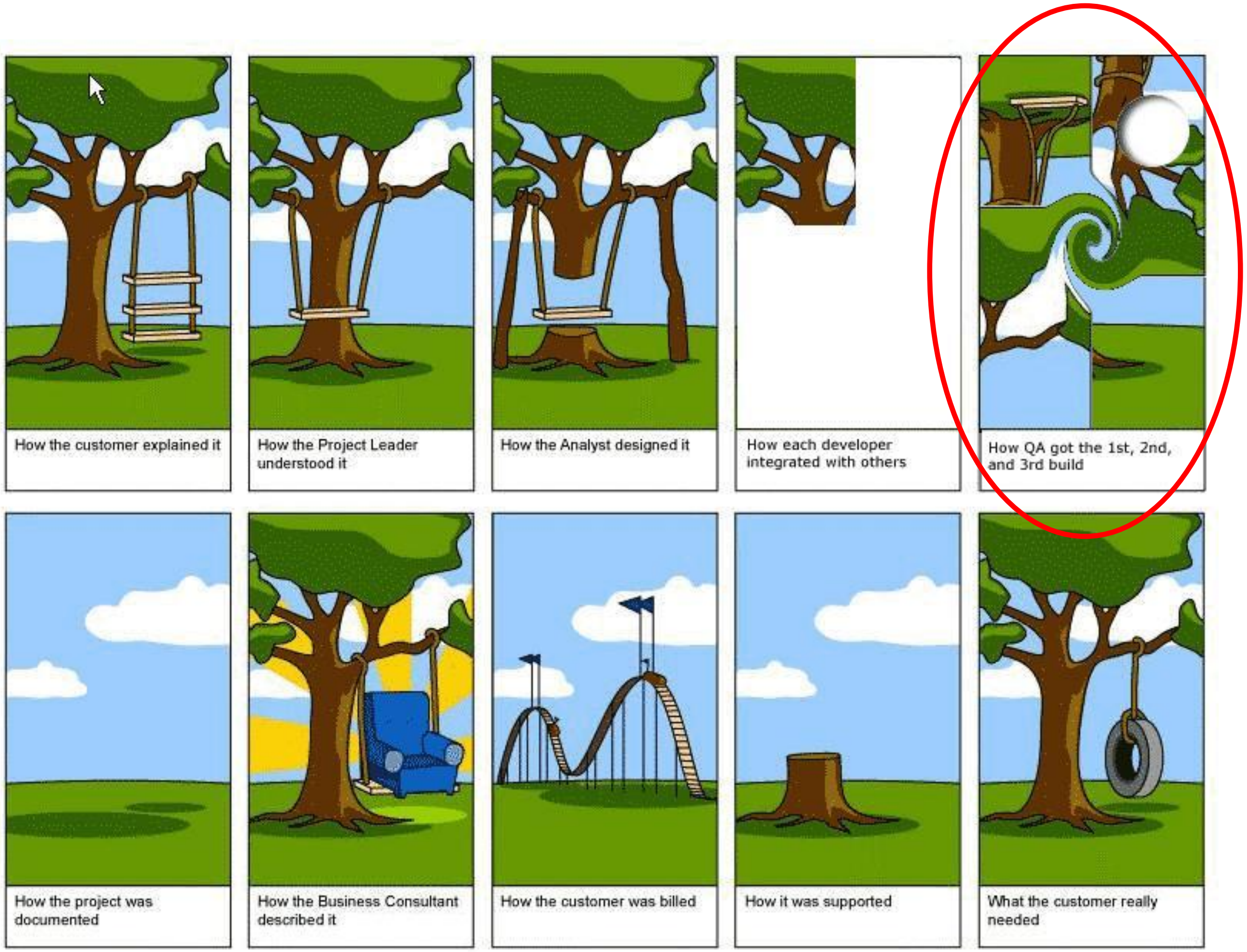
Testing – a journey from ad hoc to a *Center of Excellence*

Content

- An alarming discovery
- Some basics:
  - Models & Levels of Testing
- So where do you start, in a broken environment?
- Unit Testing, & a word about Automation
- Functional/Integration testing
- User Acceptance Testing
- Regression Testing
- Test Metrics & Reporting
- Testing in an Agile environment
- It might end up with a Testing Center of Excellence…….The story so far……

A testing funny


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How each developer integrated with others
How QA got the 1st, 2nd, and 3rd build
How the project was documented
How the Business Consultant described it
How the customer was billed
How it was supported
What the customer really needed

And just one more……

YOU ELIMINATED THE BUDGET FOR AUTOMATED TEST SOFTWARE. HOW ARE WE GOING TO TEST OUR NEW CODE?
scottadams@aol.com
www.dilbert.com
GO WRITE SOME AUTOMATED TEST SOFTWARE, YOU BIG BABY. I ALREADY PAY YOU, SO IT'S FREE.
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
7-18-07
TODAY I LEARNED YOU CAN END ANY CONVER–SATION BY CALLING THE OTHER PERSON A BIG BABY.
WAA–WAA! DO YOU WANT YOUR BOTTLE?

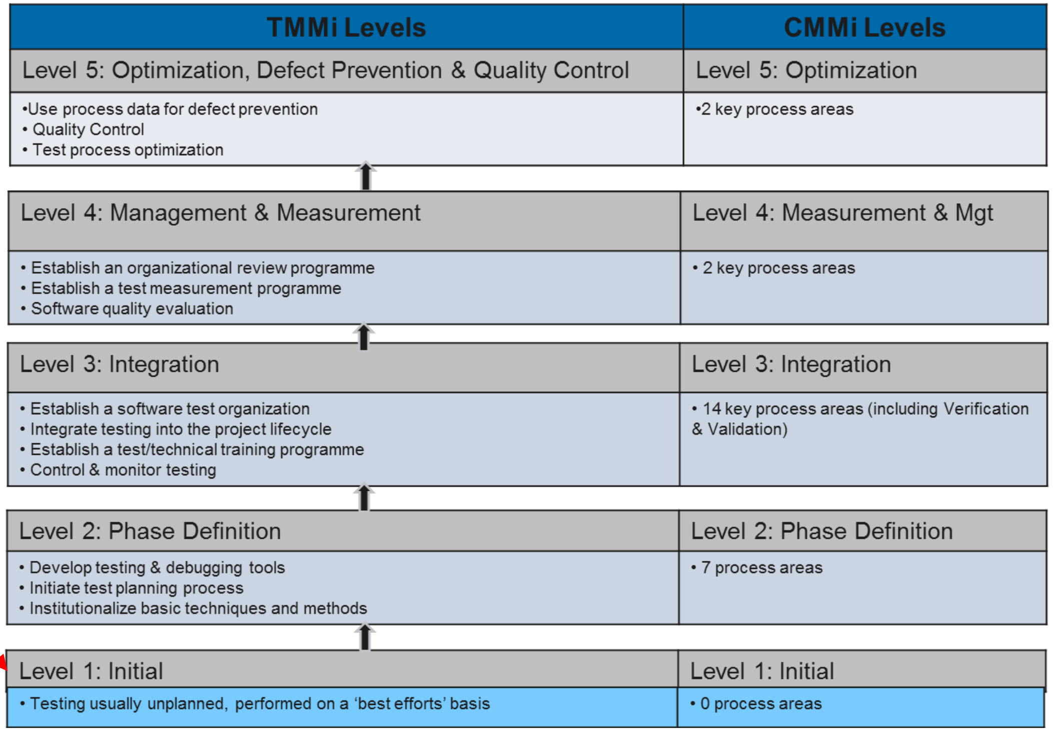
The Testing Maturity Model integrated (TMMi)

