Agenda

- Software Engineering Process Models - Pat O'Sullivan (20 mins)
- Software Test Automation in a RAD World - Garreth Browne (20 Mins)
- Questions
Agenda - Software Engineering Process Models

- Definition - Software Engineering
- Measuring Software Process Maturity
- "Classic" Software Engineering Process Models
  - The Linear Sequential Process Model
  - The Prototyping Process Model
  - The Rapid Application Development Process Model
- Evolutionary Software Engineering Process Models
  - Incremental Model
  - Spiral Model
  - Component Assembly Model
- Summary
Definition - Software Engineering

- There are various definitions of the term Software Engineering
- A concise definition can be found in IEEE [1993]:
  1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, i.e. the application of engineering to software.
  2. The study of approaches as in (1)
- To solve problems in an industry setting an engineer must incorporate an engineering strategy
  - ......one that encompasses the process, methods and tools
  - This strategy is often referred to as a Software Engineering Paradigm
- A S/W Engineering process model is chosen based on:
  - the nature of the project and the application
  - the methods and tools used
  - the control and deliverables required
## Evolution of Software Engineering

- Software Systems have evolved over time
  - Development & Testing strategies have had to evolve accordingly

<table>
<thead>
<tr>
<th>Evolution of the Software Industry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Era</strong></td>
</tr>
<tr>
<td>Batch Oriented</td>
</tr>
<tr>
<td>Limited Distribution</td>
</tr>
<tr>
<td>Custom Software</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Software Process Maturity

- In recent years there has been a significant emphasis on process maturity
- The SEI has established a comprehensive model based on a set of software engineering capabilities (CMM)
  - measured as organisations reach different levels of proficiency.
- To assess current state the SEI uses an assessment questionnaire and a five point grading system:
  - Initial - Software process is unplanned or perhaps chaotic. Few processes are defined and success depends on individual efforts.
  - Repeatable - Basic processes established, repeatability is possible if projects are similar
  - Defined - Process is documented, standardised and integrated into an organisation-wide software process. Process is applied in an identical way across all projects.
  - Managed - Detailed measurements of the software process and product quality are collected, and projects are controlled using these detailed measurements.
  - Optimising - Continuous process improvement via quantitative feedback from the process
- The ability to measure and gauge an organisation's level is significant
- Process maturity is important to further improvement/optimisation
Software Engineering Process Models

- Software Engineering Process models are important in Development & Test
- Process Models are generally described as Sequential, Evolutionary or a hybrid of the two
The Linear Sequential Process Model

- The most popular process model is the Linear Sequential Process Model
  - ......also called the Classic Life Cycle or Waterfall Model
- Implies a structured and iterative approach to software development
  - begins at the initial level and progresses through analysis, design, development, quality assurance and maintenance
The Linear Sequential Process Model

- The linear sequential model is by far the oldest and most widely used.
- There are criticisms of the model:
  - Real projects rarely follow a true sequential flow - software developers can become confused when changes are introduced as projects evolve.
  - Model stipulates that all requirements be stated up front - difficult to achieve.
  - Model has difficulty in accommodating the uncertainty that exists at the start of projects.
  - A working version of the software application is generally not available until late - mistakes can be expensive to rectify.
  - Model leads to "blocking states" - where project members must wait for others to complete dependent tasks.
    - Blocking states tend to be more prevalent at the beginning and end of a project.
- While the model does have weaknesses it is far better than an ad-hoc software engineering approach!!
The Prototyping Process Model

- The Prototyping Process Model was developed to improve on shortfalls in the linear sequential model
  - begins with requirements gathering where developers and customers meet
  - the overall objectives for the software are defined
  - whatever requirements are known are identified
  - areas outlined where further definition is required

- A quick design then occurs - focusing on a representation of those aspects of the software which will be visible to the customer

- The design leads to the construction of a prototype which is then evaluated by the customer and used as a basis to refine requirements

- Iteration occurs as the prototype is tuned, at the same time enabling the developer to understand better what is required.
The Prototyping Process Model

