



—| |—
david consulting group



CAST AIP – Pilot Results

*Large US Bank– Asset Servicing
November 18, 2008*

Business Objectives and CAST value

- **Risk mitigation – Continuous quality management**
 - ▲ *Reduce application risks that could seriously damage the business*
 - ◆ an application outage, data corruption and serious performance problems
 - ▲ *Increase end-user satisfaction*

- **Increased control – better visibility**
 - ▲ *Gain visibility over the work done by the teams (outsourced)*
 - ▲ *Provide business owners the mean to measure the deliveries to assess both risk and costs of the outsourced work*

Objectives of the POC

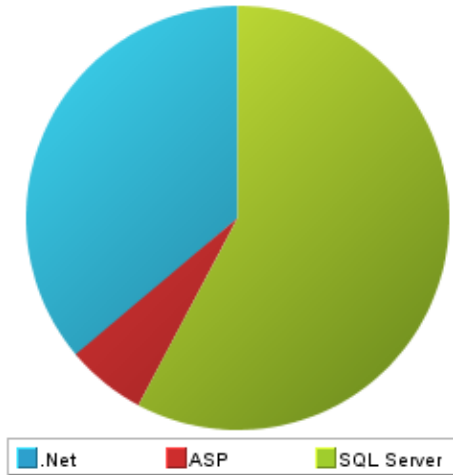
- **Demonstrate that CAST AIP can help Big Bank to achieve the above objectives**

- **Structure of the POC**
 - ▲ *Technical proof that CAST AIP can take into account a representative application of Big Bank*
 - ◆ Inventory of the "XXX" application
 - ◆ Provides the basis to demonstrate the value
 - ▲ *Exploration of the findings*
 - ◆ Value delivered in regards of the objectives
 - Risk mitigation
 - Increased control

Inventory of the "XXX" application

Technical proof

Technology & Sizing Data Overview



Quantity Metrics Summary		Metric Value
Technical Size		
Number of Code Lines		158,335
Number of Files		283
Number of Classes		360
Number of Programs		0
Number of Forms		16
Number of SQL Artifacts		670
Number of Tables		112
Functional Weight		
Backfired IFPUG Function Points		2,252.79
Automated IFPUG Function Points Estimation		595
Number of Decision Points		25,532

Key Highlights of Analysis

- Runtime: 30 minutes baseline, 15 minutes subsequent run
- Violations:
 - ▲ Total No: 21,837
 - ▲ Critical No: 3,747
- Large number of critical violations introduced between versions
- Overall health of "XXX" declining between versions
- Substantial Complexity drift indicating worsening maintenance costs

Risk mitigation – Continuous Quality Management

Value proof

■ Summary

▲ Objective

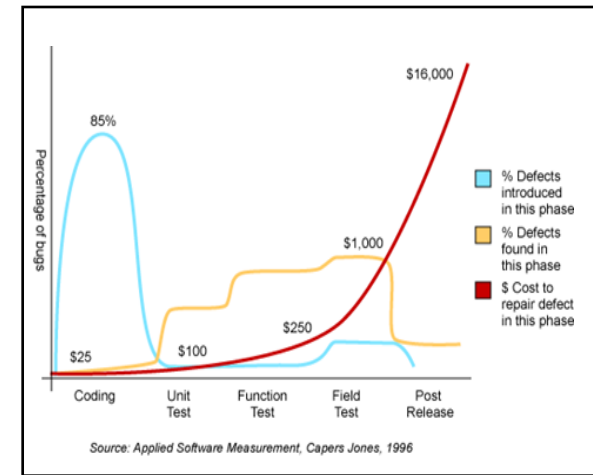
- ◆ Prevent deprecating user experience and costs associated with application downtime

▲ Means

- ◆ Early detection of possible defects
- ◆ Reuse or better use of safe components/frameworks

▲ Proof

- ◆ Verification of latest delivery specifically the new/modified code
- ◆ Examination of pertinent diagnostics
 - Error & Exception management
 - Possible performance issues
 - Others (not illustrated here): Secure coding, Risk of data corruption, Memory Management, Expensive loops