TMMi Levels	CMMi Levels
Level 5: Optimization, Defect Prevention & Quality Control	Level 5: Optimization
•*Use process data for defect prevention* • *Quality Control* • *Test process optimization*	•*2 key process areas*
Level 4: Management & Measurement	Level 4: Measurement & Mgt
• *Establish an organizational review programme* • *Establish a test measurement programme* • *Software quality evaluation*	• *2 key process areas*
Level 3: Integration	Level 3: Integration
• *Establish a software test organization* • *Integrate testing into the project lifecycle* • *Establish a test/technical training programme* • *Control & monitor testing*	• *14 key process areas (including Verification & Validation)*
Level 2: Phase Definition	Level 2: Phase Definition
• *Develop testing & debugging tools* • *Initiate test planning process* • *Institutionalize basic techniques and methods*	• *7 process areas*
Level 1: Initial	Level 1: Initial
• *0 process areas*	

You can’t go lower than Level 1!

The Testing Maturity Model integrated (TMMi)

My client was here – but only just!



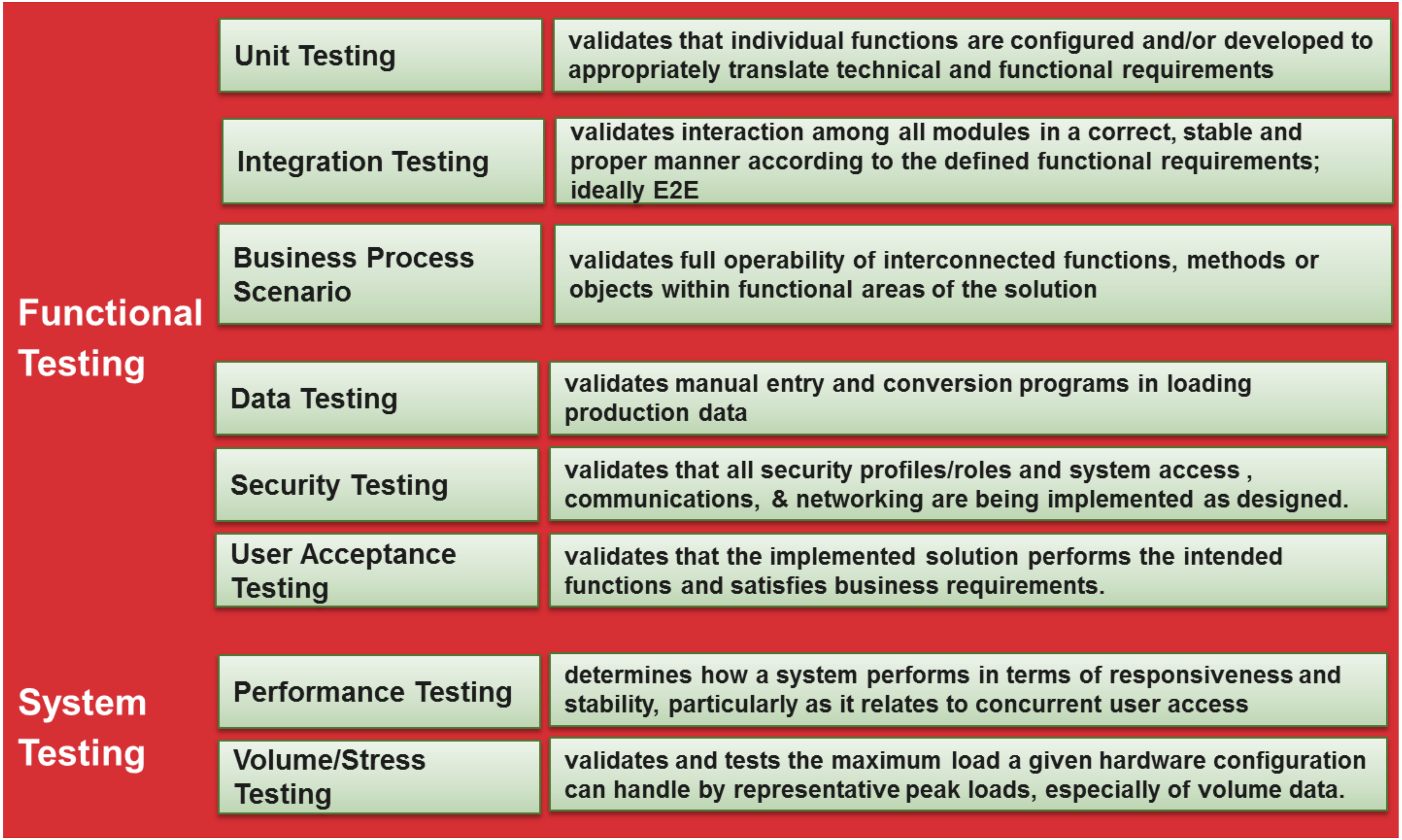
TMMi Levels	CMMi Levels
Level 5: Optimization, Defect Prevention & Quality Control	Level 5: Optimization
•Use process data for defect prevention • Quality Control • Test process optimization	•2 key process areas
Level 4: Management & Measurement	Level 4: Measurement & Mgt
• Establish an organizational review programme • Establish a test measurement programme • Software quality evaluation	• 2 key process areas
Level 3: Integration	Level 3: Integration
• Establish a software test organization • Integrate testing into the project lifecycle • Establish a test/technical training programme • Control & monitor testing	• 14 key process areas (including Verification & Validation)
Level 2: Phase Definition	Level 2: Phase Definition
• Develop testing & debugging tools • Initiate test planning process • Institutionalize basic techniques and methods	• 7 process areas
Level 1: Initial	Level 1: Initial
• Testing usually unplanned, performed on a ‘best efforts’ basis	• 0 process areas

Testing

“**Testing** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.”