- The prototyping paradigm also has noted weaknesses
  - Customer sees what "appears" to be a working version of the software
    - but is unaware that it is only partially developed
  - Once customer understand the need to rebuild, the opportunity to request new features arises
  - Too often developers give in to customer requests
    - sometimes difficult to say "No!"
    - makes it more difficult to complete the project on time
  - Subordinate algorithms/methodology may have been chosen to get a prototype ready quickly
    - Could quickly get "stuck" with shipping with the limitations

- Prototyping process model can be effective
  - IFF the rules of engagement are fully understood on all sides in advance
The Rapid Application Development Process Model

- RAD model is a variant of the linear sequential development process model that emphasises an extremely short development cycle.
- The model is a "high-speed" adaptation of the linear sequential model where rapid development is achieved by using a component-based construction approach.
Like the previous process models, the RAD approach has drawbacks

- For large projects it requires sufficient human resources
  - to create the right number of sub teams
- The model requires developers and customers who are committed to the rapid-fire development activities
- If commitment is lacking from key players, projects will fail
- Not all types of applications are appropriate for RAD
  - If a system cannot be properly modularised, then building the components necessary for RAD will be difficult
- Least "path-of-resistance" changes can be made early on
  - ______ which may imply poor architectural decisions later
Evolutionary Process Models

- Growing recognition that all complex systems evolve over a period of time
- Business and product requirements often change
  - Engineers need a process model that accommodates change
  - Linear sequential model is designed for straight line development
  - Prototyping model is designed to assist the customer in understanding requirements
  - Neither are designed to deliver a production system
- Evolutionary Model Theory was subsequently born!
Evolutionary Process Models

- Evolutionary nature of S/W is not considered in conventional software engineering process paradigms
- Evolutionary models subsequently evolved
  - they promote iteration and incremental evolution
- Enable S/W engineers to develop increasingly more complete versions of S/W
- A number of key evolutionary models have evolved - all are very similar
  - Incremental Model
  - Spiral Model
  - Component Assembly Model
Incremental Process Model

- Combines elements of linear sequential with iterative nature of prototyping
- Each linear sequence encapsulates the "analysis + design + code + test" stages to deliver an increment of the software
  - e.g. a graphics application developed using the incremental paradigm might provide basic drawing as well as graphic manipulation facilities in the first increment.
  - More sophisticated manipulation capabilities may be developed in the second increment.
  - Colour may be added in the third increment etc etc
- First increment aims to meet the basic requirements and is then reviewed by the customer for feedback for the subsequent increment.
- The iterative delivery is repeated until the completed product is produced.
Incremental Process Model - Weaknesses

- If key customer requirements are considered as components that must be delayed until a later increment then this is clearly bad.
  
  f the risk for fundamental architectural changes is encouraged.

- On the other hand, the model has tremendous benefit if such concerns are fully considered in the initial increments.

  f Assuming that this is done then subsequent increments can address and refine any areas of concern.
Spiral Process Model

- Couples the iterative nature of prototyping with the systematic aspects of the linear sequential model
- Provides the potential for rapid development of incremental versions of the software
- Software is developed in a series of incremental releases
  - During early iterations the incremental release might be a prototype or paper model
  - Later iterations usually engineer more complete versions of the system.
Component Assembly Process Model

- Incorporates many of the characteristics of the spiral model
- Evolutionary in nature and implies an iterative approach
- Model involves building applications from prepackaged software components
- Object technologies usually provide the technical framework for the building of such components
- Leads to large software reuse
  - Reusability provides a number of measurable benefits
- Careful:
  - If any of the components fall short in any way then the correction process may be complex
Hybrid Perspective

- You are not compelled to stick with any one Process Model
- Hybrid process model approach may offer significant advantages
  - if the project elements and dependencies are well understood
- Example
  - in a typical S/W project the UI design process could follow a RAD approach
    - and this could facilitate requirements gathering efforts in an incremental and iterative way
  - Development of the underlying data model could follow a linear sequential approach
    - as this would not have the same requirements for iterative customer feedback as the UI model
  - Specific project components could be sourced from third party component providers
    - might be seen as non-core competencies
    - e.g. Security (SSO, SSL), Encryption,
- Careful:
  - Hybrid perspective requires a fundamental competence in terms of understanding the architectural relationships between each of the project's functional and non-functional components
  - Requires a profound understanding of the various integration points and integration challenges
Summary - IBM's perspective

