Speaker's Biographical Sketch
John M. Medellin, IBM GBS Executive & Director Strategic Services Mexico

BBA, MBA, MPA Accounting & Computer Systems
UTD MS Software Engineering Program
SMU PhD Computer Science Program

American Institute of CPAs
Texas Society of CPAs
IBM Hispanic Engineers Association

Patent of Letters in Real Time Environments
Assistant Inventor in other patent efforts

Prior Roles at IBM Include:
Application Innovation Leader Travel & Transportation (VP Level)
Retail SAP North America Leader (VP Level)
Global Leader Center of Excellence for SOA (VP Level)

Prior Roles at PricewaterhouseCoopers Include:
Managing Partner Aerospace, Aviation and Travel, North America
Managing Partner SAP Financial Services Consulting, North America

Prior Roles at Bank of America:
Vice President, FDIC SW Plan Manager for Liquidations S. Texas
Vice President, Financial Information Systems, SW USA

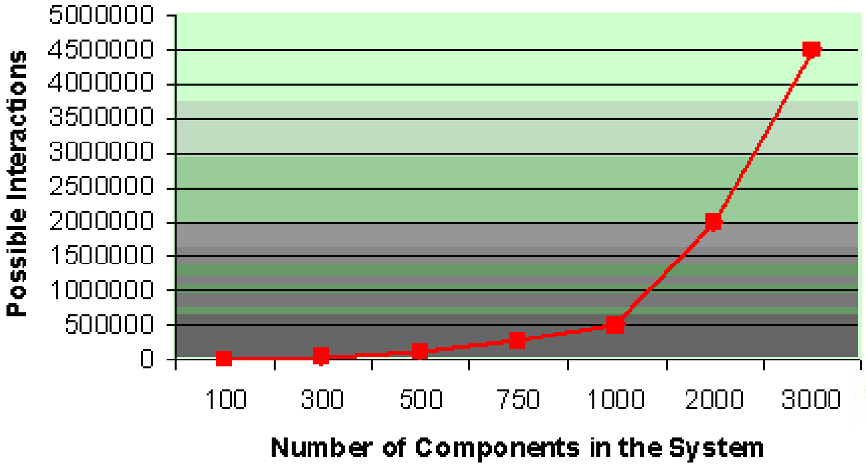
Today’s Agenda – May 4, 2013

- Software Quality Metrics – Primer on Best Practices
- ERP Testing Tools
- Legacy Rehabilitation Incorporating SE Testing Techniques

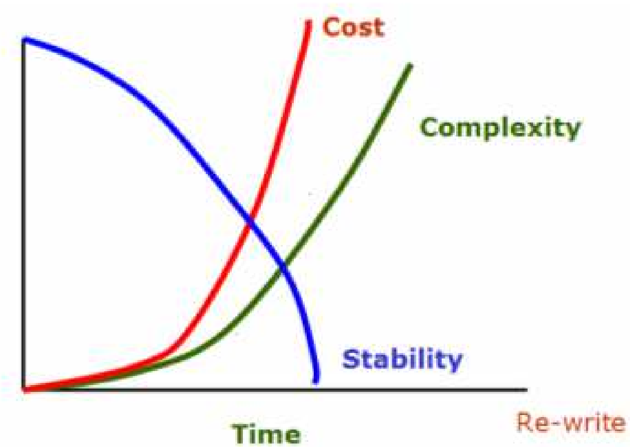
Quality Analysis with Metrics

Why do we care about Quality?

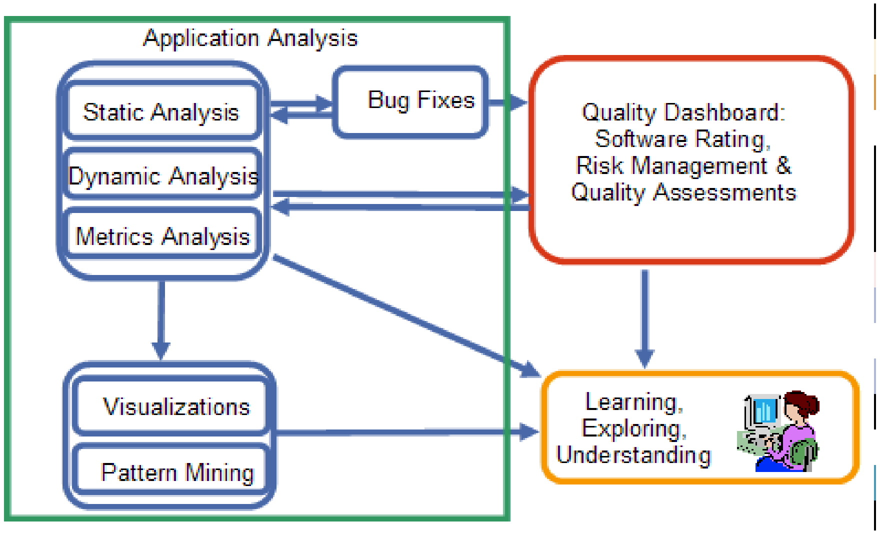
Software may start small and simple, but it quickly becomes complex as more features and requirements are addressed. As more components are added, the potential ways in which they interact grow in a non-linear fashion.



Typical System Life Span



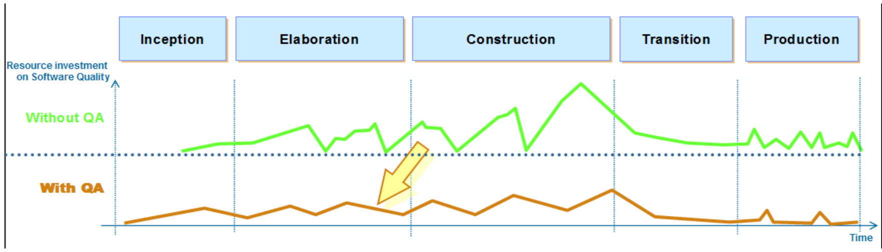
Quality Analysis Stack



Quality Analysis Phases

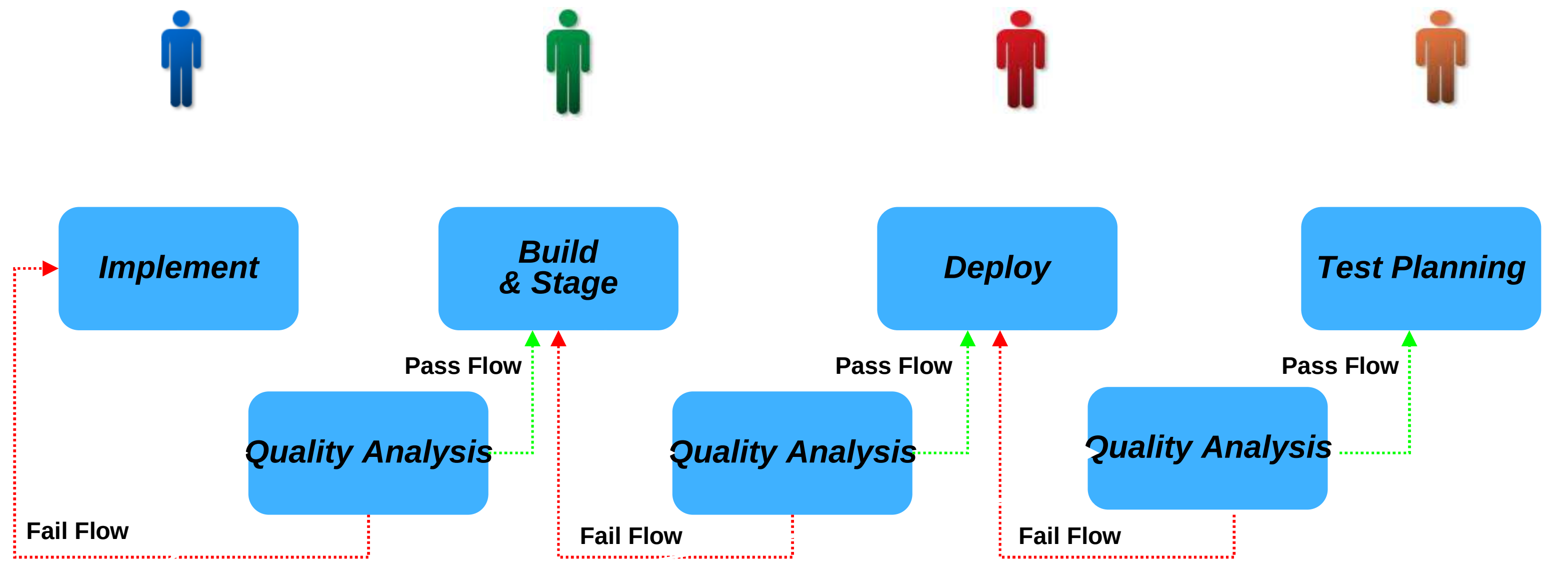
- **Assess Quality**
  - Static
    - Architectural Analysis
    - Software Quality Metrics – Rolled UP in to 3 categories
      - Stability
      - Complexity
      - Compliance with Coding Standards
  - Dynamic
    - Performance Criteria
      - Performance,
      - memory consumption