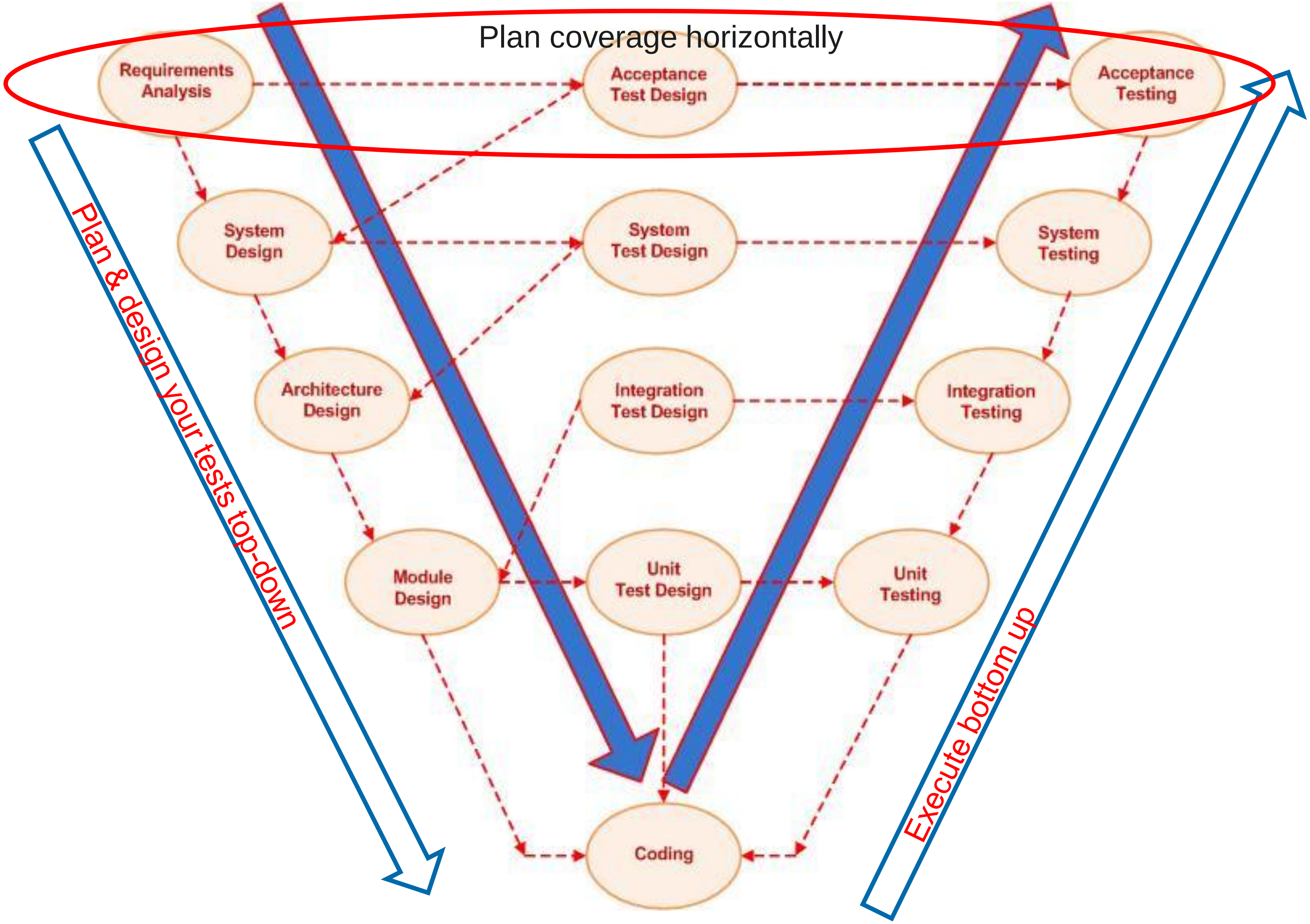
“You can have it good, fast, or cheap. Choose any 2 of the three.”

Levels of testing (a sampling)

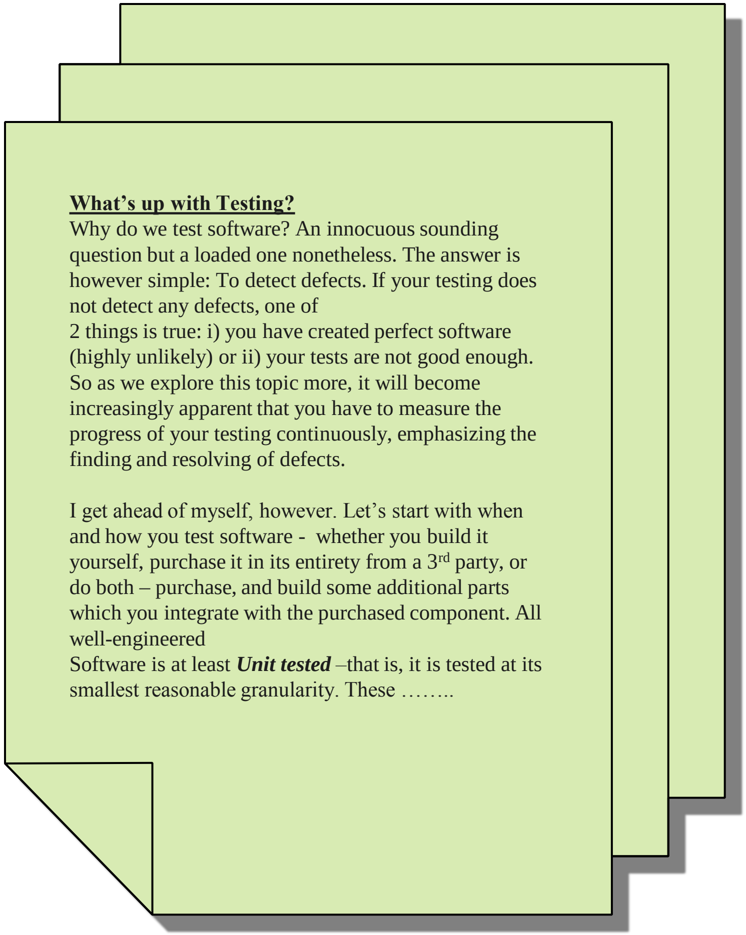


Functional Testing	Unit Testing	validates that individual functions are configured and/or developed to appropriately translate technical and functional requirements
	Integration Testing	validates interaction among all modules in a correct, stable and proper manner according to the defined functional requirements; ideally E2E
	Business Process Scenario	validates full operability of interconnected functions, methods or objects within functional areas of the solution
	Data Testing	validates manual entry and conversion programs in loading production data
	Security Testing	validates that all security profiles/roles and system access , communications, & networking are being implemented as designed.
	User Acceptance Testing	validates that the implemented solution performs the intended functions and satisfies business requirements.
System Testing	Performance Testing	determines how a system performs in terms of responsiveness and stability, particularly as it relates to concurrent user access
	Volume/Stress Testing	validates and tests the maximum load a given hardware configuration can handle by representative peak loads, especially of volume data.

