Content

- Recapitulation
  - Java MOP
  - SOA
- QoS
- Functional Testing
  - Unit Testing
- Non-Functional Testing
  Load, stress, scalability, security, recovery testing
CipherOp(ASPECipher cipher, ASPEPoint point, PublicKey pKey) {
    event publicKeyLoad after(ASPECipher cipher, PublicKey pKey) :
    call(public void loadPublicKey(PublicKey) throws *Exception )
    && args(pKey) && target (cipher) {System.err.println("MOP public key called");}

    event encrypt before(ASPECipher cipher, ASPEPoint point) :
        call(public ASPEPoint encrypt(ASPEPoint) throws *Exception )
        && args(point) && target (cipher) {}
CipherDecryptionOp(ASPECipher cipher, ASPEPoint point, PrivateKey pKey) {

    event privateKeyLoad after(ASPECipher cipher, PrivateKey pKey) : call(public void 
        loadPrivateKey(PrivateKey) throws *Exception ) && args(pKey) && target (cipher) {}

    event decrypt before(ASPECipher cipher, ASPEPoint point) :   call( public ASPEPoint 
        decrypt(ASPEPoint) throws *Exception ) && args(point) && target (cipher) {}

    event clear after(ASPECipher cipher) : call(public void clear() ) && target(cipher) { }

er : (privateKeyLoad decrypt decrypt decrypt* )*

    @fail {
        System.err.println( 
            "! The sequence: load privatekey, decrypt, clear not caled in this fashion!");
        __RESET;
    }
}
R – Service-oriented architecture

- SOA is a flexible set of design principles used during the phases of systems development and integration in computing.

- A deployed SOA-based architecture will provide a loosely-integrated suite of services that can be used within multiple business domains.

- SOA defines how to integrate widely disparate applications for a Web-based environment and uses multiple implementation platforms.

- XML is commonly used for interfacing with SOA services, though this is not required.
R – SOA – How?

- SOA separates functions into distinct units, or services, over a network in order to allow users to combine and reuse them in the production of applications.
- SOA developers associate individual SOA objects by using orchestration.
SOA – Example (1)

Deploy

Presentation layer
- JSR 168 Portlets
- WebSphere Portal Server V6
- REST-style facade
  - GET http://jivarobank.mortgage
  - WebSphere Process Server V6

Business layer
- BPEL processes
- Web services
- WCC Web services
- SCA components
- WebSphere Process Server V6.0
- WebSphere Process Server V6.0
- JDBC

Data layer
- DB2 V9
- Data stage
- DB2

Manage

Security
- Tivoli Directory Server
- Tivoli Access Manager
- Tivoli Directory Integrator
- Tivoli Federated Identity

Supporting services
- WebSphere Services Registry and Repository
Quality of service – Definition

- Quality of service (QoS) is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.
- QoS refers to resource reservation control mechanisms rather than the achieved service quality.
- QoS enables you to provide better service to certain flows.
Real–time streaming multimedia applications:
- voice over IP, online games, network support sistems
- IP–TV, cellular data communication
- Videoconferencing, circuit emulation service
- Industrial control systems (used for RT control of machinery)

In these cases a required **bit rate, delay, jitter** (the deviation in or displacement of some aspect of the pulses in a high–frequency digital signal), **packet dropping probability** and/or **bit error rate** may be guaranteed
Quality of service guarantees are important if the network capacity is insufficient or if we require a fixed bit rate and are delay sensitive.


How? A network or protocol that supports QoS may agree on a traffic contract with the application software and reserve capacity in the network nodes.

Example: it can monitor the data rate and delay, and dynamically control scheduling priorities in the network nodes => the most important data gets through the network as quickly as possible.
QoS – Traffic contract

- If a service wants to use a broadband network to transport a particular kind of traffic, it must first inform the network about what kind of traffic is to be transported, and the performance requirements of that traffic (this information is a traffic contract).

