Software Performance Engineering Failure Modes and Effects Analysis

Presented by Kevin Mobley
SPE FMEA Agenda

- Definition
- Problem solved
- SPE FMEA anatomy
  - Risk
  - Likelihood of occurrence
  - Voice of the customer
  - Detection
- Life cycle
Modeling

**Problem/Goal/Scope**

**Problem** – In order to simulate the day to day and peak day operations of an application, an accurate identification of the most critical business process is required.

**Goal** – Define the top 20% business processes that create 80% of the server requests, as well as outlier business processes that are severe performance risks.

**In Scope** – Definition of business processes, Anti-Pattern analysis, business frequency, willingness to wait and detection review.

**Resource Plan**

- Business Analyst
- Development Architect and Leads
- Performance Architects

**Milestones**

- Gather the business processes used currently by client(s)
- Rank anti-patterns for each business process
- Analyze historical business process usage
- Complete willingness to wait with client (implementations only)
- Complete detection section
- Review business process selection with stakeholders
- Sign off on business processes

**Business Case**

- Ensures performance engineering focuses on the most critical user activity that will impact the application and system performance.
- Sources of Financial Benefits: Ensures investment in performance delivers the operational readiness of application.
- First Year Annualized Benefits: Establishes a traceable and defensible methodology of how and why business processes were and were not included in the performance engineering effort. The confidence in this BP set will be tested during the first year of production.
SPE FMEA is a structured approach

- Identifies how software can fail
- Estimates risk
- Evaluates current control plan
- Prioritizes actions
SPE FMEA Anatomy

- Risk: Anti-Pattern Assessment
- Frequency: Likelihood of Occurrence
- W to W: Voice of the Customer
- Detection: Control Plan

<table>
<thead>
<tr>
<th>Business process</th>
<th>Risk</th>
<th>Frequency</th>
<th>Willingness to wait</th>
<th>Detection</th>
<th>Risk Priority Number (RPN)</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>Customization to Core Product</th>
<th>Parse Cycles</th>
<th>% of Returned Data Used</th>
<th>RT - Browser to Web Server</th>
<th>DB Inserts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - used out of box</td>
<td>1 - no parse cycles</td>
<td>1 - 100%</td>
<td>1 - 1 or less round trips</td>
<td>1 - 1 or less inserts</td>
</tr>
<tr>
<td>2 - minor</td>
<td>3 - one parse cycle</td>
<td>2 - 90%</td>
<td>2 - 2 round trips</td>
<td>2 - 2 inserts</td>
</tr>
<tr>
<td>5 - modest</td>
<td>10 - 2 or more parse cycles</td>
<td>3 - 80%</td>
<td>5 - 3 round trips</td>
<td>3 - 3 inserts</td>
</tr>
<tr>
<td>9 - substantial</td>
<td>5 - 70%</td>
<td></td>
<td>8 - 4 round trips</td>
<td>4 - 4 inserts</td>
</tr>
<tr>
<td>10 - new functionality</td>
<td>7 - 60%</td>
<td></td>
<td>10 - 5 or more round trips</td>
<td>5 - 5 inserts</td>
</tr>
<tr>
<td></td>
<td>8 - 50%</td>
<td></td>
<td></td>
<td>6 - 6 inserts</td>
</tr>
<tr>
<td></td>
<td>9 - 40%</td>
<td></td>
<td></td>
<td>7 - 7 inserts</td>
</tr>
<tr>
<td></td>
<td>10 - 30% or less</td>
<td></td>
<td></td>
<td>8 - 8 inserts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9 - 9 inserts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 - 10 or more inserts</td>
</tr>
</tbody>
</table>
### SPE FMEA – Risk Section

### TP Anti-Patterns

<table>
<thead>
<tr>
<th>Content/Message Size %</th>
<th>Sort Tier</th>
<th>Debug Configuration</th>
<th>RT ES to DB</th>
<th>DB Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 95 - 100%</td>
<td>1 - no sort</td>
<td>1 - full debug admin</td>
<td>1 - 5 or less round trips</td>
<td>1 - 1 or less reads</td>
</tr>
<tr>
<td>2 - 90 - 95%</td>
<td>2 - client sort</td>
<td>3 - limited debug admin</td>
<td>3 - 10 round trips</td>
<td>2 - up to 10 reads</td>
</tr>
<tr>
<td>3 - 85 - 90%</td>
<td>3 - database sort</td>
<td>8 - no debug admin</td>
<td>7 - 15 round trips</td>
<td>3 - up to 20 reads</td>
</tr>
<tr>
<td>4 - 80 - 85%</td>
<td>6 - application layer sort</td>
<td>10 - no debug data</td>
<td>10 - 20 or more round trips</td>
<td>4 - up to 30 reads</td>
</tr>
<tr>
<td>5 - 70 - 80%</td>
<td>10 - web server sort</td>
<td></td>
<td></td>
<td>5 - up to 50 reads</td>
</tr>
<tr>
<td>6 - 60 - 70%</td>
<td></td>
<td></td>
<td></td>
<td>7 - up to 100 reads</td>
</tr>
<tr>
<td>7 - 50 - 60%</td>
<td></td>
<td></td>
<td></td>
<td>10 - up to 500 reads</td>
</tr>
<tr>
<td>8 - 40 - 50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 - 30 - 40%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 - 30% or less</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## SPE FMEA – Risk Section

