ARGO

Quality Assurance Illustrations and Principles

argodata.com

Mark Bentsen,
CTAL, CSTE, PMP, ASQ CMQ/OE
QA Manager

ARGO Mission Statement

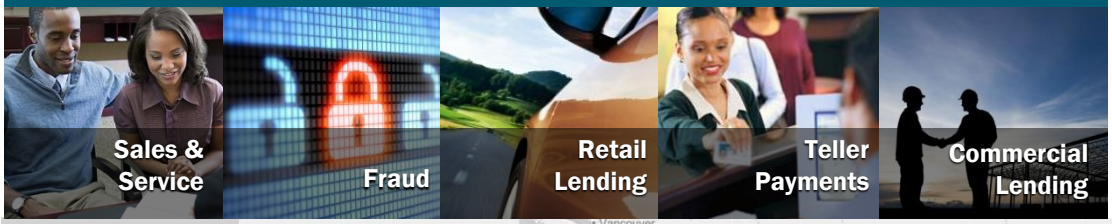
To improve business processes for the financial services and healthcare industries using software with mission-critical, real-time, and analytical competencies, resulting in revenue expansion, cost reduction, better patient and customer experience, and greater efficiency.

© 2016 ARGO Data Resource Corporation. All rights reserved.

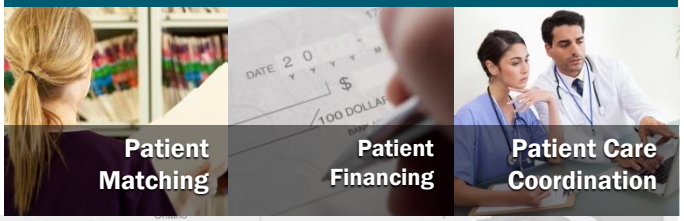
ARGO at a Glance – What We Do

Mission-critical application software

FINANCIAL SERVICES



HEALTHCARE



Operational Footprint

32,500 operating locations
301,500 workstations
100 million daily transactions
35 billion annual transactions

Strength and Longevity

Founded in 1980

Privately held

Revenue: $58 million

Assets: $134 million

No debt

44% of revenue invested in product R & D over last five years

ARGO Customers

- Financial services – Direct
- Fraud – Direct and resale
- Healthcare – Direct
- Financials services - Resale

EDMS Monitoring
22,613 geographic points in 45 states
150,000 workstations and servers
79 million daily/25 billion annual transactions

© 2016 ARGO Data Resource Corporation. All rights reserved.

Financial Services Customers

ARGO has over 100 additional customer installations through our resale partners.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Fraud Customers

ARGO has over 200 Fraud customers in the U.S. and Canada through direct and resale partners including…

BancFirst Corporation
Bank of the West — BNP Paribas Group
BOK Financial Corporation
BNP Paribas
CIBC — For What Matters.
City National Bank — The way up.
Deutsche Bank
EastWest Bank
First Republic Bank
Flagstar Bank
Hancock Bank
KeyBank
The Private Bank
Rabobank
Scotiabank
ScottradeBank
Signature Bank
svb Silicon Valley Bank

© 2016 ARGO Data Resource Corporation. All rights reserved.

Healthcare Customers

© 2016 ARGO Data Resource Corporation. All rights reserved.

Technical Assets & Capabilities

User Experience

B2B	Internet Banking	CSR/ Teller	Banker Mobility	Back-Office Support	Smartphone App	Commercial Banker
HTML/JS	HTML/JS	HTML/JS	Universal Win Platform	HTML/JS, Winform/ MFC	Native IOS/ Android	HTML/JS, Semantic Web

Application Services

Workflow & Collaboration
- Configurable Workflow
- Dynamic Routing & Assignment
- Notifications
- Doc Generation
- Doc Management
- eSignature
- Image Analytics
- OCR

Interoperability
- Web Services Integration (REST/SOAP)
- Legacy I/O
- 3rd Party Interfaces
- Adapters & Extensions
- Data Access Layer
- Device Integration

Transaction Processing
- Reusable Core Transaction Services
- High Availability (5-9) Processing
- Highly Scalable to Enterprise Demands
- Offline Processing

Decision Support & Analytics
- ARGO Decision Engine
- Intelligent Questionnaire Machine Learning
- AI + Language Understanding (Chat-bot)
- Entity Matching & Resolution

Security
- SSO (AD/LDAP)
- Multi-Factor Authentication
- OAuth 2.0
- TLS/Digital Cert
- Sql Encryption
- OWASP Positive Web-Security Model
- DMZ Abstraction
- User/System Audit

Deployment
- On-Premise
- Cloud/Hosted
  - Kubernetes
  - Docker
- Windows Server
- Internet Information Server
- SQL Server
- Virtualization (VMWare ESXI)

Configuration
- Configurable Business Rules
- Application Editors
- Systems Management Console
- Intelligent Scripting
- Extensibility

Data Services

Management Insight (OLAP)
- Analytics: SSAS
- Insight: Power BI
- Reporting: SSRS

OnlineTransaction Processing (OLTP)
High Availability Clustered SQL Server, AlwaysOn

Operational Reliability
- Proactive, Real-Time Monitoring of 250+ KPI
- Predictive –Reactive –Recovery
- Weekly / Monthly Reporting with Peer Analysis

© 2016 ARGO Data Resource Corporation. All rights reserved.

Presenter

Mark Bentsen

- Mark Bentsen is the Manager of Quality Assurance at ARGO Data, a software development company providing mission-critical and analytical solutions for financial services and healthcare. He leads a team of 18 engineers and three managers. Mark's org delivers products for fraud, teller payments, consumer lending, sales & service in banking, patient entity matching, patient care, and others. ARGO products use analytic driven technologies with a decision based engine. Certain products are currently transforming to a machine learning model.

- In 2003, Mark started at FedEx where he spent a decade in a variety of roles of increasing responsibility including his first management role.

- In 2015, he became part the Advanced Research Center for Software Testing and Quality Assurance at the University of Texas in Dallas (UTD). Mark presents on QA leadership, KPIs, and root cause analysis in local, national, and international software conferences. Mark is a PMP & CTAL (Full) from ISTQB.

- Mark & his wife Melissa are the two time, past President Couple of 'Better Marriages Texas' and have been active in Marriage Enrichment since they said "I do" in 2001. Prior to working in technology, he worked in YWAM & Mercy Ships in Switzerland and Namibia. He lives in Dallas with his wife and two boys' ages 12 and 16.

© 2016 ARGO Data Resource Corporation. All rights reserved.

What do we do?

- Reduce Risk & Eliminate Waste

- Effective **software testing** teams:
  - Build confidence
  - Reduce “Risk & Surprises”
  - Detect defects early
  - Provide valuable information

© 2016 ARGO Data Resource Corporation. All rights reserved.

What do we do?

- Reduce Risk & Eliminate Waste

- Effective **Quality Assurance** teams:
  - Identify risks
  - Prevent defects
  - Focus on continuous improvement of SDLC quality
  - Guard the company brand

© 2016 ARGO Data Resource Corporation. All rights reserved.

Test Progress Monitoring and Control

The Value of Testing – Capers Jones

Another poor measurement practice that has concealed the economic value of software quality is the usage of the cost-per-defect metric. It has become an urban legend that "it costs 100 times as much to fix a bug after delivery as during development." Unfortunately, the cost-per-defect metric actually penalizes quality and achieves its lowest values for the buggiest software. As quality improves, cost per defect rises until a level of zero defects is reached, where the cost-per-defect metric cannot be used at all.

The real economic value of high quality is only partially related to defect repair costs. It is true that high quality leads to fewer defects and therefore to lower defect repair costs. But its major economic benefits are due to the fact that high quality

- Reduces the odds of large-system cancellations
- Reduces the odds of litigation for outsourced projects
- Shortens development schedules
- Lowers development costs
- Lowers maintenance costs
- Reduces warranty costs
- Increases customer satisfaction

© 2016 ARGO Data Resource Corporation. All rights reserved.

Quality

"Quality in a product or service is not what the supplier puts in. It is what the customer gets out and is willing to pay for. A product is not quality because it is hard to make and costs a lot of money, as manufacturers typically believe. This is incompetence. Customers pay only for what is of use to them and gives them value. Nothing else constitutes quality."

– Peter Drucker

© 2016 ARGO Data Resource Corporation. All rights reserved.

How do you know Quality is Important?

- **If** quality **then** quality drives the
  - Thinking
  - Decisions
  - Actions

© 2016 ARGO Data Resource Corporation. All rights reserved.

When Quality is not Important

- When quality **is not** important:
  - Calling a release GA (ready) when you cannot deliver its primary functionality to a customer
  - The team is committed to releasing the code on a specified date at all costs
  - Incrementally adding significant defects to the product/code base release over release – sprint over sprint
  - Looking to QA for owning quality. It's not my problem.
  - When quality is a discussion topic, there is silence across the project team

© 2016 ARGO Data Resource Corporation. All rights reserved.

When Quality is Important

- Quality Activities:
  - Preventing defects is a priority
  - Training for the team
  - Root cause analysis and follow through
  - Retrospectives have results
  - Improving the quality of stories, specifications, requirements, is important. Time is made to do it right.
- Improving SDLC quality
  - Eliminating rework
  - Metrics used to make continuous improvements. Measuring yourself regularly. Wanting to improve.
  - Increasing ‘First time, done right’.
- Accountability – Everyone appreciates their role in quality and is active in doing it

© 2016 ARGO Data Resource Corporation. All rights reserved.

What is Testing?

The Types of Testing (Green Circles)

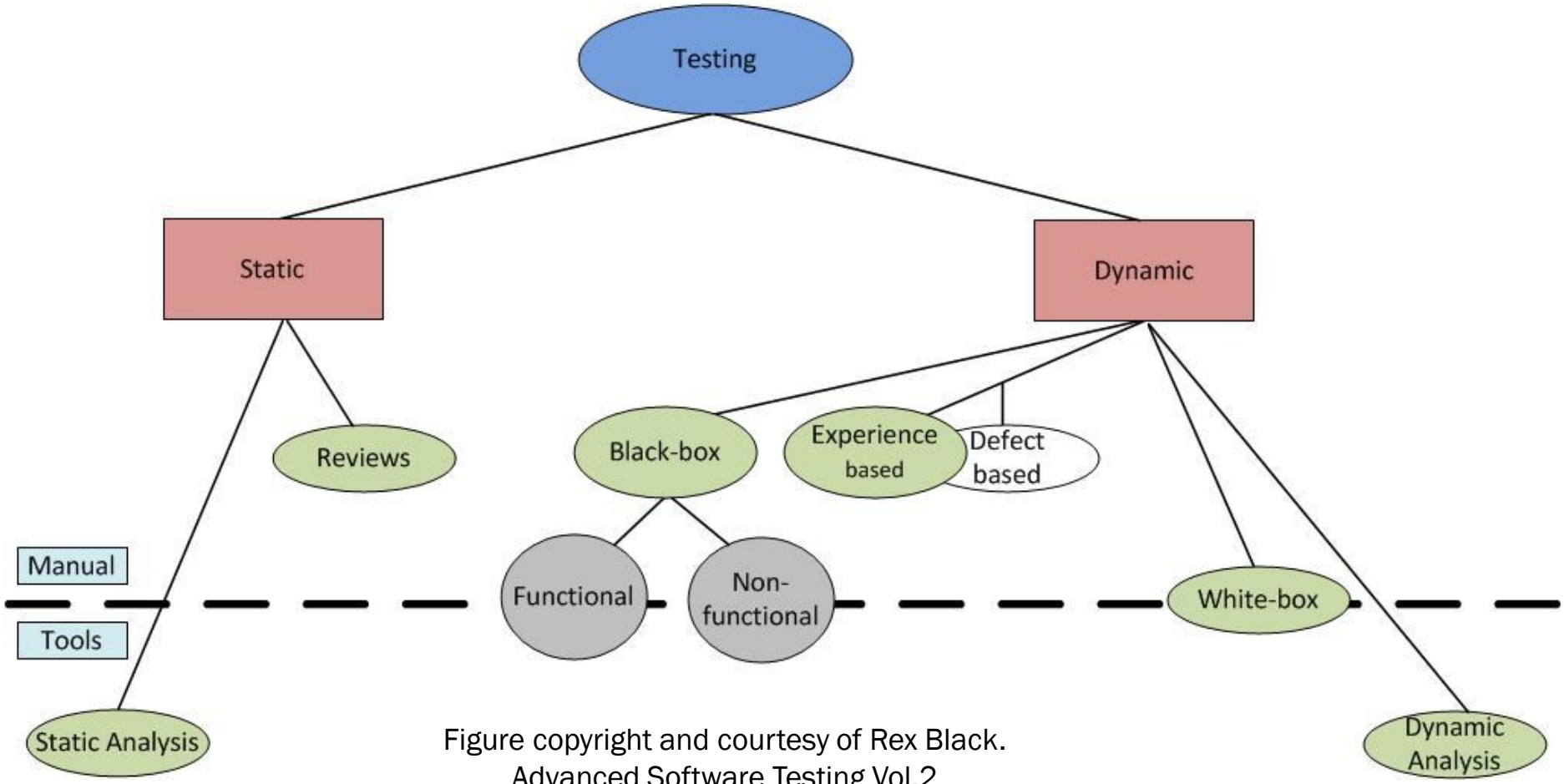
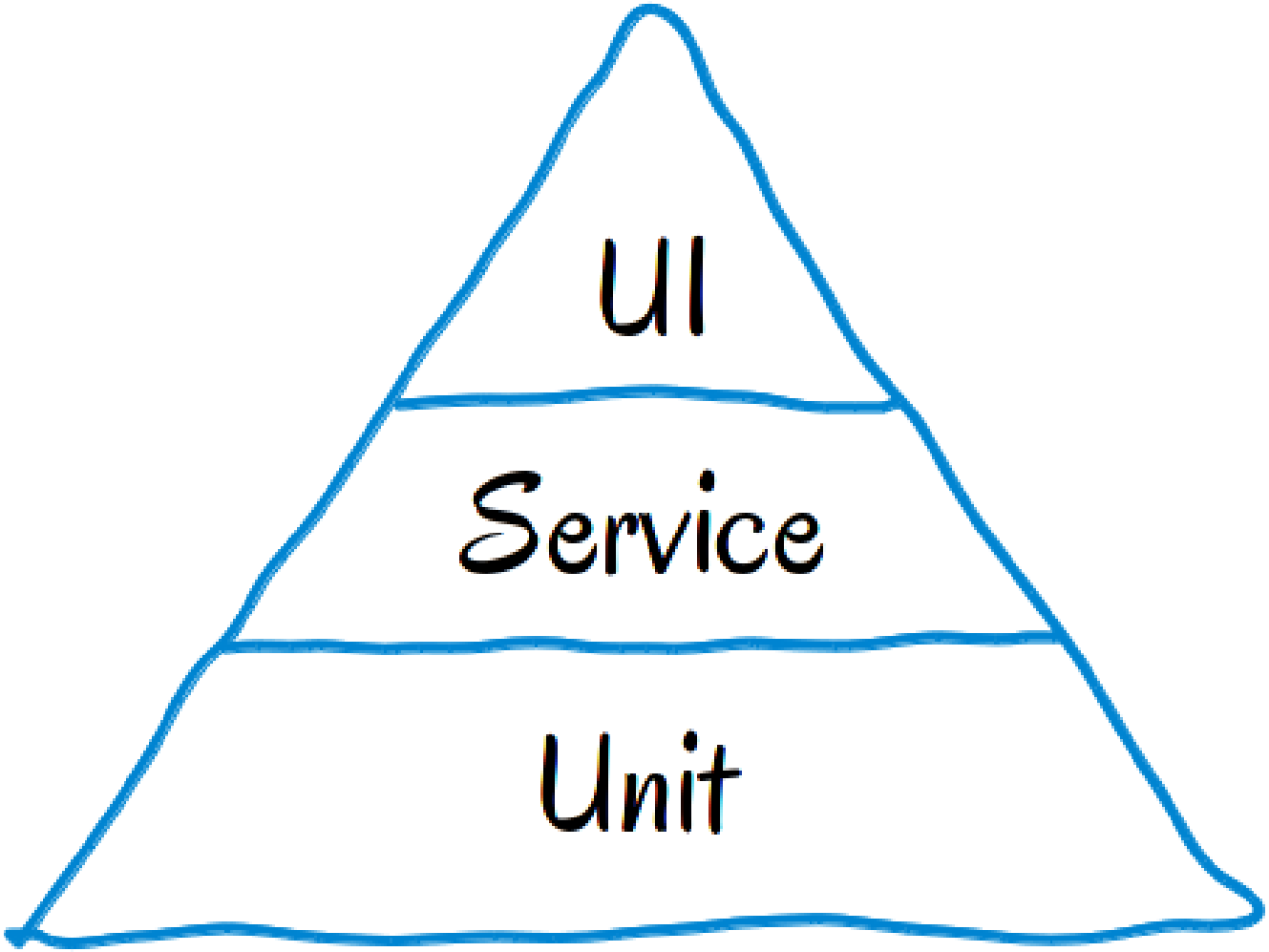


Figure copyright and courtesy of Rex Black.
Advanced Software Testing Vol.2

© 2016 ARGO Data Resource Corporation. All rights reserved.