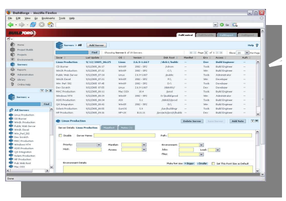
- **Maintain Quality**
  - Static Analysis, Metrics Analysis, Architectural Analysis on every build
  - Testing Efforts
    - Static
      - Statically check test coverage
      - Analyze quality of test cases
      - Prioritize and Compute Testing Activities
    - Dynamic
      - Assess Test Progress
      - Assess Test Effectiveness
      - Dynamically determine code coverage
      - Run Dynamic Analysis with Static Analysis Combination during Testing phase
  - Track the basic project related metrics
    - Churn Metrics (requirements, test cases, code)
    - Defects Metrics(fix rate, introduction rate)
    - Agile metrics for Process
    - Customer Satisfaction (based on surveys, etc.)
    - Costs



- **Forecast Quality**
  - Number of open defects per priority
  - Defect creation rate
  - Code, requirements churn
  - Defect density compared to project history

Continuous Quality Analysis

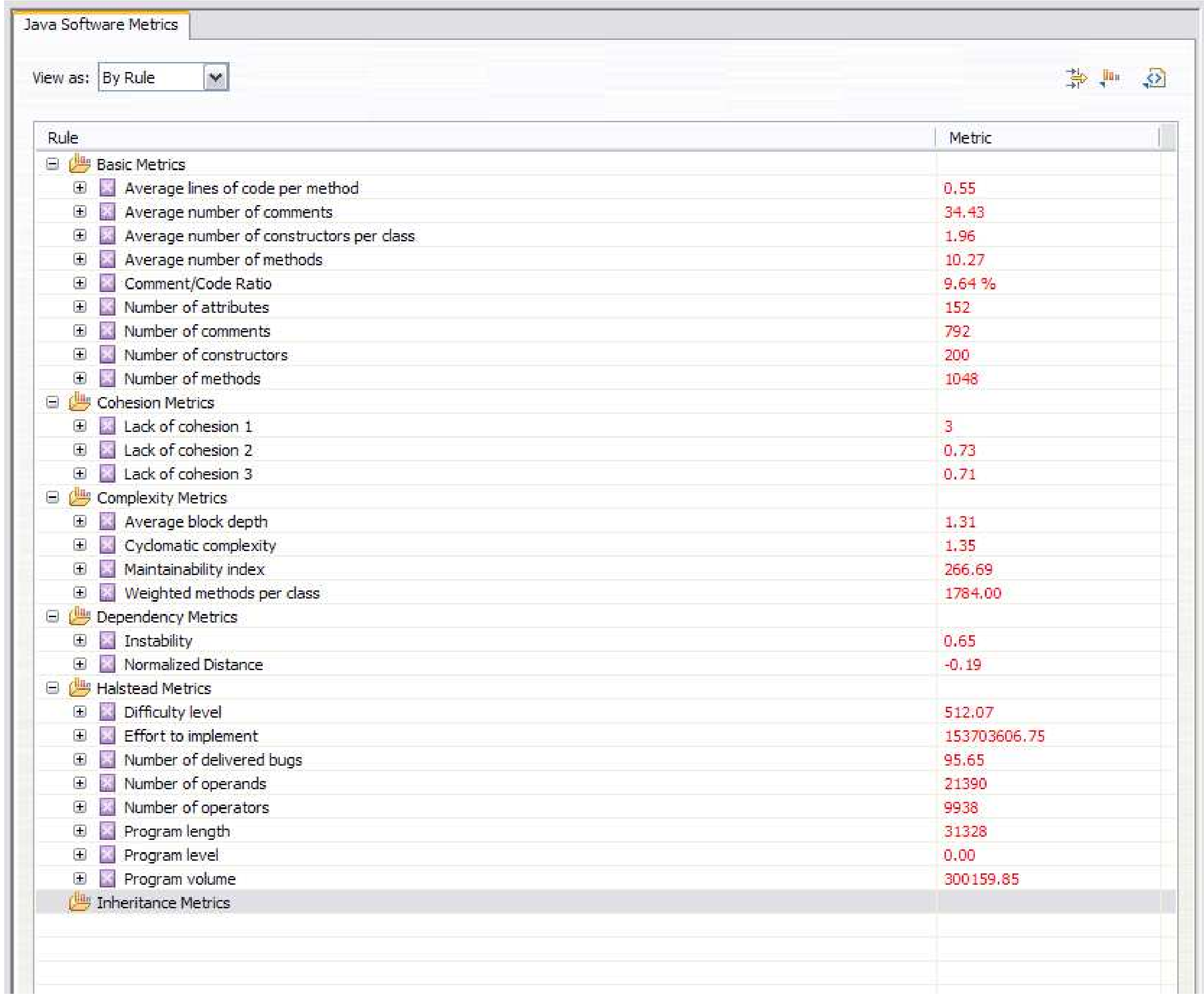




1. Configures/Deploys Tool and Rules
2. Defines Pass/Fail Criteria as a function of N metric buckets and thresholds
3. Runs the analysis tool
4. Tool persists the analysis artifacts into DB
5. Tool produces and aggregates metrics for available buckets
6. QA Lead sets up checkpoints, thresholds and pass/fail criteria

Assess Quality via Metrics Analysis

Property	Value
Number of Objects	12
Number of Packages	2
Number of Relationships	52
Maximum Dependencies	14
Minimum Dependencies	0
Average Dependencies	4.33
Maximum Dependents	11
Minimum Dependents	0
Average Dependents	4.33
Relationship To Object Ratio	4.33
Affects on Average	6.8



Rule	Metric
Basic Metrics	
Average lines of code per method	0.55
Average number of comments	34.43
Average number of constructors per class	1.96
Average number of methods	10.27
Comment/Code Ratio	9.64 %
Number of attributes	152
Number of comments	792
Number of constructors	200
Number of methods	1048
Cohesion Metrics	
Lack of cohesion 1	3
Lack of cohesion 2	0.73
Lack of cohesion 3	0.71
Complexity Metrics	
Average block depth	1.31
Cyclomatic complexity	1.35
Maintainability index	266.69
Weighted methods per class	1784.00
Dependency Metrics	
Instability	0.65
Normalized Distance	-0.19
Halstead Metrics	
Difficulty level	512.07
Effort to implement	153703606.75
Number of delivered bugs	95.65
Number of operands	21390
Number of operators	9938
Program length	31328
Program level	0.00
Program volume	300159.85
Inheritance Metrics	

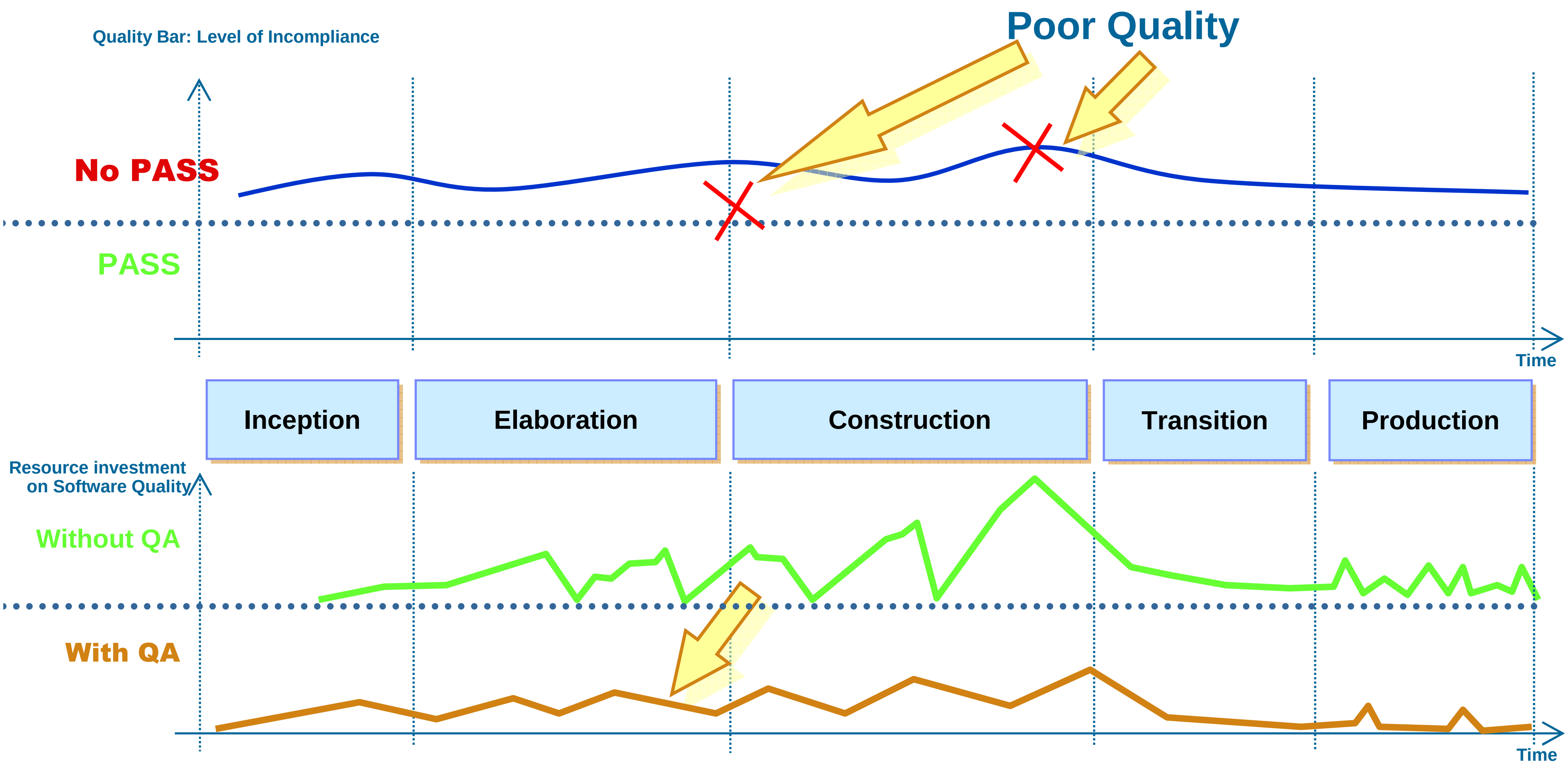
Maintain Quality through Metrics Analysis