The traditional ‘V’ Model in software testing



A paper on Testing……



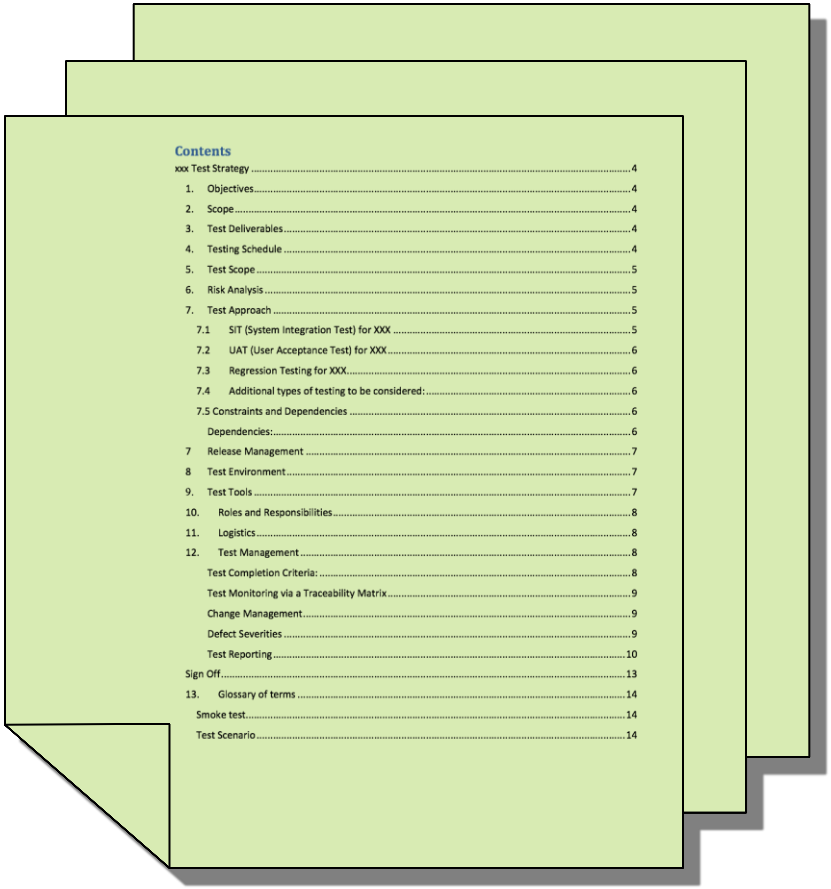
What's up with Testing?
Why do we test software? An innocuous sounding question but a loaded one nonetheless. The answer is however simple: To detect defects. If your testing does not detect any defects, one of
2 things is true: i) you have created perfect software (highly unlikely) or ii) your tests are not good enough. So as we explore this topic more, it will become increasingly apparent that you have to measure the progress of your testing continuously, emphasizing the finding and resolving of defects.

I get ahead of myself, however. Let's start with when and how you test software - whether you build it yourself, purchase it in its entirety from a 3rd party, or do both – purchase, and build some additional parts which you integrate with the purchased component. All well-engineered
Software is at least ***Unit tested*** –that is, it is tested at its smallest reasonable granularity. These ……..

Let’s start with a Strategy…..

Test Strategy

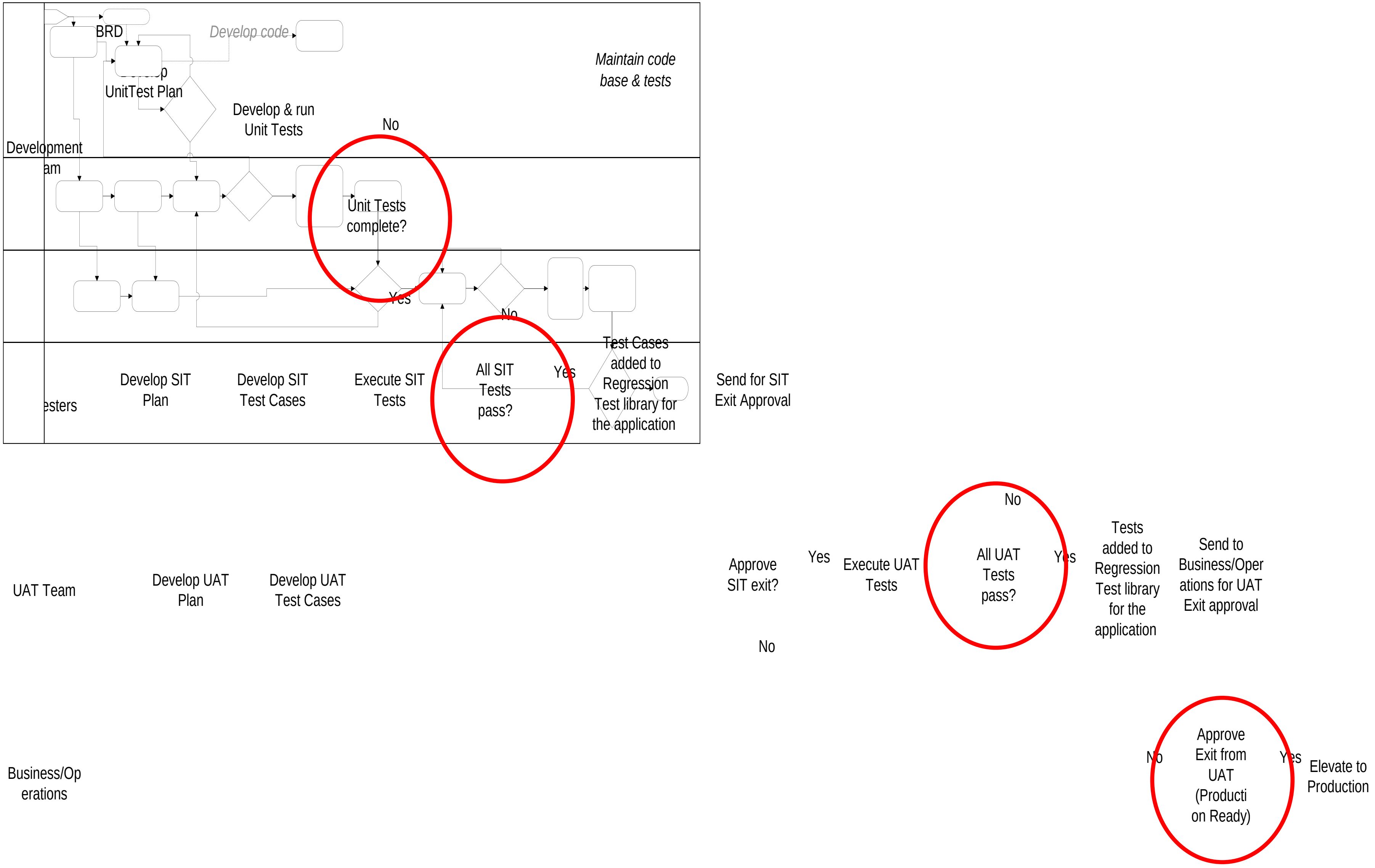
- From SIT to UAT -



Contents

xxx Test Strategy 4
1. Objectives 4
2. Scope 4
3. Test Deliverables 4
4. Testing Schedule 4
5. Test Scope 5
6. Risk Analysis 5
7. Test Approach 5
7.1 SIT (System Integration Test) for XXX 5
7.2 UAT (User Acceptance Test) for XXX 6
7.3 Regression Testing for XXX 6
7.4 Additional types of testing to be considered: 6
7.5 Constraints and Dependencies 6
Dependencies: 6
7 Release Management 7
8 Test Environment 7
9. Test Tools 7
10. Roles and Responsibilities 8
11. Logistics 8
12. Test Management 8
Test Completion Criteria: 8
Test Monitoring via a Traceability Matrix 9
Change Management 9
Defect Severities 9
Test Reporting 10
Sign Off 13
13. Glossary of terms 14
Smoke test 14
Test Scenario 14

A Generic Testing Process flow – for new projects



Development Team

BRD

Develop code

Develop UnitTest Plan

Develop & run Unit Tests

No

Unit Tests complete?

Yes

Maintain code base & tests

esters

Develop SIT Plan

Develop SIT Test Cases

Execute SIT Tests

All SIT Tests pass?

No

Yes

Test Cases added to Regression Test library for the application

Send for SIT Exit Approval

UAT Team

Develop UAT Plan

Develop UAT Test Cases

Approve SIT exit?