- Type of engineering process model used is dependent on a number of factors
  - The complexity of the project may require a more careful and planned approach
    - implying a linear sequential or prototyping process approach (as opposed to RAD)
  - The time frame to complete development and testing may be variable
    - If time is short then linear sequential may not be suitable & RAD approach may be best
  - If products are in a mature stage then an evolutionary approach may be suitable
  - The size of the team is also an important factor
    - If large, then a rapid development approach may be appropriate
    - If small, then this will not be feasible

- Key drivers for the use of process models:
  5 enablement of effective communication regarding the process
  6 facilitation of process reuse
  7 motivating process change and process evolution
  8 facilitation of process management
  9 Ongoing measurement
    - e.g. various costs/benefits to assess the implications of process changes
Implication of Process Models on Test Engineering

Joe Fitzpatrick
Program Director
Collaborative Learning

IBM Software Group
Software Testing - WHY

- Poll of Software Developers
  - To verify that the code is working

- Poll of Test Engineers
  - To verify that "things" are working the way they are supposed to

- Poll of Development Managers
  - Because we will never get to add new features in the next release, if we have to keep diverting developers to do customer fixes on this release
Software Testing - WHO

- Specification
  - Agreed with end customer (ie tested)

- Design
  - Subject to an "independent" review (ie tested)

- Code
  - In an ideal world developers write test modules before application code
  - Code Review by peer developer (ie tested)

- Traditional Test
  - Planned by test engineering function
  - Exit criteria defined by owner of the next stage
  - Execute
  - Triage
  - Iterate
Linear Sequential Process Model

- Generally has stable requirements and specification
- Facilitates
  - Test Planning
  - Test Case Generation
  - Problem Triage
- Straightforward process
- (from a testing perspective)
The Prototyping Process Model

- When is the specification written and frozen
- As requirements change - test cases change
- Effective
  - as long as the rules of engagement are fully understood on all sides in advance
- Not generally the model of choice for business logic
The Prototyping Process Model - UI

- **Functionality**
  - Can all functions be accessed

- **Usability**
  - Are key features easy to access
  - Does this conform to standards

- **Branding**
  - Visual impact, Cultural implications

- UI is often more than User Interface
  - May impose workflow
  - May provide bounds checking
RAD / Component Assembly Process Models

- Large, complex, interdependent systems

Key decisions are:
  - when to start testing
  - what **NOT** to test
  - exit criteria from one phase to next

- Build Verification Test identifies when to start

- Each component must be tested - Functional Test

- Aggregations of components must be tested - System Test

- Integrated systems must be tested - Solution Test

- Load/Performance Test
Software Testing - WHAT

- Testing produces results
  - Need predetermined courses of action based on the type of results obtained

- In different process models – results may have different impacts
  - Sequential: a problem found in test usually means a code fix is required
  - Prototyping: a problem found in test may force a complete project re-evaluation
  - Component: a problem found in test may be "switched off" and addressed later

- Classify results by likely action
  - What can be deferred
  - What merits a code change
  - What merits a design change
  - What merits a spec change

- What merits a schedule change
Summary

- **Key drivers for the use of process models:**
  1. enablement of effective communication regarding the process
  2. facilitation of process reuse
  3. support of process evolution
  4. facilitation of process management

- **Review your own engineering process**
  - Do you have one - Is it the same as it was when you last thought about it
  - Ensure everyone involved is aware of the impact

- **Review the impact of reuse**
  - Reusing code reduces standalone functional test load - but increases system test load

- **Manage the process**
  - Know what you are looking for - and what you will do if you find it
  - Know what your options are when you encounter the unknown
END

Implication of Process Models on Test Engineering

Joe Fitzpatrick
Program Director
Collaborative Learning

IBM Software Group
System Testing & Workload Test Automation

Garreth Browne

IBM Software Group
Agenda

- System Verification Testing
  - Basic Definitions
  - Describing A System
  - System Test Strategies
  - System Test in IBM
  - System Test Focus

- Workload Automation
  - Basic Definitions
  - Operational Profiles
  - Workload Development
  - Workload Execution
  - Reliability
Once upon a time….

- A system meant a product.
- Customers installed products.
- Products ran on single computers.