Early detection of defects - How

Value proof

Critical comparison

Identifying critical violations

- ▲ Violation with high probability of introducing problems during later development phases
- ▲ Verify the critical violations introduced in latest build
- ▲ List of artifacts in violation

Critical Violations by Module					
Latest Delivery (new and modified)	RBST	PERF	SECU	TRSF	CHNG
DMA .NET Analysis-.NET Analyze	701	384	226	477	509
DMA TSQL-T-SQL Analyzer	215	344	215	-	7
DMA ASP-ASP Analyzer	-	-	-	-	-
Entire Application (whole code)	RBST	PERF	SECU	TRSF	CHNG
DMA .NET Analysis-.NET Analyze	2,265	543	412	1,855	1,926
DMA TSQL-T-SQL Analyzer	449	488	449	-	15
DMA ASP-ASP Analyzer	-	-	-	-	-

Violations list				
Application	Module	Health Factor	Violation	Status
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.oboe	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.isps	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.boscat.BOSCATClient.getRemoteHandle	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.isps.CallRCom.processEvent	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.ejb.session.BOSOrderBean.ejbCreate	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.ejb.session.BOSOrderHome.create	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.boscat.BOSCATClient.callBOSCAT	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.ejb.session.BOSOrderBean.processOrderDetails	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.util.XMLParser.xmlReader	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.ejb.session.BOSOrderBean.processOrderDetails	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.data.OrderData.updateDetails	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.util.XMLParser.xmlReader	Added
New application	BOSB-J2EE Analyzer	Security	com.bt.bos.bosb.ejb.session.BOSProcessorsHome.create	Added

Early detection of defects - What

Value proof

Error management

■ UC: Effects of poor error and except management

- ▲ *Application instability*
- ▲ *Deteriorating user experience*
- ▲ *Extended time for root cause analysis and time to fix (MTTR)*

■ Application shows issues

- ▲ *No error management in T-SQL referenced*
- ▲ *Generic exceptions thrown, rather than specific exceptions*
- ▲ *Presence of unhandled exceptions; empty catch blocks*

Metric grade	1
Name	Programming Practices - Error and Exception Handling
Description	Respect of error handling practices

Child Metric Weight	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade
8	Yes	T-SQL: Avoid Functions and Procedures doing an Insert, Update or Delete without including error management	Very High Risk	4
6	No	T-SQL: Avoid Stored Procedures not returning a status value	Very High Risk	1
6	No	.NET: Avoid catching an exception of type Exception	High Risk	2.72
6	No	.NET: Avoid throwing an exception of type Exception	Low Risk	4
6	No	.NET: Avoid empty finally blocks	Low Risk	4
6	Yes	.NET: Avoid empty catch blocks	Low Risk	4
4	No	ASP: Use of error handling page	Very High Risk	1

Critical rule with many violations

Non critical rule with few violations

Early detection of defects - What

Value proof

Error management

- **Example: Avoid throwing an exception of type Exception**
 - ▲ *Rationale: caller cannot perform a specific recovery process that is needed*
 - ▲ *Issue: Root cause analysis more difficult*

Metric grade	2.95
Name	Java: Avoid throwing an exception of type Exception
Rationale	Whenever a method throws an exception of type Exception, it prevents its callers from doing the specific recovery process that is needed and as a consequence this will threaten both application robustness and security. For example, each exception related to resource allocation whose catch does not explicitly release the resource might create a "resource leak". When such a leak happens on a limited set of available resources, such as database connection, the application can then become unusable because no resources can be allocated any more. The application also becomes difficult to support and run in production as root-cause analysis is made more difficult. The support teams might not even aware that something went wrong (by catching Exception, RuntimeException might not be visible any more).
Description	The exception Exception should never been thrown. Always Subclass Exception and throw the Subclassed Classes.
Remediation	The method must throw a Subclass of the generic Exception that is provide valuable information about the exception that happened in order to help programmers calling this method to write the appropriate recovery or error management code.
Output	This report lists all Methods throwing an exception of type Exception. It provides the following information: Method full name

