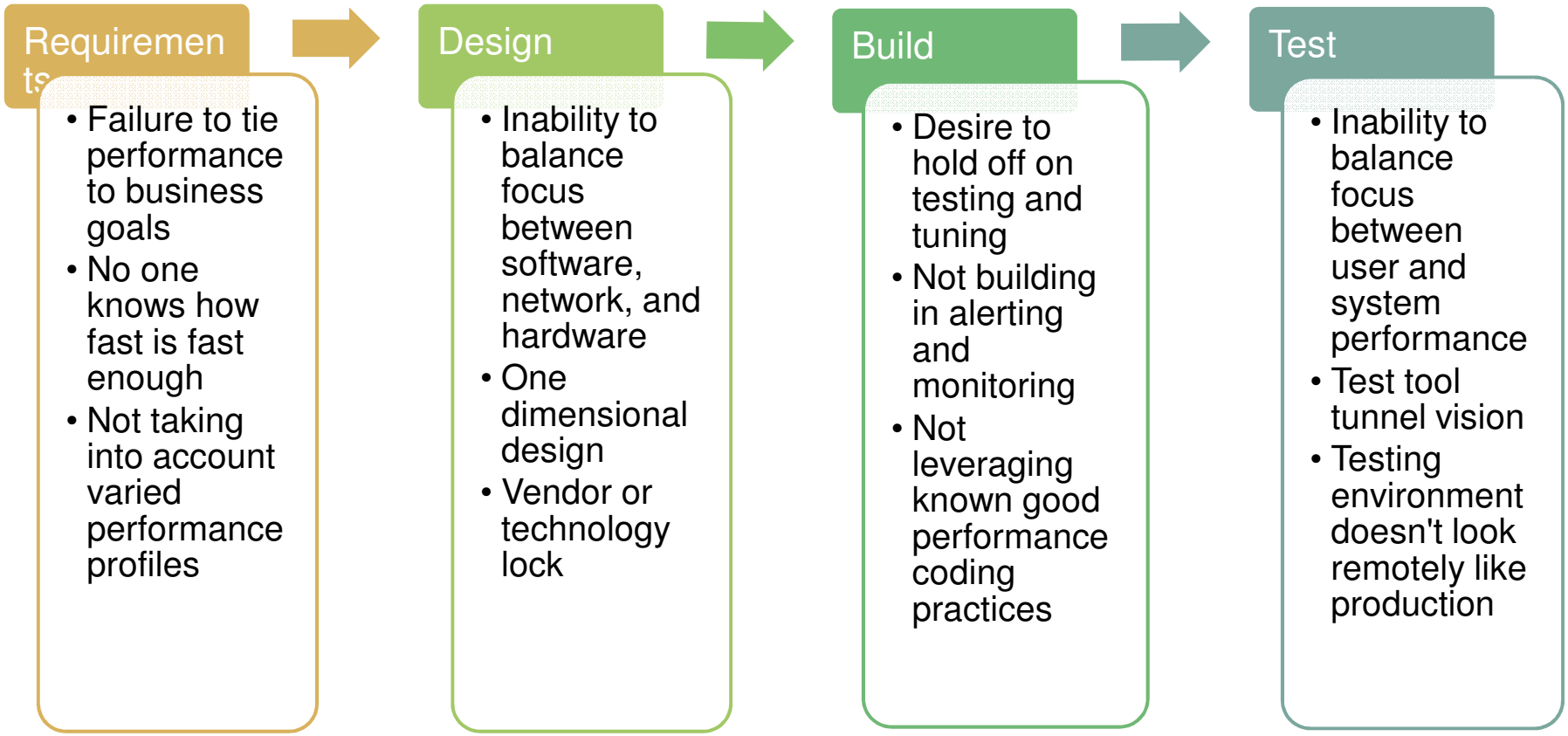
PERFECTING APPLICATION PERFORMANCE BUILDS

Advice on issues relating to requirements, design, build and test phases

It's about tradeoffs

- Application performance is often a series of tradeoffs that occur throughout requirements, design, build, and test phases:
  - You have classic project pressures: timelines and budgets
  - You have process tradeoffs: waterfall vs. agile
  - You have technology tradeoffs: open source vs. commercial
  - You have feature tradeoffs: "ability to upload documents of unlimited size"
  - You have people tradeoffs: differing areas of expertise

Common Problems



Requirements

Michael Kelly, www.MichaelDKelly.com

Failure to tie performance to business goals

The problems

- Few teams take the time to tie application performance to business metrics: (revenue, user satisfaction, repeat visits, etc…)
- Examples:
  - Bing
  - Yahoo!
  - Shopzilla
  - Netflix

Some solutions

- Take the time to define both top-line and bottom-line application business metrics
- Work to prioritize that list of metrics to better understand what's most important (creating tiers can help)
- Identify what processes and transactions will affect those key metrics

No one knows how fast is fast enough

The problems

- Failure to analyze production
- No one knows how fast is fast enough
- Even if they do, no one knows how fast an individual component needs to perform to achieve overall performance goals
- Performance requirements, goals, and targets become guesses

Some solutions

- Build a cross-functional teams to provide performance recommendations
- Tie performance to business goals, not to “industry standards”
- Define SLAs across the application