Striving for:

- Above 90% Code Coverage
- Above 90% Complexity Stability
- Above 90% Compliance with Major SE Metrics
- Above 90% Static Analysis Compliance

Recipe for successful release:

- SA & Unit testing run on every build
- Break flow on checkpoints – do not allow failures
- Continue only when passed



Forecast Quality via Metrics Analysis

Internal Tools

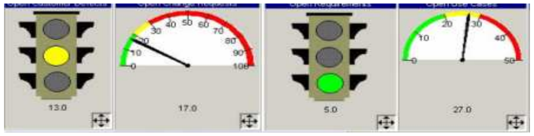
Tests

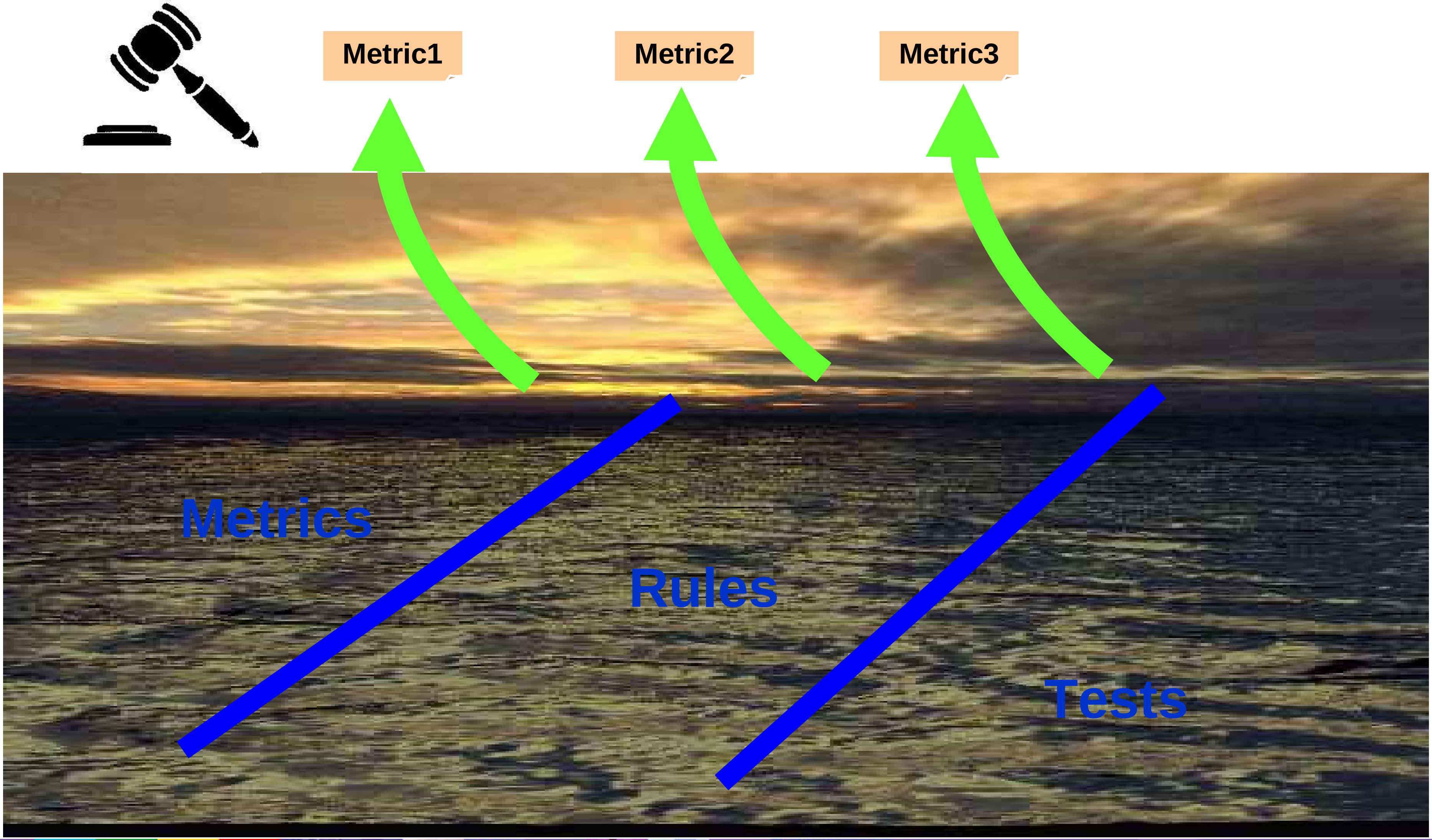
3rd Party Tools

CQ			# open defects per priority (defect backlog)
CQ			Defect arrival rate
CQ			Defect fix rate
PjC (CC)			Code churn per class, package, application
CQ, RP			Requirements churn
CQ, CC			Defect density

Dashboard

Metrics from Static Analysis





Assess, Maintain and Forecast Quality through Metrics Roll-up

Project Management Metrics

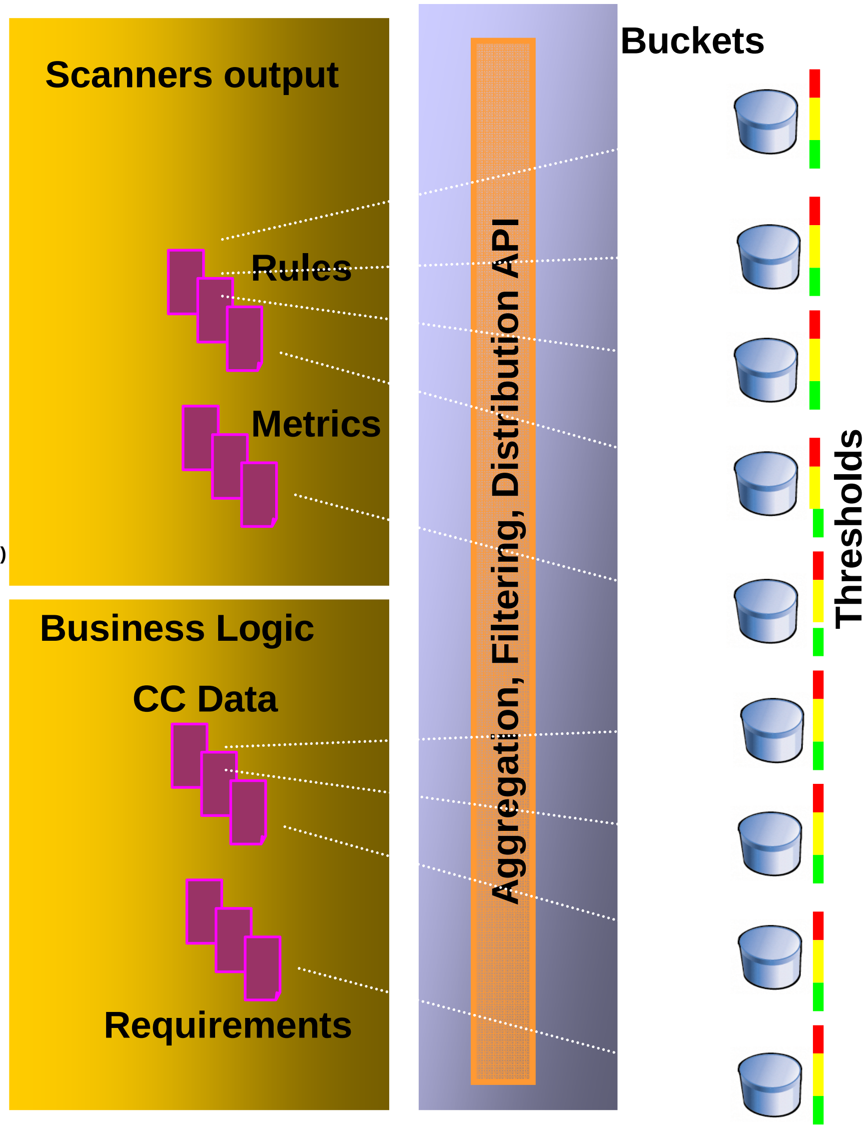
- Forecast quality readiness
  - Number of open defects per priority
  - Defect creation rate
  - Code, requirements churn
  - Defect density compared to project history

Test Management Metrics

- Assess Test Progress
  - Attempted vs. planned tests
  - Executed vs. planned tests
- Assess Test Coverage
  - Code coverage rate (Current, Avg., Min/Max)
  - Object map coverage rate (Current, Avg., Min/Max)
  - Requirements coverage
- Assess Test Effectiveness
  - Test/Case pass/fail rate per execution
  - Coverage per test case
- Prioritize Testing Activities
  - Open defects per priority
  - Planned tests not attempted
  - Planned tests attempted and failed
  - Untested requirements

