Software Test & Performance

April 17-19, 2007

Applying Software Performance Engineering to Java EE Application Design, Development and Deployment

William Louth, JINSPIRED

What is SPE?

- **Pro-active** performance assurance
- **Process** spanning software life cycle
- **Quantitative** method for software design
- **Details** effective data gathering and performance measurement **techniques**
- **Describes** performance-oriented design **patterns**

Importance of Performance

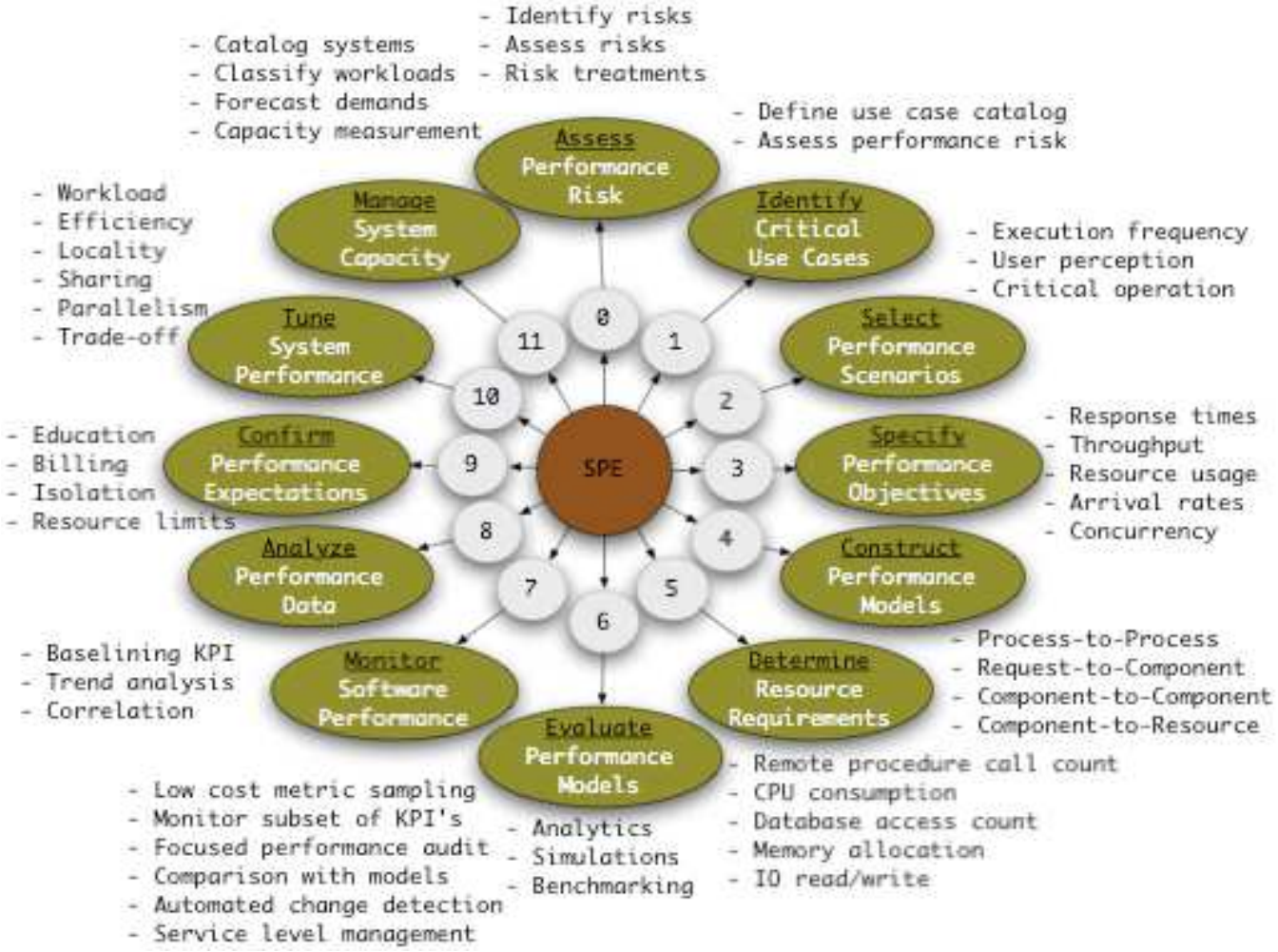
There are many negative consequences associated with poor performance:

- **Damaged customer relations**
- **Lost income**
- **Reduced competitiveness**
- **Failed projects**

Benefits of SPE?

