



Architectural Software Risk Analysis

A Software Engineering Approach to Software Assurance

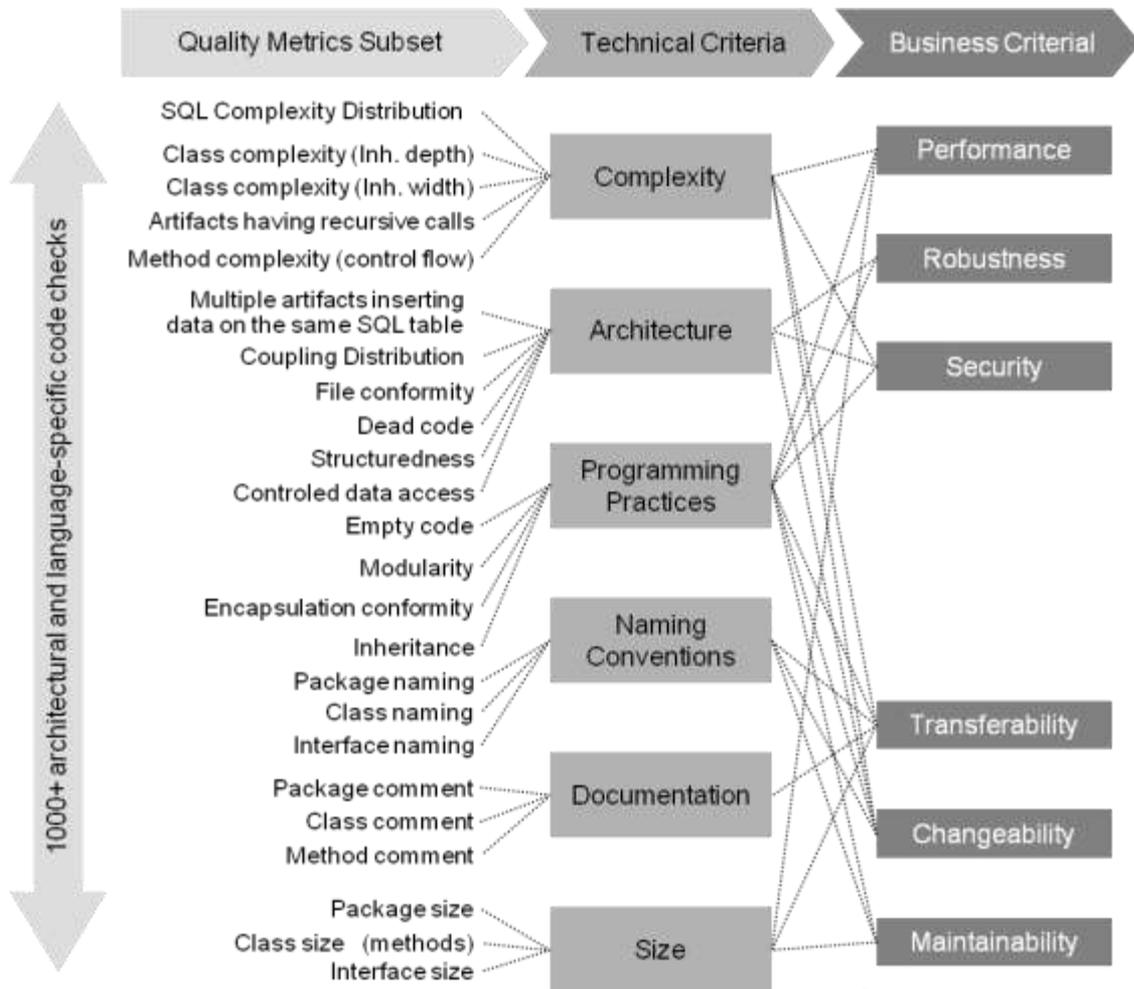
September 2012

15 years in software analysis & measurement



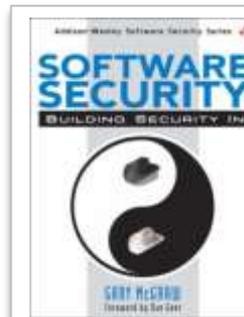
Our approach to software assurance

- Security as an important component of overall structural quality
- Structural quality must be viewed at whole software system level



“Software Assurance is 5 parts Code Quality with 2 parts Software Security.”

- John Keane, *Military Health*



“Architectural flaws account for 50% of security problems.”