	Count	%
Failed Checks	42	2.12
Successful Checks	1935	97.88
Total	1977	100

Item	Object Name	Module Name	Value	New Violation?	Changed Numerical Value?
<input type="checkbox"/>	com.bt.bos.bosb.data.OrderData.prepareGetRegradeProducts	BOSB-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosb.ispss.ValidateDomainName.processEvent	BOSB-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosp.ajax.DomainAvailabilityCheckServlet.doGet	BOSP-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosp.ajax.ValidateUserNameServlet.doGet	BOSP-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosp.locator.BOSServiceLocator.readBOSBProperties	BOSP-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosp.locator.BOSServiceLocator.readBOSPProperties	BOSP-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosp.util.BECommon.ValidateDomainName	BOSP-J2EE Analyzer	java.lang.Exception	Yes	No
<input type="checkbox"/>	com.bt.bos.bosp.util.BECommon.validateUserNameFunction	BOSP-J2EE Analyzer	java.lang.Exception	Yes	No

Early detection of defects – What

Value proof

Performance issues

- **UC: Expensive calls in Loops**
 - ▲ *Causes poor performance due to bad garbage collection*
 - ▲ *Can cause extreme use of network and database I/O caching*

- **"XXX" Application shows various deviations**

Child Metric Weight	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade
10	No	Efficiency - Memory, Network and Disk Space Management	Moderate Risk	3.7
10	No	Efficiency - Expensive Calls in Loops	Very High Risk	1.78
9			Very High	
5				
Child Metric Weight	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade
8	Yes	.NET: Avoid String concatenation in loops	Very High Risk	1
8	No	Avoid method invocation in a loop termination expression	Moderate Risk	3.97
8	No	.NET: Avoid instantiations inside loops	Very High Risk	1
8	No	.NET: Avoid using exception handling inside loops	High Risk	2.06
7	No	.NET: Avoid calling properties that clone values in loops	Low Risk	4
7	Yes	Avoid using SQL queries inside a loop	High Risk	2.5
7	Yes	T-SQL: Avoid Cursors inside a loop	Low Risk	4
7	Yes	ADO.NET: Avoid doing select on Datatable in loop	High Risk	2.49

Early detection of defects - What

Value proof

SQL Performance

- **SQL handling**
 - ▲ *Incorrectly written SQL queries cause for performance issues*

- **"XXX" Application**

Child Metric Weight	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade
10	No	Efficiency - Memory, Network and Disk Space Management	Moderate Risk	3.7
10	No	Efficiency - Expensive Calls in Loops	Very High Risk	1.78
9	No	Efficiency - SQL and Data Handling Performance	Very High Risk	1.65
5	No	Complexity - SQL Queries	Very High Risk	1.62

Child Metric Weight	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade
8	Yes	ADD.NET: Avoid call to AcceptChanges in a loop	Very High Risk	1.6
9	Yes	Never use SQL queries with a cartesian product	High Risk	2.49
9	Yes	Avoid SQL queries not using the first column of a composite index in the WHERE clause	Very High Risk	1.7
9	Yes	Avoid SQL queries on XXL tables not using the first column of a composite index in the WHERE clause	Low Risk	4
9	Yes	Avoid SQL queries that no index can support	Very High Risk	1.8
9	No	SQL: Prefer UNION ALL to UNION	Very High Risk	1



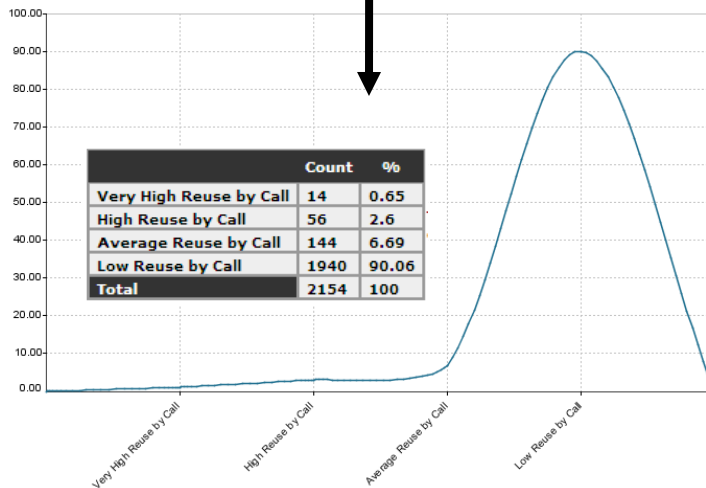
Better use of safe components/frameworks