- Traffic descriptor:
  - Type of service (CBR – constant bite rate, rt VBR – real time variable, nrt VBR, ABR – available, UBR – unspecified)
  - Traffic parameters (in both directions) (PCR – peak cell rate, SCR – sustainable (avg), MCR – minimum, MBS)
  - QoS parameters (in both directions) (CTD – cell transfer delay, CDV – variation, CLR – cell loss ratio)

- What is the traffic descriptor for following applications: chat, TV, e-mail, VoIP? (see example)
QoS – Attention

- QoS is sometimes used as a quality measure, with many alternative definitions, rather than referring to the **ability to reserve resources**
- Quality of service sometimes refers to the level of quality of service, i.e. the **guaranteed service quality**
- High QoS is often confused with a high level of performance or achieved service quality, for example high bit rate, low latency and low bit error probability
QoS – Technologies

- Frame relay and ATM (Asynchronous Transfer Mode) added QoS tags to the data.
- Today, Ethernet may offer QoS. In Ethernet, Virtual LANs can be used to separate different QoS levels.
- For example in fibre-to-the-home switches, typically offer several Ethernet ports connected to different VLAN:s. One VLAN may be used for Internet access (low priority), one for IP–TV (higher priority) and one for IP telephony (highest priority).
QoS – Example

- Voice, Video, and Data Transmission Requirements
QoS – Factors

- QoS can be affected by human or technical factors
- Human factors: stability of service, availability, delays, user information
- Technical factors: reliability, scalability, effectiveness, maintainability, grade of service
- Traffic problems:
  - Throughput – the bit rate can be too low for RT multimedia services
  - Dropped packets – routers might fail to deliver some packets
  - Errors – packets are corrupted (in wireless communication)
  - Latency – a long time for each packet to reach its destination
  - Jitter – packets from the source will reach the destination with different delays
  - Out-of-order delivery – packets arrive in a different order than they were sent
QoS – Financial systems

- Key criteria for Financial Systems: Responsiveness, scalability, availability, usability and security
QoS – How to obtain it?

- **In advance**: by prioritizing traffic

- **Reserving resources**: Resources are reserved at each step on the network for the call as it is set up

- **Over provisioning**: a network capacity is based on peak traffic load estimates

- **Integrated services**: reserving network resources
QoS – Solutions
QoS – Projects

- Multi Service Access Everywhere (MUSE) has aim to research and development of low cost, multi-service access network [http://www.ist-muse.org/](http://www.ist-muse.org/)

QoS – Cisco

- The **capability of a network** to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET, and IP-routed networks.

- Important is making sure that providing priority for one or more flows does not make other flows fail.

- QoS technologies provide the elemental building blocks that will be used for future business applications in campus, WAN, and service provider networks.
QoS – Cisco QoS Toolset

- It address the diverse needs of voice, video and multiple classes of data applications
- The user can effectively control bandwidth, delay, jitter, and packet loss
Untangle: can perform routing tasks or simply pass traffic as a transparent bridge
http://www.untangle.com/Routing-QoS

Enable administrators to:
- Provide the basics like NAT, DMZs, DHCP & DNS
- Get fancy with multiple NAT spaces, routing tables and configurable MTU
- Prioritize traffic with QoS
- Support SIP & IAX VoIP traffic
Zeroshell: a Linux distribution for servers and embedded devices aimed at providing the main network services a LAN requires.
MOD_QoS: implements control mechanisms that can provide different priority to different requests
Testing

- Testing is usually *incomplete* – like runtime verification
- **Test case**: finite sequence of input/output actions
- **Test suite**: finite collection of test cases
- **Test execution**: check whether output is as expected when input sequence is given to the system
- **Test oracle**: monitor checking behavior of the system

- sounds like runtime verification!
R – Test Automation

- A process of writing a computer program to do testing that would otherwise need to be done manually process
- The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process
Graphical user interface testing. A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.

Code-driven testing. The public (usually) interface to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.
Testing tools can help automate tasks such as *product installation, test data creation, GUI interaction, problem detection, defect logging*, etc.

Important points when thinking at TA:
- Platform and OS independence
- Data driven capability (Input Data, Output Data, Meta Data)
- Customizable Reporting (DB Access, crystal reports)
- Email Notifications
- Easy debugging and logging
- Version control friendly
- Extensible & Customizable
- Support distributed execution environment
- Distributed application support
R – Non Functional Software Testing

- Verifies that the software functions properly even when it receives invalid or unexpected inputs
- Example: software fault injection (fuzzy form)
- Methods:
  - **Performance testing** or **Load Testing** checks to see if the software can handle large quantities of data or users (software scalability).
  - **Usability testing** checks if the user interface is easy to use and understand.
  - **Security testing** is essential for software which processes confidential data and to prevent system intrusion by hackers.
  - **Internationalization and localization** is needed to test these aspects of software, for which a pseudo localization method can be used.
Software Performance Testing