Today:

- Customers deploy solutions that consist of collections of products
- Products operate across many servers in a distributed environment.
- What we mean by “system” today is very different than what we meant 20 years ago

**System testing must reflect the customer experience of systems today**
Basic definitions

- System Test = System Verification Testing = SVT
- A system is:
  - A point of view.
  - A way of looking at the world.
  - Weinberg, An Introduction to General Systems Thinking
- From a testing perspective:
  - A system is a collection of hardware and software components around which we can reasonably draw a boundary. The act of drawing a boundary is an expression of a point of view.
- Questions we can ask about the system:
  - Do the components, when taken together, form a functional whole?
  - Do component interactions cause undesirable side effects?
  - Is the system resilient to changes in the environment?
  - Is the system stable over a wide range of inputs?
  - Does the system adversely impact the environment?
  - What are the properties of the system?
  - Reliability, performance, serviceability, etc.

We can infer very little about system properties from component properties.
Deployments are complex (so test environments must also be complex)
Componentisation introduces new challenges

- A family of systems exists when:
  - Customers can “mix and match” components from a product line
  - There is variation in the customer environment (directories, platforms, and so on)

- The testing challenges:
  - Component behavior may not be consistent in every system in the family
  - Component properties can “leak through” component interfaces
  - The number of possible systems explodes combinatorially
  - Autonomic systems dynamically change their topologies and configurations (can be thought of as a trajectory through the space of possible systems)
## Non-functional Attributes

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 6.0</td>
<td>Domino 5.0.12</td>
</tr>
<tr>
<td>Mozilla 1.3, SuSe 7.2+</td>
<td>Domino 6.02</td>
</tr>
<tr>
<td>Opera</td>
<td>File System</td>
</tr>
<tr>
<td></td>
<td>Lotus Workplace</td>
</tr>
<tr>
<td></td>
<td>Lotus Workplace</td>
</tr>
<tr>
<td></td>
<td>Java/J2EE</td>
</tr>
<tr>
<td></td>
<td>J2EE Version 1.3</td>
</tr>
<tr>
<td></td>
<td>JDK Version 1.3.1 SR2</td>
</tr>
<tr>
<td></td>
<td>JDK Version 1.4</td>
</tr>
<tr>
<td></td>
<td>Netegrity Policy Server 4.6</td>
</tr>
<tr>
<td></td>
<td>Netegrity Policy Server 5.5</td>
</tr>
<tr>
<td></td>
<td>RSA ClearTrust</td>
</tr>
<tr>
<td></td>
<td>Tivoli Access Manager 4.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client Operating Systems</th>
<th>Reverse Proxy Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux SuSe 7.2+</td>
<td>Tivoli WebSEAL</td>
</tr>
<tr>
<td>Mac OS9 (Learning only)</td>
<td>WebSphere Edge Server 2.0</td>
</tr>
<tr>
<td>Mobile Browsers</td>
<td>Virus Scanner</td>
</tr>
<tr>
<td>WinXP</td>
<td>Granite</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>Network Associates/MacAfee</td>
</tr>
<tr>
<td>Windows 2003</td>
<td>Symantec</td>
</tr>
<tr>
<td></td>
<td>Trend Micro</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mail Clients POP</th>
<th>LDAP Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lotus Notes 6.02 - on WinXp and Win2K</td>
<td>Domino 5.0.12 and above</td>
</tr>
<tr>
<td>MS Outlook XP/2002 -- on WinXp</td>
<td>Domino 6.0.2</td>
</tr>
<tr>
<td>MS Outlook Express 6 - on WinXp and Win2K</td>
<td>IBM Directory Server Version 4.1</td>
</tr>
<tr>
<td>WSP4.2 Internet Mail Box Portlet - on WinXp and Win2K</td>
<td>IBM Directory Server Version 5.1</td>
</tr>
<tr>
<td>WSP5.0 Internet Mail Box Portlet - on WinXp and Win2K</td>
<td>Microsoft Active Directory 2000</td>
</tr>
<tr>
<td>Mozilla 1.3 - on Linux SuSe v7.2+</td>
<td>Microsoft Active Directory 2003</td>
</tr>
<tr>
<td></td>
<td>Novell eDirectory V8.6</td>
</tr>
<tr>
<td></td>
<td>Sun ONE Directory V5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP Servers</th>
<th>Network Monitoring Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache 1.3.20</td>
<td>NetIQ Identity Manager 2.0</td>
</tr>
<tr>
<td>Apache 1.3.26</td>
<td>IBM Tivoli Access Manager 4.1</td>
</tr>
<tr>
<td>IBM HTTP</td>
<td>IBM Tivoli Monitoring 5.4</td>
</tr>
<tr>
<td>iPlanet Web Server Enterprise Edition 4.1 SP7, SP8, SP9</td>
<td>IBM Tivoli Monitoring 5.2</td>
</tr>
<tr>
<td></td>
<td>IBM Tivoli Monitoring 5.1</td>
</tr>
<tr>
<td></td>
<td>IBM Tivoli Monitoring 5.0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Firewall Software</th>
<th>Single Sign-On Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM AIX 5.2</td>
<td>Netegrity Policy Server 5.5</td>
</tr>
<tr>
<td></td>
<td>RSA ClearTrust</td>
</tr>
<tr>
<td></td>
<td>Tivoli Access Manager 4.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load Balancers</th>
<th>Database Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Edge Server 2.0</td>
<td>Cloudscape</td>
</tr>
<tr>
<td></td>
<td>(client disconnect use only)</td>
</tr>
<tr>
<td></td>
<td>DB2: Enterprise Server Edition Version 8.1</td>
</tr>
<tr>
<td></td>
<td>DB2: iSeries V5.1</td>
</tr>
<tr>
<td></td>
<td>MS SQL Server 2000 SP2 (Learning only)</td>
</tr>
<tr>
<td></td>
<td>Oracle 9i R2</td>
</tr>
</tbody>
</table>