Value proof

■ Missed opportunity for re-use in the application

- ▲ *Distribution shows very limited used based on actual use in the application*
- ▲ *Many copy pasted artifacts present*

Child Metric Weight	Critical contribution	Child Metric Name	Child Metric Status	Child Metric Grade
9	No	Avoid Too Many Copy Pasted Artifacts	High Risk	2.16
9	No	Avoid High Volume of Copy Pasted Code	High Risk	2.15
1	No	Reuse by Call Distribution	Very High Risk	1
1	Yes	JSP: Avoid direct definition of JavaScript Functions in a Web page	Very High Risk	1
1	No	JSP: Use of style sheets	Very High Risk	1



	Count	%
Failed Checks	274	27
Successful Checks	741	73
Total	1015	100

Risk mitigation – Quality Management

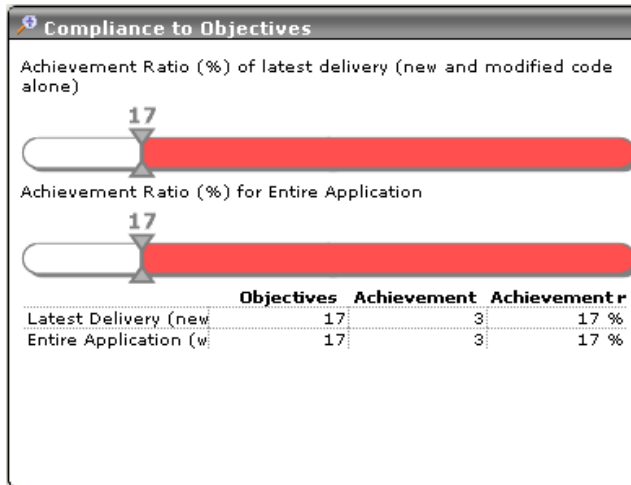
Value proof

■ Immediate value delivery

Impact Area	Key Issues	Immediate Effect on Business Objectives
Robustness	- Inconsistent database access	- Risk of data corruption
	- Issues error and exception management	- Exceptions are not caught which could make the application come to a full stop - Deteriorating user experience
Performance	- Expensive calls in Loops	- Deteriorating user experience
	- Weak memory management	
Security	- Encapsulation of data fields	- Sensitive data exposure
	- Dangerous data access from presentation layer	- Sensitive data exposure

Objective measurement and critical points follow up *Value proof*

■ Technical Quality gate



Topmost Violations

Critical Rule name	Latest	All Viol	Object	Observ
ADO.NET: Avoid using untyped DataSet	477	1,855	99 %	0.00 %
T-SQL: Avoid Functions and Procedures doing too much	208	742	99 %	16.25 %
Avoid SQL queries that no index can support	128	317	99 %	58.18 %
Avoid SQL queries not using the first column	88	231	99 %	52.07 %
.NET: Avoid String concatenation in loops	75	157	99.99 %	5.80 %
Avoid using SQL queries inside a loop	65	150	99 %	80.21 %
Never use SQL queries with a cartesian product	62	153	99 %	79.82 %
.NET: Avoid direct access to database Tables	25	56	99 %	86.92 %
.NET: Avoid empty catch blocks	18	31	99 %	99.09 %
ADO.NET: Avoid doing select on DataTable in loops	13	14	99 %	79.71 %
T-SQL: Avoid Tables access directly from client	7	30	99 %	85.78 %
ADO.NET: Avoid call to AcceptChanges in a loop	4	7	99 %	0.00 %
T-SQL: Avoid Cursors inside a loop	1	2	99 %	99.78 %

- **Achievement Ratio for the Second Code Drop – V2 = 17%**
 - ▲ 17 objectives = 17 critical rules
 - ▲ 3 achieved objectives = 3 critical rules have achieved the expected compliance ratio
 - ▲ 14 failed objectives = 14 critical rules under expected compliance ratio
 - ▲ → **delivery doesn't meet expectations and possibly SLA requirements**