### TP Anti-Patterns

<table>
<thead>
<tr>
<th>Message Size</th>
<th>Cache Hit Ratio</th>
<th>RT ES to Host</th>
<th>DB Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - small -- &lt; = 5K</td>
<td>1 - no cache used</td>
<td>1 - 3 or less round trips</td>
<td>1 - 1 or less updates</td>
</tr>
<tr>
<td>3 - average -- &lt; = 20k</td>
<td>1 - 100%</td>
<td>5 - 5 or less round trips</td>
<td>2 - 2 updates</td>
</tr>
<tr>
<td>5 - high -- &gt; 20k and &lt;= 50K</td>
<td>2 - 90%</td>
<td>10 - 6 or more round trips</td>
<td>3 - 3 updates</td>
</tr>
<tr>
<td>8 - very high -- &gt; 50k and &lt;= 100K</td>
<td>3 - 80%</td>
<td></td>
<td>4 - 4 updates</td>
</tr>
<tr>
<td>10 - extreme -- &gt; 100k</td>
<td>4 - 70%</td>
<td></td>
<td>5 - 5 updates</td>
</tr>
<tr>
<td></td>
<td>5 - 60%</td>
<td></td>
<td>6 - 6 updates</td>
</tr>
<tr>
<td></td>
<td>6 - 50%</td>
<td></td>
<td>7 - 7 updates</td>
</tr>
<tr>
<td></td>
<td>7 - 40%</td>
<td></td>
<td>8 - 8 updates</td>
</tr>
<tr>
<td></td>
<td>8 - 30%</td>
<td></td>
<td>9 - 9 updates</td>
</tr>
<tr>
<td></td>
<td>9 - 20%</td>
<td></td>
<td>10 - 10 updates</td>
</tr>
<tr>
<td></td>
<td>10 - 10% or less</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## SPE FMEA – Risk Section

<table>
<thead>
<tr>
<th>Bandwidth Impacts</th>
<th>XSL Transformation</th>
<th>RT WS to ES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 0% empty XML tags</td>
<td>1 - 0 transformations</td>
<td>1 - 5 or less round trips</td>
</tr>
<tr>
<td>2 - 5% empty XML tags</td>
<td>3 - 1 transformation</td>
<td>3 - 10 round trips</td>
</tr>
<tr>
<td>5 - 10% Empty XML tags</td>
<td>10 - 2 or more transformations</td>
<td>7 - 15 round trips</td>
</tr>
<tr>
<td>8 - 15% empty XML tags</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 - 20% or greater empty XML tags</td>
<td></td>
<td>10 - 20 or more round trips</td>
</tr>
</tbody>
</table>
SPE FMEA Life Cycle
SPE FMEA During Design

• Guessing is okay
• Review code and interview app developers
• Survey clients and product management
• Advantages
  – Redesign with smaller impact
  – Localize the concept of performance
  – Establish early
• Challenges
  – Lower confidence in SPE FMEA
  – High resistance
  – Not standard in software development
Risk Example

- Message size is 24 kb – Risk Rank is 5
- Content to Message Ratio is 11.2% -- Risk Rank is 10
- A snippet of XML Message
  
  ```xml
  <InterestDueFromClosingAmount>0.00</InterestDueFromClosingAmount>
  <DailySimpleInterestOverdueInterestAmount>0.00</DailySimpleInterestOverdueInterestAmount>
  <PiggybackPrincipalBalance>0.00</PiggybackPrincipalBalance>
  <BuydownSubsidyRemainingBalance>0.00</BuydownSubsidyRemainingBalance>
  <AccruedLateChargeBalance>0.00</AccruedLateChargeBalance>
  <RuleOf78sUnearnedInterestUnpaidBalance>0.00</RuleOf78sUnearnedInterestUnpaidBalance>
  <RuleOf78sOriginalUnearnedInterestDueBalance>0.00</RuleOf78sOriginalUnearnedInterestDueBalance>
  ```
SPE FMEA During Development

• Measure data values
• Code changes
• Developer Anti-Pattern spec sheets
• Advantages
  – Analysis of software
  – More comprehensive with each release
• Challenges
  – Architecture changes
  – Software budget
Risk Example

We wrote a bridge parse tool

Message Name: SMValidateUser (REQUEST)

Total Tags: 5
Total Empty Tags: 0

Content/Message Size Ratio: 15.13%
Empty Tag/Total Tag Ratio: 0.00%
SPE FMEA During Performance Testing and Optimization

• Anti-patterns and performance correlation
• Key driver
• Advantages
  – High risk areas
  – Paper trail
  – Performance monitoring review
  – Performance and anti-patterns analysis
• Challenges
  – Early generations less accurate
  – Anti-pattern weights
Thank You
Calibrate Workload Model
for
Accurate Performance Testing