Prioritizing Automation

A test automation vision



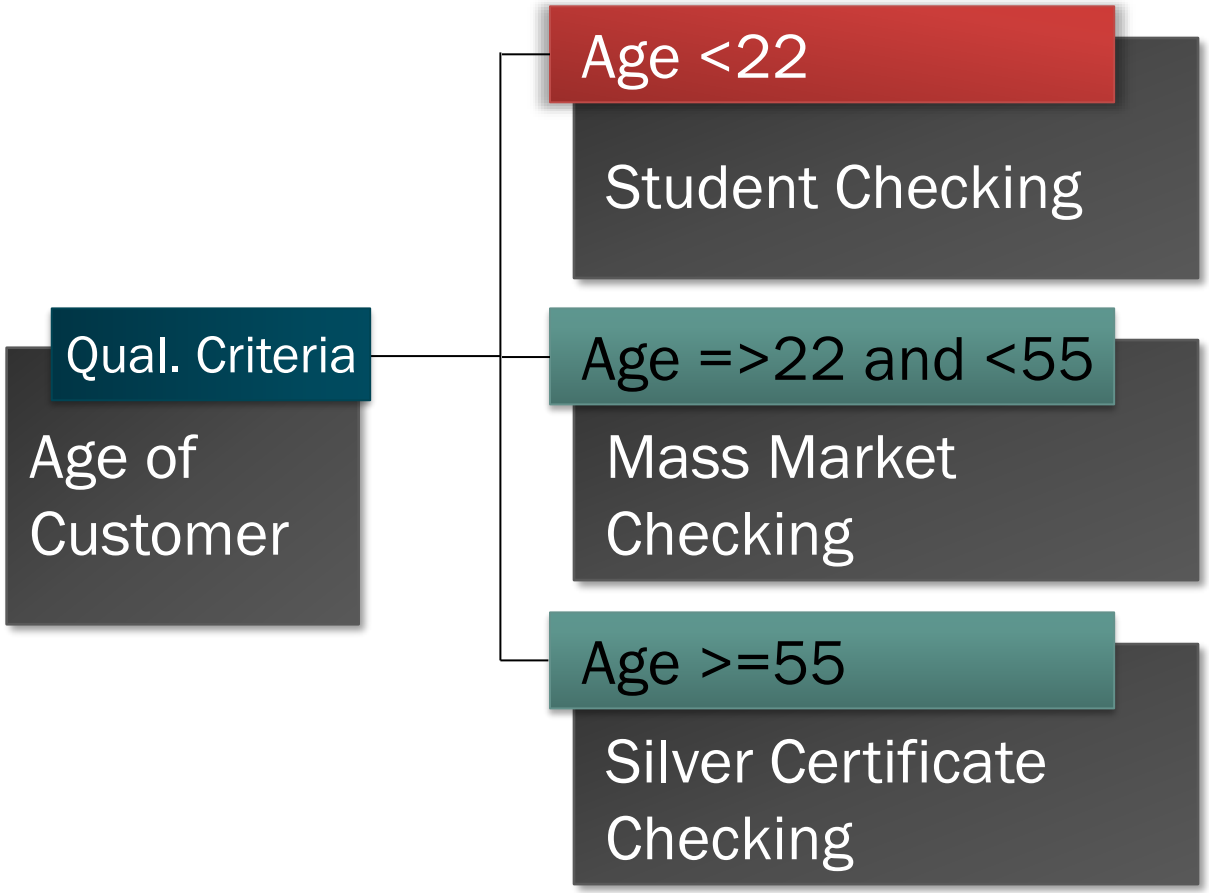
© 2016 ARGO Data Resource Corporation. All rights reserved.

The value proposition of unit/component testing

How to Get development Peers ‘On-Board’ with Quality Practices in Development Workflows.

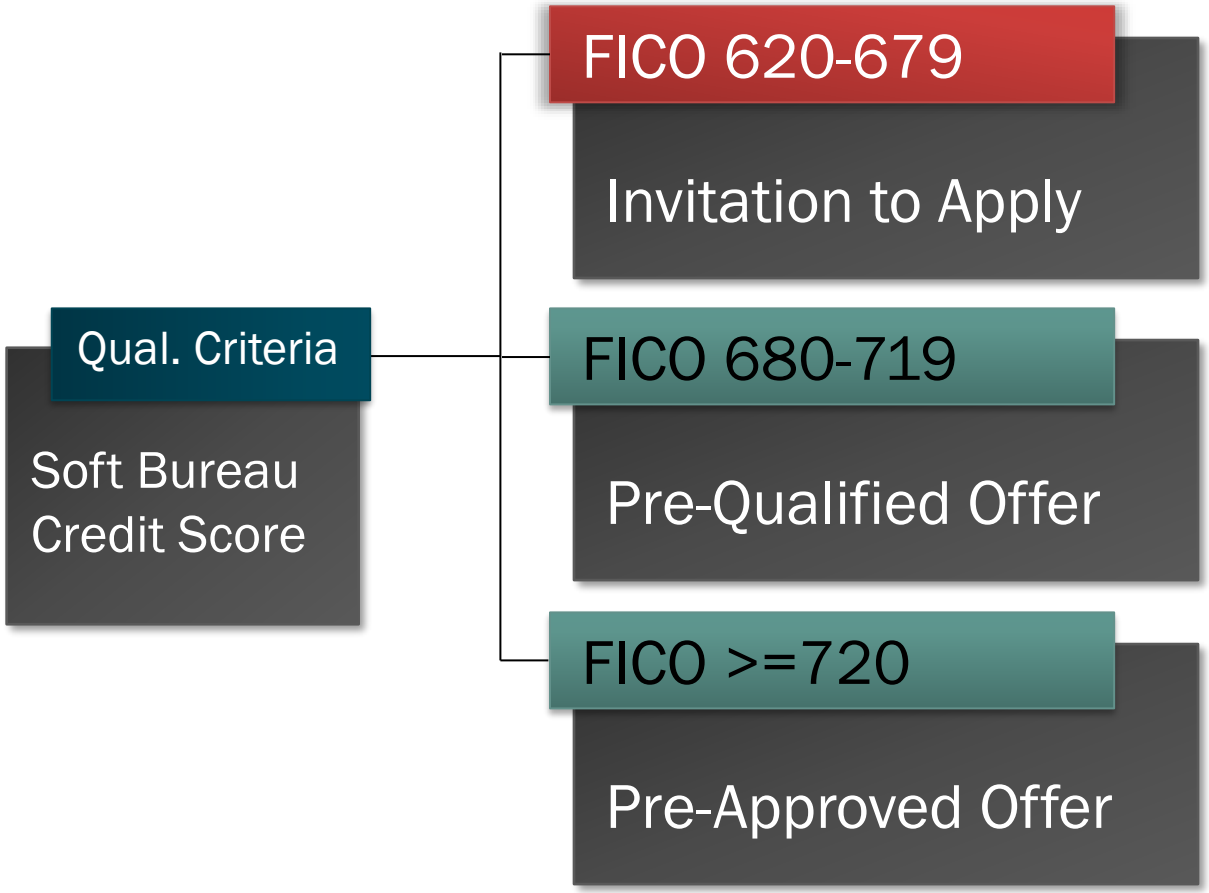
© 2016 ARGO Data Resource Corporation. All rights reserved.

Unit Tests - Checking Account Type



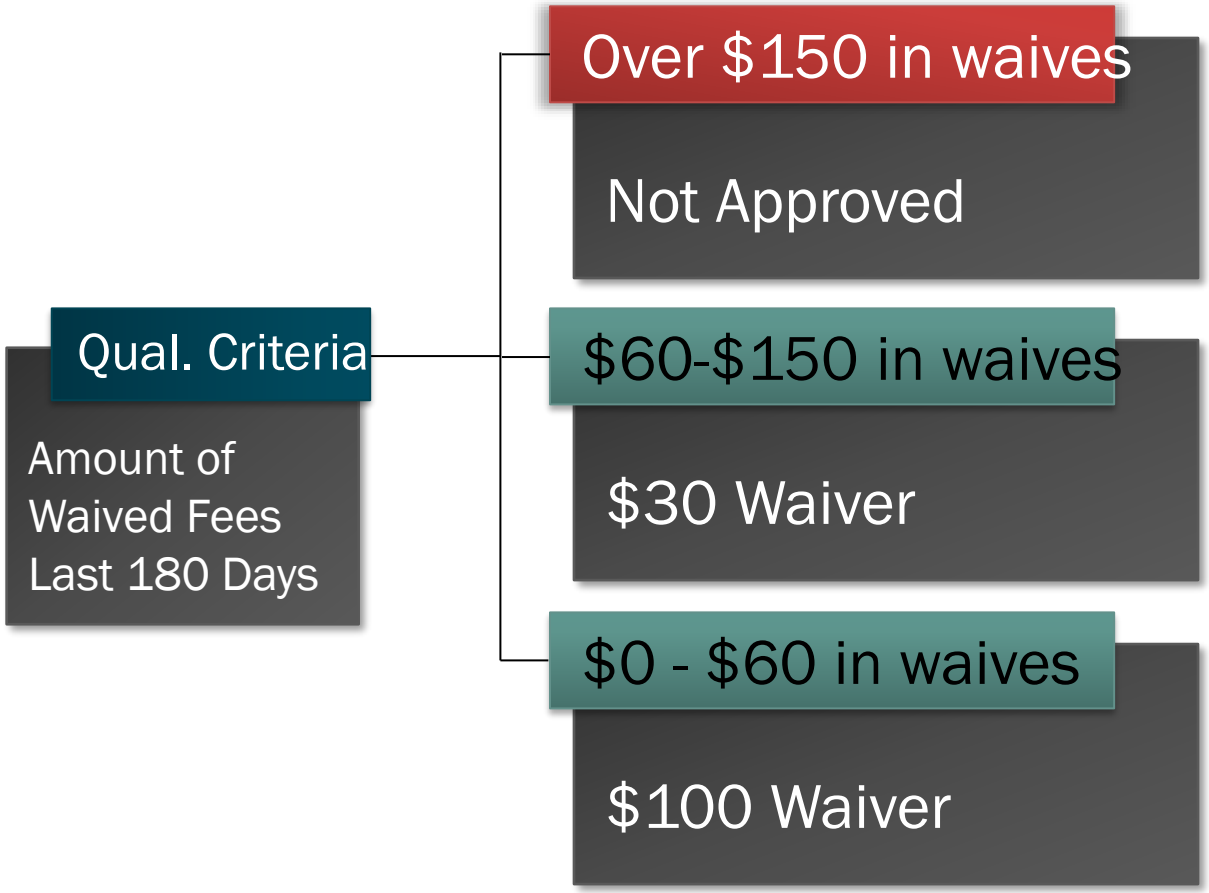
© 2016 ARGO Data Resource Corporation. All rights reserved.

Credit Card Prequal



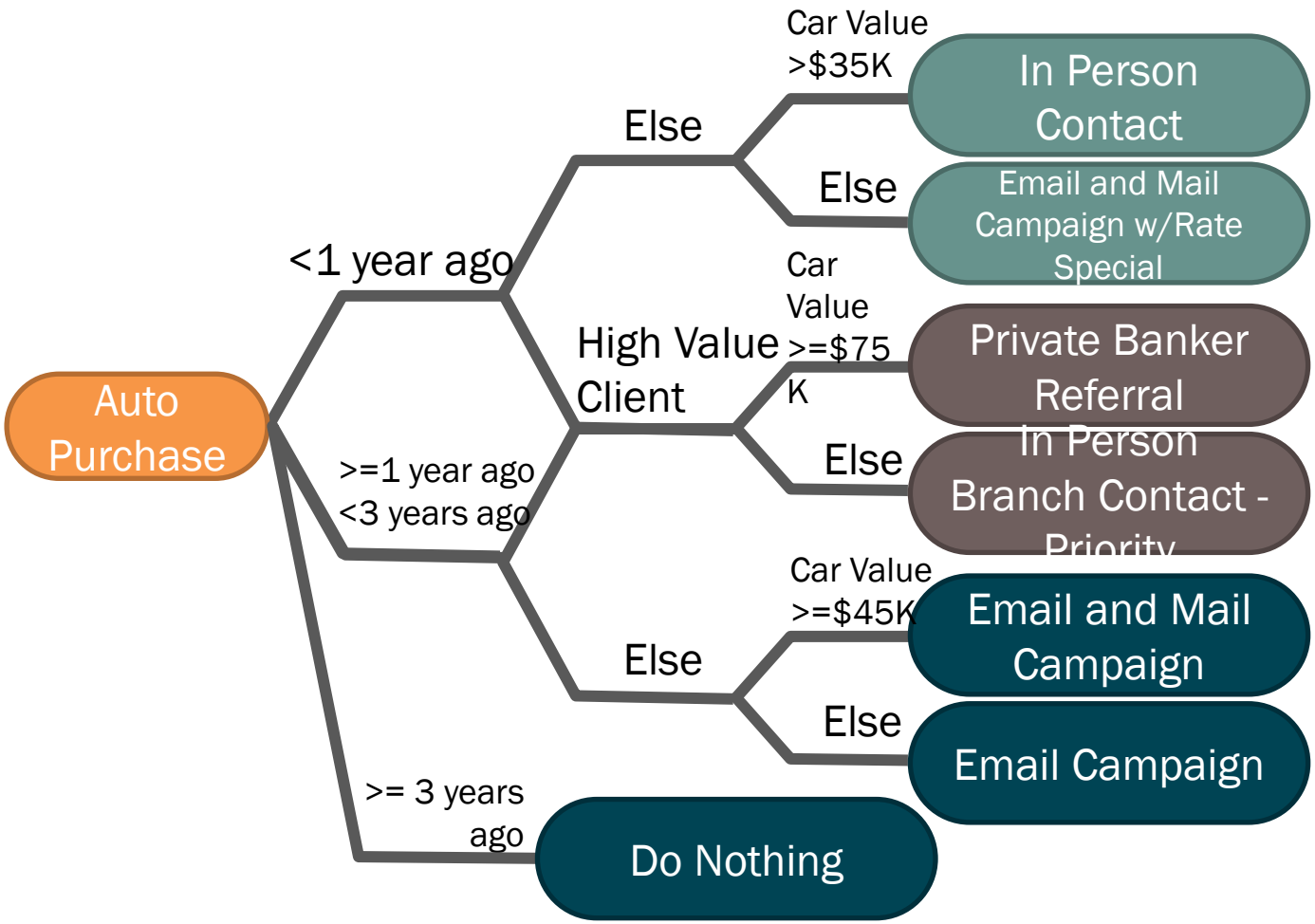
© 2016 ARGO Data Resource Corporation. All rights reserved.

Waive Fee Decisioning



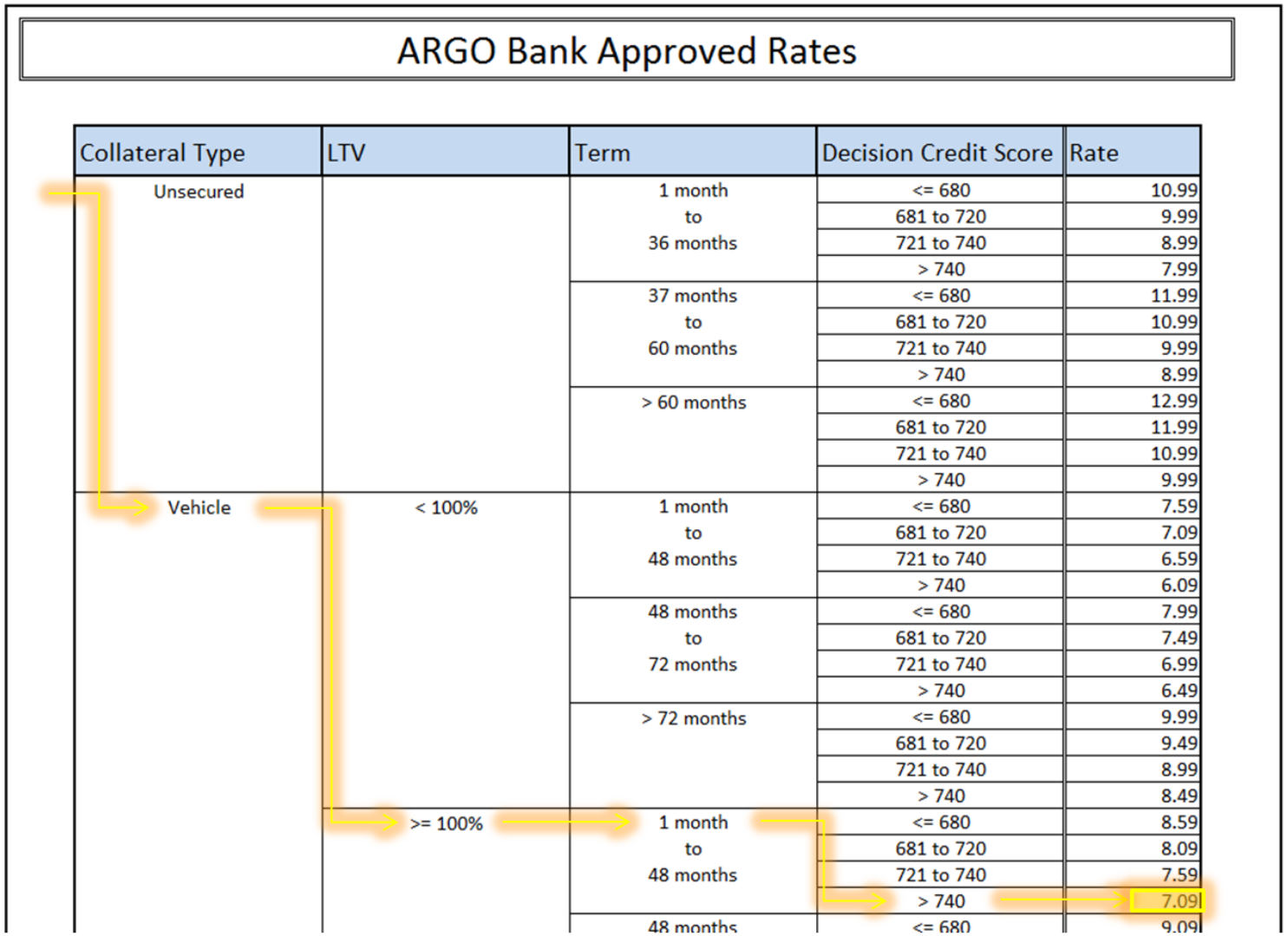
© 2016 ARGO Data Resource Corporation. All rights reserved.

Auto Refi Campaign



© 2016 ARGO Data Resource Corporation. All rights reserved.