Yes

No

Execute UAT Tests

All UAT Tests pass?

No

Yes

Tests added to Regression Test library for the application

Send to Business/Operations for UAT Exit approval

Business/Operations

Approve Exit from UAT (Production Ready)

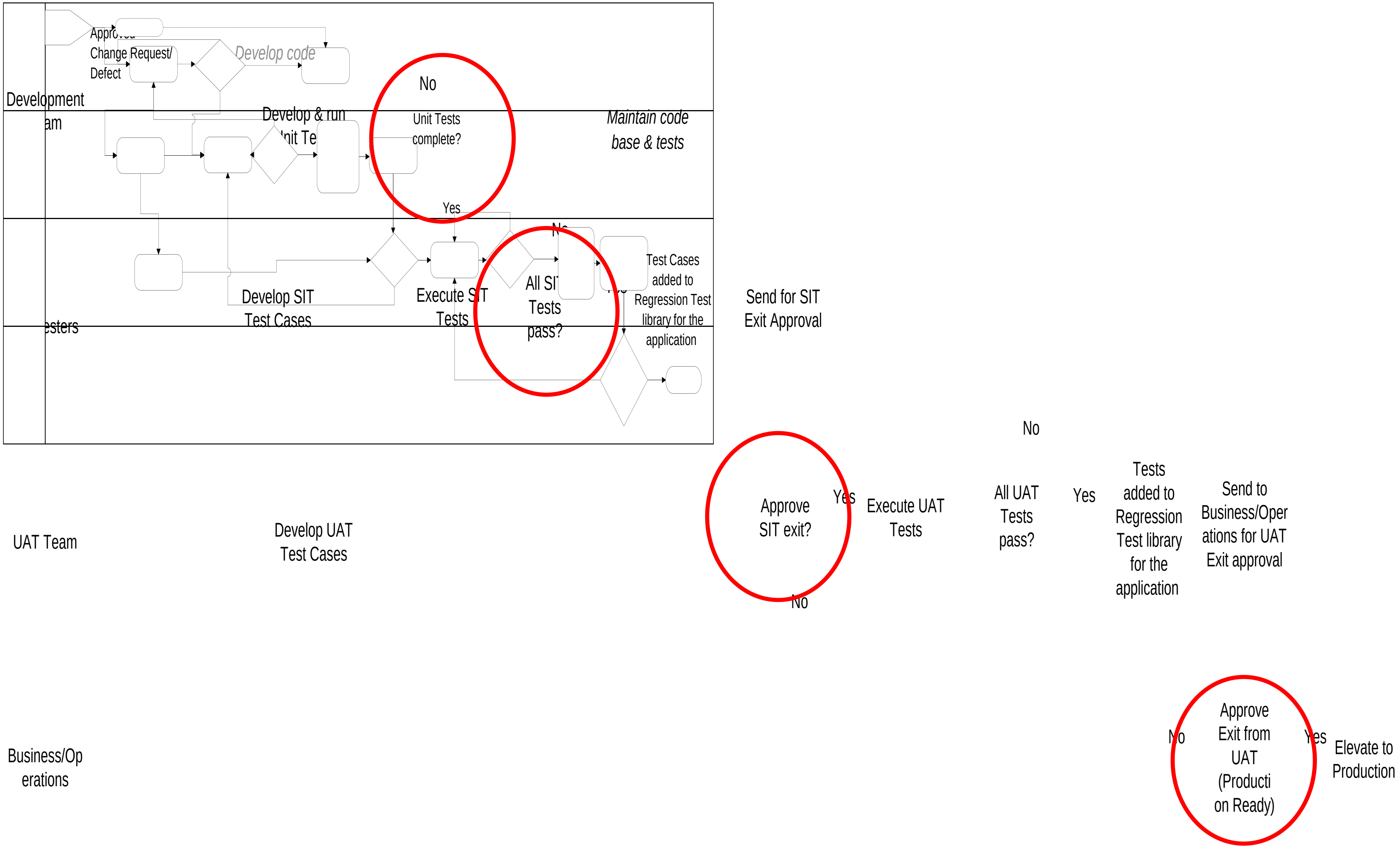
No

Yes

Elevate to Production

Testing – from ad hoc to TCoE: Tony Hutchings

A Testing Process flow – enhancement/Keep-The-Lights-On projects



Development
am

Approved
Change Request/
Defect

Develop code

Develop & run
Unit Te

No

Unit Tests
complete?

Yes

Maintain code
base & tests

esters

Develop SIT
Test Cases

Execute SIT
Tests

No

All SI
Tests
pass?

Yes

Test Cases
added to
Regression Test
library for the
application

Send for SIT
Exit Approval

UAT Team

Develop UAT
Test Cases

Approve
SIT exit?

Yes

No

Execute UAT
Tests

No

All UAT
Tests
pass?

Yes

Tests
added to
Regression
Test library
for the
application

Send to
Business/Oper
ations for UAT
Exit approval

Business/Op
erations

No

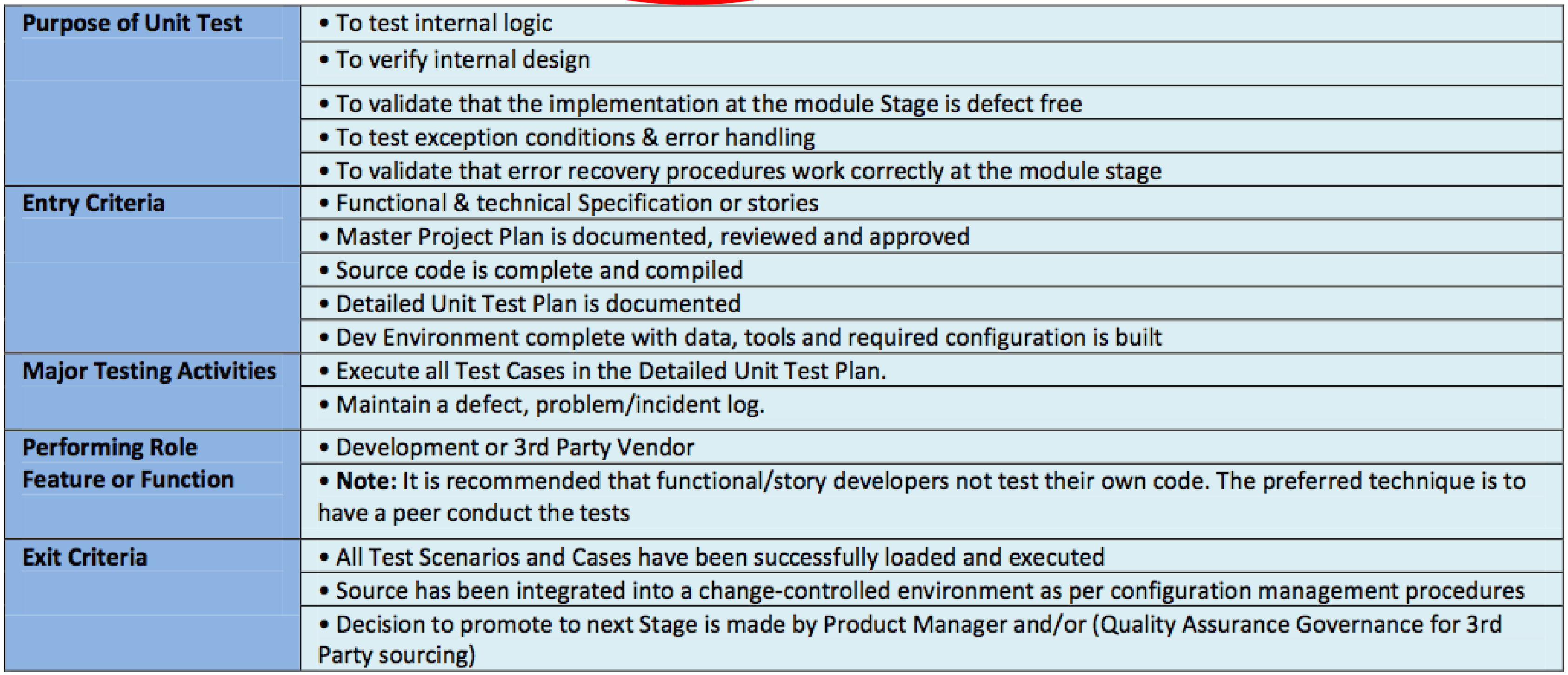
Approve
Exit from
UAT
(Producti
on Ready)