- **Achievement Ratio for the entire application = 17%**
 - SLA could specify an improvement of this achievement ratio over a period of time

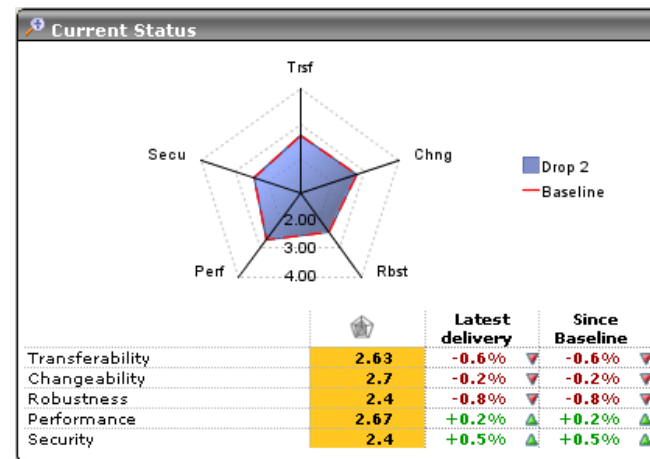
Objective measurement and critical points follow up *Value proof*

■ Health Factors and Violations

- ▲ *Combine the assessment of the overall quality (the TQI and Health Factors) with a close follow-up of critical points that can cause severe failures.*

Statistics	Prev.	Cur.
Violations	3,177	▲ 3,747
- per File	14.44	▼ 13.24
- per kLOC	23.48	▲ 23.67
Complex Obj.	373	▲ 474
- avg violations	867	▲ 1,142

Cur. Snapshot	Drop 2		
Prev. Snapshot	Baseline		
First Snapshot	Baseline		
Snapshots	2		
Quality Indexes	Max	Min	Cur.
TQI	2.57	2.57	2.57
SEI	2.94	2.89	2.89



■ Between V1 and V2:

- ▲ +570 violations (+15%)
- ▲ +101 complex objects (+22%)
- ▲ Significant degradation of the Robustness, Changeability and Transferability
- ▲ Major deterioration of the compliancy to the Architectural Design
- ▲ Maintainability and Technical quality indexes also sliding

Objective measurement and critical points follow up *Value proof*

■ Latest delivery statistics

- ▲ *Quick detection of latest delivery of the application*
- ▲ *How many violations are added?*
- ▲ *Do I need to take action on the Health Factor indications?*

Added Artifacts	15 %
Deleted Artifacts	3 %
Updated Artifacts	10 %

Statistics	Prev.	Cur.
Violations	3,177	▲ 3,747
- per File	14.44	▼ 13.24
- per kLOC	23.48	▲ 23.67
Complex Obj.	373	▲ 474
- avg violations	867	▲ 1,142

Cur. Snapshot	Drop 2
Prev. Snapshot	Baseline
First Snapshot	Baseline
Snapshots	2

Quality Indexes	Max	Min	Cur.
TQI	2.57	2.57	2.57
SEI	2.94	2.89	2.89

Functional Weight	Prev.	Cur.
BFP	1,919	▲ 2,252
FP (Est.)	549	▲ 595
Total CC	20,952	▲ 25,532

Technical Size	Prev.	Cur.
kLOCs	135	▲ 158
Files	220	▲ 283
Classes	340	▲ 360
Programs	0	0
Forms	16	16
SQL Art.	584	▲ 670
Tables	99	▲ 112

Metric Name	DMA - Baseline	DMA - Drop 2
Technical Size		
Number of Comment Lines	42,386	46,249
Number of Commented-out Code Lines	0	0
Number of Code Lines	135,332	158,335
Number of Artifacts	4,385	4,951
Number of Files	220	283
Number of Classes	340	360
Number of Programs	0	0
Number of Forms	16	16
Number of SQL Artifacts	584	670
Number of WEB Pages	31	34
Number of Interfaces	0	0
Number of Methods	3,165	3,544
Number of Functions	0	0
Number of Tables	99	112
Number of Views	0	0
Number of Triggers	8	8
Number of Packages	0	0