Risk Based Pricing



ARGO Bank Approved Rates

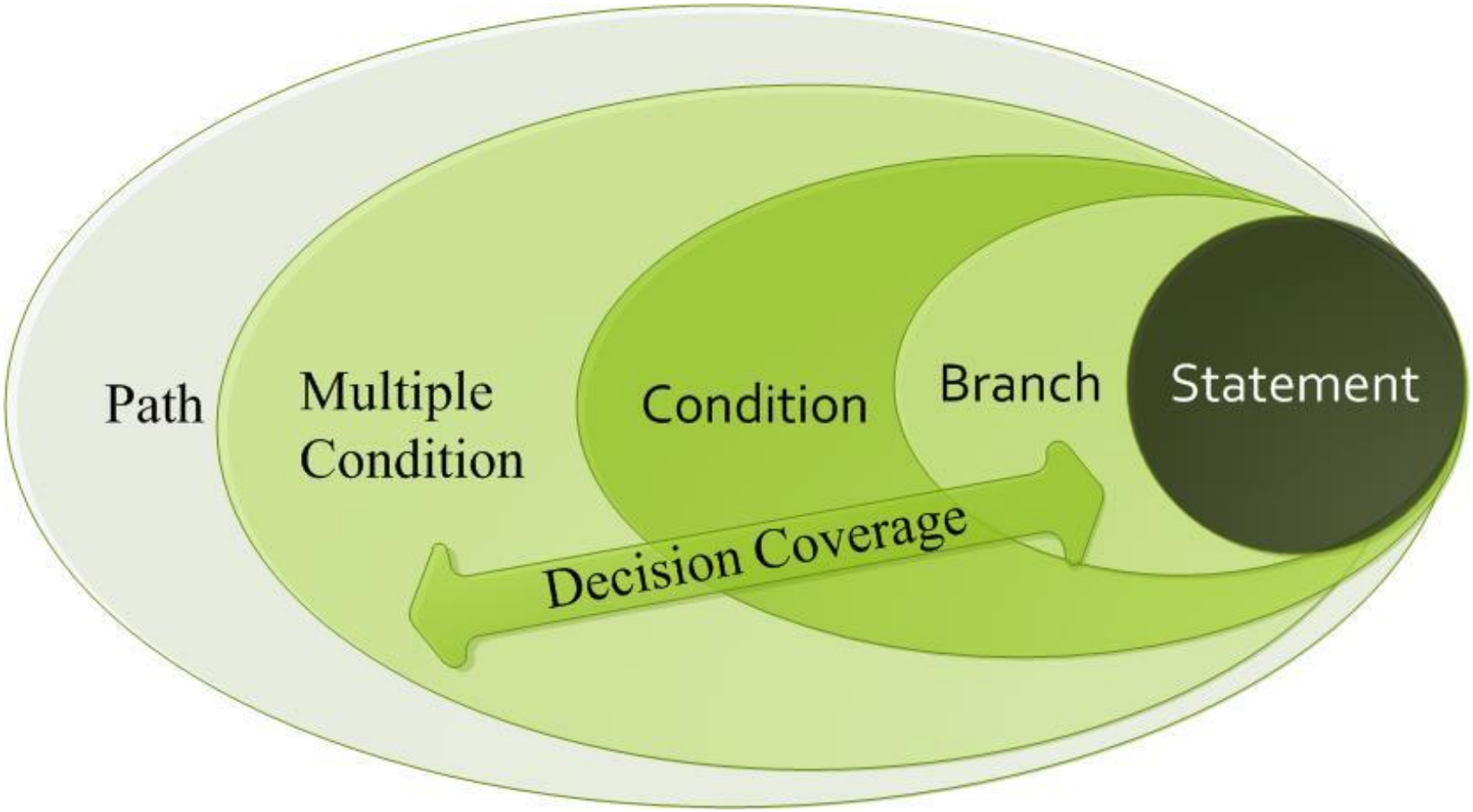
Collateral Type	LTV	Term	Decision Credit Score	Rate
Unsecured		1 month to 36 months	<= 680	10.99
			681 to 720	9.99
			721 to 740	8.99
			> 740	7.99
		37 months to 60 months	<= 680	11.99
			681 to 720	10.99
			721 to 740	9.99
			> 740	8.99
		> 60 months	<= 680	12.99
			681 to 720	11.99
			721 to 740	10.99
			> 740	9.99
Vehicle	< 100%	1 month to 48 months	<= 680	7.59
			681 to 720	7.09
			721 to 740	6.59
			> 740	6.09
		48 months to 72 months	<= 680	7.99
			681 to 720	7.49
			721 to 740	6.99
			> 740	6.49
		> 72 months	<= 680	9.99
			681 to 720	9.49
			721 to 740	8.99
			> 740	8.49
	>= 100%	1 month to 48 months	<= 680	8.59
			681 to 720	8.09
			721 to 740	7.59
			> 740	7.09
		48 months	<= 680	9.09

© 2016 ARGO Data Resource Corporation. All rights reserved.

Structure-based or White-box Techniques

Describe the concept and value of code coverage

Structural Test Coverage Levels



© 2016 ARGO Data Resource Corporation. All rights reserved.

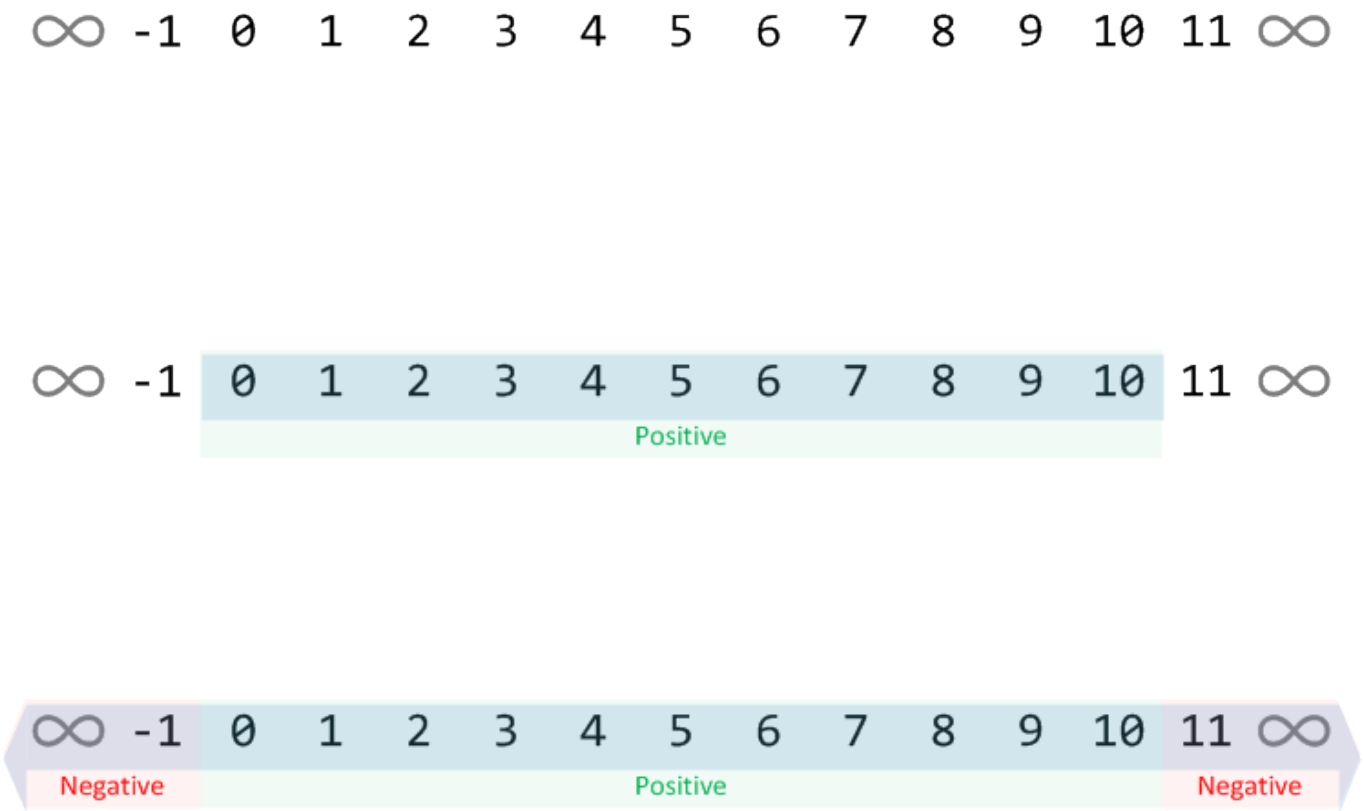
Equivalence & Boundary | Positive & Negative

© 2016 ARGO Data Resource Corporation. All rights reserved.

Equivalence & Boundary | Positive & Negative

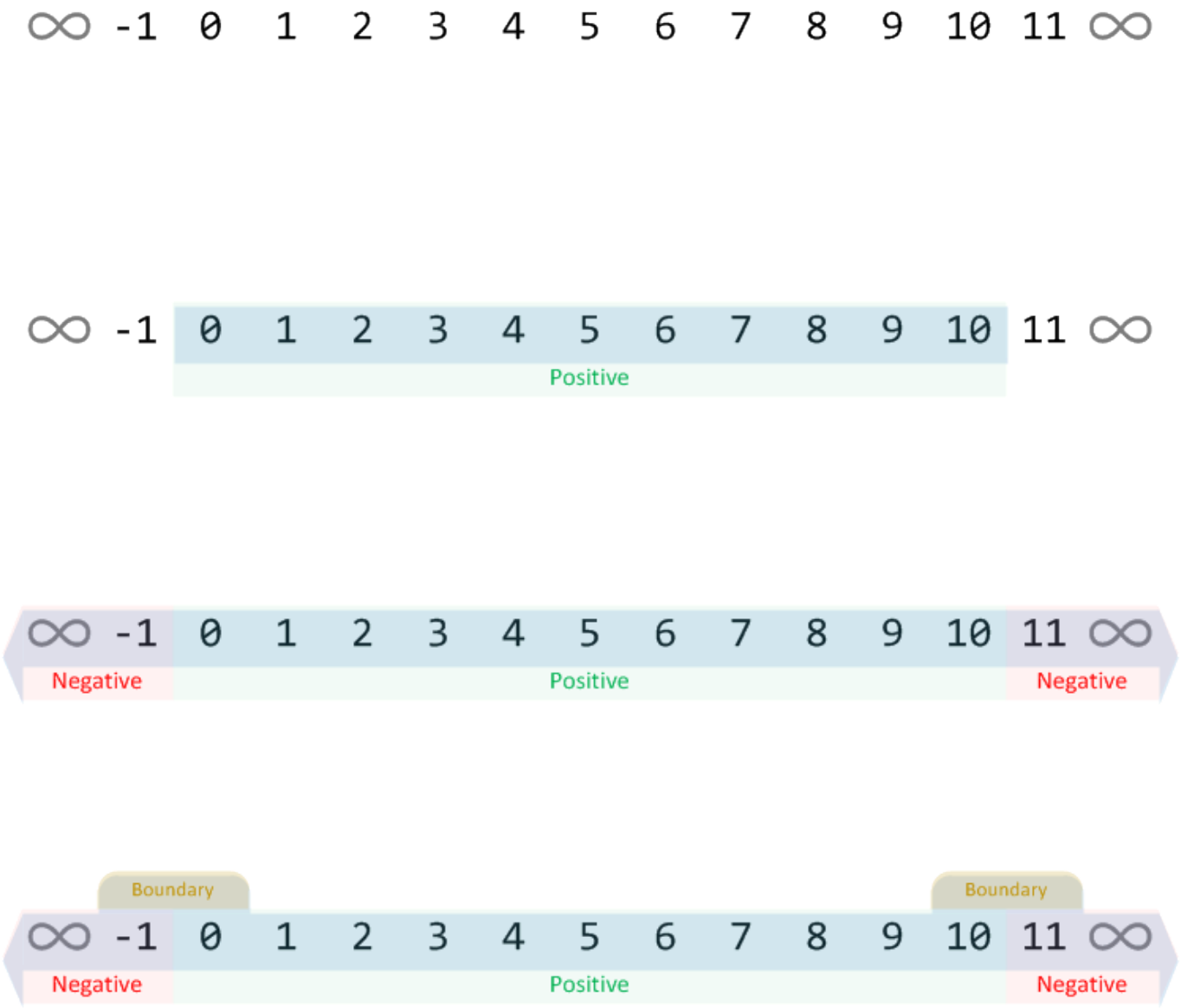
© 2016 ARGO Data Resource Corporation. All rights reserved.

Equivalence & Boundary | Positive & Negative



© 2016 ARGO Data Resource Corporation. All rights reserved.

Equivalence & Boundary | Positive & Negative



© 2016 ARGO Data Resource Corporation. All rights reserved.

Unit Testing Template

	A	B	C	D	E	F	G	H	I
1	Test Case #	Purpose	Step #	Action (Design Step)	Expected Result	Pass / Fail	Defect Tracking #	Notes	Test Data
28	24	Ensure proper display of telephone fields. For enterable fields, the user is only required to type the numbers. Formatting occurs when the user tabs off the control. The output edit used is PHONE, which has the pattern (ZZZ) 999-9999. All numbers have a length of 10 digits; leading zeros are applied if the user enters less than 10 digits.	1	Enter 10 digit phone number. Enter telephone number in data set, <TAB> Repeat for each telephone number.	Field is displayed in the pattern (ZZZ) 999-9999.				
29	24	Ensure proper display of telephone fields. For enterable fields, the user is only required to type the numbers. Formatting occurs when the user tabs off the control. The output edit used is PHONE, which has the pattern (ZZZ) 999-9999. All numbers have a length of 10 digits; leading zeros are applied if the user enters less than 10 digits.	2	Enter less than 10 digits (0, 1, 9)	Field is displayed in the pattern (ZZZ) 999-9999 with leading zeroes.				
30	25	Ensure proper edits of tax ID number fields. Tax ID numbers are greater than or equal to 001000000.	1	Enter tax ID number in data set, <push button to validate> Repeat for each tax ID number.	Field is displayed as ?				
31	25	Ensure proper edits of tax ID number fields. Tax ID numbers are greater than or equal to 001000000.	2	Enter each invalid SSN/TIN number.	Entry is not allowed.			The following SSN/TIN numbers are invalid for entry/search: 000000000, 111111111, 222222222, 333333333, 444444444, 555555555, 666666666, 777777777, 888888888, 999999999, 123456789, 987654321.	
32	25	Ensure proper edits of tax ID number fields. Tax ID numbers are greater than or equal to 001000000.	3	Enter SSN/TIN number less than 001000000.	Entry is not allowed.				

© 2016 ARGO Data Resource Corporation. All rights reserved.

Unit Test - Maturity Model

CMM	Unit Test Level	Details
Level 1 Initial	Level 0 - Unaware	Unaware of unit testing concepts or missing fundamental skills to develop unit test.
	Level 1 - Ignored	A belief that not enough time is available for unit testing or that it would not bring benefit to the specific work at hand.
	Level 2 - Experimental	Experimentation of basic unit test concepts, typically positive scenarios. Missing strategy as to coverage areas. Typically used by creator of test and not others within the organization. Likely not maintained for reuse..
Level 2 Repeatable	Level 3 - Intentional	Intentional effort to build some unit test in places throughout the development lifecycle. May not represent test scenarios outside positive (happy path) testing.
	Level 4 - Positive/Negative Test	Intentional effort to build positive and negative unit test throughout the development lifecycle. Understanding of testing principals beyond positive (Happy Path) testing techniques.
Level 3 Defined	Level 5 - Positive/Triangulation Test	Specific test with different input and expected results than the positive test to ensure no hard coded return results.
	Level 6 - Positive/Negative/Boundary Test	Intentional effort to build effective unit test leveraging appropriate testing principals such as Positive, Negative and Boundary testing. Effective communication channels in place between development and QA.
	Level 7 - Mocks and Stubs	Mocks and Stubs in place to replicate dependent functionality.
	Level 8 - Designed for Testability	Code that is easier to test due to development design. Clear delineation and simplicity in design.
	Level 9 - Test Driven Development	Begin development process by building unit test which evolve with primary code development. Designed for testability. Red, Green, Refactor. Never write a line of code that doesn't have a failing test.
Level 4 Managed	Level 10 - Code Coverage	Intentional effort to build unit test to measurably cover functionality, logic and lines of code across the development.
	Level 11 - Unit Test in the Build	Automated unit testing during the build process (CI). All Unit Test must pass in order to consider the build successful.
	Level 12 - Code Coverage Awareness	Awareness of Unit Test code coverage across an organizations landscape ensuring consistency in testing practices. High level dashboards showing metrics down to individual projects regarding code coverage and last execution times.
Level 5 Optimizing	Level 13 - Automated Builds and Tasks	Fully automated build and reporting process. Bringing awareness to the collective and individual health of the SDLC process.

Web Services Tests

- Read the Web services description language (WSDL)
- Give the user a GUI to submit data
- Create an XML file from the selections that the user has made
- Send the XML request to the Web Service
- Receive the XML response from the Web Service
- Display the results

© 2016 ARGO Data Resource Corporation. All rights reserved.

Web Services Tests

Assertions – Automated Test Scripts

- Data
- Read the Web services description language (WSDL)
- Give the user a GUI to submit data
- Create an XML file from the selections that the user has made
- Send the XML request to the Web Service
- Receive the XML response from the Web Service
- Display the results
- Certainty of the Expected Result

© 2016 ARGO Data Resource Corporation. All rights reserved.