Yes

Elevate to
Production

Unit Test

- Unit Test must be executed in the **development environment**.
- Defects that slip through this Stage of testing will cost more in the next Stage of testing, both in detection and removal.
- Unit Test Stage tests individual system components, programs or modules against documented detail functional & technical specifications and requirements.
- While it utilizes all five Testing Techniques, it is the only Stage that has the detailed knowledge required to perform significant white box testing.



Purpose of Unit Test	• To test internal logic
	• To verify internal design
	• To validate that the implementation at the module Stage is defect free
	• To test exception conditions & error handling
	• To validate that error recovery procedures work correctly at the module stage
Entry Criteria	• Functional & technical Specification or stories
	• Master Project Plan is documented, reviewed and approved
	• Source code is complete and compiled
	• Detailed Unit Test Plan is documented
	• Dev Environment complete with data, tools and required configuration is built
Major Testing Activities	• Execute all Test Cases in the Detailed Unit Test Plan.
	• Maintain a defect, problem/incident log.
Performing Role	• Development or 3rd Party Vendor
Feature or Function	• **Note:** It is recommended that functional/story developers not test their own code. The preferred technique is to have a peer conduct the tests
Exit Criteria	• All Test Scenarios and Cases have been successfully loaded and executed
	• Source has been integrated into a change-controlled environment as per configuration management procedures
	• Decision to promote to next Stage is made by Product Manager and/or (Quality Assurance Governance for 3rd Party sourcing)

Unit Tests – a definition

A cool quote from Roy Osherove[1]:

A unit test *should* have the following properties:

- It should be automated and repeatable.
- It should be easy to implement.
- It should be relevant tomorrow.
- Anyone should be able to run it at the push of a button.
- It should run quickly.
- It should be consistent in its results (it always returns the same result if you don't change anything between runs).
- It should have full control of the unit under test.
- It should be fully isolated (runs independently of other tests).
- When it fails, it should be easy to detect what was expected and determine how to pinpoint the problem.

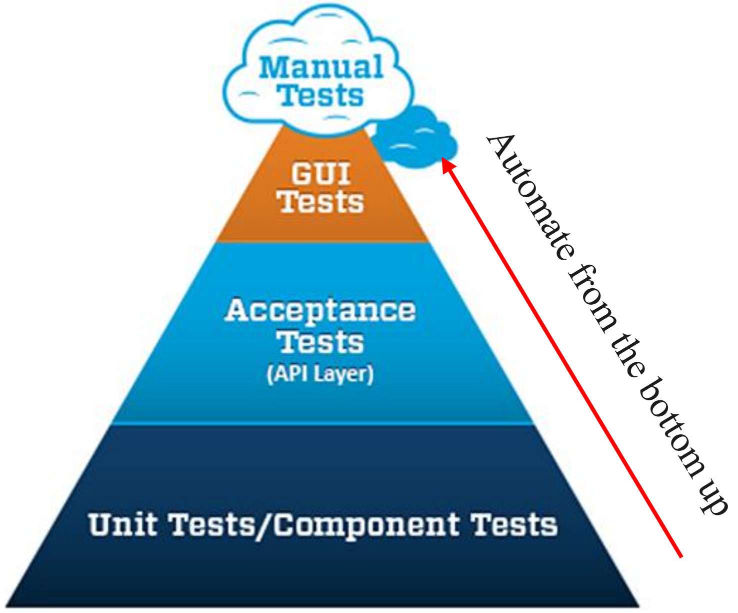
[1] *"The Art of Unit Testing"*

Test Automation

Capture/Playback

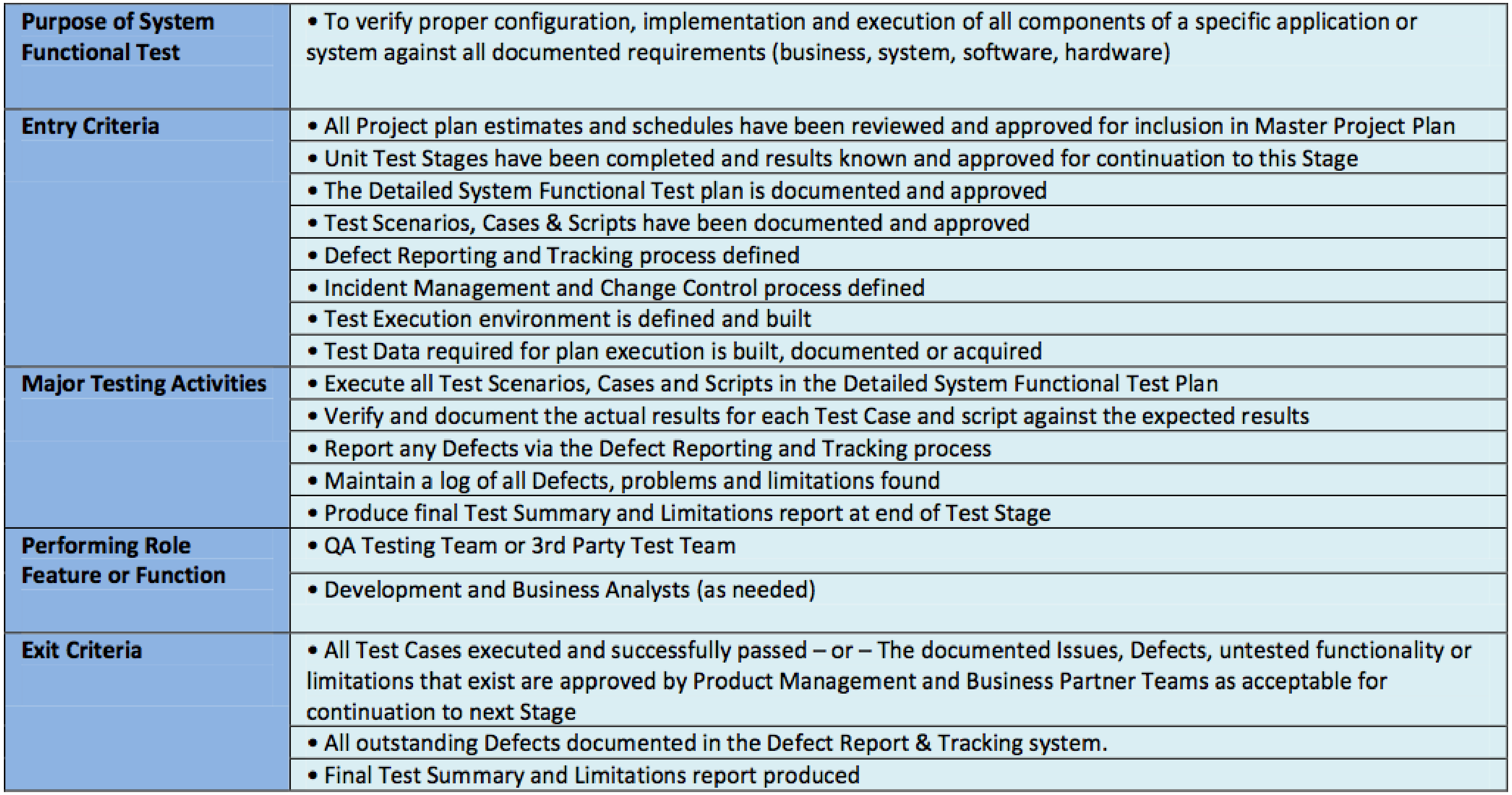
The capture/playback approach means that tests are performed manually while the inputs and outputs are captured in the background. During subsequent automated playback, the script repeats the same sequence of actions to apply the inputs and compare the actual responses to the captured results; differences are reported as errors. Capture/playback is available from almost all automated test tools, although it may be Implemented differently.