Objective measurement and critical points follow up *Value proof*

■ Verification of effort and complexity

- ▲ *How much has been added and what was the associated effort/cost?*
- ▲ *Complexity of delivery*
 - ◆ 2% Highly complex
 - ◆ 6% Complex

DMA	Baseline	Variation	(% or details)	Drop 2
⌵ Functional Weight				
Backfired IFPUG Function Points	1,919.25	333.54	17.38%	2,252.79
Number of Decision Points	20,952	4,580	21.86%	25,532
⌵ Technical Size				
Number of Code Lines	135,332	23,003	17%	158,335
Number of Artifacts	4,385	566	12.91%	4,951
⌵ Number of Processed Artifacts		1,441	(870/351/90/130)	
Added (L/M/H/VH Complexity)		748	(545/145/26/32)	
Updated (L/M/H/VH Complexity)		511	(180/179/58/94)	
Deleted (L/M/H/VH Complexity)		182	(145/27/6/4)	
⌵ Effort				
⌵ Estimated Effort		404.62	(45/39/62/258)	
Addition (L/M/H/VH Complexity)		130.85	(30/20/15/64)	
Updated (L/M/H/VH Complexity)		246.47	(6/16/39/183)	
Deletion (L/M/H/VH Complexity)		27.3	(8/2/6/9)	
Average Estimated Effort per Processed Artifact		1		
⌵ Cost				
⌵ Estimated Cost		247,145	(23,335/22,395/37,035/164,380)	
Addition (L/M/H/VH Complexity)		78,220	(15,735/11,330/9,315/41,840)	
Updated (L/M/H/VH Complexity)		153,205	(3,250/9,715/23,940/116,300)	
Deletion (L/M/H/VH Complexity)		15,720	(4,350/1,350/3,780/6,240)	
Average Estimated Cost per Processed Artifact		1		

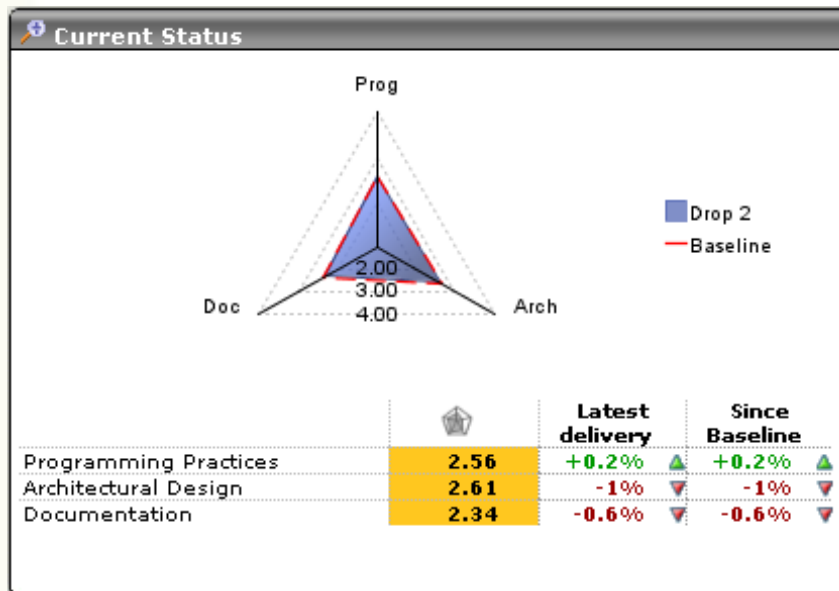
Monitoring of deliveries

Value proof

■ UC: Compliance view

▲ *Disrespect of coding standard and architectural standards cause problems in the long run*

- ◆ Bad performance
- ◆ Illegal data access
- ◆ Poor transition capability
- ◆ Difficult to change applications