Web Services Tests

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:parameters.sellyourjunk.com"
xmlns:xm="http://www.w3.org/2005/05/xmlmime">
 <soapenv:Header/>
 <soapenv:Body>
   <urn:SubmitItem>
     <urn:attachmentInfo>
       <urn:fileName>GadgetPhoto.zip</urn:fileName>
       <urn:itemPrice>1000</urn:itemPrice>
       <urn:itemDescription>Really cool gadget!</urn: itemDescription>
       <urn:accountNumber>ABC123DEF</urn: accountNumber>
      </urn:attachmentInfo>
      <urn:attachmentFile
xm:contentType="application/?">cid:123604199920</urn:attachmentFile>
   </urn:SubmitItem>
 </soapenv:Body>
</soapenv:Envelope>
```

Listing 1

© 2016 ARGO Data Resource Corporation. All rights reserved.

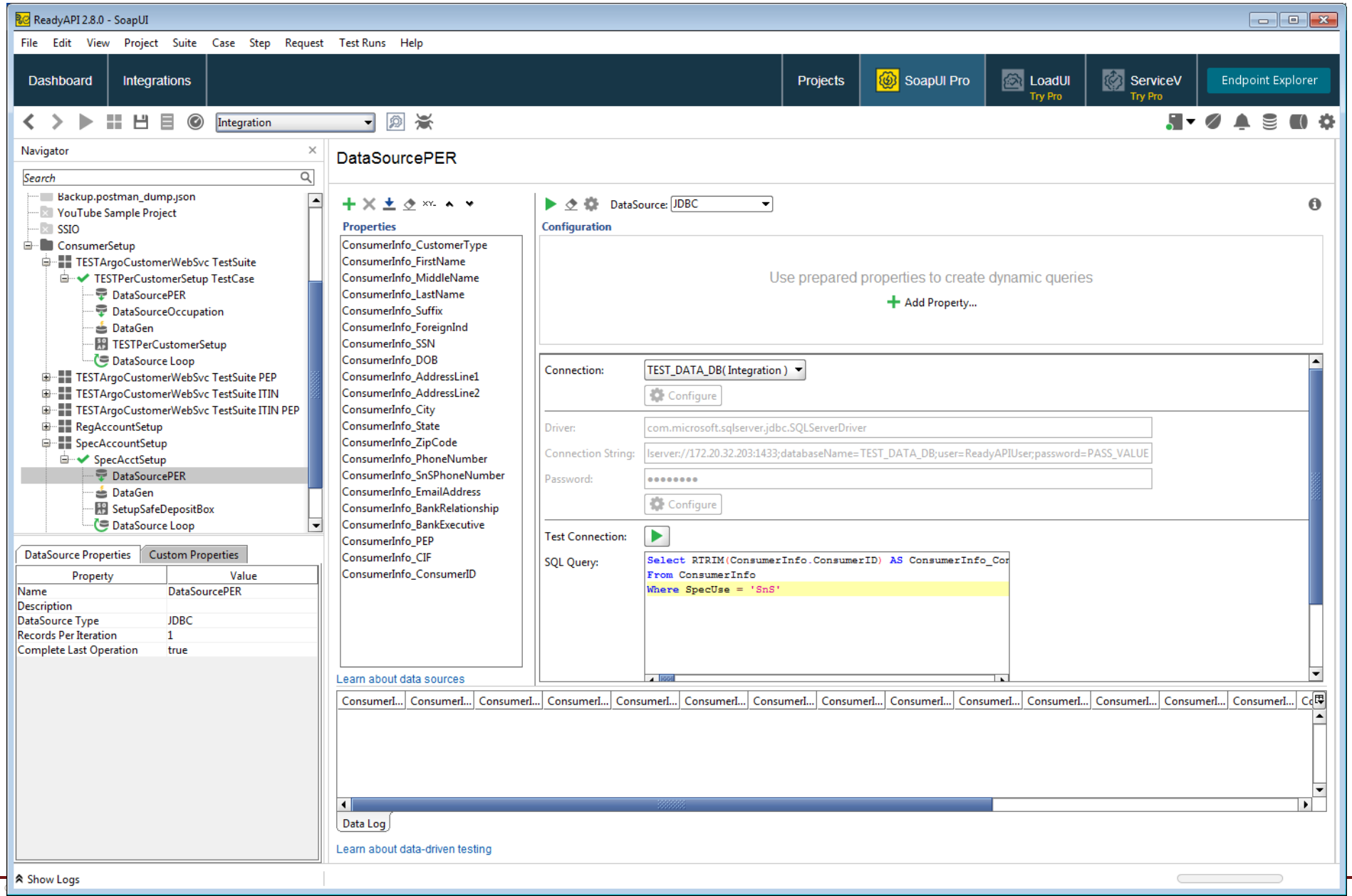
Web Services Tests

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns1:ReceiptResponse xmlns:ns1="urn:sellyourjunk.com">
      <ns1:return>
        <ns1:statusCode>0</ns1:statusCode>
      </ns1:return>
     </ns1:ReceiptResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

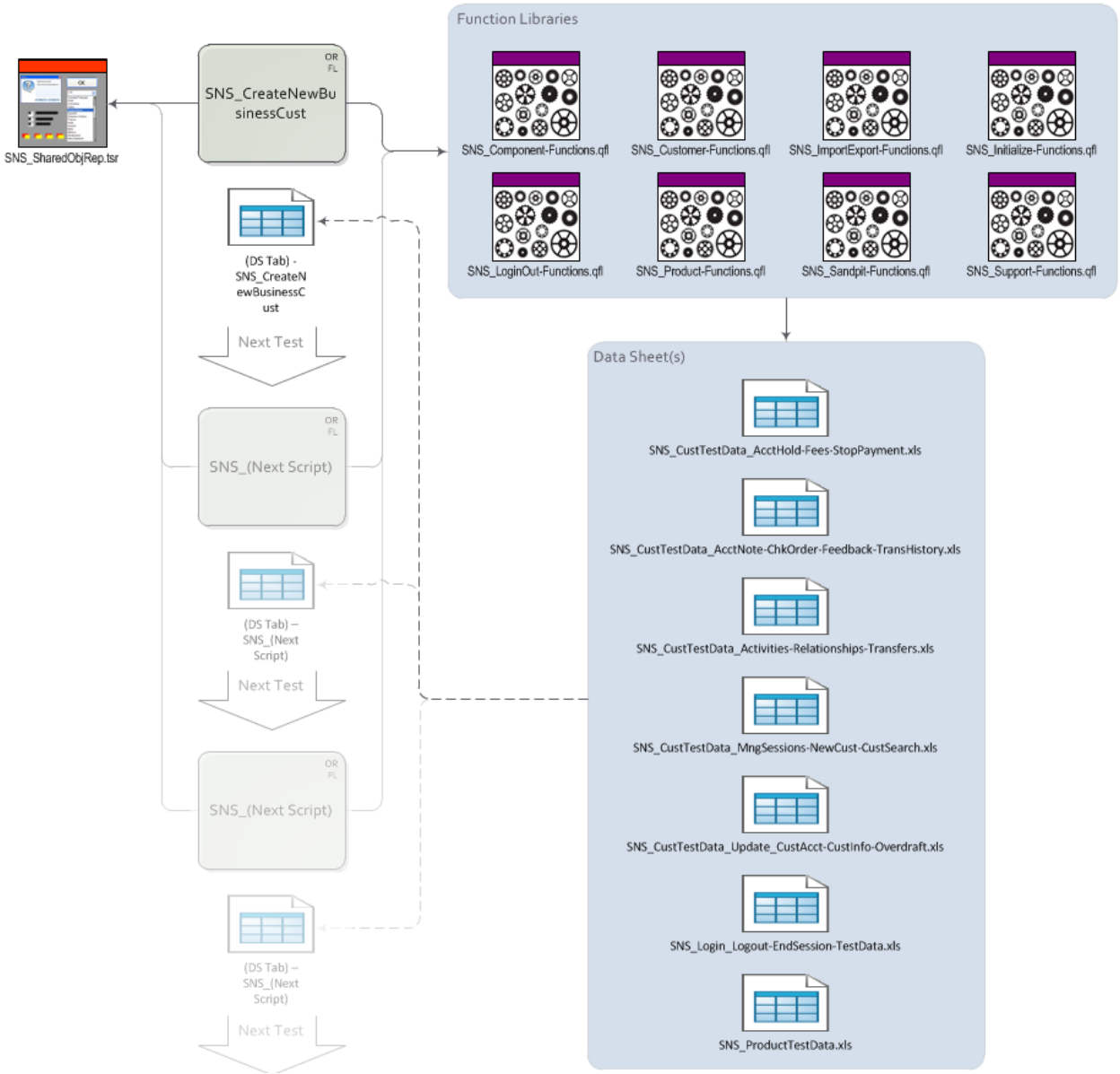
Listing 2

© 2016 ARGO Data Resource Corporation. All rights reserved.

Web Services Data Integrity

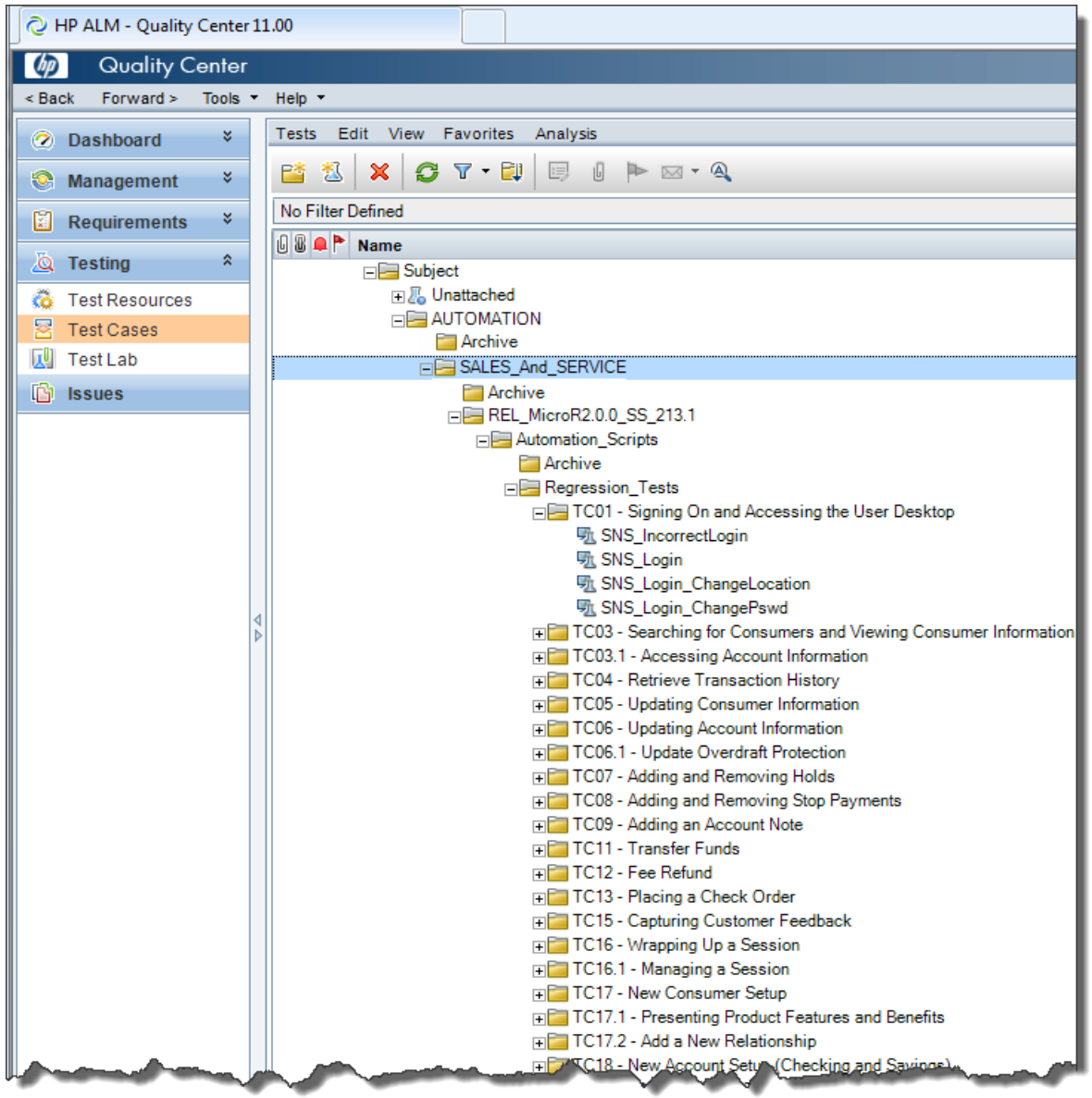


UI Automation Tests



© 2016 ARGO Data Resource Corporation. All rights reserved.

UI Automation Tests



© 2016 ARGO Data Resource Corporation. All rights reserved.

UI Automation Tests

Name	Test: Test Name	Type
[1]SNS_Login	SNS_Login	QUICKTEST_TEST
[1]SNS_CIFSearchCust	SNS_CIFSearchCust	QUICKTEST_TEST
[1]SNS_ManualAddCustToSession	SNS_ManualAddCustToSession	QUICKTEST_TEST
[1]SNS_SearchCustByAcct	SNS_SearchCustByAcct	QUICKTEST_TEST
[1]SNS_SearchCustByTaxIDNum	SNS_SearchCustByTaxIDNum	QUICKTEST_TEST
[1]SNS_SearchProspect	SNS_SearchProspect	QUICKTEST_TEST
[1]SNS_SearchCustReqFieldErr	SNS_SearchCustReqFieldErr	QUICKTEST_TEST
[1]SNS_ManagingCustSession	SNS_ManagingCustSession	QUICKTEST_TEST
[1]SNS_RemoveCustFromSession	SNS_RemoveCustFromSession	QUICKTEST_TEST
[1]SNS_EndCustSession	SNS_EndCustSession	QUICKTEST_TEST
[1]SNS_EndSessionAddCust	SNS_EndSessionAddCust	QUICKTEST_TEST
[1]SNS_IncludeProfilesEndSession	SNS_IncludeProfilesEndSession	QUICKTEST_TEST
[1]SNS_Logout	SNS_Logout	QUICKTEST_TEST

Name	Test: Test Name	Type
[1]SNS_Login	SNS_Login	QUICKTEST_TEST
[1]SNS_SearchCustByName	SNS_SearchCustByName	QUICKTEST_TEST
[1]SNS_SearchCustByAcct	SNS_SearchCustByAcct	QUICKTEST_TEST
[1]SNS_Logout	SNS_Logout	QUICKTEST_TEST

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **Test automation should use designated machines**
- Test Automation can easily be interrupted from executing properly. The most common reason for this type of interruption is when a user is trying to use their everyday PC for daily common task while at the same time trying to execute test automation from it. At least one machine should be setup for kicking off test automation within automation tool and separate machine(s) setup to execute the automation code against the application under test.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **automation tool to instantiate the application under test (AUT)**
- It's critical that the automation tool instantiates the application it is interacting with to ensure the automation tool has full visibility of the application objects. To resolve situations whereby the automation engineer is encountering scenarios when objects seem to be recognized sometimes and blind to those objects at other times.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **The realities of developed software testing developed software**
- A significant challenge in any developed test automation is the fact that you are using developed software (test automation) to test developed software (AUT). Bugs can exist both in the test automation as well as the application under test. Try to keep the test automation as simple and straight forward as possible. Consider refining overly complicated test automation code into simpler approaches. Add logging when appropriate to track what the automation is doing and what the results of the test verification/validation have been.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **Overly abstract test automation**