- Gary McGraw

Measurement based on standards

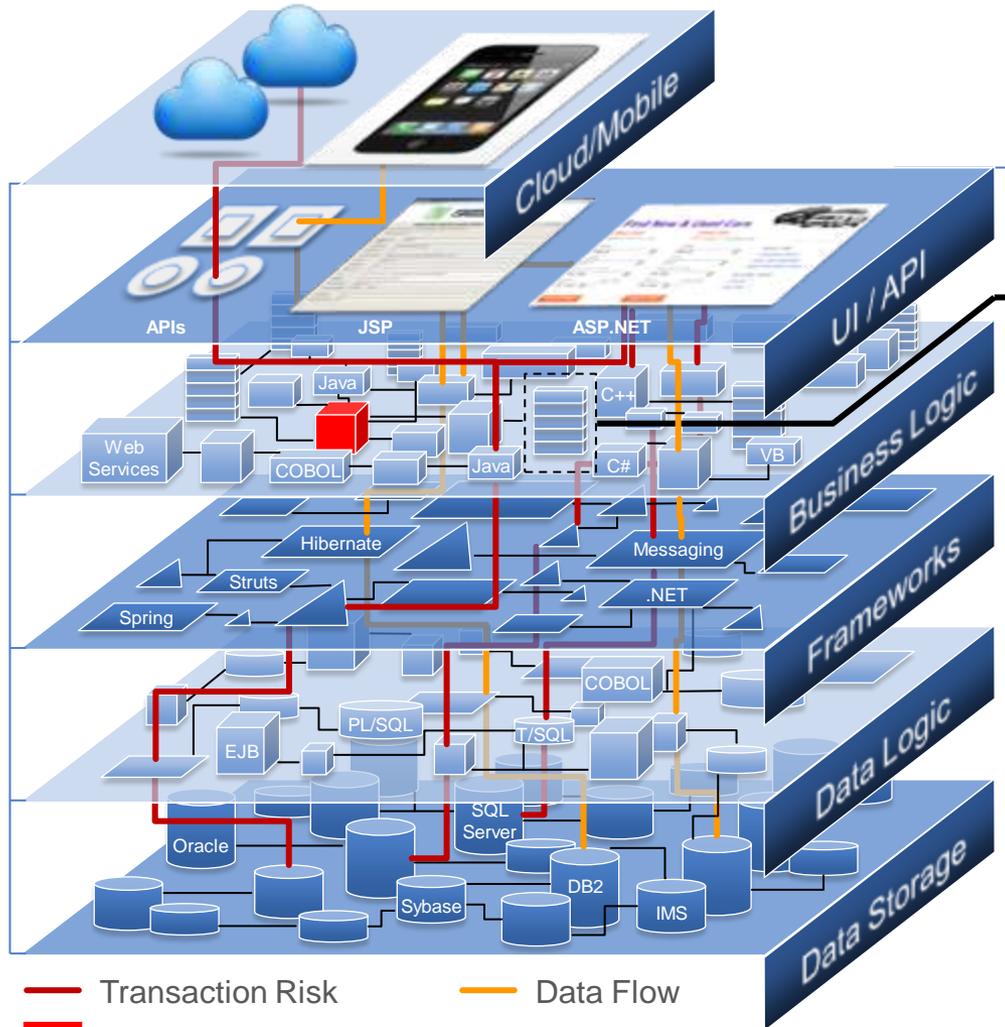
Consortium for IT Software Quality: CISQ



Characteristic	Architectural & System Level Flaws	Coding & Component Level Flaws	
RELIABILITY	Multi-layer design compliance Software manages data integrity and consistency Exception handling through transactions Class architecture compliance	Protecting state in multi-threaded environments Safe use of inheritance and polymorphism Patterns that lead to unexpected behaviors Resource bounds management, Complex code Managing allocated resources, Timeouts, Built-in remote addresses	
PERFORMANCE EFFICIENCY	Appropriate interactions with expensive and/or remote resources Data access performance and data management Memory, network and disk space management Centralized handling of client requests Use of middle tier components versus stored procedures and database functions	Compliance with Object-Oriented best practices Compliance with SQL best practices Expensive computations in loops Static connections versus connection pools Compliance with garbage collection best practices	
SECURITY	Input validation SQL injection Cross-site scripting Failure to use vetted libraries or frameworks Secure architecture design compliance	Error and exception handling Buffer overflows Missing initialization Improper locking Uncontrolled format string	Use of hard-coded credentials Broken or risky cryptographic algorithms Improper validation of array index References to released resources
MAINTAINABILITY	Strict hierarchy of calling between architectural layers Excessive horizontal layers	Tightly coupled modules Cyclomatic complexity Encapsulated data access Hard coding of literals Excessive component size	Unstructured and Duplicated code Controlled level of dynamic coding Over-parameterization of methods Commented out instructions Compliance with OO best practices

Enterprise-grade IT systems are complicated

Architecture Compliance



- Transaction Risk
- Data Flow
- Propagation Risk

1 Program Level

- Code style & layout
- Expression complexity
- Code documentation
- Class or program design
- Basic coding standards

2 Module Level

- Intra-technology architecture
- Intra-layer dependencies
- Module complexity & cohesion
- Design & structure
- Inter-program invocation
- Security Vulnerabilities

3 System Level

- | | |
|-------------------------------|------------------------------------|
| ▪ Integration quality | ▪ Function point & EFP measurement |
| ▪ Architectural compliance | ▪ Effort estimation |
| ▪ Risk propagation simulation | ▪ Data access control |
| ▪ Application security | ▪ SDK versioning |
| ▪ Resiliency checks | ▪ Calibration across technologies |
| ▪ Transaction integrity | |

Beyond static analysis – towards architecture

Static Analysis

Understanding of language syntax and grammar using source code parsing

Simulation

Analysis of some run-time behaviors to understand dynamic behaviors of applications

Dependencies

Understanding of cross-layer and cross-technology links between application components

Code Pattern Scanning

Finding patterns and anti-patterns in application control flow

Content Updater

Adjustment of analysis results to better match application advanced behaviors

Data Flow

Tracking the use of the content of variables such as user inputs along static and dynamic call stacks

Intelligent Configuration

Capability to build object sets based on object properties, links, etc. to support layers, modules, and scope definition

Architecture Checking

Identification of invalid calls and references between application architectural layers

Rules Engine

Analysis of knowledge base against quality rules, metrics and constraints to identify violations (non-compliant objects or situations)

Transaction Scoping

Identification and configuration of cross-layer and cross-technology transactions from UI down to data entities

Function Points

Estimation of Function Points functional sizing, relying on data entities and Application-wide transactions

Aggregation & Consolidation

Aggregation and calibration of results along the quality model and consolidation across applications

Simulating runtime behavior to resolve links in code

Simulation

Analysis of some run-time behaviors to understand dynamic behaviors of applications

```
...  
public void f(){  
...  
    String sMySql = "Select Title "  
...  
    sMySql = sMySql + " from Authors";  
    g(..., sMySql);
```

```
public void g(..., String sSql, ...){  
...  
    sSql = sSql + ' where Author = ' + sAuthor  
...  
    execSQL(connection, sSql);
```

quasi-runtime behavior

Consider "Select Title from Authors where Author = " as a SQL statement

Use (select) link between Java method "f()" and SQL table "Author"

Multi-tier analysis for dependencies (1/2)

Dependencies

Understanding of cross-layer and cross-technology links between application components