---

**Lotus software**
Strategies for System Testing

- Orthogonal array decomposition
  - Assume that high-order interactions between components are not a common source of failure. (Picking the right set of interactions to test is the tough part.)

- Customer-facing prioritization
  - Allocate test resources (and configurations) based on expected customer usage

- Constrain deployments
  - Ensure that customers deploy what’s been tested, instead of testing what customers might deploy
This diagram depicts the testing phases from IBM's perspective

System test is just one phase of the whole product test cycle

- **Unit Test**
  - Execution and verification of all statements and paths in the individual piece of code against the specifications and design documents for that code
  - Analysis of the source code by tools which check for common defects and weaknesses in the code.

- **Function Test**
  - Verification of component functionality accessible to an intended user of the component based on technical design-requirement-specification documents for that component.

- **System Test**

- **Performance Test**
  - Confirms that the software's performance actually meet its performance objective.

- **Integration Test**
  - Identifies problems related to the use of combinations of systems, products and/or solutions that were developed separately but are being used together.
System Testing in IBM (continued)

- Translation Verification Test (TVT)
  - Reviewing translated Program Integrated Information (PII) after build or integration and compilation

- Service Test
  - The testing activity associated with APARs, defect fixes, etc.

Predefined entry criteria and exit criteria control movement from one testing phase to the next.
Focus of SVT

- System Testers must focus on:
  - Being the customer's advocate
  - Proving the product meet specifications
  - If the product can be seamlessly implemented, operated, and managed in a customer production enterprise.
Focus of SVT (continued)

- Objectives:
  
  1. **Load/Stress** - load and stress key elements of the system to and beyond the levels expected in production environments.
  
  2. **Regression** - SVT's objective is to show that 'what used to work, still works'. In regression test we verify that new functionality introduced into the system does not affect existing functions.
  
  3. **Recovery** - Our customers expect fault tolerant, robust, reliable products. SVT verifies that the product(s) are resilient to errors on the system and recovers as designed. Error inject and detailed recovery scenarios are typically part of SVT.
  
  4. **Migration** - Typically new products are introduced into an already existing IT infrastructure that supports our customers' businesses. Seamless migration from where that infrastructure is to where the customer wishes to go is an important aspect of successful deployment of new products. SVT's objective is to verify seamless migration capability into existing system infrastructures.
Focus of SVT (continued)

5. **Usability** - Although not normally a formal usability test, SVT's objective is to verify the usability of the product. System testers believe that if something is hard for the tester then it will be hard for our customer also.