Topmost Violations

Critical Rule name	Latest	All Viol	Object	Observ
ADO.NET: Avoid using untyped DataSet	477	1,855	99 %	0.00 %
T-SQL: Avoid Functions and Procedures doing	208	742	99 %	16.25 %
Avoid SQL queries that no index can support	128	317	99 %	58.18 %
Avoid SQL queries not using the first column	88	231	99 %	52.07 %
.NET: Avoid String concatenation in loops	75	157	99.99 %	5.80 %
Avoid using SQL queries inside a loop	65	150	99 %	80.21 %
Never use SQL queries with a cartesian product	62	153	99 %	79.82 %
.NET: Avoid direct access to database Tables	25	56	99 %	86.92 %
.NET: Avoid empty catch blocks	18	31	99 %	99.09 %
ADO.NET: Avoid doing select on DataTable in a	13	14	99 %	79.71 %
T-SQL: Avoid Tables access directly from client	7	30	99 %	85.78 %
ADO.NET: Avoid call to AcceptChanges in a	4	7	99 %	0.00 %
T-SQL: Avoid Cursors inside a loop	1	2	99 %	99.78 %

Monitoring of deliveries

■ Enforce Big Bank standards and Big Bank framework usage

Metric grade	1.6
Name	ADO.NET: Avoid call to AcceptChanges in a loop
Rationale	From a performance point of view, it is better to call AcceptChanges only once at the end of a loop rather than at each iteration.
Description	Avoid call to AcceptChanges (of DataSet, DataTable etc) in loop as this have a negative impact on performance.
Remediation	Move the Acceptchanges call at the end of the loop. Check that the obtained results are the same.
Reference	See .Net 2005 sample provided. Second loop is about 50 times quicker.
Sample	<pre>// Define a dataset with a datatable and three columns (keep default values for all) ... void DisplayMessage(String szAdditionalMsg, long lStart) { String szMsg = szAdditionalMsg + Convert.ToString((new System.TimeSpan(DateTime.Now.Ticks)).Ticks - (new System.TimeSpan(lStart)).Ticks); MessageBox.Show(szMsg, "Information", MessageBoxButtons.OK, MessageBoxIcon.Information); } void f() { DataSet1 ds1 = new DataSet1(); int i = 0; long lStart = 0; String szMsg = ""; lStart = DateTime.Now.Ticks; for (i=0; i <1000; i++) { ds1.DataTable1.Rows.Add(new object[] { i.ToString(), i.ToString(), i.ToString() }); ds1.AcceptChanges(); } DisplayMessage("#Ticks with AcceptChanges in the loop: ", lStart); } // Define a dataset with a datatable and three columns (keep default values for all)</pre>

	Count	%
Failed Checks	7.0	53.85
Successful Checks	6.0	46.15
Total	13.0	100

Object Name	Module Name	Value	New Violation? ▾
[BNYM.Business.Processes.Pricing].BNYM.Business.Processes.Pricing.VendorPriceRequest.BuildIA	DMA .NET Analysis-.NET Analyzer	N/A	Yes
[BNYM.Business.Processes.Pricing].BNYM.Business.Processes.Pricing.VendorPriceRequest.BuildIALondon	DMA .NET Analysis-.NET Analyzer	N/A	Yes
[BNYM.DMA.UI.Web.AppVB].BNYM.DMA.UI.Web.AppVB.EditDefaultPreference.imgDeletPrice_Click	DMA .NET Analysis-.NET Analyzer	N/A	Yes
[BNYM.Dma.Data.Framework].BNYM.Dma.Data.Framework.DataAccess_HoldingFile.InsertRawData	DMA .NET Analysis-.NET Analyzer	N/A	No
[BNYM.DMA.UI.Web.AppVB].BNYM.DMA.UI.Web.AppVB.EditPreference.imgDeletPrice_Click	DMA .NET Analysis-.NET Analyzer	N/A	No
[BNYM.DMA.UI.Web.AppVB].BNYM.DMA.UI.Web.AppVB.maint_pricingData_CPRMAccountSetUp.imgDeletPrice_Click	DMA .NET Analysis-.NET Analyzer	N/A	No
[BNYM.DMA.UI.Web.AppVB].BNYM.DMA.UI.Web.AppVB.OverrideCPRM.DeleteAddedPriceTypes	DMA .NET Analysis-.NET Analyzer	N/A	No

Summary & Next Steps

- **Several critical issues found = immediate value**

- **Next steps**
 - ▲ *Extended Pilot – continued use of CAST on "XXX" platform*
 - ▲ *Incorporate into larger SDLC*