Hibernate mapping.dtd

```
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.castsoftware.data.entity.Address" table="ADDRESS">
<id name="addrId" type="long">
<column name="ADDR_ID" precision="10" scale="0" />
<generator class="assigned" />
</id>
<many-to-one name="street" class="com.castsoftware.data.entity.Street" fetch="select">
<column name="STR_ID" precision="10" scale="0" not-null="true" />
</many-to-one>
<property name="addrNumber" type="int">
<column name="ADDR_NUMBER" precision="5" scale="0" not-null="true" />
</property>
<property name="addrMore" type="string">
<column name="ADDR_MORE" length="50" />
</property>
<property name="addrInsertDate" type="timestamp">
```



Address.java

```
1
2 Address instance = (Address) sessionFactory.getCurrentSession()
3 .get("com.castsoftware.data.entity.Address", new Long(id));
```



Table oracle address

```
-- *****
-- * DDL to create table ADDRESS to store Developer information *
-- *****
CREATE TABLE HR_DB.ADDRESS
(ADDR_ID          LONG          NOT NULL,
STR_ID           INTEGER
ADDR_NUMBER      INTEGER      NOT NULL,
ADDR_MORE        VARCHAR2(50)
ADDR_INSERT_DATE TIMESTAMP
ADDR_INSERT_USER LONG          NOT NULL,
ADDR_LAST_UPD_DATE TIMESTAMP
ADDR_LAST_UPD_USER_ID LONG
PRIMARY KEY (ADDR_ID))
```



Create links between Java Class and Sql Table

Multi-tier analysis for dependencies (2/2)

Dependencies

Understanding of cross-layer and cross-technology links between application components

Struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config SYSTEM "struts-config_1_1.dtd">
...
<action path="/paymentMethod"
type="com.castsoftware.utils.struts.paymentmethod.ActionPaymentMethod" >
    <forward name="paymentMethod" path="Payment.jsp">
    </forward>
</action>
```

Struts

Payment.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
...
<a href="paymentMethod.perform">New credit card</a>
```

JSP
java server pages

ActionPaymentMethod.java

```
Package com.castsoftware.utils.struts.paymentmethod
...
public class ActionPaymentMethod {
```

Java

Create links between JSP page and Action mapping
Create links between Action mapping and Java class

Data flow – across distributed architecture

Data Flow

Tracking the use of the content of variables such as user inputs along static and dynamic call stacks

```
...  
public void handleRequest(WebSession s) throws ParameterNotFoundException, ValidationException  
{  
...  
String employeeld = null;  
try  
{  
    employeeld = s.getParser().getStringParameter(SQLInjection.EMPLOYEE_ID);  
    String password = s.getParser().getRawParameter(SQLInjection.PASSWORD);  
    // Attempt authentication  
    boolean authenticated = login(s, employeeld, password);  
    updateLessonStatus(s);  
...  
}
```

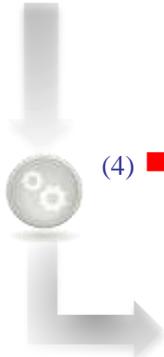
(1) →

(2) →

```
...  
public boolean login(WebSession s, String userId, String password)  
{  
...  
    String query = "SELECT * FROM employee WHERE userid = " + userId + " and password = " +  
    password + "";  
    // System.out.println("Query:" + query);  
    try  
    {  
        Statement answer_statement = WebSession.getConnection(s)  
            .createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,  
                ResultSet.CONCUR_READ_ONLY);  
        ResultSet answer_results = answer_statement.executeQuery(query);  
        if (answer_results.first())  
        {  
...  
}
```