Software Engineering Metrics

- Complexity
- Rules Output Rollup
- Metrics Rollup



Project Management Buckets

- Core Measure Categories
  - Schedule and Progress
  - Resources and Cost
  - Product Size and Stability
  - Product Quality
  - Process Performance
  - Technology Effectiveness
  - Customer Satisfaction

Test Management Buckets

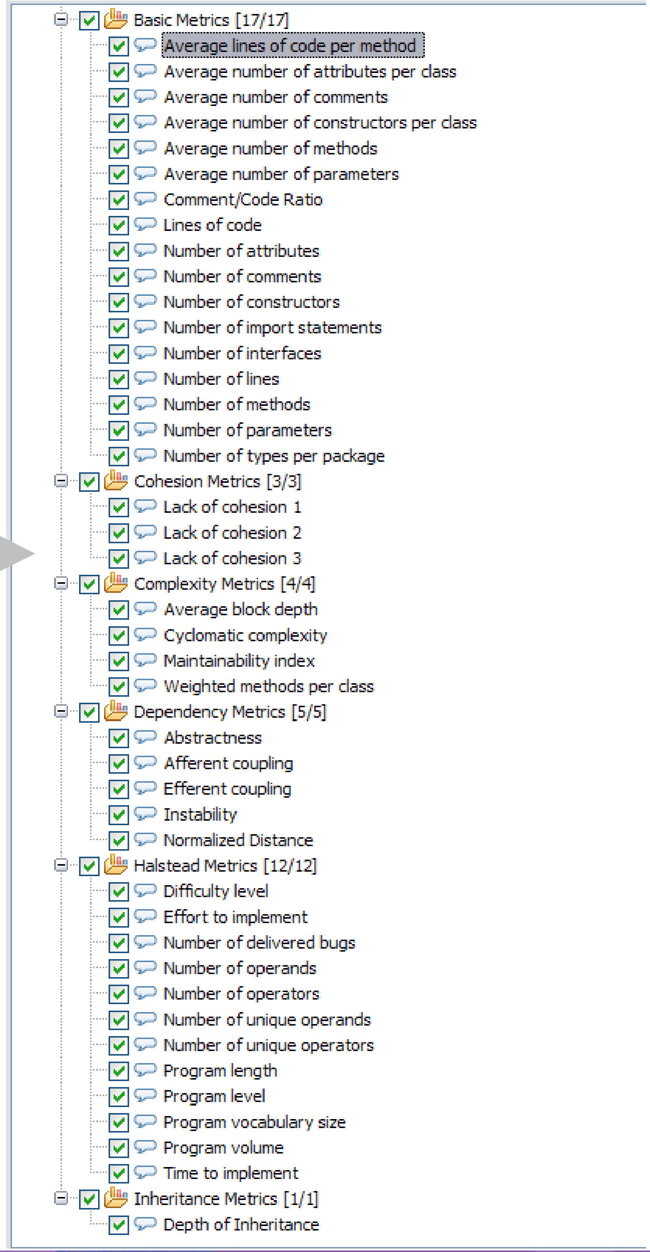
- Core Measure Categories
  - Test Thoroughness
  - Test Regression Size
  - Fail-through Expectance

Software Quality Buckets

- Core Measure Categories
  - Complexity
  - Maintainability
  - Globalization Score
  - Size
  - Stability
  - Adherence to Blueprints

SE Metrics

Assess software quality	
CQ	# of defects per severity
RAD, RPA, P+	Runtime metrics per method, class, package, application, and test case
RAD, RPA, P+	Execution time (avg. or actual)
RAD, RPA, P+	Memory consumption (avg. or actual)
RSA	SE Metrics
RAD, RSA	# static analysis issues



Project Management Metrics

	Forecast quality readiness
CQ	# open defects per priority (defect backlog)
CQ	Defect arrival rate
CQ	Defect fix rate
PjC (CC)	Code churn per class, package, application
CQ, RP	Requirements churn
CQ, CC	Defect density
	Adjust process according to weaknesses (ODC)
CQ (ODC schema)	Defect type trend over time
CQ, CC	Component/subsystem changed over time to fix a defect
CQ, CC	Impact over time
CQ	Defects age over time
	Assess Unit Test Progress
RAD	cumulative # test cases
RAD	Code coverage rate (Current, Avg., Min/Max)
	Agile Metrics (http://w3.webahead.ibm.com/w3ki/display/agileatibm)
Agile Wiki	% of iterations with Feedback Used
Agile Wiki	% of iterations with Reflections

Test Management Metrics

	Assess Test Progress (assume that UnitTests are not scheduled, planned, traced to requirements)
CQ, RFT, RMT, RPT	cumulative # test cases
CQ	# planned, attempted, actual tests
CQ	Cumulative planned, attempted, actual tests in time
CQ	Cumulative planned, attempted, actual tests in points
	Assess Test Coverage
RAD, RPA, P+	Code coverage rate (Current, Avg., Min/Max)
RFT	Object map coverage rate (Current, Avg., Min/Max)
CQ, RP	Requirements coverage (Current, Avg., Min/Max)
	Assess Test Effectiveness
CQ, RFT, RMT, RPT	Hours per Test Case
CQ	Test/Case pass/fail rate per execution
	Coverage per test case
CQ, RAD, RPA, P+	Code coverage
CQ, RFT	Object map coverage
CQ, RP	Requirements coverage
	Prioritize Testing Activities
CQ	Open defects per priority
CQ	# planned tests not attempted
CQ	# planned tests attempted and failed
CQ, RP	# untested requirements

Coupling Metrics

Afferent Couplings	Afferent Couplings This is the number of members outside the target elements that depend on members inside the target elements.
Efferent Couplings	Efferent Couplings This is the number of members inside the target elements that depend on members outside the target elements.
Instability	Instability (I) Description: I = (Ce ÷ (Ca+Ce))
Number of Direct Dependents	Includes all Compilation depdencies
Number of Direct Dependencies	Includes all Compilation depdencies
Normalized Cumulative Component Dependencies	Normalized Cumulative Component Dependency(NCCD) Normalized cumulative component dependency, NCCD, which is the CCD divided by the CCD of a perfectly balanced binary dependency tree with the same number of components. The CCD of a perfectly balanced binary dependency tree of n components is (n+1) * log2(n+1) - n. http://photon.poly.edu/~hbr/cs903-F00/lib_design/notes/large.html
Coupling between object classes	Coupling between object classes(CBO). According to the definition of this measure, a class is coupled to another, if methods of one class use methods or attributes of the other, or vice versa. CBO is then defined as the number of other classes to which a class is coupled. Inclusion of inheritance-based coupling is provisional. http://www.iese.fraunhofer.de/Products_Services/more/faq/MORE_Core_Metrics.pdf Multiple accesses to the same class are counted as one access. Only method calls and variable references are counted. Other types of reference, such as use of constants, calls to API declares, handling of events, use of user-defined types, and object instantiations are ignored. If a method call is polymorphic (either because of Overrides or Overloads), all the classes to which the call can go are included in the coupled count. High CBO is undesirable. Excessive coupling between object classes is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is to reuse it in another application. In order to improve modularity and promote encapsulation, inter-object class couples should be kept to a minimum. The larger the number of couples, the higher the sensitivity to changes in other parts of the design, and therefore maintenance is more difficult. A high coupling has been found to indicate fault-proneness. Rigorous testing is thus needed. A useful insight into the 'object-orientedness' of the design can be gained from the system wide distribution of the class fan-out values. For example a system in which a single class has very high fan-out and all other classes have low or zero fan-outs, we really have a structured, not an object oriented, system. http://www.aivosto.com/project/help/pm-oo-ck.html
Data Abstraction coupling	Data Abstraction Coupling DAC is defined for classes and interfaces. It counts the number of reference types that are used in the field declarations of the class or interface. The component types of arrays are also counted. Any field with a type that is either a supertype or a subtype of the class is not counted. http://maven.apache.org/reference/metrics.html

Information Complexity Metrics

Depth Of Looping	**Depth Of Looping (DLOOP)** **Depth of looping equals the maximum level of loop nesting in a procedure. Target at a maximum of 2 loops in a procedure.** **http://www.aivosto.com/project/help/pm-complexity.html**
Information Flow	**Information Flow (IFIO)** **Fan-in IFIN = Procedures called + parameters read + global variables read** **Fan-out IFOUT = Procedures that call this procedure + [ByRef] parameters written to + global variables written to** **IFIO = IFIN * IFOUT** **http://www.aivosto.com/project/help/pm-complexity.html**
Information Flow Cohesion	**Information-flow-base cohesion (ICH)** **ICH for a method is defined as the** **number of invocations of other methods of the same class, weighted by** **the number of parameters of the invoked method (cf. coupling measure** **ICP above). The ICH of a class is the sum of the ICH values of its methods.** **http://www.iese.fraunhofer.de/Products_Services/more/faq/MORE_Core_Metrics.pdf**