- Automation tools can bring great opportunities to a test team when the automation is sustainable and maintainable over the long haul. A pressing challenge for any automation effort is to not let the development get so complex that it is no longer easy to work with and understand. Test assets that reference other test assets can easily add abstraction to the automation effort and make it more and more difficult to understand. Strive to keep the automation as straight forward and simple as possible with the best advice being to follow the automation vendors intended way to use the tool as it was designed.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **Trying to account for every contingency (Exception Handling)**
- A common mistake in the development of test automation is to try to account for every contingency that the automation might encounter during the test execution. This can easily lead to more exception handling code than the primary code used to execute the test cases. Excessive error handling code can also mask real errors encountered with the application under test. Keep exception handling to a minimum, erring on the side of the test automation stalling if an unknown exception is encountered. When coupling this approach with good logging, it will bring awareness to where bugs may exist within the application under test.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **Common code blocks**
- When designing an automation framework (or any automation for that matter) it's important to consider maintainability and sustainability to ensure the automation can be reused and kept updated as easily as possible. One way to do this is to develop a consistent approach to the way the code is built. By building very **reusable code blocks**, it's possible to easily modify or relocate areas of the script that may need maintenance, enhancements or updates. It's important when making changes to the automation to continue in this methodology of using common code blocks to enable future automation engineers to easily maintain the solution going forward.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **Disable system locking**
- A challenge encountered with many automation environments is to prevent it from going into a Locked or Logged Off state when system inactivity is encountered. Working with system administrators, test machines can be configured to never go into a Locked or Logged Off state. Configure the automation tool as well as the test execution machines to not go into the Locked or Logged Off state unless this is intentional by the user.

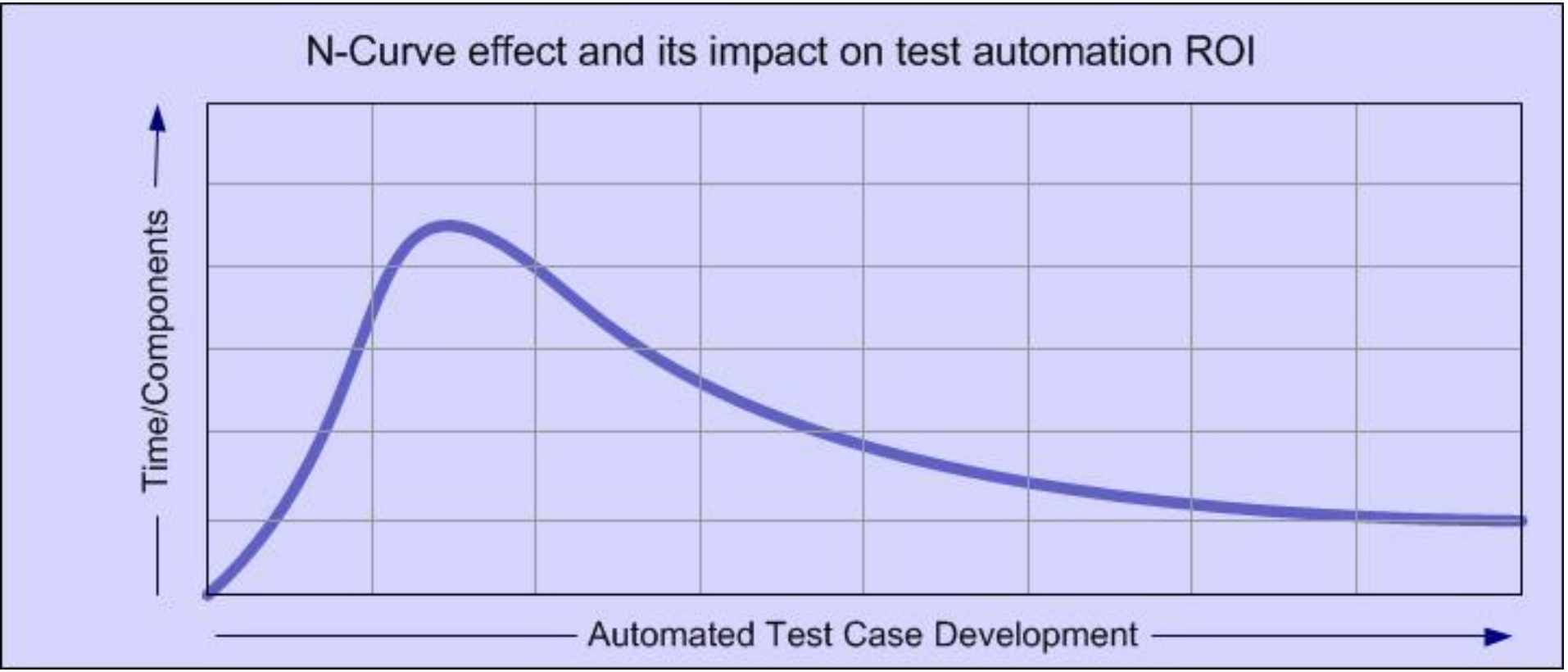
© 2016 ARGO Data Resource Corporation. All rights reserved.

Lessons Learned

- **Stub Scripts – Pulling the test resources together**
- A stub script essentially pulls the resources together and gets the test connected to the Automation Framework. From a code perspective, it's very minimal but it's critical to connect all the pieces that are necessary to perform the test.

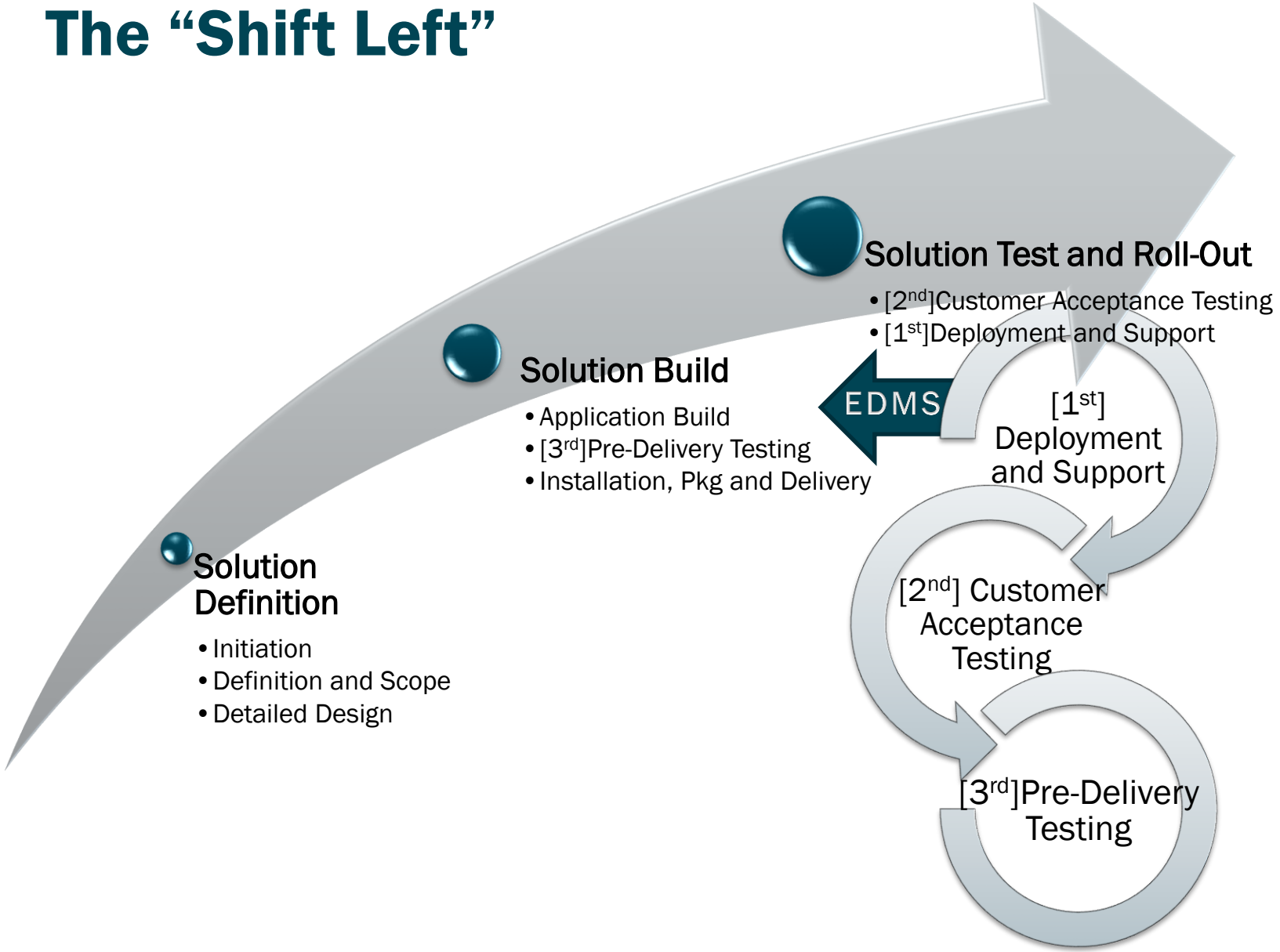
© 2016 ARGO Data Resource Corporation. All rights reserved.

UI Automation Tests



© 2016 ARGO Data Resource Corporation. All rights reserved.

The "Shift Left"



© 2016 ARGO Data Resource Corporation. All rights reserved.

Testing Scanner

- ARGO manual testers rely heavily on visual UI verification to find errors and defects.
- There are other mechanisms available for verification that are not at the UI and are available to assist testers in finding defects.
  - Logs and traces are available to be monitored by EDMS at ARGO.
  - This data provides insight into events that alert of errors in the system.
  - This intelligence is not exposed to testers today.

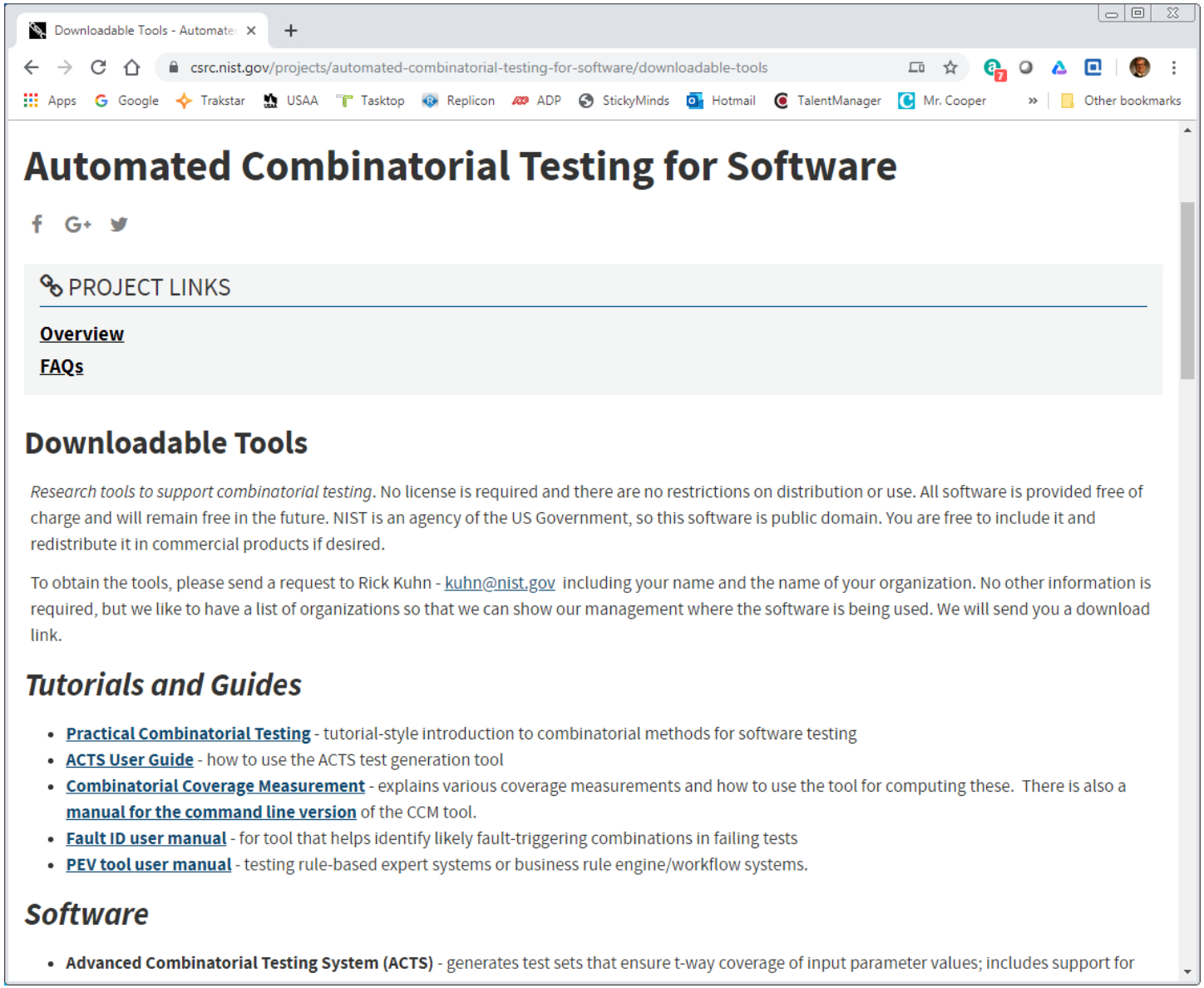
© 2016 ARGO Data Resource Corporation. All rights reserved.

Testing Scanner

- An application that provides the tester with insight to defects that is not available today.
- Objectives:
  - Detect events
  - Determine the source user
  - Notify the user
  - Capture trace data

© 2016 ARGO Data Resource Corporation. All rights reserved.

Automated Combinatorial Testing for Software