(3) →

(4) →

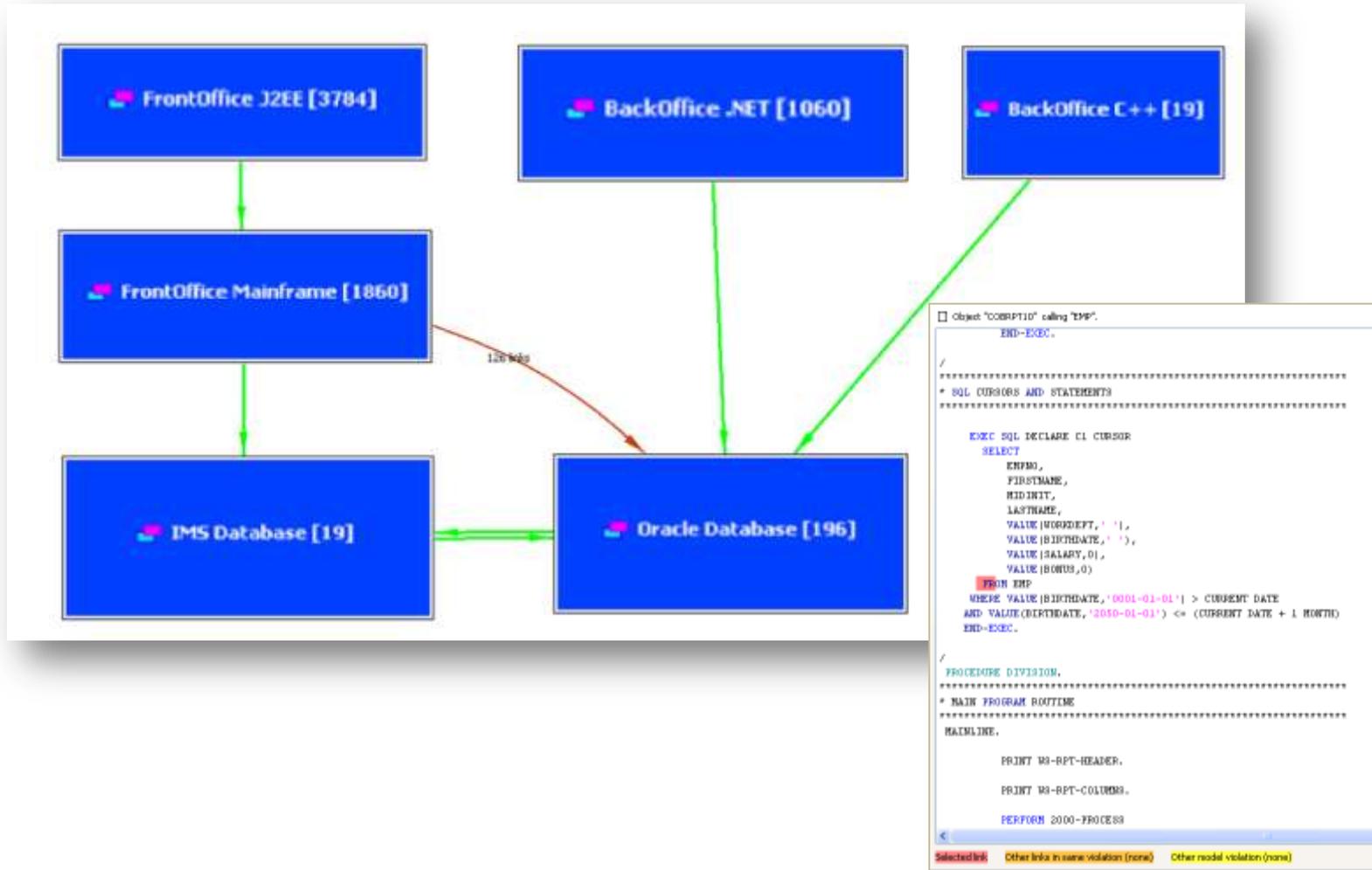


SQL injection vulnerability – CWE-89

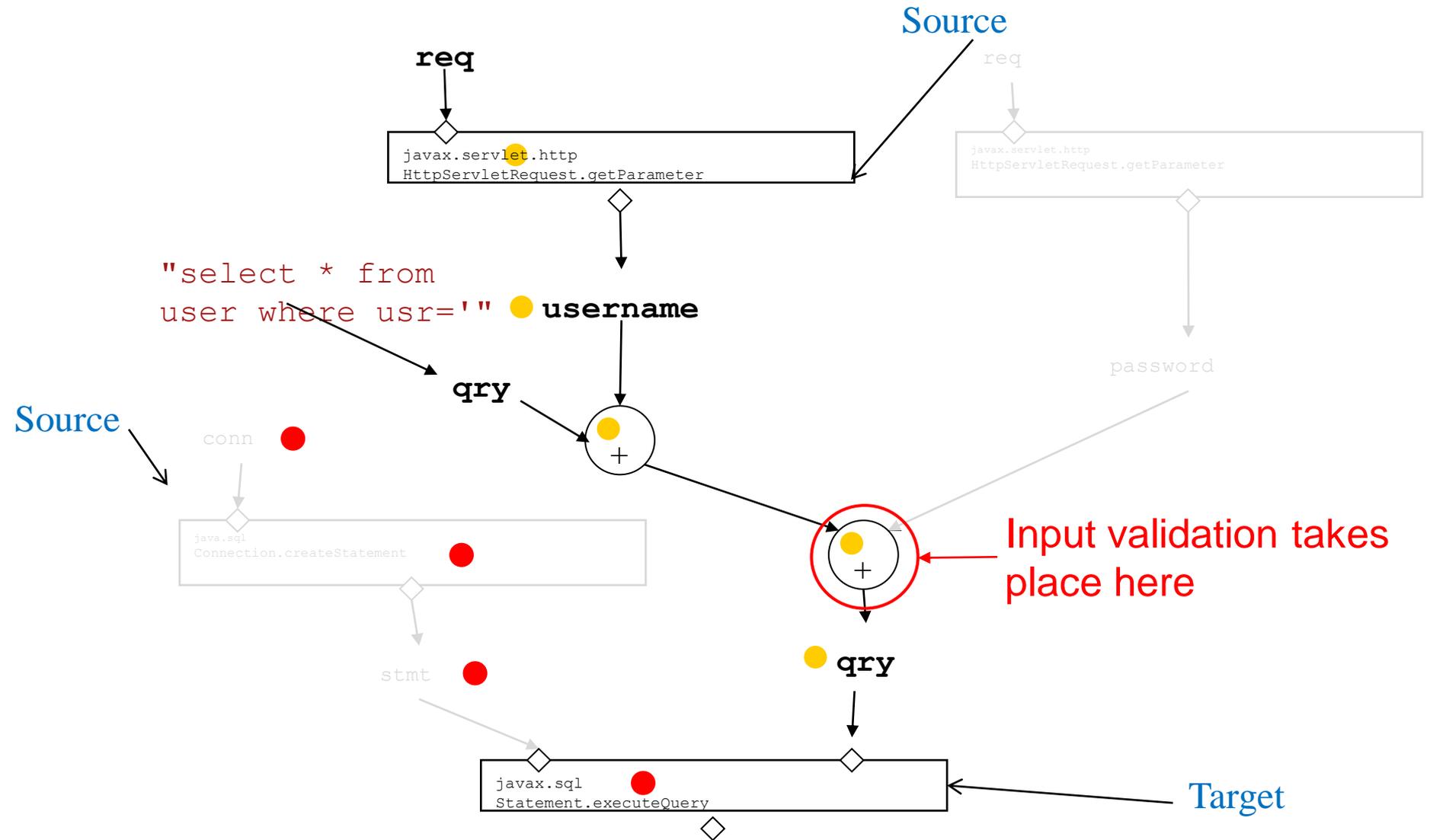
Configuring rules specific to enterprise architecture

Architecture Checking

Identification of invalid calls and references between application architectural layers