Class Cohesion

Lack of Cohesion	Lack Of Cohesion (LCOM) A measure for the Cohesiveness of a class. Calculated with the Henderson-Sellers method. If (m (A) is the number of methods accessing an attribute A, calculate the average of m (A) for all attributes, subtract the number of methods m and divide the result by (1-m). A low value indicates a cohesive class and a value close to 1 indicates a lack of cohesion and suggests the class might better be split into a number of (sub) classes. http://metrics.sourceforge.net
Lack of Cohesion1	LCOM1 is the number of pairs of methods in the class using no attribute in common.http://www.iese.fraunhofer.de/Products_Services/more/faq/MORE_Core_Metrics.pdf
Lack of Cohesion2	COM2 is the number of pairs of methods in the class using no attributes in common, minus the number of pairs of methods that do. If this difference is negative, however, LCOM2 is set to zero. http://www.iese.fraunhofer.de/Products_Services/more/faq/MORE_Core_Metrics.pdf
Lack of Cohesion3	LCOM3 Consider an undirected graph G, where the vertices are the methods of a class, and there is an edge between two vertices if the corresponding methods use at least an attribute in common. LCOM3 is then defined as the number of connected components of G. http://www.iese.fraunhofer.de/Products_Services/more/faq/MORE_Core_Metrics.pdf
Lack of Cohesion4	LCOM4 Like LCOM3, where graph G additionally has an edge between vertices representing methods m and n, if m invokes n or vice versa. http://www.iese.fraunhofer.de/Products_Services/more/faq/MORE_Core_Metrics.pdf

Halstead Complexity

The Halstead measures are based on four scalar numbers derived directly from a program's source code:

n1 = the number of distinct operators

n2 = the number of distinct operands

N1 = the total number of operators

N2 = the total number of operands

From these numbers, five measures are derived:

Measure	Symbol	Formula
Program length	N	N= N1 + N2
Program vocabulary	n	n= n1 + n2
Volume	V	V= N * (LOG2 n)
Difficulty	D	D= (n1/2) * (N2/n2)
Effort	E	E= D * V

Cyclomatic Complexity

The cyclomatic complexity of a software module is calculated from a connected graph of the module (that shows the topology of control flow within the program):

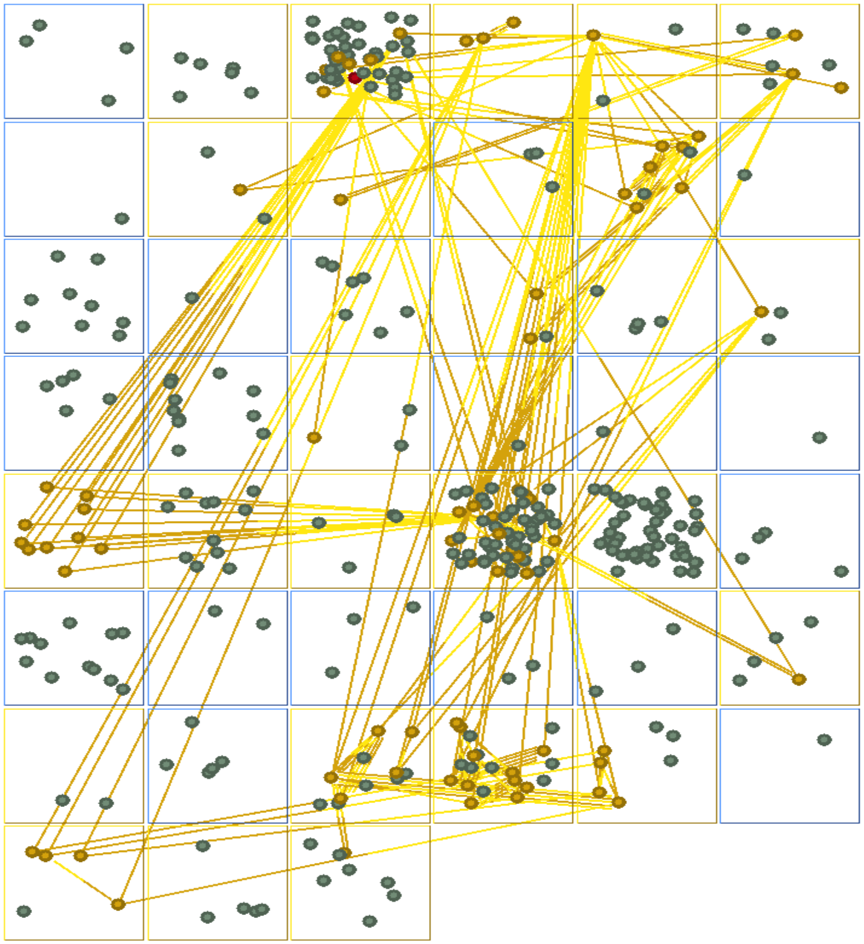
Cyclomatic complexity (CC) = E - N + p
where E = the number of edges of the graph
N = the number of nodes of the graph
p = the number of connected components

Cyclomatic Complexity	Risk Complexity
1-10	a simple program, without much risk
11-20	more complex, moderate risk
21-50	complex, high risk
51+	untestable, very high risk

Cyclomatic Complexity	Cyclomatic complexity (Vg) Cyclomatic complexity is probably the most widely used complexity metric in software engineering. Defined by Thomas McCabe, it's easy to understand, easy to calculate and it gives useful results. It's a measure of the structural complexity of a procedure. V(G) is a measure of the control flow complexity of a method or constructor. It counts the number of branches in the body of the method, defined as: while statements; if statements; for statements. CC = Number of decisions + 1 http://www.aivosto.com/project/help/pm-complexity.html http://maven.apache.org/reference/metrics.html
Cyclomatic Complexity2	Cyclomatic complexity2(Vg2) CC2 = CC + Boolean operators CC2 includes Boolean operators in the decision count. Whenever a Boolean operator (And, Or, Xor, Eqv, AndAlso, OrElse) is found within a conditional statement, CC2 increases by one. The reasoning behind CC2 is that a Boolean operator increases the internal complexity of the branch. You could as well split the conditional statement in several sub-conditions while maintaining the complexity level. http://www.aivosto.com/project/help/pm-complexity.html

SmallWorlds Stability (SA4J)

The stability is calculated as follows. For every component C (class/interface) in the system compute
Impact(C) = Number of components that which potentially use C in the computation. That is it is a transitive closure of all relationships. Then calculate Average Impact as Sum of all Impact(C) / Number of components in the system. The stability is computed as an opposite of an average impact in terms



University of Texas at Dallas
Erik Jonsson School of Engineering & Computer Science

Dr. W. Eric Wong, Professor
Mr. Ricky Gao, Assistant

"Evaluating SAP Testing Tools in Light of Software Engineering Principles; Version 2, Including Market Potential Extrapolations"

John M. Medellin

May 4, 2013

Contents

- ERP System Testing Failures
- ERP Testing Market and Potential for Products
- ERP Systems Overview
- Testing Tools
- Approach To Testing
- Third Party Testing Tool Overview
- Testing Tool Evaluation
- Scaling Analysis Example
- Potential Enhancements Recommended

ERP System Testing Blunders

- CPG Corporation, 2002…production system issuing wrong order fulfillment, 200+ trucks stuck without correct delivery, estimated cost in the millions, reputational impact for integrator.
- Chemicals corporation, 2000's, incorrect MRP set in several plants in europe requires additional months of testing, chemical/additional losses become part of financial statement disclosures to the Bourse
- Major telecom producer writes off several million dollars due to miss matching between telephone set production and order fulfillment.
- Fixed % of Airwaybills are written off by major logistics player due to incorrect pricing and/or billing instructions.
- And the defects go on……

ERP System Testing Market

ERP Advanced Testing Techniques Estimation Model
Version 1, JM Medellin, UTD Software Engineering Program
May-13

Commercial Testing Sector Vendors

	SAP---------------------------		Oracle---------------------		Other (Infor etc.)-----------		Totals-------------------------	
	Low End	High End	Low End	High End	Low End	High End	Low End	High End
IBM	10,000	15,000	5,000	7,000	1,000	1,500	16,000	23,500
HP/EDS	8,000	12,000	4,000	5,600	800	1,200	12,800	18,800
TCS	8,000	12,000	4,000	5,600	800	1,200	12,800	18,800
Wipro	8,000	12,000	4,000	5,600	800	1,200	12,800	18,800
Sapient	2,400	3,600	1,200	1,680	240	360	3,840	5,640
Hitachi	2,400	3,600	1,200	1,680	240	360	3,840	5,640
Indra	1,200	1,800	600	840	120	180	1,920	2,820
Others (next 5)	2,400	3,600	1,200	1,680	240	360	3,840	5,640
Estimated Resources	42,400	63,600	21,200	29,680	4,240	6,360	67,840	99,640
Hours/resource/year	1,750	1,750	1,750	1,750	1,750	1,750	1,750	1,750
Hours Total Per Year	74,200,000	111,300,000	37,100,000	51,940,000	7,420,000	11,130,000	118,720,000	174,370,000
$USD Per Hour	20	25	20	25	15	20		
$USD MM Per Year	1,484.00	2,782.50	742.00	1,298.50	111.30	222.60	2,337.30	4,303.60
In House Testing Departments								
Ratio of in house to comm hrs	50%	50%	50%	50%	50%	50%		
Ratio of in house to comm cost	75%	75%	75%	75%	75%	75%		
In house hours	37,100,000	55,650,000	18,550,000	25,970,000	3,710,000	5,565,000	59,360,000	87,185,000
In house cost per	15	19	15	19	11	15		
In house spend estimate	556.50	1,057.35	278.25	493.43	40.81	83.48	875.56	1,634.26
Combined Market Costed Capacity							3,212.86	5,937.86
Rounded Estimate							3200	5900

What if…a company was created to help?