Mike Cohn's Test Automation Pyramid:



Functional Testing

- Verifies proper configuration, implementation & execution of all the components of a specific application or system against all documented stories/requirements (business, system, software, hardware).
- Executes in a ***SIT*** environment, mirroring as closely as possible, the production environment
- Validates that the system functionality, standards compliance and stability meet the business need and any other system it interfaces with.
- Also to verify that the system is structurally sound and can perform the intended tasks.
- Ensures that the technology has been used properly and the system can be deployed as planned.

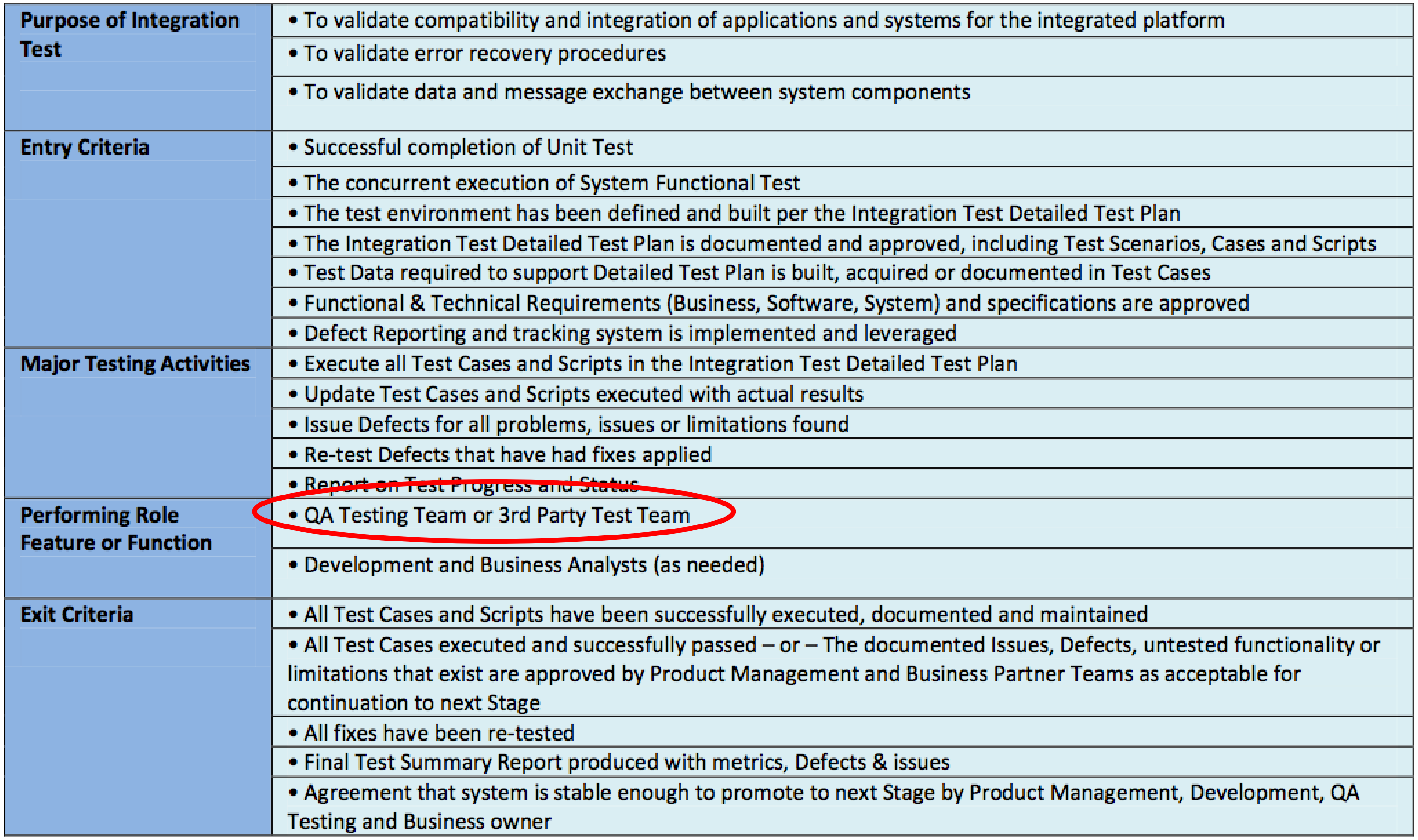


Purpose of System Functional Test	• To verify proper configuration, implementation and execution of all components of a specific application or system against all documented requirements (business, system, software, hardware)
Entry Criteria	• All Project plan estimates and schedules have been reviewed and approved for inclusion in Master Project Plan
	• Unit Test Stages have been completed and results known and approved for continuation to this Stage
	• The Detailed System Functional Test plan is documented and approved
	• Test Scenarios, Cases & Scripts have been documented and approved
	• Defect Reporting and Tracking process defined
	• Incident Management and Change Control process defined
	• Test Execution environment is defined and built
	• Test Data required for plan execution is built, documented or acquired
Major Testing Activities	• Execute all Test Scenarios, Cases and Scripts in the Detailed System Functional Test Plan
	• Verify and document the actual results for each Test Case and script against the expected results
	• Report any Defects via the Defect Reporting and Tracking process
	• Maintain a log of all Defects, problems and limitations found
	• Produce final Test Summary and Limitations report at end of Test Stage
Performing Role Feature or Function	• QA Testing Team or 3rd Party Test Team
	• Development and Business Analysts (as needed)
Exit Criteria	• All Test Cases executed and successfully passed – or – The documented Issues, Defects, untested functionality or limitations that exist are approved by Product Management and Business Partner Teams as acceptable for continuation to next Stage
	• All outstanding Defects documented in the Defect Report & Tracking system.
	• Final Test Summary and Limitations report produced

End To End/Integration Test

- Tests the integration of two or more applications or systems into an integrated software product platform.
- Executes in a ***SIT*** environment & focuses on the end-to-end aspects of the integrated platform verifying that all interfaces, API’s or data exchange points are functioning as designed.
- May also include additional test cases for performance and standards compliance.