© 2016 ARGO Data Resource Corporation. All rights reserved.

Five Whys

- Work backward from the problem to identify the root cause.
- Ask “Why does this happen?”
- For each answer ask why again.
- Continue until the reason is no longer related to the problem.
- Typically requires asking “Why” five times.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Non-technical Example

- I have a flat tire
- Because I have a nail in my tire
- Because I drove through a construction site on my way to work.
- Because it's the only way to get to work.
- Root Cause: I have a flat tire because I drove through a construction site on my way to work and drove over a nail.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Appendix I - Primary Contributing Cause

"**Primary Contributing Cause**", found in the Quality Control Status tab, captures the root cause for the defect. Additional supporting info is to be included in the defect's **Comments**. Primary Contributing Cause is to be assigned during or before dev's **Fixed** status.

Root Cause – Primary Contributing Causes Defined | Quick Reference Ver. 02

Data Handling/Data Management – Data handled improperly, causing issues where data is not validated, defined, transformed, masked/encrypted consistent with ARGO published standards or specifications.

Environment – Error produced by combination of hardware, configuration, or code version discrepancy. Includes compilation, build errors and failed code pushes where application runtime files not updated properly.

Exception Handling – Architectural, global or functional exception handling contaminated. Missed exceptions causing issues in otherwise issue-free logic. Unusual situations not handled non-destructively creating cascading issues.

Insufficient Skill / Knowledge – Issue that originated from programmer's lack of skill or knowledge on line of business, application, or development methodology resulting in failure to effectively complete the task.

Logic / Decisioning Errors – Business logic not correctly interpreted programmatically. Application therefore does not follow decisions, policies, or explicit intent in specified requirements.

Failed to Conform to Specification – Issue transpired from developer misinterpreting accurately specified requirement, programming per own understanding. Application functions without error, but not as designed.

Failed to Conform to Standards – Failed to conform to published standards for UI or other physical attribute behavior.

Performance – Issue pertaining to memory leaks, data volume, architectural complexities, and ineffective processes, generally discovered during performance testing.

Process Deficiency – Process in SDLC is incomplete, ambiguous or too tolerant of errors resulting in issues that degrade quality of deliverables, communication and permits defects to manifest in application.

Serialization, Timing, Sequence – Issues exposed or created when dependencies between functions are not identified prior to subsequent development.

Requirements Unclear / Inadequate – Gaps in functional requirement specifications and inadequate design definitions cascading into additional issues further in the SDLC.

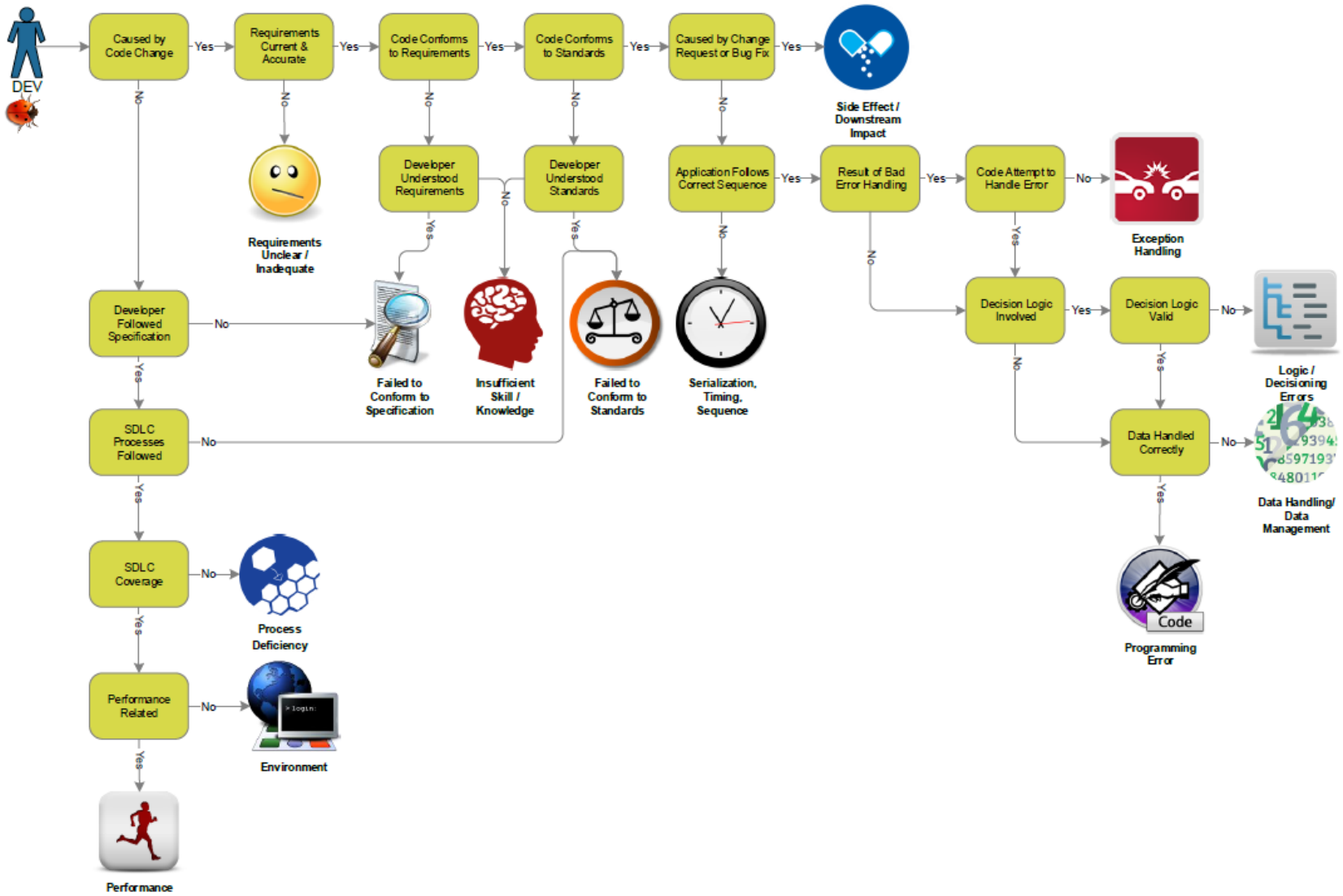
Side Effect / Downstream Impact – Issue caused inadvertently while making changes either in development of another function or while addressing another issue.

Programming Error – Error that originated during development which caused specified requirement to not function as designed.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Appendix I – Decision Flow to Assign Primary Contributing Cause

Root Cause – Primary Contributing Cause Decision Flow | Quick Reference Ver. 03

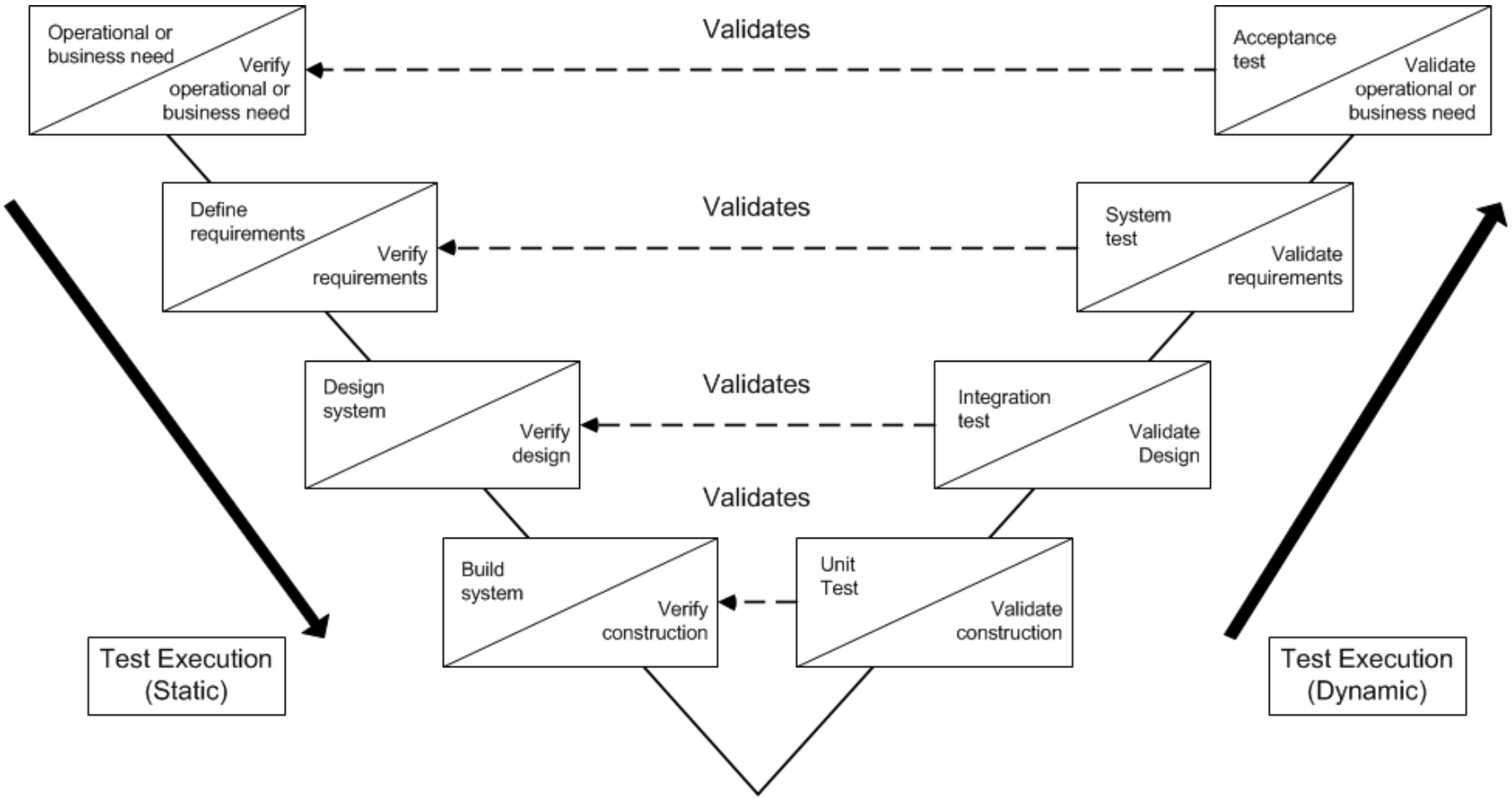


© 2016 ARGO Data Resource Corporation. All rights reserved.

Sequential Development V-Model

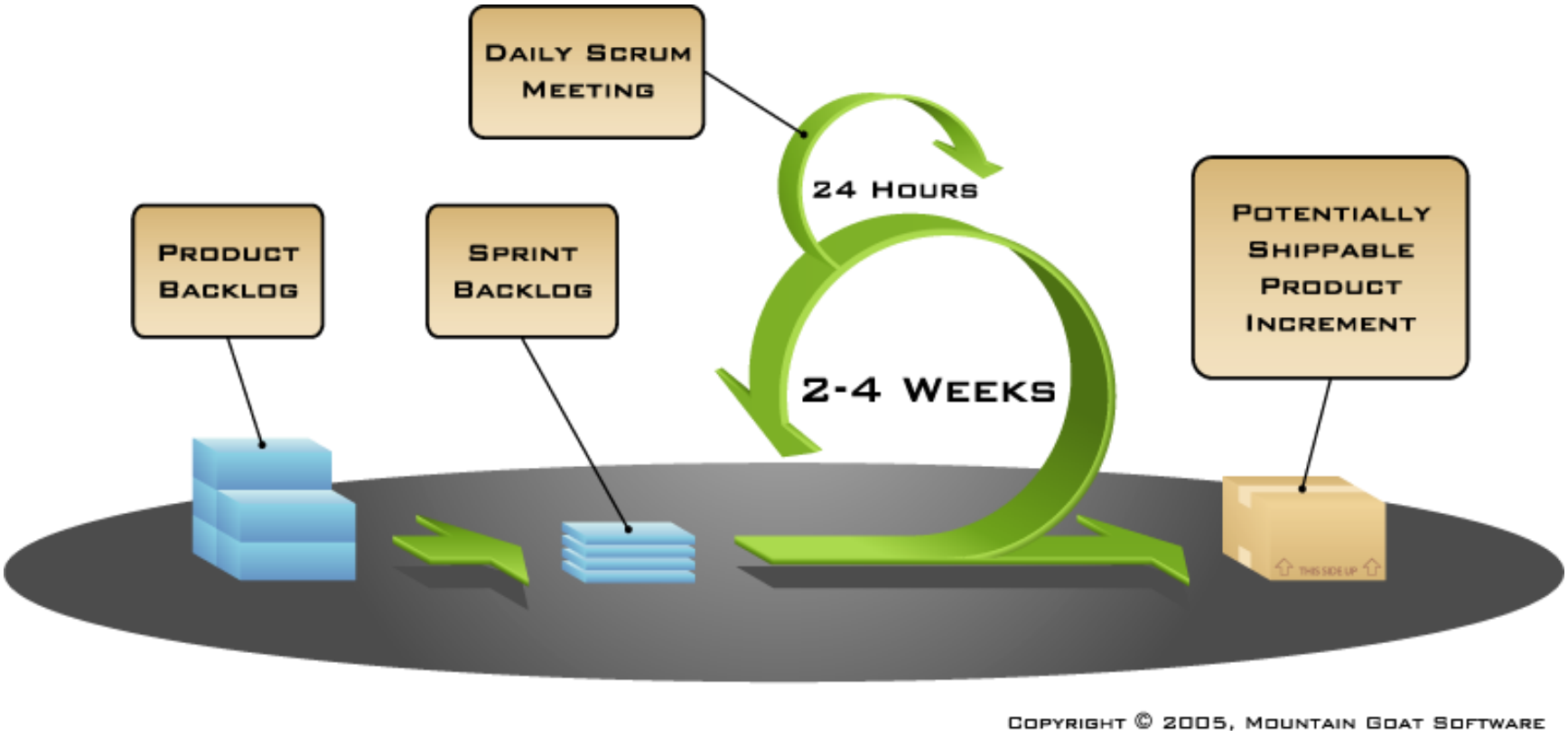
Testing Throughout the Software Life Cycle

Testing in the lifecycle



© 2016 ARGO Data Resource Corporation. All rights reserved.

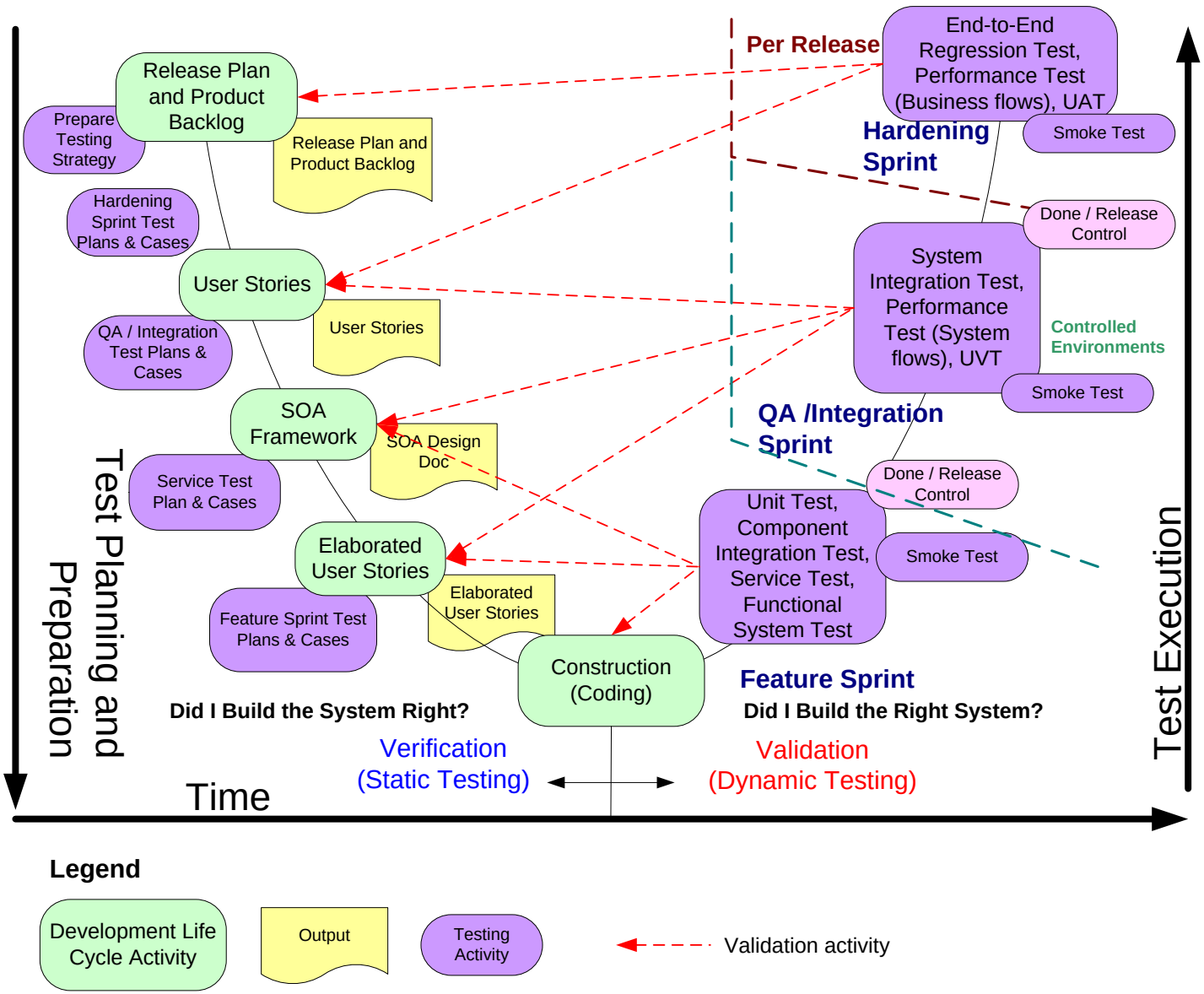
Scrum Overview



http://www.mountaingoatsoftware.com/scrum_figures

© 2016 ARGO Data Resource Corporation. All rights reserved.