6. **Serviceability** - SVT uses the tools, procedures, and diagnostic capabilities shipped with the product(s) to debug and service the problems uncovered during system test. If SVT has issues with servicing the product then, so will the customers of the product(s). SVT's mission is to ensure the product can be serviced and maintained.

7. **Functional Completeness** - As the Customer's Advocate, system test focuses on determining what is missing from the product(s) as much as determining that the product(s) performs to previously-documented specifications.

8. **Hardware and Software interaction** - Since SVT is executed in realistic environments, the interaction of HW and SW components is a focus of SVT. It is this view of total systems that provides much of the uniqueness of SVT over Function Test. The view of HW and SW interaction is pervasive throughout SVT and at least a part of the previous seven objectives listed above.
Workload Test Automation

Garreth Browne

IBM Software Group
Basic definitions

- Workload is the means by which an artificial load is generated on a system under different scenarios with the goal of flushing out issues in:
  - Reliability
  - Availability - the probability that a system is performing properly at an arbitrary time.
  - Scalability - the ability of a system to add additional users or services
- The process of exercising a system by emulating actual client/server interaction
Operational Profiles

- Operational profiling is an industry standard practice of determining which functions the customers use and tailoring tests to them.
  
  The operational profile is a quantitative characterization of how customers will use a system. The operational profile describes the expected frequency of operations and their distribution over time.

- The principle was pioneered by Dr. John Musa.

- Workload scripts are developed for test cases implied by the operational profiles.

- When used in the context of reliability, the operational profile specifies the conditions under which system reliability will be evaluated. It can also help to prioritise the development of workloads (the highest probability operations should be automated first).
Workload Development

- The workload scripts are used as a base for focusing on these three structural test types:
  
  1. **Volume** - focuses on "numbers of things". So the goal is to define test cases that involve the creation of large numbers of things, or large amounts of data, and then drive the features that might break when faced with large numbers of things.
  2. **Load** - we apply load to the system in accordance with the operational profile. Here we are not trying to break the system, but simply trying to keep the system under a long-term, sustainable load.
  3. **Stress** - attempts to break the system, either by raising the number of users involved in the load testing, or by reducing available system resources.

- Scripts constitute the 'building blocks' that are used to create the load, volume and stress suites. Volume, load and stress suites are then just a subset of the overall operational profile.
Workload Execution

- Scripts are grouped together in suites.
- Suites are developed at a component and/or portlet level. These suites may contain a number of scenarios as per the operations identified in the profile. The scenarios are weighted as per the probabilities outlined in the operational profiles.
- This concept is best illustrated by an example. Imagine this simple operational profile for the system each of which has an associated script to drive the operation.
- We can load the system or just individual components as the execution scenario dictates.
Reliability

- Reliability is the conditional probability at a given confidence level that a system will perform its intended function properly without failure.
- Software reliability is measured in terms of the failure rates in the execution time domain, not the calendar time domain. In other words failures per number of transactions - not failures per hour of system operation.
- The calendar time domain does not properly account for system load - and we know that the only system that never fails is one that never runs.
- A controlled system load needs to be imposed on the system reflective of the system’s natural busy usage pattern.
- We do not need to concentrate on writing and developing ALL possible test cases to simulate the load. Instead we need to influence the operational profile to define the required load which will allow us to make meaningful evaluations by way of data and systems analysis.
What Reliability Criteria are System Test interested in measuring?

- System Test use workloads to make reliability assessments using a scenario based approach:
  - testing components in isolation
  - testing all components with a parallel workload

- Areas that we measure include:
  - Mean time to first failure
  - Failure rate
  - Mean time between failures
  - Mean time between recoveries
  - Mean down time
  - The balance of the unreliability contribution of each subsystem
  - Instantaneous availability—the probability that the system is operational at given time
In Summary

- We have looked at System Test definitions and strategies, and IBM's take on system test. We also highlighted some system test focus areas.
- We also explored the workload automation topic. We discussed how to use operational profiles to model usage of and some aspects of the development and executions phases of workload.
- The underlying process model used in developing the product has a significant impact on the actual workload script development and execution phases.
- Workload development in a project which exploits a RAD process model is difficult. Development of the workload scripts needs a lead time. This buffer is condensed using this process model.
- It is possible to adapt to the other process models but this required an intimate knowledge of the other models and be sensitive to their workings.
Questions