- **Development cost reductions:**
  - **Decreased need for tuning and/or redesign**
- **Deployment cost reductions:**
  - **Less expensive hardware**
- **Management cost reductions:**
  - **Faster problem diagnosis**
  - **Efficient usage of available resources**
  - **Reduced operational costs**

What is SPE?

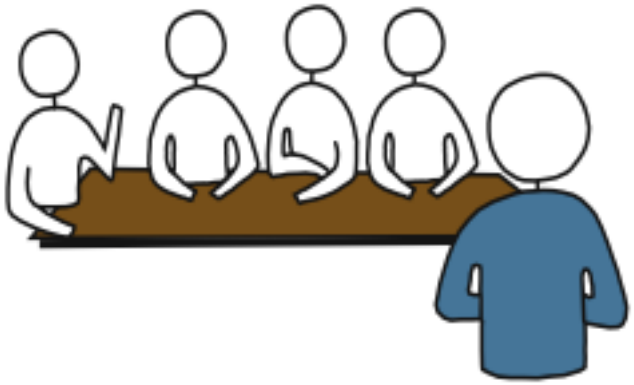


Fundamental Activities

- **Assess** Risk
- **Identify** Use Cases
- **Select** Scenarios
- **Specify** Objectives
- **Construct** Models
- **Determine** Resource Needs
- **Evaluate** Models
- **Monitor** Software
- **Analyze** Data
- **Confirm** Expected Results
- **Tune** Performance
- **Manage** Capacity

Assess Risk

- **Identify** possible risks
- **Assess** impact severity and probability
- **Prescribe** risk reduction treatments



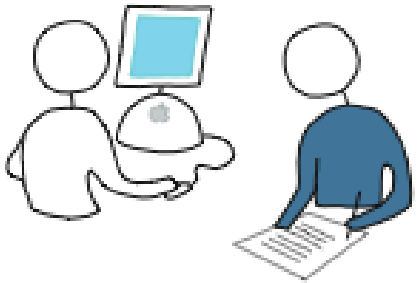
Identify Use Cases

- **Define** use case catalogue
- **Assess** performance risk

Select Scenarios

- **Frequency of execution**
- **Critical to overall performance perception**
- **Workload characteristics**
- **Performance constraints**

Demonstration



Software Execution Models

Processing Steps

Specify Objectives

- **Good** performance objectives must be:
  - Realistic
  - Reasonable
  - Quantifiable
  - Measurable
- Performance objectives must **balance** *individual*, *community* and *enterprise* goals

Construct Models

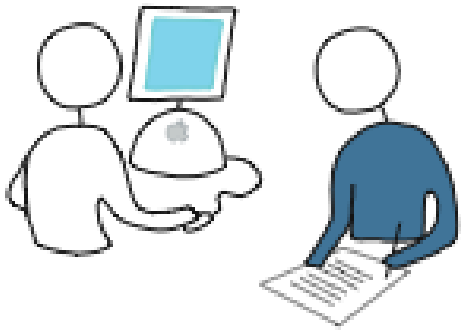
- **Determine** processing steps:
  - Process-to-Process: WebServer..EJBContainer
  - Request-to-Component: HTTP..Servlet
  - Component-to-Component: EJB..EJB
  - Component-to-Resource: EJB..JMS, EJB..JCA
- **Classify** interaction styles
  - Synchronous or Asynchronous: EJB, MDB
  - Local or Remote: EJB, CORBA

Resource Needs

Determine resource requirements for each scenario and step:

- Remote procedure call count: RMI, JMS
- CPU consumption: Process and Thread
- Database access count: SQL/JDBC
- Memory allocation: Object Sizes (GC!)
- IO read/write bytes: JMS, JDBC, RMI, SOAP
- Etc...