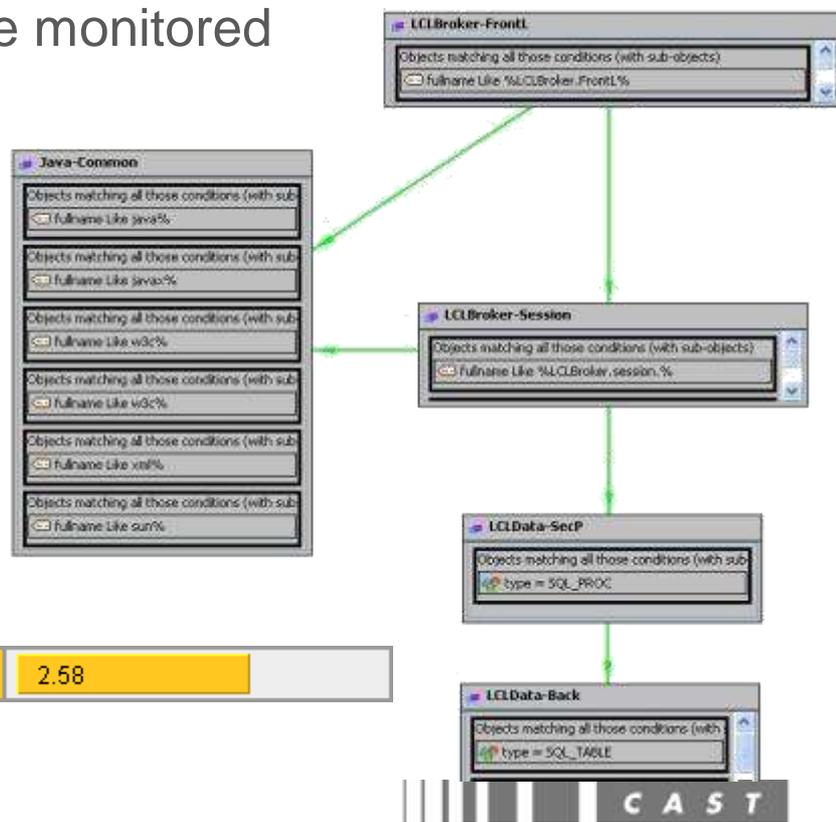


Input validation with dataflow & configuration



Security breach due to architecture misuse

- For example: banking application, for monitoring reasons, all database calls must go through stored procedures
- Investigations showed:
 - Many transactions developed offshore did not comply with secure architecture framework
 - Without automation, this could not be monitored
 - 100 UI elements (250 kloc)
 - 2000 mid-tier programs (1 mloc)
 - 250 tables, 350 kloc of PL/SQL
- Use of Architecture Checker
 - to define the desired architecture
 - To generate and enforce the appropriated quality rules