Not taking into account varied performance profiles

The problems

- Many applications have different performance profiles
  - Go-live looks different than steady state
  - Different quarters have different performance profiles
  - Etc…

  and they failure to capture different requirements based on these different needs

Some solutions

- When you define your usage models, take into account different aspects of your business:
  - Seasonal variance
  - New product release
  - Advertising campaign or new partnership
  - New company acquisition
  - Etc…

Design

Michael Kelly, www.MichaelDKelly.com

Inability to balance focus between software, network, and hardware

The problems

- Separate teams for networking, hardware, databases, application development
- The application team is overly focused on the software, not thinking of deployment hardware and configuration
- Develop software solutions to problems that can be solved by other methods

Some solutions

- Get people moving between teams to help build a more balanced perspective
- When designing performance solutions (clustering, load balancing, monitoring, etc…) involve other teams

One dimensional design

The problems

- Overly focused on how "beautiful" the application will be
- Overly focused on replicating the design of an existing system
- Overly focused on leveraging a specific technology

Some solutions

- Try using mocked up side-by-side comparisons
  - performance impact
  - A/B testing
  - Etc…
- Involve outside consultants
- Design workshops with experts in different technologies

Vendor or technology lock

The problems

- Inability to evaluate other vendors or technologies to solve the problem
  - application servers
  - queuing technologies
  - databases
  - service providers
  - switches/routers
  - etc…

Some solutions

- Design workshops with experts in different technologies
- When you've got your final technology stack, ask yourself if you really "selected" each technology or if you're only using it because it's "the default"

Build

Michael Kelly, www.MichaelDKelly.com

Desire to hold off on testing and tuning

The problems

- No performance testing at the unit, component, service level
- An attitude that says tuning should be held off until testing
- Compartmentalized development teams

Some solutions

- Provide developers with the tools and time to do unit level performance testing
- Make sure teams can hit component level-performance goals *before* integration begins
- When you review performance results, invite the consumers of your software to attend and provide feedback

Not building in alerting and monitoring

The problems

- Didn't plan for alerting and monitoring in the design
- You plan for alerting and monitoring, but the project falls behind and it's an "optional" feature
- Trusting that alerting and monitoring are features of your tool or technology selections
- Not leveraging the alerting and monitoring features of your tool or technology selections

Some solutions

- Do sequence diagram walkthroughs
- Have some backup monitoring tools
- Add project tasks to explore the alerting and monitoring features of the tools/technologies you're using

Not leveraging known good performance coding practices

The problems

- Not leveraging known good performance practices
  - reduce DNS lookups
  - optimizing images
  - working effectively with Ajax
  - etc…

Some solutions

- Just-in-time training
- External audit
- Contests to optimize code

Test

Michael Kelly, www.MichaelDKelly.com

Inability to balance focus between user and system performance

The problems

- Overly focused on end user performance
- Overly focused on transactional performance
- Overly focused on web application performance

Some solutions

- For one or two early tests, take the time track a transaction through every level of the system and get individual timings under load

Test tool tunnel vision

The problems

- Inability to evaluate or leverage multiple technologies to solve the problem
  - testing tools
  - monitoring tools
  - virtual environments
  - etc…

Some solutions

- Create a portfolio of tools, grouped by technology
- Leverage multiple tools at the same time
- Develop the habit of validating numbers using multiple methods

Testing environment doesn't look remotely like production

The problems

- Hardware: underpowered, miss-configured, shared deployments, etc…
- Data: too few records, non-representative, etc…
- Network: reduced bandwidth, miss-configured , etc…
- Load generation: load not distributed, IP spoofing, etc…

Some solutions

- Hardware, network, and configuration comparison spreadsheets
- Dumps of data from production or processes that generate data
- Setup a load generation configuration standard for your organization
- Bring in someone to audit your load generation configuration

Resources

Articles

- Barber, Scott. “Beyond Performance Testing.” (14-part series.) PerfTestPlus. 2006. <http://www.perftestplus.com/resources/BPT1.pdf>
- Kelly, Michael. “Testing for performance.” (3-part series.) SearchSoftwareQuality. 8 Feb. 2008. <http://searchsoftwarequality.techtarget.com/tip/0,289483,sid92_gci1298371,00.html

Presentations

- Stefanov, Stoyan. “The performance business pitch.” phpied.com. 2009. <http://www.phpied.com/the-performance-business-pitch/>

Books

- Meier, J.D., Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. Performance Testing Guidance for Web Applications. Microsoft, 2007.
- Menasce, Daniel A., Virgilio A. F. Almeida. Capacity Planning for Web Services. Prentice Hall, 2002.
- Souders, Steve. High Performance Web Sites: 14 Steps to Faster-Loading Web Sites. O’Reilly, 2007.
- Souders, Steve. Even Faster Web Sites: Performance Best Practices for Web Developers. O’Reilly, 2009.
- Smith, Connie U. Performance Engineering of Software Systems. Addison-Wesley, 1990.

Thank You

Questions?

Michael Kelly

Email: mike@michaeldkelly.com
Twitter: michael_d_kelly
LinkedIn: michaeldkelly

On SearchSoftwareQuality

Tips and Articles: http://searchsoftwarequality.techtarget.com/tips/
Expert Advice: http://searchsoftwarequality.techtarget.com/expert/Knowledgebase/
Software Quality Insights Blog: http://itknowledgeexchange.techtarget.com/software-quality/