Benjamin Mao
March 14, 2016
Abstract

Web performance testing, in short, is to simulate real user workloads and web system behaviors in test environment. With rapid evolvement, the real web user access patterns have been changed dramatically in today’s public facing websites. It has been providing many challenges to the performance testing process. Performance engineers often spent significant time to build performance test workload model with their best effort and knowledge. The reality is the workload models of performance testing often missed their real-world workload targets due to missing “significant non critical” web usages and lack of accurate workload model validation process. The accuracy of workload models is directly reflecting the understanding and simulation of prod real web app usage pattern in performance testing environment. Without an accurate workload model, not only performance test results will be inaccurate, but also many following tasks, performance bottleneck recreating, server performance tuning, capacity planning, or application framework validation will be at wrong targets. From this webinar, I will walk you through a practical way of building accurate performance workload models that matches to the targeted prod workload much closer comparing to the workload model process in current performance testing practice. The accurate workload models and followed performance test results will bring high confidence on web app performance of new releases to project and upper management teams.
Benjamin Mao is currently a Performance Architect at Randstad USA. He has been guiding large clients on the accurate performance testing methodology and best practice for accurate performance assessment of web apps of all sizes. Benjamin Mao has 20 years of experiences in IT where he performed various roles of developer, software engineer, lead, architect in web app development and performance tuning. He had developed an accurate performance testing methodology and practice to fill out a gap in performance testing practice when its moving from mainframe/client-server/intranet to public facing internet world. Besides being a web performance thinker, he enjoys poem writing, custom home building, singing, dancing.
Agenda

- Performance testing as a computer simulation to real world
- Review current user access workload modelling practice of PTLC
- Missing a step in current PTLC of SDLC or agile
- Accurate user access pattern and workload analysis
- Workload model validation iteration
- Acceptance range of accuracy
- From server performance testing to end user performance assessment
- Q&A
Performance testing as a simulation

Simulate real user behaviors in performance testing environment

- Simulate workload volume of real user traffic in full/scale-down
- Simulate real user access patterns in a statistical significant way
  - User group distribution
  - Page access distribution
- User access pattern validation - load test vs target
  - Iteration process
  - Acceptance range of accuracy
Poll Question #1

Have you ever reproduced a real prod performance issues in test environment?

1) Not really but want to
2) Can reproduce some
3) Do not care
Modelling real user access workload

Real user access traffic pattern changed for a public facing web sites

- Key transactions
- User group distribution
- Concurrent users

Performance workload also affected by

- Seasons
- Events - promotions, Ads, campaigns, coupons, social medias
- Robots
Missing from workload modeling of current PTLC

Significant but non-critical web usage

- Bounce traffic
  - Search engines, wrong access
- Bot traffic
- Media traffic
  - ads, promotions, coupons, social medias

Workload model validation

- Validate the performance workload into acceptance of accuracy
- Add compensating synthetic traffic
Calibrating performance workload model

Calibration process

- Workload calibration - the first phase of performance testing
- Calibrating iterations to make sure perf workload model close to target
- User 80/20 rule to evaluate the load test workload model

Perf test workload evaluation

- Graphic comparison
- Least squares
- Confidence interval
Source of inaccurate workload modeling

- Scripting load test scenarios in simplest and shortest ways
- Length of scenarios are much different
- Simulate arrival rates for different user group
- Flat user access transaction/operation mixes
- Efforts being put in model analysis not validation
Challenges to theoretical perf workload modeling

- User access paths do not fit into stochastic form well
  - Ads, promotions, champions, robots
- Prod operational environments are not ideal or unlimited
  - limited capacity
  - caching, CDN
- Model normal peak cases not extreme cases where top leaders are very interested
  - very bad end user experience,
  - system failure
  - capacity budget
- Many assumptions limited model coverage to many extreme cases
  - normal distributions
  - stochastic form
Acceptance range of accuracy

Accuracy - closeness of the agreement between the result of a measurement and a true value (VIM)

- Quantitative comparison - 80/20 Pareto rule
- Avoid systematic errors
- Minimize random errors
- When to stop the load test calibration process
  - Meet pre-determined range of accuracy
  - Calibration process timeframe expired
Benefits of accurate performance testing

- Reproduce performance issues in perf test environment
  - Root cause analysis on web site crashes, bot traffic detection
- Server performance tuning
- Accurate capacity planning
- Application framework validation
- Accurate end user performance experience assessment
- Marketing/champion effective analysis
- Top user access pattern analysis
Adoption of accurate performance testing practice

Pros

● Worth the efforts
● March less performance issues in prod
● Benefit downstream apps on performance assessment and capacity planning
● Quick root cause analysis on prod performance issues
● Work well with agile development

Cons

● Need extra efforts and time to make it right at the beginning
● Need extra skills and patience for performance engineers
● Need support from upper management teams
Thank you for attending!

Benjamin Mao
Performance Architect at Randstad USA
maob92@gmail.com
Benjamin.Mao@nike.com