Architecture - Architecture Models Automated Checks

High Risk

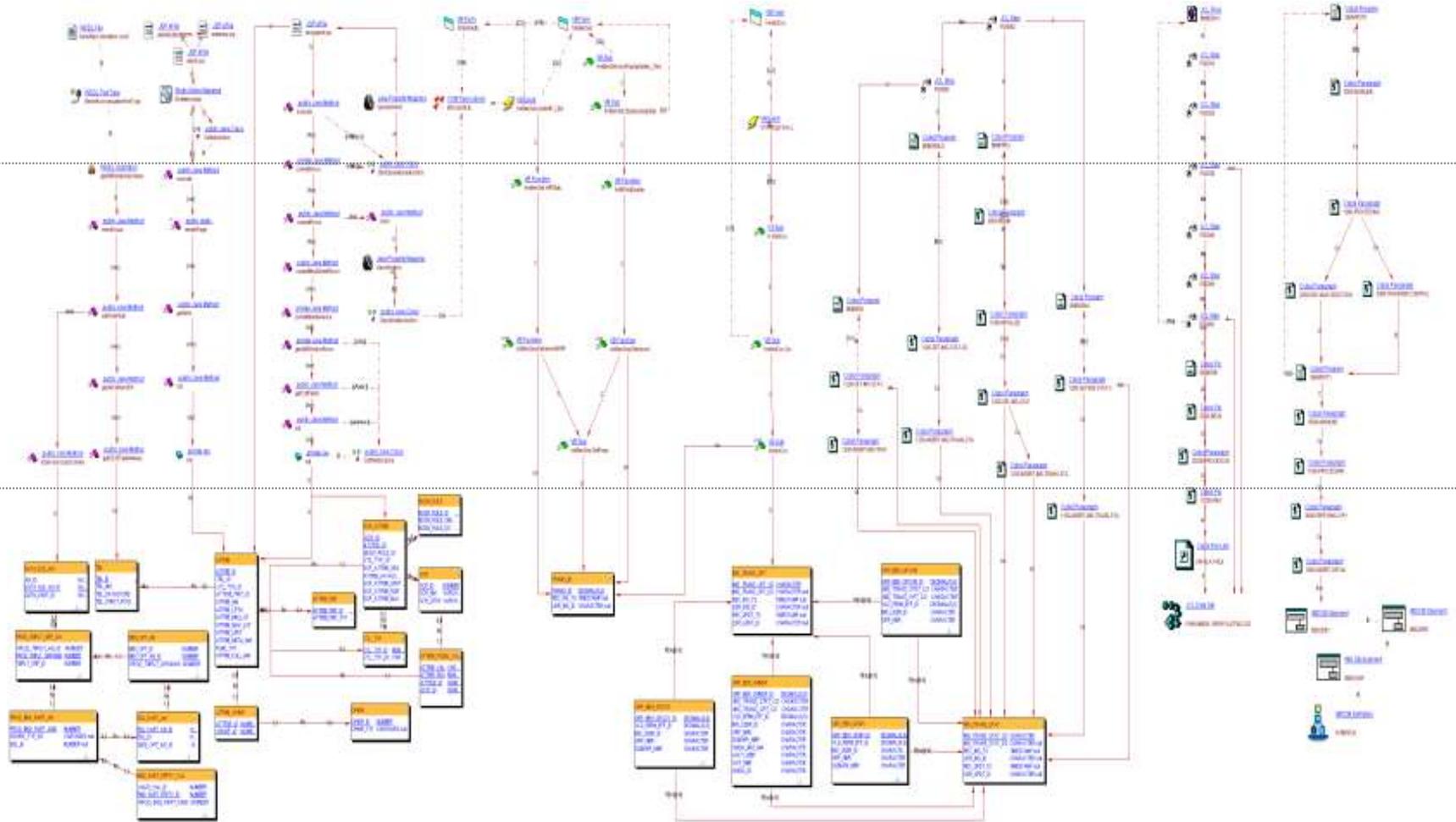
2.58

Use of blueprints for large systems

UI Layer

Logic

Data Layer



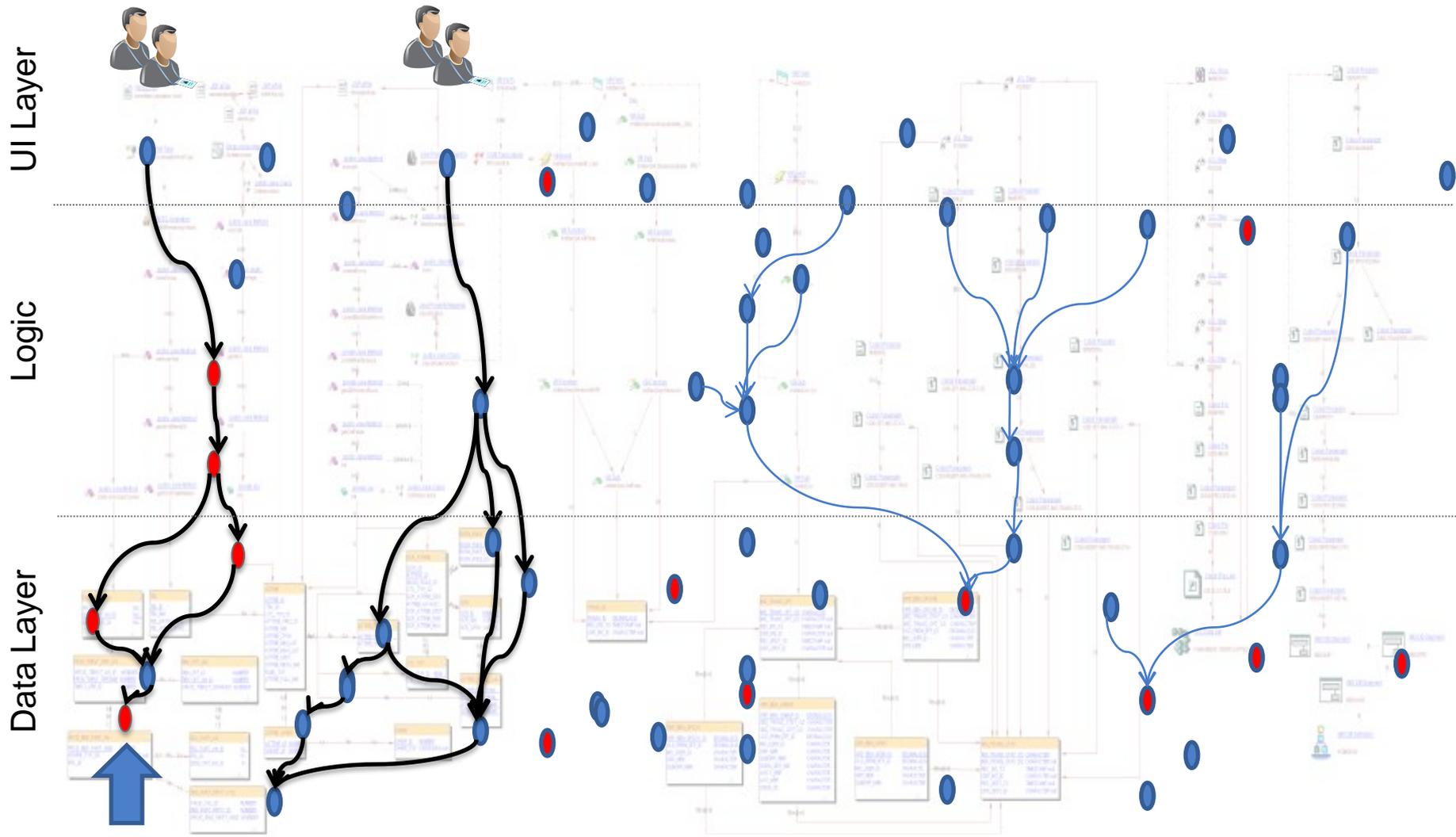
Proactive threat analysis from an architectural standpoint

Propagated Risk Index (PRI)



Violation with the largest impact on the rest of the application, regarding Robustness, Performance, or Security

Transaction Risk Index (TRI)



Transaction with largest number of Robustness, Performance or Security violations

TRI to prioritize user-facing elements

- Identify the riskiest transactions for testing, remediation
- Sum of Violation Indices (VIs) of the objects along a specific transaction: Robustness, Performance or Security

Transaction View

Transaction Details View

Transaction List

Transaction	Total VI
com.myfeko.ui.hrManagement.EmployeeDetailsAction	2134
com.myfeko.ui.hrManagement.EmployeeDetailsAction	2133
com.myfeko.ui.hrManagement.EmployeeDetailsAction	1948
com.myfeko.ui.hrManagement.EmployeeDetailsAction	1948
com.myfeko.ui.hrManagement.EmployeeDetailsAction	1672
com.myfeko.ui.hrManagement.EmployeeDetailsAction	1536
com.myfeko.ui.hrManagement.EmployeeDetailsAction	1172
com.myfeko.ui.hrManagement.EmployeeDetailsAction	997
com.myfeko.ui.hrManagement.EmployeeDetailsAction	994
com.myfeko.ui.hrManagement.EmployeeDetailsAction	924
com.myfeko.ui.hrManagement.EmployeeDetailsAction	882
com.myfeko.ui.hrManagement.EmployeeDetailsAction	832
com.myfeko.ui.hrManagement.EmployeeDetailsAction	682
com.myfeko.ui.hrManagement.EmployeeDetailsAction	632
com.myfeko.ui.hrManagement.EmployeeDetailsAction	609
com.myfeko.ui.hrManagement.EmployeeDetailsAction	609
com.myfeko.ui.hrManagement.EmployeeDetailsAction	562
com.myfeko.ui.hrManagement.EmployeeDetailsAction	562
com.myfeko.ui.hrManagement.EmployeeDetailsAction	334
com.myfeko.ui.hrManagement.EmployeeDetailsAction	334
com.myfeko.ui.hrManagement.EmployeeDetailsAction	484
com.myfeko.ui.hrManagement.EmployeeDetailsAction	398
com.myfeko.ui.hrManagement.EmployeeDetailsAction	392
com.myfeko.ui.hrManagement.EmployeeDetailsAction	345
com.myfeko.ui.hrManagement.EmployeeDetailsAction	345
com.myfeko.ui.hrManagement.EmployeeDetailsAction	332
com.myfeko.ui.hrManagement.EmployeeDetailsAction	324