- **Types**
  - **load testing** – can be the expected concurrent number of users on the application (database is monitored)
  - **stress testing** – is used to break the application (2 x users, extreme load) (application's robustness)
  - **endurance testing** – if the application can sustain the continuous expected load (for memory leaks)
  - **spike testing** – spiking the number of users and understanding the behavior of the application whether it will go down or will it be able to handle dramatic changes in load
R – Usability testing steps
R – Security testing

- The Process to determine that an Information System protects data and maintains functionality as intended.
- The six basic security concepts that need to be covered by security testing are:
  - **Confidentiality**,  
  - **Integrity** – information which it receives has not been altered in transit or by other than the originator of the information,  
  - **Authentication** – validity of a transmission, message, or originator,  
  - **Authorization** – determining that a requester is allowed to receive a service or perform an operation,  
  - **Availability** – Assuring information and communications services will be ready for use when expected,  
  - **Non-repudiation** – prevent the later denial that an action happened, or a communication that took place
Testing

Functional Testing
- Integration
- User Acceptance

Non-Functional Testing
- Usability, Security, Disaster Recovery

Compatibility Testing
- OS, Browser, Certification, Localization, Backward Compatibility

Regression & Automation Testing

Load Testing

Code: Maintainability, Dead Code, Standards, Performance
Functional Testing

- Testing conducted on a complete, integrated system to evaluate the system's compliance with its **specified requirements**
- Unit Testing
- Integration Testing
- Ad-Hoc Testing
- Regression Testing
- User Acceptance Testing
- Installation Testing
- Interface & Usability Testing
- System Testing
- White Box & Black Box Testing
A unit test is a **piece of code** that tests a specific functionality in the code which is tested.

In Java Unit Testing is done with **JUnit**.

JUnit is a test framework which uses annotation to identify the test methods. To write a test with JUnit:

- Annotate a method with `@org.junit.Test`
- Use a method provided by JUnit to check the expected result of the code execution versus the actual result.
package math;

public class BasicOperations {

    public int add(int x, int y){
        return x + y;
    }

    public int min(int x, int y){
        return x + y;
    }

    public int mul(int x, int y){
        return x * y;
    }

    public int div(int x, int y){
        return x / y;
    }

    /**
     * @param args
     */

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        BasicOperations bc = new BasicOperations();
        System.out.println(bc.add(3, 5));
    }
}
Unit Testing – Example (2)

```java
package test;

import static org.junit.Assert.*;

public class BasicOperationsTest {

    @Test
    public void testAdd() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 8, bo.add(3, 5));
    }

    @Test
    public void testMin() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 2, bo.min(5, 3));
    }

    @Test
    public void testMul() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 15, bo.mul(3, 5));
    }

    @Test
    public void testDiv() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 2, bo.div(4, 2));
    }
}
```
Unit Testing – Example (3)

```java
package test;

import static org.junit.Assert.*;

public class BasicOperationsTest {

    @Test
    public void testAdd() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 8, bo.add(3, 5));
    }

    @Test
    public void testMin() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 2, bo.min(5, 3));
    }

    @Test
    public void testMul() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 15, bo.mul(3, 5));
    }

    @Test
    public void testDiv() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 0, bo.div(4, 0));
    }
}
```
Unit Testing – Example (4)

```java
package test;

import static org.junit.Assert.*;

public class BasicOperationsTest {

    @Test
    public void testAdd() {
        BasicOperations bo = new BasicOperations();
        assertTrue("Result", 8 == bo.add(3, 5));
    }

    @Test
    public void testMin() {
        BasicOperations bo = new BasicOperations();
        assertFalse("Result", ! (3 != bo.min(5, 3))); // Corrected operator
    }

    @Test
    public void testMul() {
        BasicOperations bo = new BasicOperations();
        assertEquals("Result", 15, bo.mul(3, 5));
    }

    @Test
    public void testDiv() {
        BasicOperations bo = new BasicOperations();
        if (bo.div(4, 2) == 3)
            fail("Incorrect result!");
    }
}
```
Non-Functional Testing (1)

- What is NFT?
  - Testing of those requirements that do not relate to functionality

- NFT – The Myths
  - Non Functional testing can only be performed after functional testing
  - Only highly technical testers can perform this type of testing
  - The Return on Investment is not favorable
Non-Functional Testing (2)

- **Reality**
  - Significant NF testing can be performed even before software is fully developed
  - Can iteratively develop appropriate tests
  - Can assist the functional testing
  - De-risk final testing