Combined Market Costed Capacity		3,212.86	5,937.86
Rounded Estimate		3200	5900
Productivity Financial Effect			
	1%	32	59
	3%	96	177
	5%	160	295
Average Individual Ranges		96	177
Estimate Productivity Per Year (average low/high)			137
Market Potential for Offering (5 Year Revenue, 35% of productivity estimate, client keeps 65%, company stays with 35% as revenue)			240
Cost of Organization, see below….			103
5 Year EBITDA in Millions of USD			**137**
SW Multiplier			10
Target Market Capitalization at IPO			**1,370.00**

Cost of Organization

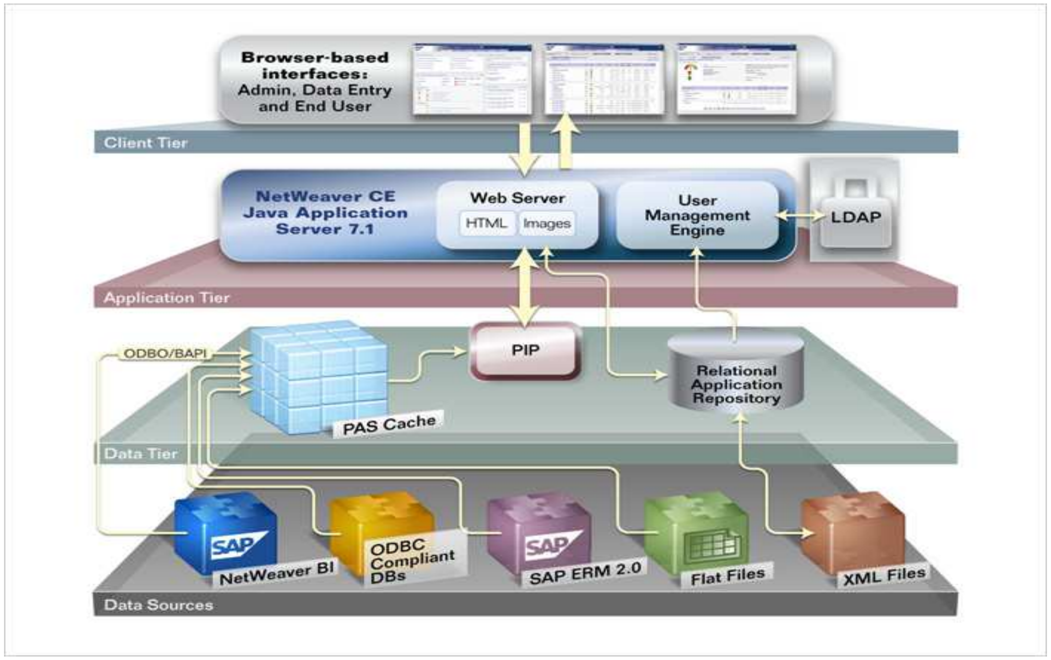
	Assumptions	Staffing	Cost Per	Extended	Benefits (30%)	OH 20%	FF&E (10%)	Totals @ 5Y
Number of Product Lines	3							
Manager per Prod Line	1	3	150000	450000	135000	90000	45000	3,600,000
Developers/Prod Line	6	18	90000	1620000	486000	324000	162000	12,960,000
Help Desk/Prod Line	8	24	50000	1200000	360000	240000	120000	9,600,000
Consultants		50	90000	4500000	1350000	900000	450000	36,000,000
Executive Mgt		10	250000	2500000	750000	500000	250000	20,000,000
Numbers per year in full production mode		105		10,270,000	3,081,000	2,054,000	1,027,000	
Estimated ground up cost								82,160,000
Contingency								25%
Estimated cost of Organization								102700000
Rounded Estimate								**103**

ERP Systems Overview

ERP Market:

- **$30-50B Market Depeding of how it is counted**
- **License, Maintenance, Services**
- **SAP, Oracle Leaders (70,000 Customers, Millions of Users)**

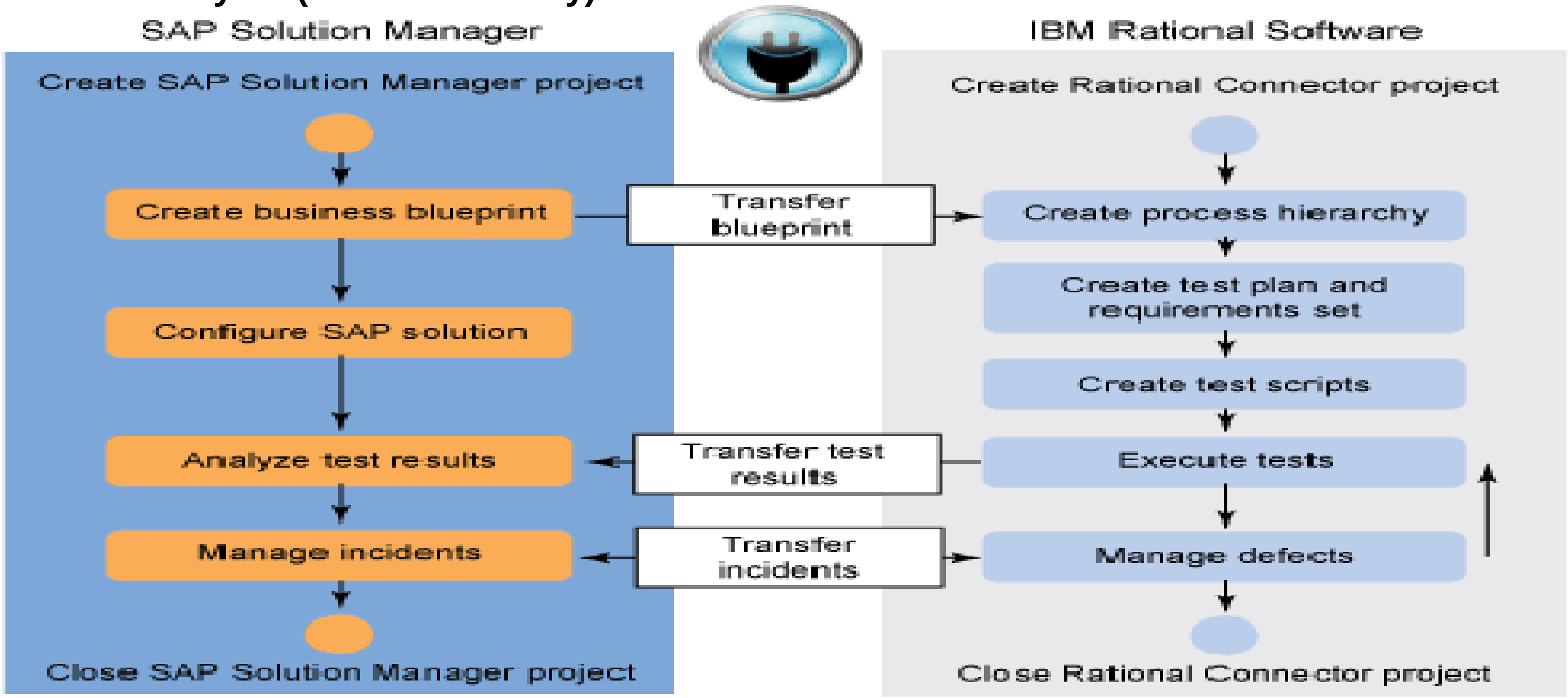
System Architecture



Testing Tools: All Integrate with Solution Manager

Tool Vendors

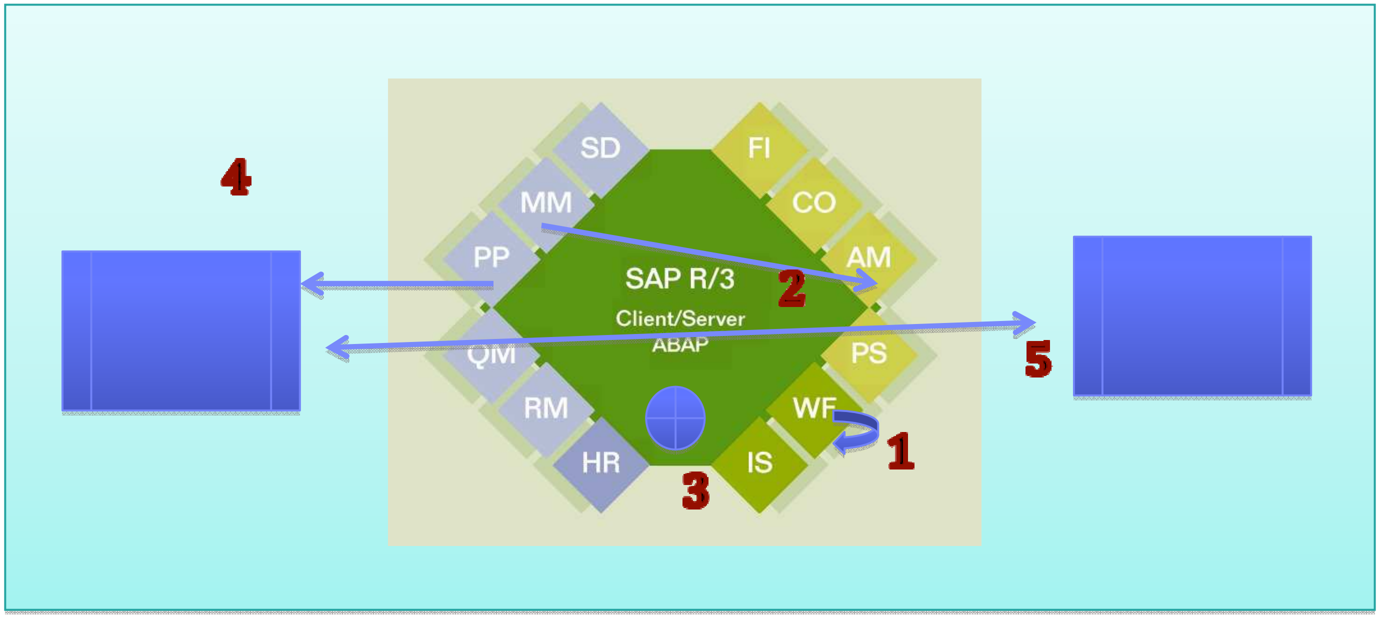
- **HP (Mercury Interactive)**
- **IBM (Rational)**
- **Borland**
- **Niche Players (Worksoft Certify)**