Transaction Details View

Object Name	VI	Object Status
com.myfeko.ui.hrManagement.EmployeeDetailsAction	474	Added
com.myfeko.ui.hrManagement.EmployeeDetailsAction	382	Added
com.myfeko.ui.hrManagement.EmployeeDetailsAction	223	Added
com.myfeko.ui.hrManagement.EmployeeDetailsAction	26	Unchanged
com.myfeko.ui.hrManagement.EmployeeDetailsAction	16	Unchanged
com.myfeko.ui.hrManagement.EmployeeDetailsAction	74	Added
com.myfeko.ui.hrManagement.EmployeeDetailsAction	6	Unchanged
com.myfeko.ui.hrManagement.EmployeeDetailsAction	6	Added
com.myfeko.ui.hrManagement.EmployeeDetailsAction	6	Added

Securing multi-tier IT applications – example

Missing Error Handling Block Across All Layers

Object type FLEX Function
Object path T:\ONDEP\Flex\src\Services\MainServices.as

User Interface - Flex

```
public static function DisplayMessage(strMessage:String):void
{
    //Show friendly message
    Alert.show(strDefaultMessage);

    //Call service to save real error
    try
    {
        var objService:RemoteObject = MainServices.GetObjService();
        objService.SVR_ERROR_LOG(currentUserId, strMessage);
    }
    catch (err:Error){}
}
```

Business Logic – C# .NET

Object full name ONDEP.BLL.Resource.MakeRollBack
Object type C# Method
Object path t:\ondep\net\bll\resource.cs

```
}
catch{}

if (strResultTemp == string.Empty)
{
    return "-All Rollbacks OK-";
}
else
{
    return "-Error in Rollbacks: " + strResultTemp + "-";
}
```

Data Access – SQL Server (T-SQL)

Object full name LACCHLDB047\SQL2008.ER...Countryrs
Object type SQL Function

```
--Insert the first value into the table.
INSERT INTO @Tabla
SELECT country_id
FROM ER_COUNTRY
WHERE country_id = @country

--Insert the children nodes from the seed.
WHILE (@sw > 0)
BEGIN

    INSERT INTO @t
    SELECT country_id
    FROM ER_COUNTRY
    WHERE parent_country_id IN (SELECT Cou

    SELECT @sw = IsNull(count(1), 0) FROM @

    IF @sw > 0
    BEGIN

        INSERT INTO @Tabla
        SELECT country_id
        FROM @t AS A
        WHERE NOT EXISTS (SELECT 1 FROM @T

        SET @sw = @@ROWCOUNT

    END
    DELETE @t
END

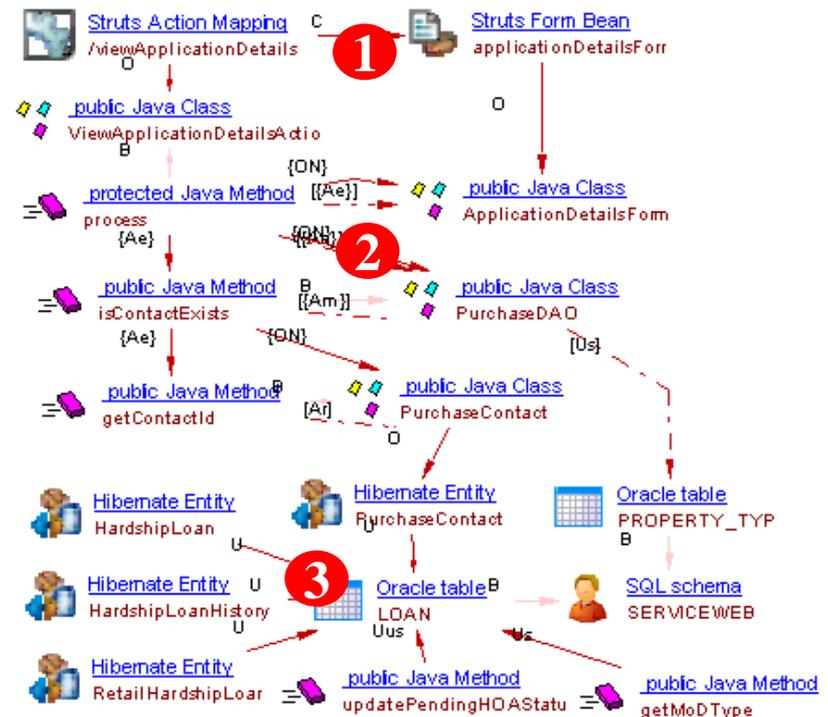
RETURN
END
```

Securing multi-tier IT applications – another example

Multiple violations across the same transaction make end-user facing applications more vulnerable

Child Metric Name	Child Metric Status
Programming Practices - Unexpected Behavior	Very High Risk
Programming Practices - Error and Exception Handling	Moderate Risk
Secure Coding - Input Validation	Very High Risk
Architecture - Multi-Layers and Data Access	Very High Risk

- 1** Input validation - 4 form fields without validator in user interface
- 2** Architecture design - action class talking to data access object bypassing business layer
- 3** Database access security - multiple artifacts accessing and modifying data on the LOAN table potentially containing confidential data