Demonstration

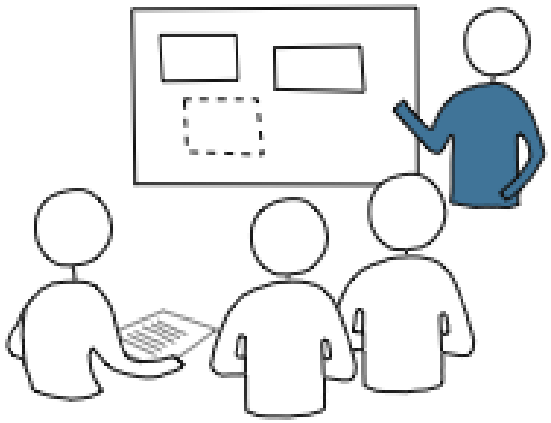


Software Execution Models

System Independent Measurements

Evaluate Models

- **Techniques**
  - Rule-of-thumb
  - Analytic
  - Simulation
  - Benchmarking

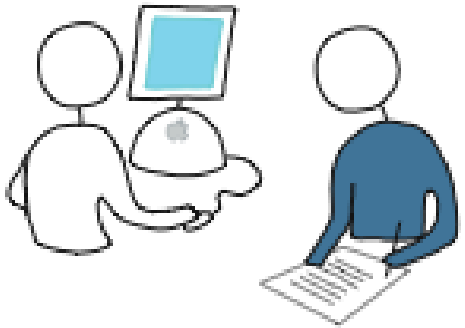


- **Variables**
  - Deployment topology: Clustering & Locality
  - Peak volumes: HTTP request, RPC calls
  - Arrival rates: Thread and JDBC Pools

Monitor Software

- **Levels** of monitoring:
  - **1** Low cost metric sampling
  - **2** Targeted performance tracking
  - **3** Focused performance testing
- **Data** sources: OS,Container,JVM,Log,DB
- **Instrumentation: JMX, AOP, JVMPI/JVMTI**
- **Metering** of resource usage: CPU, Memory
- **Dependency** analysis: EAR, WAR, RAR

Demonstration

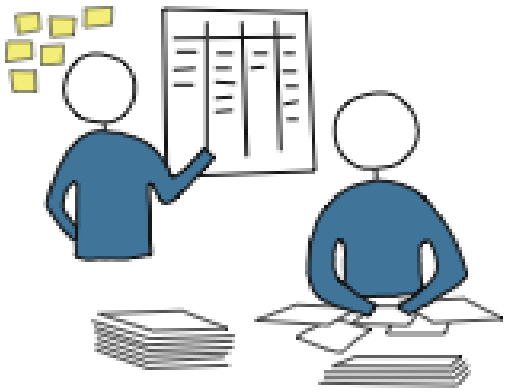


System Execution Model

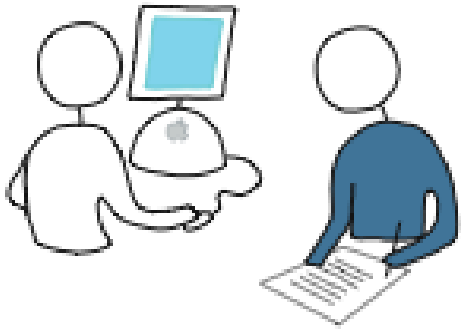
Workload and Concurrency

Analyse Data

- **Base lining** key performance indicators
- **Trend** analysis
- **Correlation**: cause and effect
- **Change** detection
- **Problem diagnostics**



Demonstration



Performance Data Analysis

Confirming Expectations

- **Users** make performance
- **Design** allows performance
- **Management** involves:
  - **Education** of user population on performance implications of particular usage patterns
  - **Billing** of resource consumption to increase awareness
  - **Partitioning** and isolation of applications and systems for improved resolution
  - **Enforcement** of resource limits

Demonstration

Customer Case Studies

Tune System Performance

- **Tuning** differs from design in that the activity occurs following a problem report and the choice of solution is limited
- **Find** the biggest bottleneck and fix it
  - Do less work
  - Improve efficiencies
  - Trade-offs
  - Revisit requirements
  - Add resources

Manage Capacity

- **Catalogue** systems: inventory
- **Classify** workloads: characteristics
- **Forecast** demands: trends
- **Measure** capacities: monitoring
- **Optimize** resources: allocation
- **Minimise** overheads: deployment