Approach to Testing (build on integration)

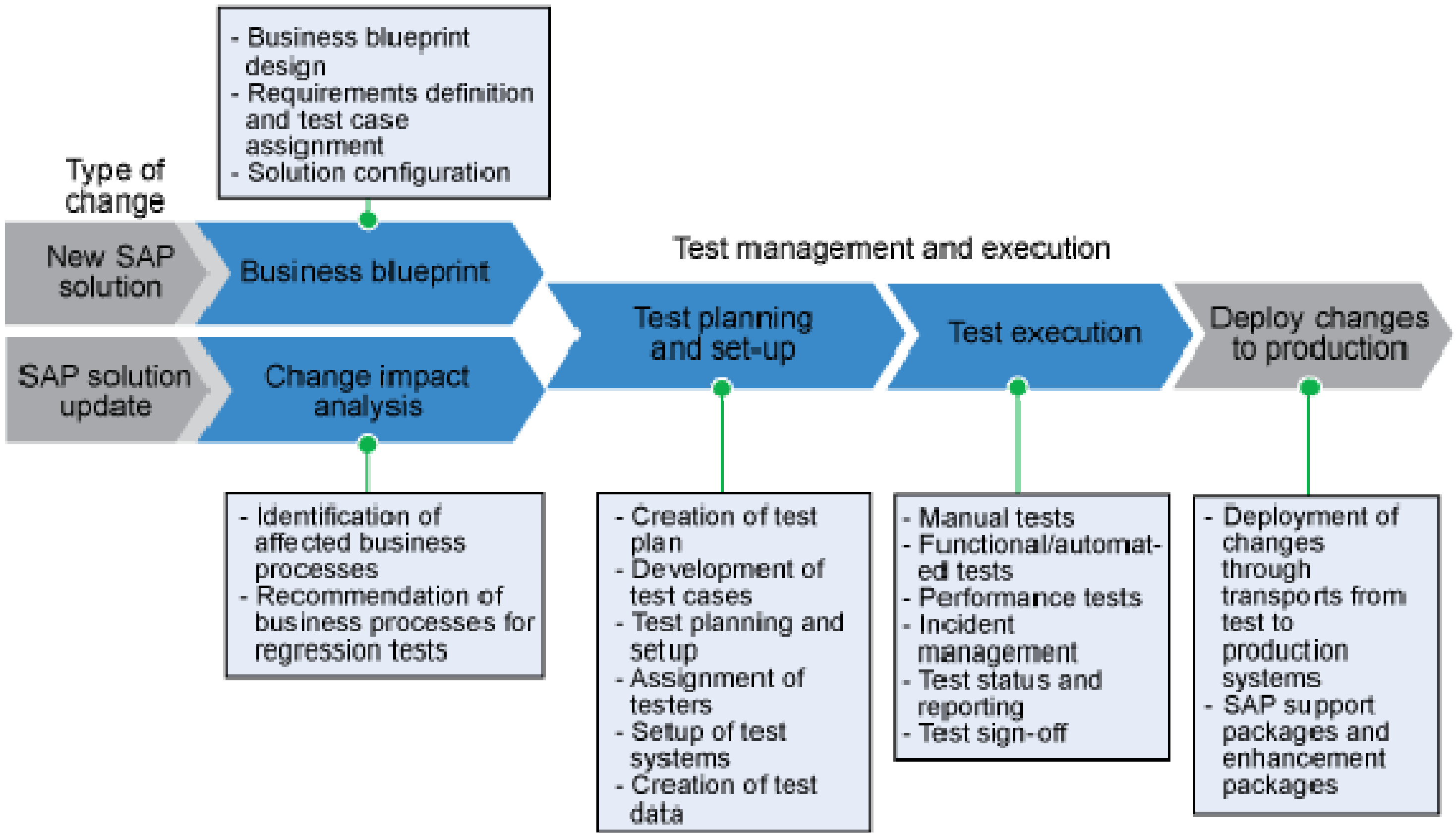
Test Script Scope

- **Intramodule Script Testing**
- **Intermodule Script Testing**
- **User Exit Script Testing**
- **SAP to Legacy Testing**
- **Legacy to SAP to Legacy Testing**



Third Party Test Tool Overview

- **Third party vendors will emphasize their integration within the SAP Methodology as complimentary to standard SAP tools.**
- **HP and IBM both will emphasize their integration with SAP and Legacy systems for end to end integration**



Characteristic / Tool	SAP Provided Tool Set	HP SAP Quality Center	Borland SILK	IBM Rational	Worksoft Certify
1. Requirements based test generation	(M) tools can generate scripts within the SAP domain	(H) can create scripts that include SAP and non-SAP systems	(H) can create scripts that include SAP and non-SAP systems	(H) can create scripts that include SAP and non-SAP systems	(M) can create specialized test cases within SAP
2. Control Flow and Data Flow Testing	(M) tools can identify path to statement within ABAP & block (no decision) execution only	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)
3. Coverage Criteria Reporting	(M) reports if the statements are covered but does not provide path to cover	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)
4. White Box Testing	(L) no Xsuds style reporting of block coverage	(L) does not contain an analyzer for ABAP (use SAP's)	(L) does not contain an analyzer for ABAP (use SAP's)	(M) Available for Java based implementations (for Java portion of ABAP)	(L) does not contain an analyzer for ABAP (use SAP's)
5. Regression Testing	(M) analyzes impact of upgrades and changes in parametrization (not code)	(H) Will analyze SAP and non-SAP Impact on scripts (not code)	(M) can perform the analyisis but not as integrated as HP or IBM	(H) Will analyze SAP and non-SAP Impact on scripts (not code)	(M) Will analyze SAP and non-SAP Impact through Rational
6. Model based test generation	(H) can generate scripts from parameter values in Solution Manager	(H) can generate scripts from parameter values in Solution Manager	(H) can generate scripts from parameter values in Solution Manager	(H) can generate scripts from parameter values in Solution Manager	(M) can create specialized test cases within SAP mostly for high performance testing
7. Debugging Support	(H) Identification through SAP ABAP Debugger	(M) Tracks bug status & reports (Interface to Sol Mgr.)	(M) Tracks bug status & reports (Interface to Sol Mgr.)	(M) Tracks bug status & reports(Interface to Sol Mgr.)	(M) Tracks bug status & reports (Interface to Sol Mgr.)
8. Test Process Management	(M) Restricted to SAP side of implementation	(H) Full best practices implemented	(H) Full best practices implemented	(H) Full best practices implemented	(H) Full best practices implemented
9. Test Artifact Management	(M) Artifacts within SAP domain well managed, can	(H) Full best practices implemented	(H) Full best practices implemented	(H) Full best practices implemented	(H) Full best practices implemented

Evaluation Part 2

Characteristic / Tool	SAP Provided Tool Set	HP SAP Quality Center	Borland SILK	IBM Rational	Worksoft Certify
10. Test Data Generation	**(M) some support for script data generation**	**(M) some support for script data generation**	**(M) some support for script data generation**	**(M) some support for script data generation**	**(H) High area of focus of the tool set**
11. Overall Cost	(L) mostly included	(M) higher cost than previous mercury due to HP overhead	(L) cost effective option, may not scale for extremely large test orgs.	(H) really a part of the integrated Rational Suite	(L-M?) Did not get pricing indication but small company probably small price
12. Overall Value	*(H) cost is low and only one to offer some features*	*(M) higher cost but could scale if the HP Suite is brought in*	*(M) Lower cost but might not scale for larger org.*	*(M) higher cost but could scale when the Rational Suite is brought in*	*(M) probably lower cost but only useful in specialized spaces*

Scale Evaluation Example

Element/Program Scope	Small	Medium	Large	Very Large
1. # of Organizational Units	1	2-3	4-7	More than 7
2. # of Locations	1	2-4	5-10	More than 10
3. # of SIPCO Level 4 Business Processes	10	11-40	41-100	More than 100
4. # of Extensions (API Usage)	1 (Vertex)	2-5	5-10	More than 10
5. # of Interfaces & Conversions	5	6-12	13-20	More than 20
6. # of test scripts (4 per business process, 10 per custom object)	100	~ 430	~ 950	~ 2000+
7. Est Test Hours w/o tool at 40 hr/script (generation, execution, remediation included)	4,000 hours	17,200 hours	38,000 hours	80,000 hours
8. Est PMO Hours (7.5%*half of Org Units+Locations over 2, topped at 50% of tester budget)	300 hours	7,000 hours	19,000 hours	40,000 hours
9. Est Total Hours without tool sets	4,300 hours	24,200 hours	57,000 hours	120,000 hours
10. Estimated benefit with testing tools	20-40% (860-1720 hrs)	20-40% (4,840-9,680 hrs)	20-40% (11,400-22,800 hrs)	20-40% (24,000-48,000 hrs
11. Cost Avoided (@25/hr, India)	$21,500-$43,000	$121,000-$242,000	$285,000-$570,000	$600,000-$1,200,000
12. Potential Strategy	Only use the tools that SAP Provides	Consider Borland	Consider Borland & HP	Consider All Tools Mentioned