Security correlation to other quality characteristics

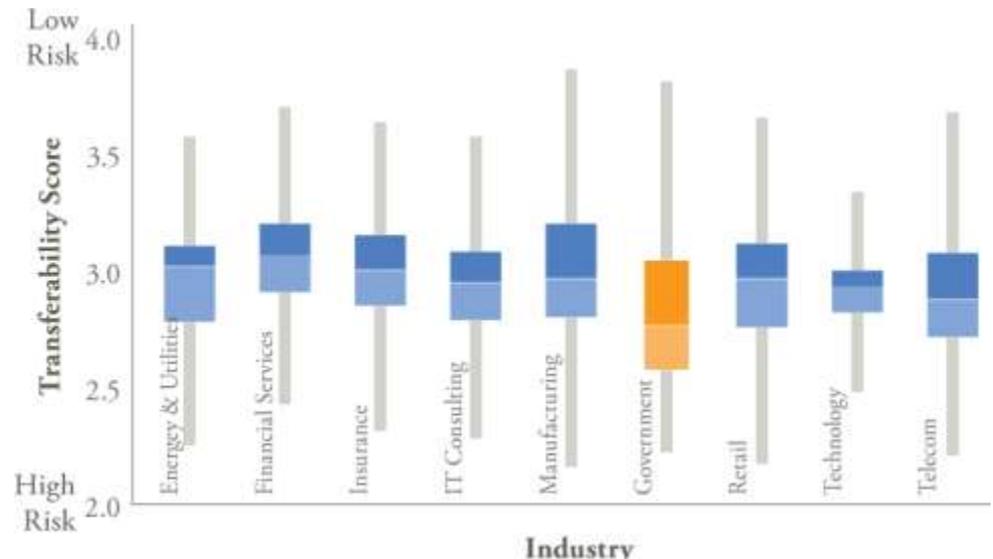
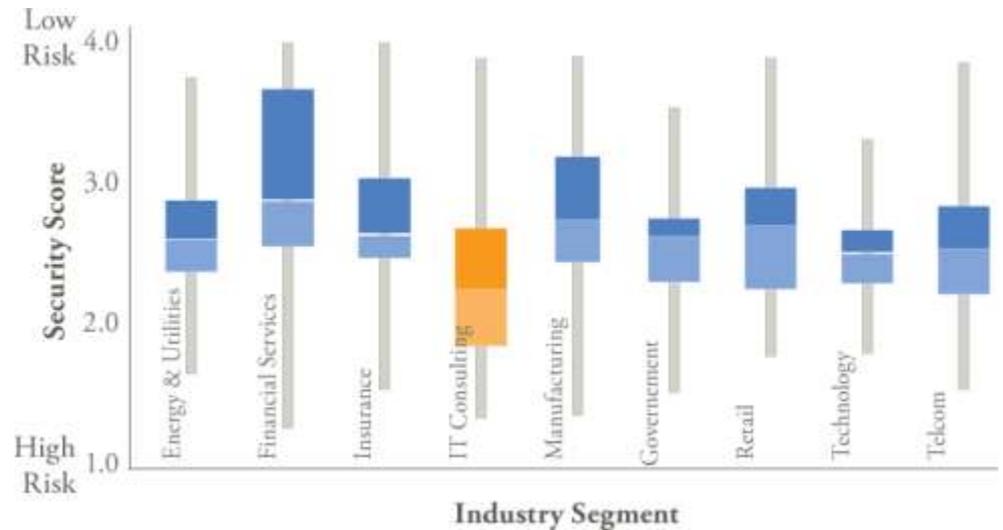
CORRELATIONS											
	ROB	PERF	SEC	TRANS	CHG	ARCH	DOC	PROG	KLOC	% High & V.High COC	% High & V.High COCast
Robustness	1	-.014	.556	.410	.616	.474	.103	.513	-.014	-.069	-.050
Performance	-.014	1	.263	.064	-.148	-.197	.140	.628	-.054	.233	.201
Security	.556	.263	1	.257	.440	.350	.413	.423	.073	.373	.428
Transferability	.410	.064	.257	1	.619	.201	.753	.392	-.034	.157	.224
Changeability	.616	-.148	.440	.619	1	.802	.401	.163	-.062	-.134	-.055
Architecture	.474	-.197	.350	.201	.802	1	.015	-.091	-.057	-.268	-.213
Documentation	.103	.140	.413	.753	.401	.015	1	.173	.058	.463	.509
Standards	.513	.628	.423	.392	.163	-.091	.173	1	-.068	.142	.119
KLOC	-.014	-.054	.073	-.034	-.062	-.057	.058	-.068	1	.132	.078
% High & V.High COC	-.069	.233	.373	.157	-.134	-.268	.463	.142	.132	1	.929
% High & V.High COCast	-.050	.201	.428	.224	-.055	-.213	.509	.119	.078	.929	1

CAST Research Labs – 2012 Industry Trends Report*

- Security
 - Prof services scores lowest in application security



- Maintainability
 - Government has lowest scores



* Full 2012 report available for purchase

CAST Research Labs – current and future directions

- Industry benchmark data published for two years running
- Presented technical debt research as part of ICSE Global Conference – Zurich, June 2012
- Publishing results in next issue of IEEE Software
- Starting research collaboration with several universities
- Next round of benchmark data research due in Q1 of 2013

- Planned research topics
 - Risk modeling based on structural application analysis: modeling likelihood of failure or break-in
 - Maintenance cost modeling based on technical debt: empirical quantification of future maintenance cost
 - Frequency, occurrence of security, robustness flaws by application typology

Summary

- Structural quality and Security are inextricably interwoven, in fact Security is an aspect of structural quality (ISO 25010)
- Structural quality must be measured at the full application level across languages, architectural layers, transactions, and data flows to detect the most insidious flaws
- Industrial practice in detecting structural quality and security flaws is behind the technology