Agile Testing V Model

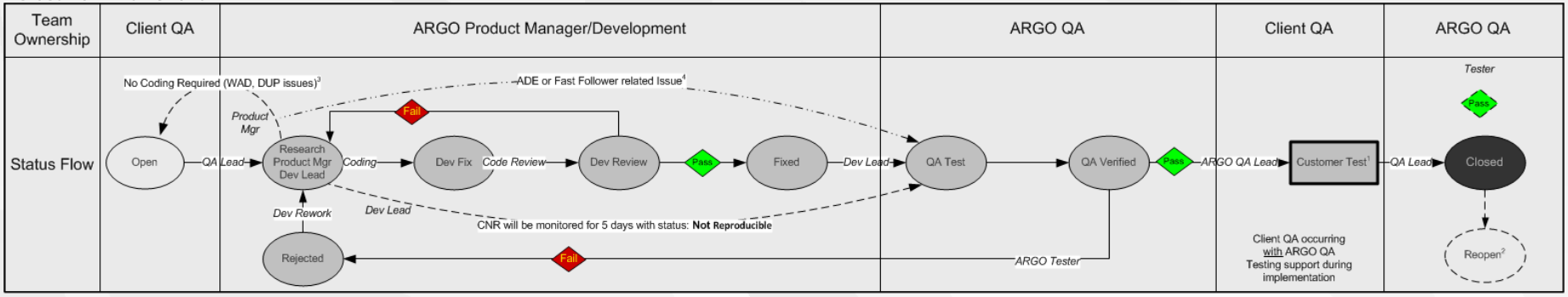


© 2016 ARGO Data Resource Corporation. All rights reserved.

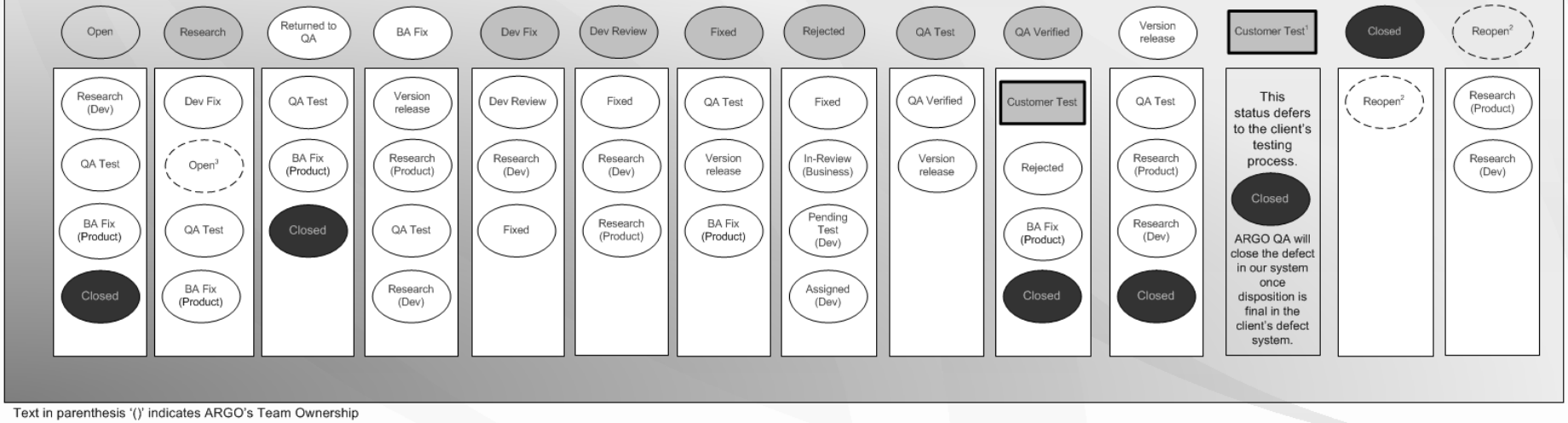
Defect Flow Client Implementations

ARGO Defect Lifecycle

Defect-flow with Client



Possible Status Changes



Text in parenthesis '()' indicates ARGO's Team Ownership

1. Customer Test is an ARGO status. At that point the client defect system is the system of record.
2. The "Reopen" and "Rejected" states are equivalent to "Research", and follow the same path.
3. The ARGO Product Manager will assign WAD Issue back to the client QA Lead with comments.
   Issue Type "CR from WAD" used if WAD contested by client QA. Status is "Assigned" Owner "client QA Lead"
4. ADE & Fast Follower issues retested when ISV testing is complete or when fast followers are implemented. Status is "Pending".

ARGO

10/16/2015

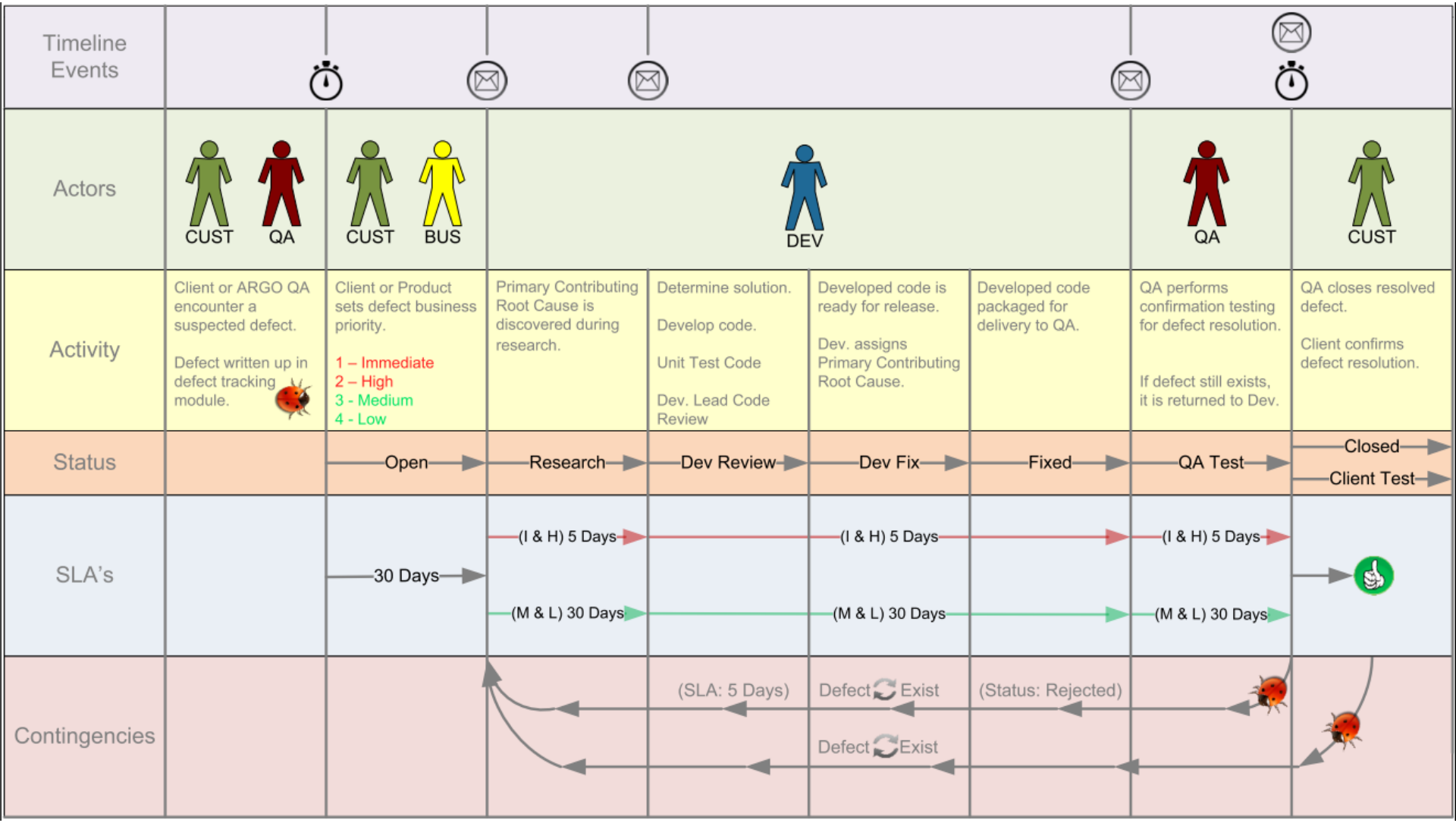
Appendix III - Defect Governance & SLAs

Lifecycle of Development & Testing – SLA's & Defect Business Priority

Ver. 1.0

To identify Issues in ALM that exceed defined service level agreement

To place additional focus on older issues that may not be valid due to product direction or implemented enhancements.



© 2016 ARGO Data Resource Corporation. All rights reserved.

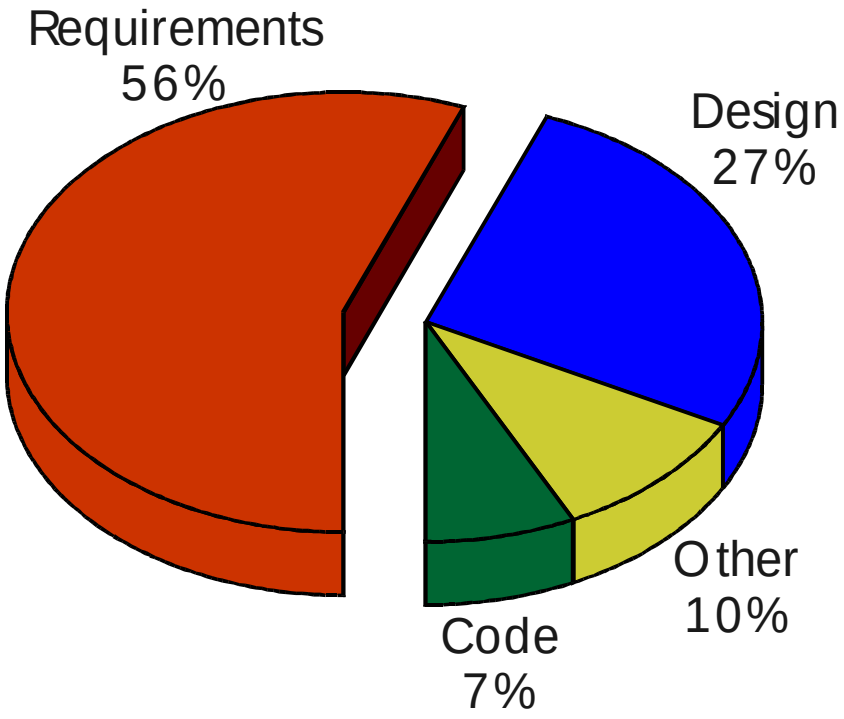
Requirements Quality

The Key to Quality

© 2016 ARGO Data Resource Corporation. All rights reserved.

Static Techniques and the Test Process

Most defects are introduced in the requirements



Typically, the defects introduced in the requirements remain undetected until the test execution phase, or worse still, until the developed system is delivered to the customer, because the original undetected defect also drives incorrect design, code development, and test case development.

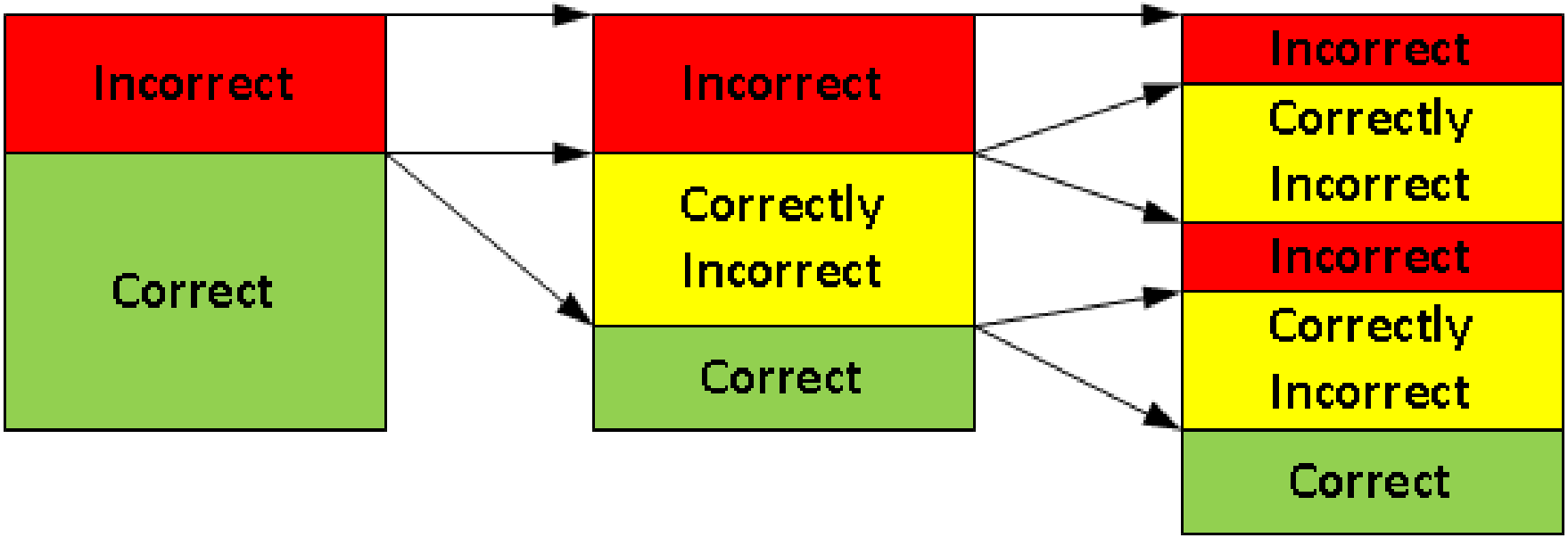
The amount of effort (and the corresponding cost) that it takes to fix defects whose origin can be traced to the requirements is even higher at 82%

© 2016 ARGO Data Resource Corporation. All rights reserved.

Static Techniques and the Test Process

Relationship between Requirements, design, and code

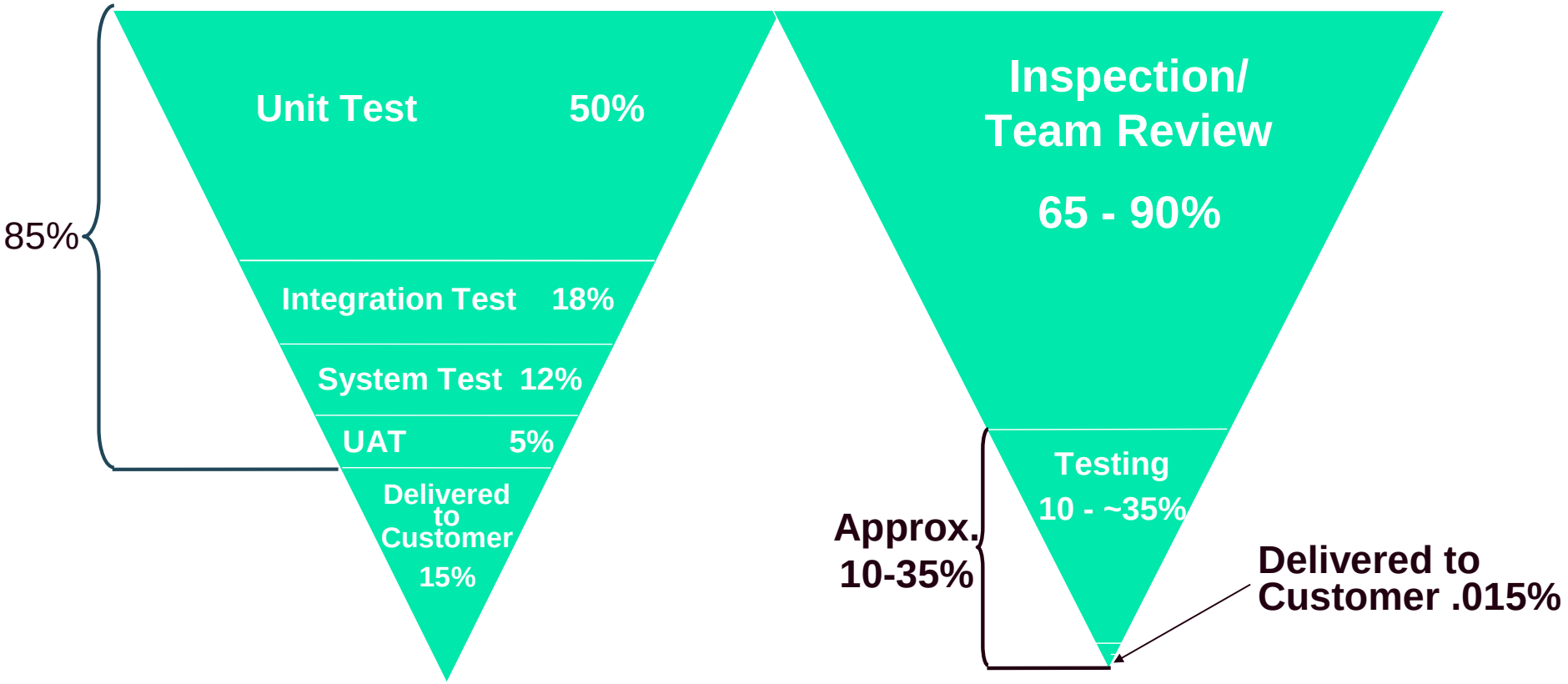
An error in requirements must be corrected not only in the requirements themselves, but also in the design, the code, and the test cases. In other words, the rework effort can almost equal the initial design, development and testing effort.



© 2016 ARGO Data Resource Corporation. All rights reserved.

Static Techniques and the Test Process

Relationship between Requirements, design and code



The typical defect discovery rate on projects that rely exclusively on code-level testing to validate application quality, and do not perform rigorous reviews for requirements quality is 85%.

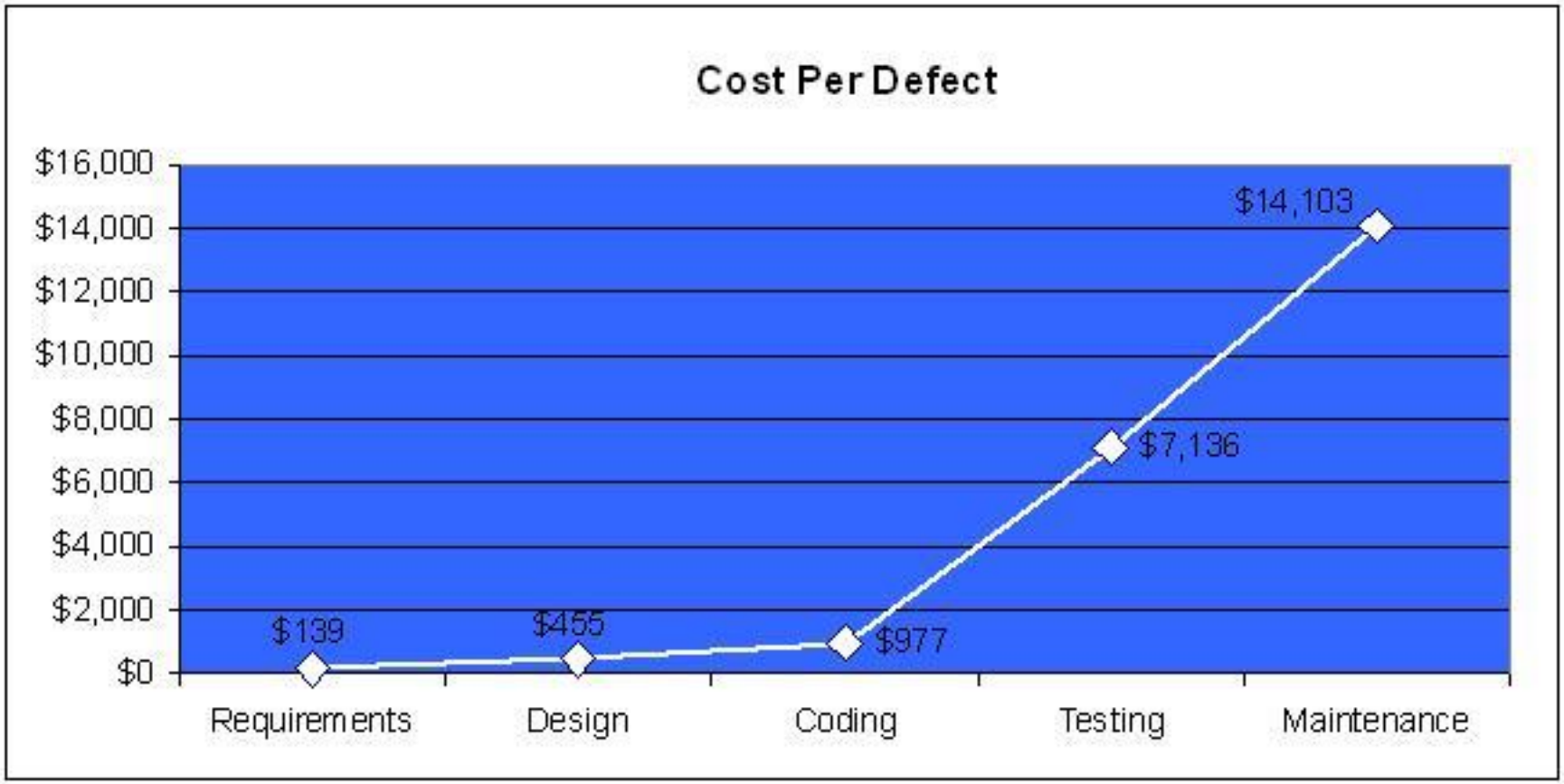
© 2016 ARGO Data Resource Corporation. All rights reserved.

Cost of Defects

Myths & Realities

© 2016 ARGO Data Resource Corporation. All rights reserved.

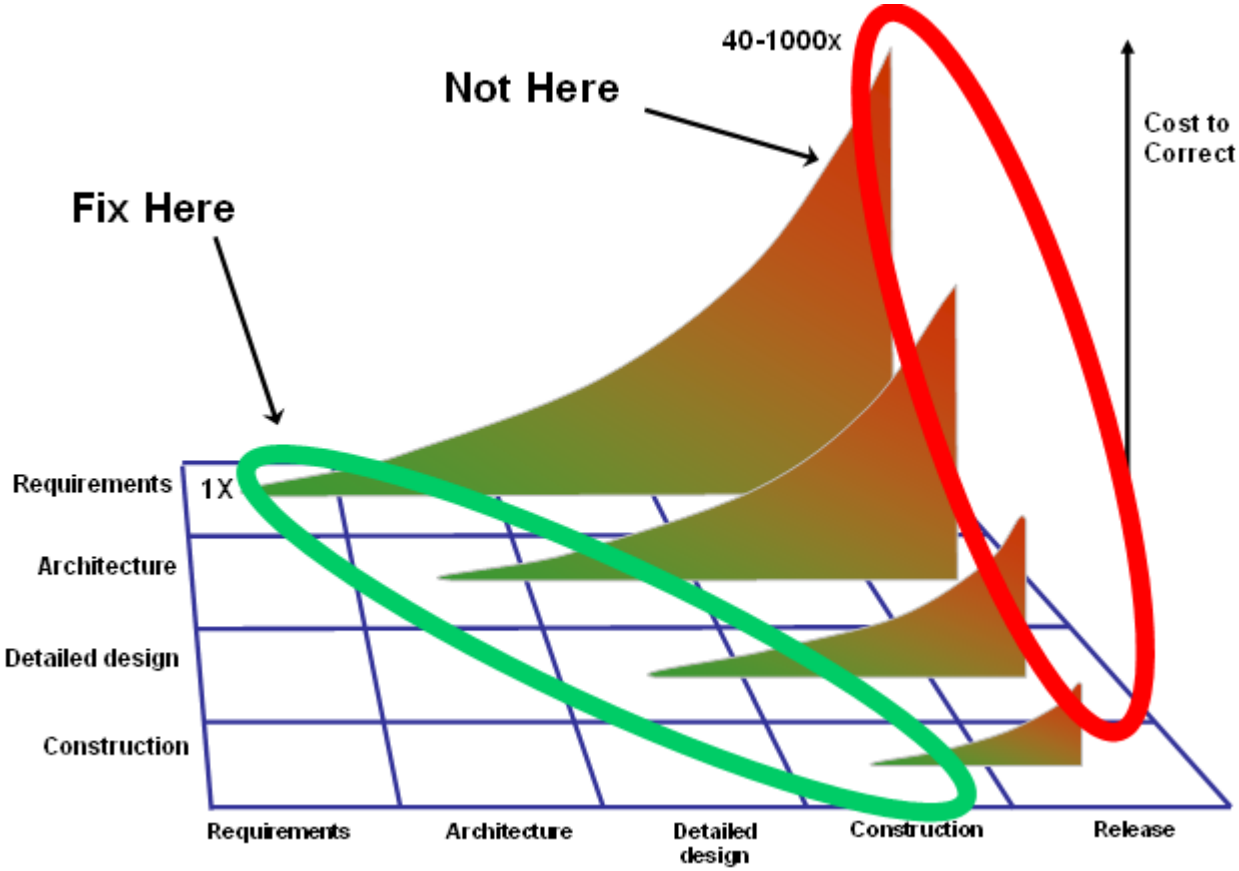
Cost of Defects at Different Stages of the SDLC



Source: Capers Jones, *Software Assessments, Benchmarks, and Best Practices*, Addison-Wesley, 2000

© 2016 ARGO Data Resource Corporation. All rights reserved.

Phase That a Defect Is Corrected

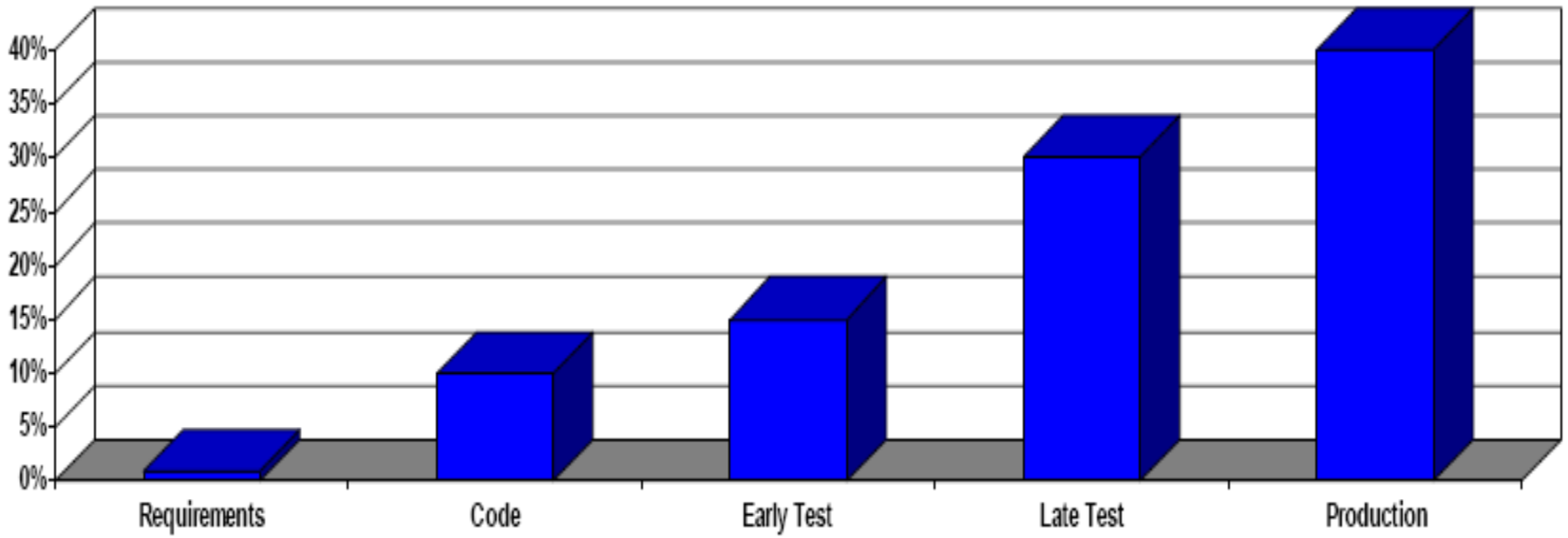


McConnell, Delivering Software Project Success: 10 Myths of Rapid Development, 2001

BST Defect Cost Analysis

© 2016 ARGO Data Resource Corporation. All rights reserved.

Pressman Cost Model



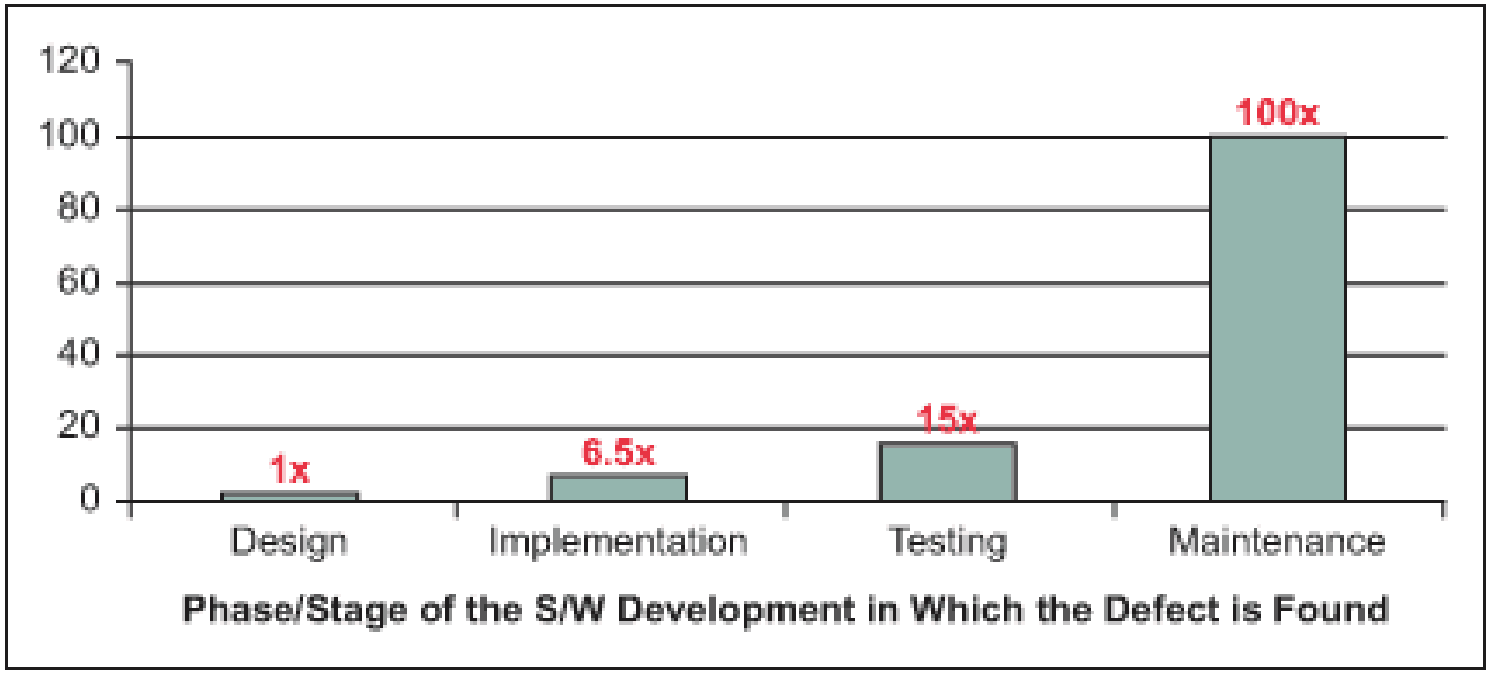
Requirements: 1X
Code: 10X
Early Test: 15-40X
Late Test: 30-70X
Production: 40-1000X

Pressman, R.S. Software Engineering: A Practioners Approach, Sixth Ed., McGraw Hill, New York, 2005

© 2016 ARGO Data Resource Corporation. All rights reserved.

IBM Cost Model

Relative Costs to Fix Software Defects



IBM Systems Sciences Institute

BST Defect Cost Analysis

© 2016 ARGO Data Resource Corporation. All rights reserved.

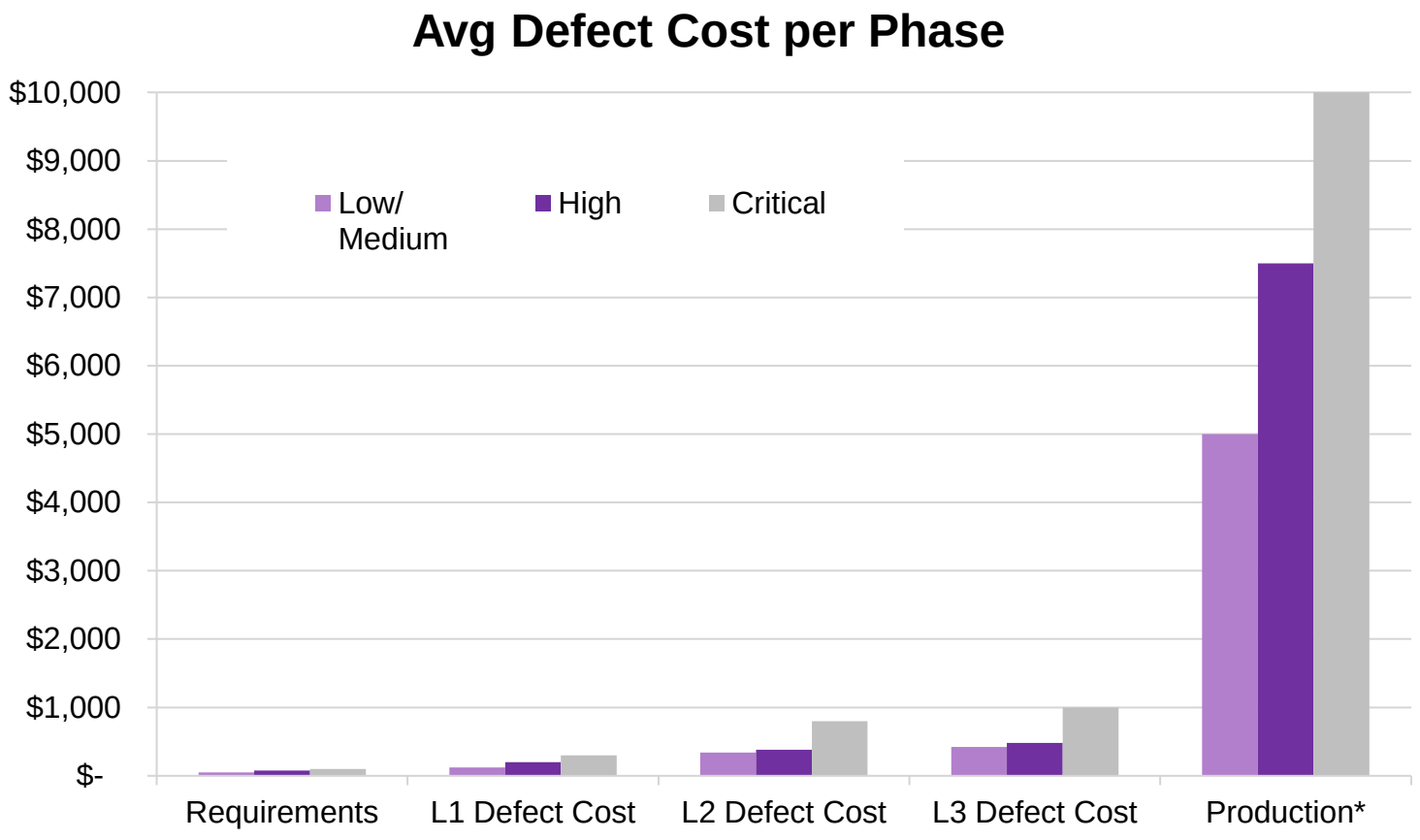
Cost of Software Quality (CoSQ)