- **Who?**
  - Modern tools are starting to make scripting more accessible
  - Often a company will have the technical resources available to assist testers
  - With the right people, governance and financial backing everything is possible
NFT – Challenges

- Environments – cost, complexity
- Planning – juggling act between availability of suitable functionality, environment and people
- Requirements – inability to define clearly and measurably what is required
- Application – stability, lack of CM, late changes
- What is NFT?
NFT – Benefits

- Have confidence in your system
- Have confidence in your offering to your customers
- Easing of support workload
- Better planning of infrastructure
- Demonstrate conformance to legal requirements
- Keep your IT group happy!
NFT – Types

- Load and Performance Testing
- Ergonomics Testing
- Stress & Volume Testing
- Compatibility & Migration Testing
- Data Conversion Testing
- Security / Penetration Testing
- Operational Readiness Testing
- Installation Testing
NFT – Load Testing

- Performed to determine a system’s behavior under both normal and anticipated peak load conditions

- **Example**: a web site (with shopping cart) is required to support 100 concurrent users. Testing activities:
  - 25 Virtual Users (VUsers) log in, browse through items and then log off
  - 25 VUsers log in, add items to their shopping cart, check out and then log off
  - 25 VUsers log in, return items previously purchased and then log off
  - 25 VUsers just log in without any subsequent activity

Once the test has started and reached a steady state, the application is being tested at the 100 VUser load
Load Testing – Causes

- Application server(s) or software
- Database server(s)
- Network – latency, congestion, etc.
- Client–side processing
- Load balancing between multiple servers
Load Testing – Tools

- **OpenSTA** – Open source web load/stress testing application
- **HP LoadRunner** – automated performance and load testing product
- **Visual Studio 2010**
Scalability Testing

- Testing of a software application for measuring its capability to scale up or scale out:
  - Can be the user load supported
  - The number of transactions
  - The data volume etc.

- Performance, scalability and reliability are usually considered together by software quality analysts
Stress testing

- Used to determine the stability of a given system or entity
- Used to break the application (2 x users, extreme load) (we can see what is the application’s robustness)
- The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space), unusually high concurrency, or denial of service attacks
Stress Testing – Tools
Process to determine that an information system protects data and maintains functionality as intended

The six basic security concepts are: confidentiality, integrity, authentication, availability, authorization and non-repudiation
Security Testing

- **Penetration Test** – simulates an attack by a malicious party. Building on the previous stages and involves exploitation of found vulnerabilities to gain further access.
- Using this approach will result in an understanding of the ability of an attacker to gain access to confidential information, affect data integrity or availability of a service and the respective impact.
**Web application penetration testing**

- **OWASP**, the Open Web Application Security Project – an open source web application

- The **Firefox browser** has plugins designed for web application penetration testing

- **Damn vulnerable web app (DVWA)** is an open source web application

- **Foundstone's Hacme Bank** simulates a banking application. It allows input validation flaws such as SQL injection and Cross Site Scripting (XSS)
Recovery testing

- How fast and better an application is able to recover from crashes, hardware failures, forced failures and other similar problems
Recovery Testing examples (1)

- While an application is running, suddenly **restart the computer**, and afterwards check the validness of the application's data integrity.
- While an application is receiving data from a network, **unplug the connecting cable**. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection disappeared.
- **Restart the system** while a browser has a definite number of sessions. Afterwards, check that the browser is able to recover all of them.
Recovery Testing examples (2)
Requirements

- Use open source tools in order to assure the QoS of your application
- Implement services in a SOA architecture
- Functional testing: Unit testing
- Non-Functional testing: load, stress, scalability, security, recovery


SOA Example 3: http://www.infoq.com/articles/soa-healthcare


IP QoS: http://www.network24.co.uk/ip-quality-of-service/

Global Knowledge: http://globalknowledgeblog.com/technology/unified-communications/qos-11-cbWFq/


**Links (2)**

- MOD_QoS: [http://en.wikipedia.org/wiki/Mod_qos](http://en.wikipedia.org/wiki/Mod_qos)
- Implementing QoS: [http://vonage.nmhoy.net/qos.html](http://vonage.nmhoy.net/qos.html)
- OpenSTA: [http://opensta.org/](http://opensta.org/)
Bibliography

- Mishra, A: Security and Quality of Service in Ad Hoc Wireless Networks, 2008,  
  http://www.cambridge.org/gb/knowledge/isbn/item1174618/?site_locale=en_GB
- Nortel Networks: Introduction to QoS, 2003  