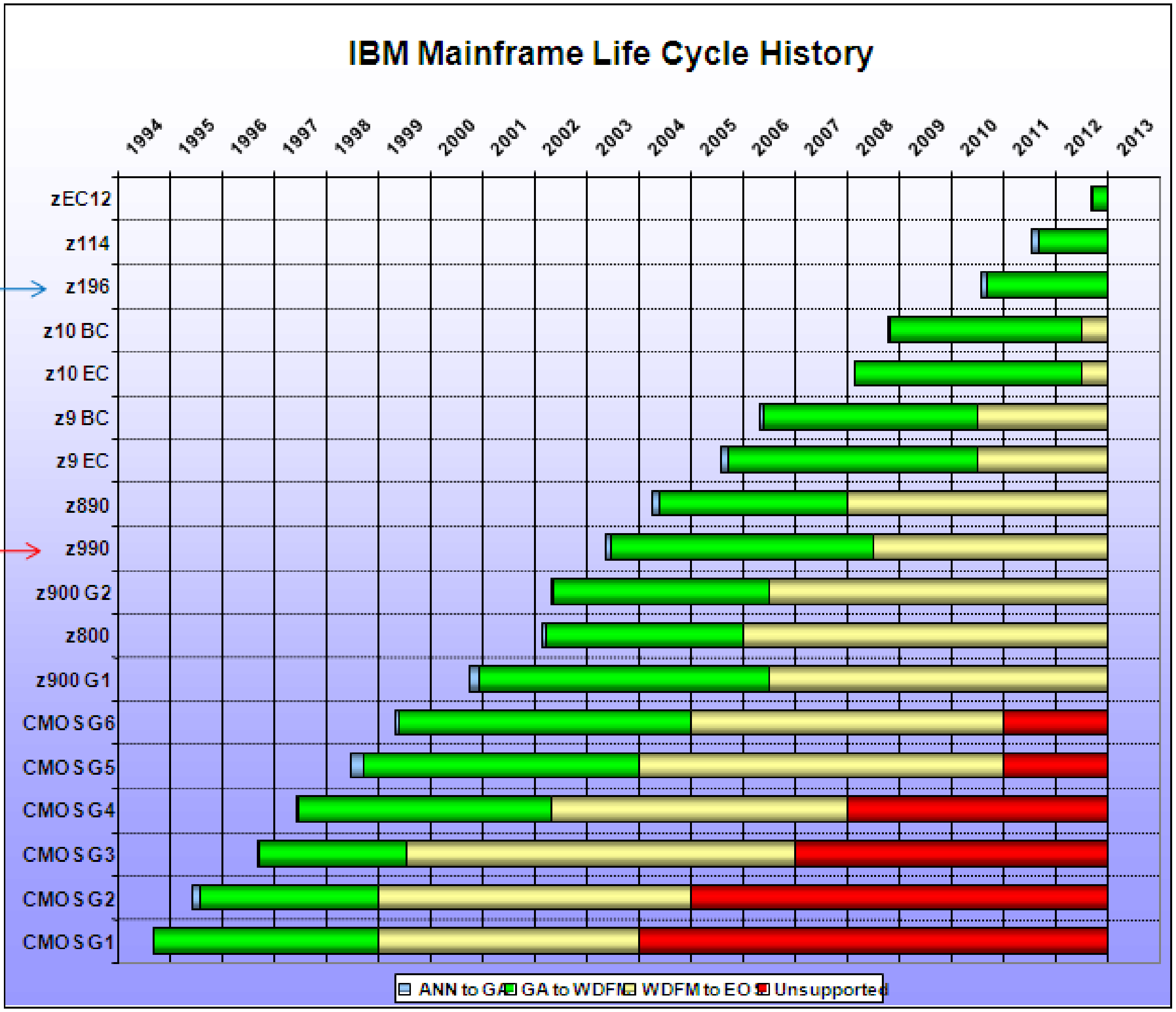
Potential Enhancements Recommended

From Class Material

- **Inclusion of block and superblock analysis in the ABAP coverage model.**
- **Identification of branch bound paths to get to a particular block or node in the code itself.**
- **Identification of the values of a test case that will exercise the block or node that has not been exercised yet.**
- **Inclusion of statistics for P-Case and C-Case usage in the Extensions and custom code generated for interfaces, conversions and other custom objects.**

Technical Discussion

Hardware Obsolescence



Legend
ANN Announcement of a new product
GA General Availability of a product
WDFM Withdrawal From Marketing
EOS End of Service

“As-Is” IT environment issues

- A good number of products are either **“out of currency”** or approaching that state. Loss of support has consequences – significant impact on problem solution, service levels...or, cost for special product support extensions

- The present MF hardware is several years old and is at increasing risk of incurring **hardware failures**

- Present software subsystems and applications might have **scalability** issues as volumes increase in the near to medium term.

- Selectively revising some applications might prevent some **performance** issues

MF Obsolescence Reduction (MFOR) Program

MF Obsolescence Reduction (MFOR) Program

Extended Support

New HW

SW Stack and Selective Application Migration

Scalability

Functionalities

Performance

Proj. A

Issues: a good number of products are either "out of currency" or approaching that state.

Actions: extend product support for products where possible.

Risks: N/A.. Just a cost.

Proj. B

- **Issues:** HW is the foundation upon which everything else depends If HW fails, everything fails.

Actions: the current system can either be encapsalated (cocooned) in a partition of new HW (via some SW layers: z-VM). This will greatly mitigate the HW failure concern given ongoing improvements in technology reliability.

Risks: The additional SW layer will introduce additional overhead. Additional capacity requirements would have to be evaluated.

Proj. C

Issue: present SW subsystems and applications might have scalability issues as volumes increase in the near to medium term. Exploitation of new or different functionalities are expected to be requested as well. Selectively revising some applications might prevent some performance issues.

Actions: more recent versions of Sw have parameters and feautres to help solve part of the above issues. Some applications need to be revised though, to make more efficient use of the system and subsystems new features.

Risks: migrations carry some risks that should be managed. Careful testing of fundions and regression evaluation should be done.

A: *Necessary... Risk of not doing it is too high.*

B: *The only way to reduce very pervasive impacts from HW failures.*

C: *Updating the SW stack is the way to help prevent the described issues over the longer-term.*

Application Defficiencies (Project C)

IN MAINFRAME (PRODUCTION)

Production Stack: Batch Cobol Code
Product
Product
Product
Product
Product
Product
Product
Product
Product

IN MAINFRAME (DEVELOPMENT)

Configuration Tool:
Product
Product
Product
Product
Product
Product
Product
Product
Product

Basic Application Architecture:

The System was ported from an RPG/AS400 code base to batch COBOL, it essentially executes the way an RPG system would but under the 370 Batch Architecture.

Each product is elaborated in the Configuration Tool by a group of business analysts and batch code is generated in that tool.

Integrator estimates that between 100K and 200K of code are generated for each product. Client has approximately 250 products so the duplicated code base is around 25-35M Lines of Code

Prototyped key strategies:

a. **Parsing, Lexical and Instrumented analysis on 4 products**
b. **Theoretical analysis of reconfiguration of the basic product to reduce application footprint & turn time in generation of new products**

The initial results yielded the following observations:

1. **By blocking out code that was never used approximately 80%+ of memmory was released.**
2. **Similar efficiency in execution of the code in production (reduction of time) results were secured.**

Rearchitecting the Application

OLD Code Base

IN MAINFRAME (PRODUCTION)

Production Stack: Batch Cobol Code

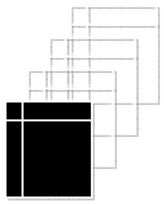
- Product
- Product
- Product
- Product
- Product
- Product
- Product
- Product
- Product

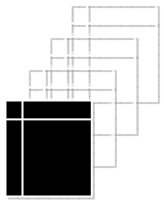
IN MAINFRAME (DEVELOPMENT)

Configuration Tool:

- Product
- Product
- Product
- Product
- Product
- Product
- Product
- Product
- Product

NEW Code Base





The application code base would be reworked into four separate code bases based on the product line supported. Preliminary studies have shown that a high degree of parameter similarity exists within the products there rather than between the other lines.

The Configuration tool would in addition be rearchitected to:

1. Make uniform the usage of each field and reducing the number of blank spaces currently found, thereby reducing the parameter footprint
2. Potentially migrating to a dedicated Power Processor (when the reduction is achieved) to fully be able to scale on that platform rather than the Mainframe.

The Production Code Base would be rearchitected to:

1. Enhance the compiler´s capabilities to add instrumentation to each product so that coverage in production could be computed & determination which variables and code could be removed.
2. Usage of special purpose tools to monitor coverage and specific testing as referenced in *X*suds (Telcordia/IBM technologies) to gain a high degree of competence space compliance prior to migrating into production.

Overall: the migration could be done either on a product by product basis, product line family or product line, depending on volume and complexity constraints. This would avoid any kind of knife edge cut over & would support elasticity in the program plan.