Cost of Prevention • Solid requirements • Management of quality & process improvement • Training • Automation	**Cost of Appraisal** • Work product reviews • Code reviews • Testing • Audit and compliance activities
Cost of Internal Failure • Analysis • Defect repair • Crisis management – • Project Time/Costs • Re-testing • Opportunity costs related to missing launch dates	**Cost of External Failure** • Service Failures • Reputation impact • Crisis management - Ops • App Support • Customer Service calls • Defect remediation • Regulatory non-compliance

$ $$ $$$ $$$$

© 2016 ARGO Data Resource Corporation. All rights reserved.

Defect Analysis- Cost per Phase



On average, 10 people touch each defect

*** Estimate**

© 2016 ARGO Data Resource Corporation. All rights reserved.

Defect Cost Analysis Results

	Cost			
	Low/ Medium	High	Critical	Average
Total Average Defect Cost	$ 293	$ 353	$ 700	$ 449
Requirements	$ 50	$ 75	$ 100	$ 75
L1 Defect Cost	$ 120	$ 200	$ 300	$ 207
L2 Defect Cost	$ 340	$ 380	$ 800	$ 507
L3 Defect Cost	$ 420	$ 480	$ 1,000	$ 633
Production*	$ 5,000	$ 7,500	$ 10,000	$ 7,500

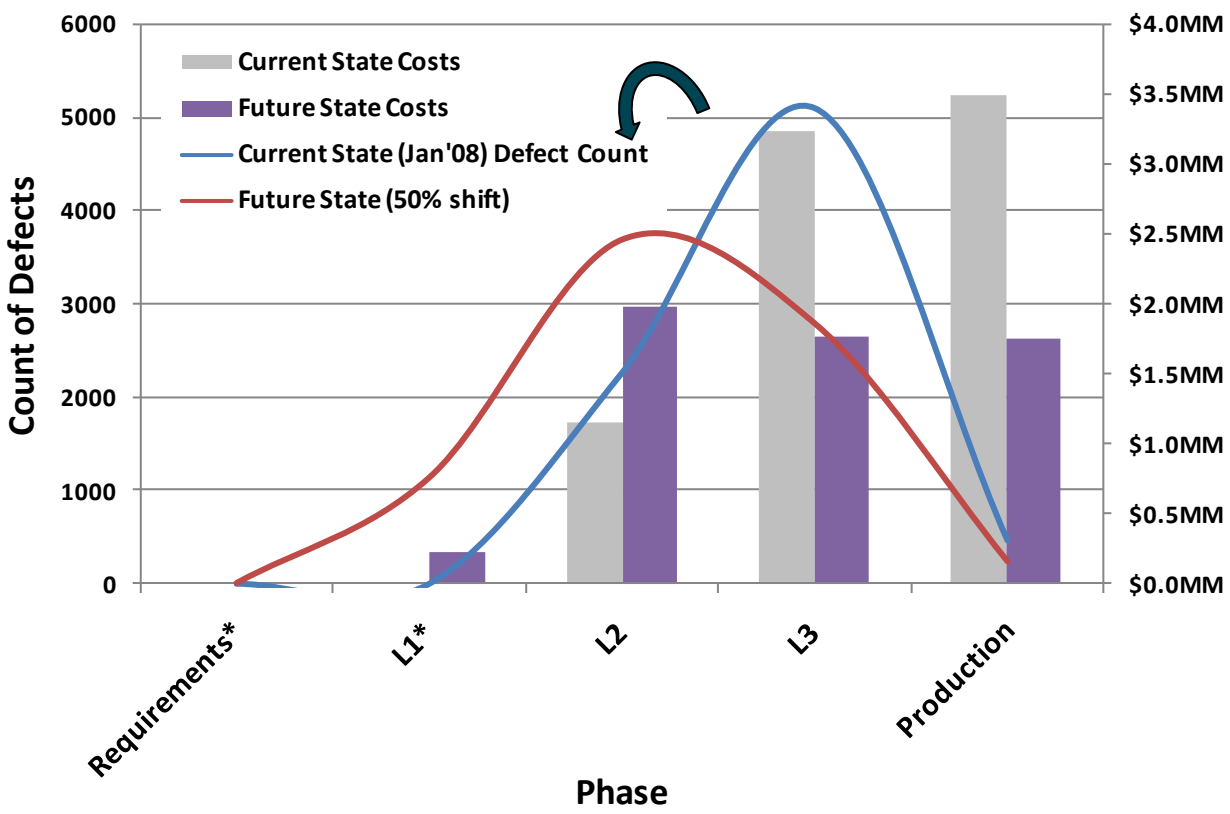
*Production used factor of 100x as an estimate, Production includes defect correction, customer impact, & lost revenue. Utilizing industry standard- low end weighting

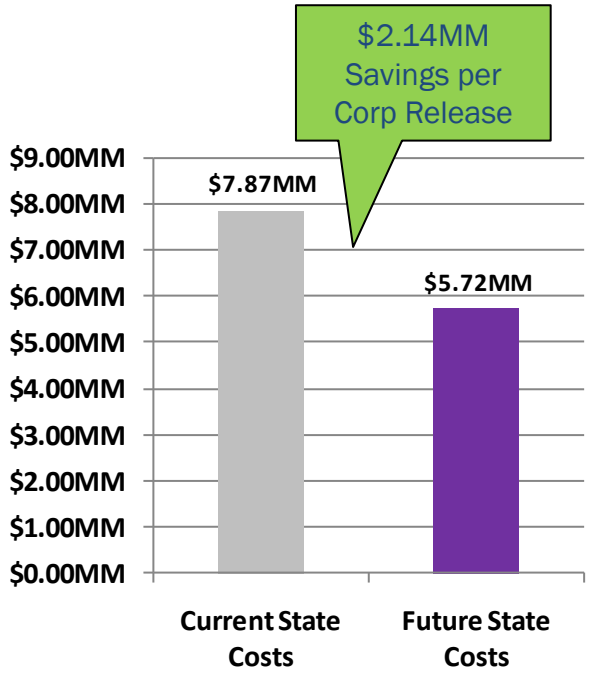
All values averages and rounded to nearest whole number

© 2016 ARGO Data Resource Corporation. All rights reserved.

January Analysis- Detecting Defects Earlier

- 50% defect shift saves $2.14MM per Corp Release
  - Finding 50% of each phases defects in earlier phase





© 2016 ARGO Data Resource Corporation. All rights reserved.

The Riskiest of the Risks

“It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so.”

Mark Twain

© 2016 ARGO Data Resource Corporation. All rights reserved.

Managing Risk

Beware Unknown Knowns

© 2016 ARGO Data Resource Corporation. All rights reserved.

Your Byproducts

Increase the Effectiveness of Your Test Coverage & Improve Code Quality

© 2016 ARGO Data Resource Corporation. All rights reserved.

NO!!

Don’t Agree to the Impossible.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Communication Heuristics

- Misconception is that status and metrics only go out in email.
- If you depend on email, you have one "**where**" and one "**how**" in your communication tool belt. There are a lot of other tools available to the wise test manager.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Test Management Trifecta

- What have you completed?
- What did you learn?
- What remains?

XBOSoft, Inc. All Rights

© 2016 ARGO Data Resource Corporation. All rights reserved.

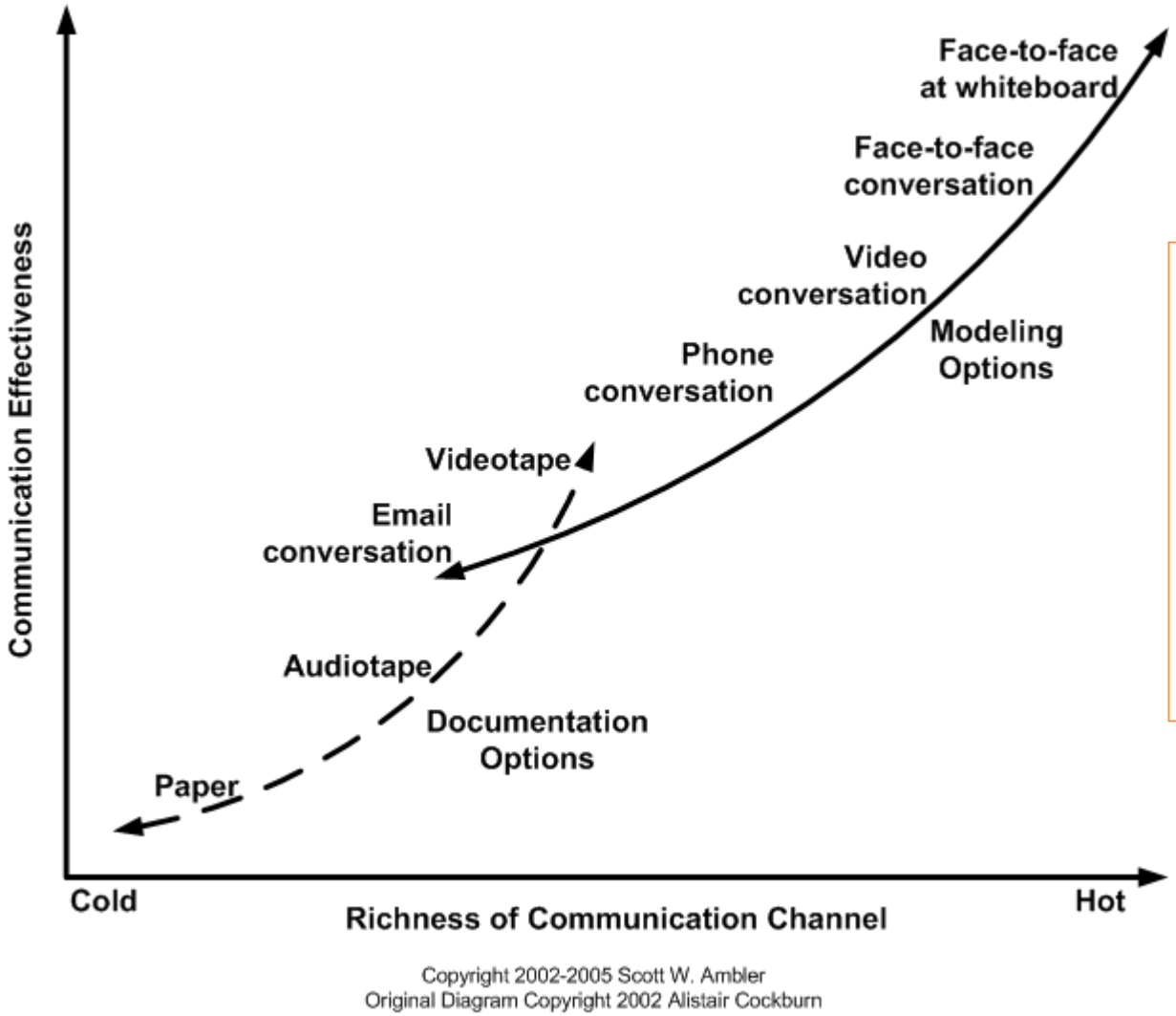
BLUF

- Bottom Line Up Front
- Follow with a “Headline”
- Impact to the triple constraint?

XBOSoft, Inc. All Rights

© 2016 ARGO Data Resource Corporation. All rights reserved.

Communication Effectiveness



Copyright 2002-2005 Scott W. Ambler
Original Diagram Copyright 2002 Alistair Cockburn

"When documents are mostly to enable handoffs, they are evil. When they capture a record of a conversation that is best not forgotten, they are valuable."

- Tom Poppendieck

© 2016 ARGO Data Resource Corporation. All rights reserved.

Credibility of the testing Organization

Credibility

Building and **keeping credibility**

- Credibility is based on trust built over time
- It can be lost in a moment
- Credibility is not perfection
- Be quick to admit mistakes and slow to make assumptions

Damage Control – **Rebuilding Lost Credibility**

Be honest and open

- Allow time to recover trust
- Keep relationships and lines of communication open
- Be able to explain your position. Don't argue.
- Document your findings carefully

© 2016 ARGO Data Resource Corporation. All rights reserved.

Test Leadership

I share my expectations for the team of testers.

Then they have the **homework** to share their expectations of me.

© 2016 ARGO Data Resource Corporation. All rights reserved.

Fundamental Test Process

A Project = Who does What by When

© 2016 ARGO Data Resource Corporation. All rights reserved.

ARGO

1500 N. Greenville Avenue, Suite 500
Richardson, TX 75081

Mark Bentsen, QA Manager
CTAL, CSTE, PMP, ASQ CMQ/OE
972.275.7240
Mark.Bentsen@argodata.com