“*Integration testing* is testing a unit of work without having full control over all of it and using one or more of its real dependencies, such as time, network, database, threads, random number generators, and so on. “



Purpose of Integration Test	• To validate compatibility and integration of applications and systems for the integrated platform
	• To validate error recovery procedures
	• To validate data and message exchange between system components
Entry Criteria	• Successful completion of Unit Test
	• The concurrent execution of System Functional Test
	• The test environment has been defined and built per the Integration Test Detailed Test Plan
	• The Integration Test Detailed Test Plan is documented and approved, including Test Scenarios, Cases and Scripts
	• Test Data required to support Detailed Test Plan is built, acquired or documented in Test Cases
	• Functional & Technical Requirements (Business, Software, System) and specifications are approved
	• Defect Reporting and tracking system is implemented and leveraged
Major Testing Activities	• Execute all Test Cases and Scripts in the Integration Test Detailed Test Plan
	• Update Test Cases and Scripts executed with actual results
	• Issue Defects for all problems, issues or limitations found
	• Re-test Defects that have had fixes applied
	• Report on Test Progress and Status
Performing Role Feature or Function	• QA Testing Team or 3rd Party Test Team
	• Development and Business Analysts (as needed)
Exit Criteria	• All Test Cases and Scripts have been successfully executed, documented and maintained
	• All Test Cases executed and successfully passed – or – The documented Issues, Defects, untested functionality or limitations that exist are approved by Product Management and Business Partner Teams as acceptable for continuation to next Stage
	• All fixes have been re-tested
	• Final Test Summary Report produced with metrics, Defects & issues
	• Agreement that system is stable enough to promote to next Stage by Product Management, Development, QA Testing and Business owner

User Acceptance Testing

- Tests all functionality from an end user’s perspective, including any administrative functions needed to manage the system

Purpose of User Acceptance Test	• To validate User visible function for the integrated platform
Entry Criteria	• Successful completion of the Integration Tests • The test environment has been defined and built per the User Acceptance Test Detailed Test Plan • The User Acceptance Test Detailed Test Plan is documented and approved, including Test Scenarios, Cases and Scripts • Test Data required to support Detailed Test Plan is built, acquired or documented in Test Cases • Business, Functional & Technical Requirements (Business, Software, System) and specifications are approved • Defect Reporting and tracking system is implemented and leveraged
Major Testing Activities	• Execute all Test Cases and Scripts in the User Acceptance Test Detailed Test Plan • Update Test Cases and Scripts executed with actual results • Issue Defects for all problems, issues or limitations found • Re-test Defects that have had fixes applied
Performing Role Feature or Function	• Users • Business Analysts
Exit Criteria	• All Test Cases and Scripts have been successfully executed, documented and maintained • All Test Cases executed and successfully passed – or – The documented Issues, Defects, untested functionality or limitations that exist are approved by Product Management and Business Partner Teams as acceptable for continuation to next Stage • All fixes have been re-tested

Regression Testing

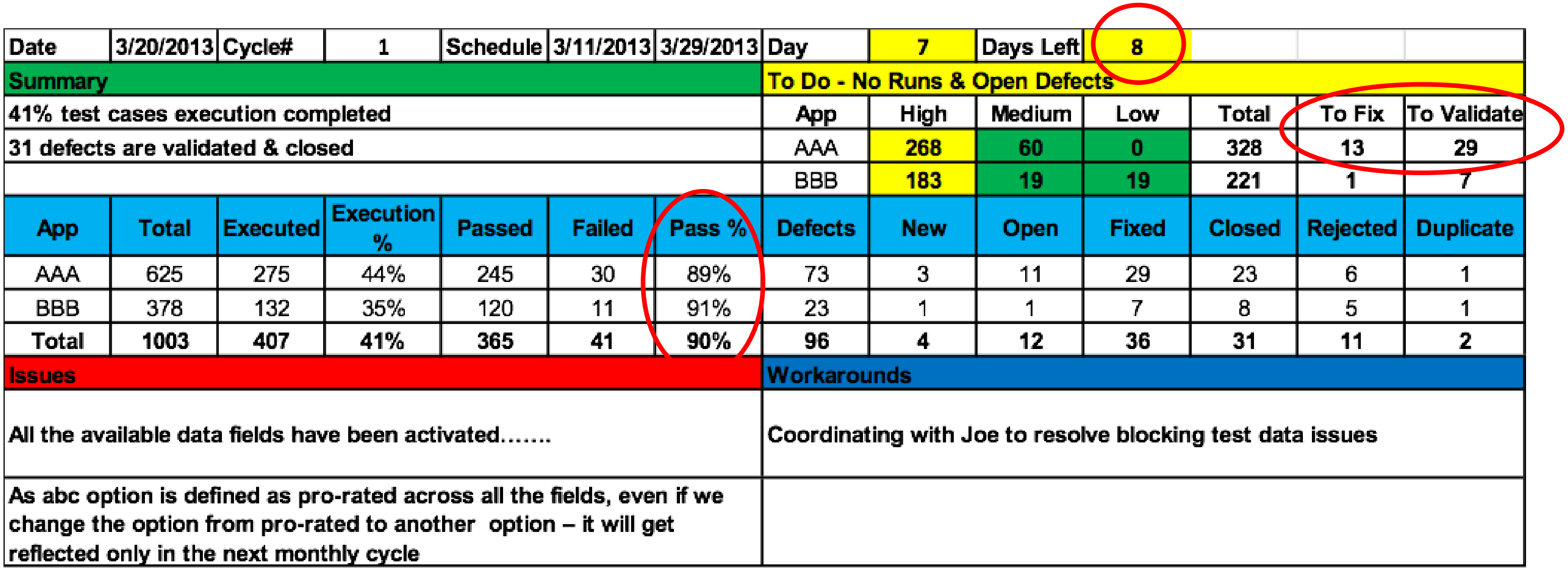
- A test performed at the Regression Test Stage validates that previously tested functionality still performs as expected when new or changed functionality is introduced into the system under test. It takes place in test environments **SIT/UAT environments)**. This test type is a good candidate for automation efforts.

- Regression Test executes the entire system in a test environment created to resemble ***as closely as possible*** the production environment to verify previous functionality still performs with changes applied to the software product. The intent of Regression Testing is to ensure that all functional features of the software product work before and after modifications to code, configuration or environment. Once the tested system has met the exit criteria for Regression Testing, the code is promoted to the next Stage for the User Acceptance Test.

- Building a Regression Test Bed is an incremental, iterative process that incorporates manual test case development, automated test cases development and repeated validation. A systematic approach to building a regression test bed is depicted below. It is recommended that static data be used throughout all Regression runs for the purpose of before and after compares. However it is understood that some applications/systems are unable to do so, due to data volume and the inability to refresh extremely large amounts of data.

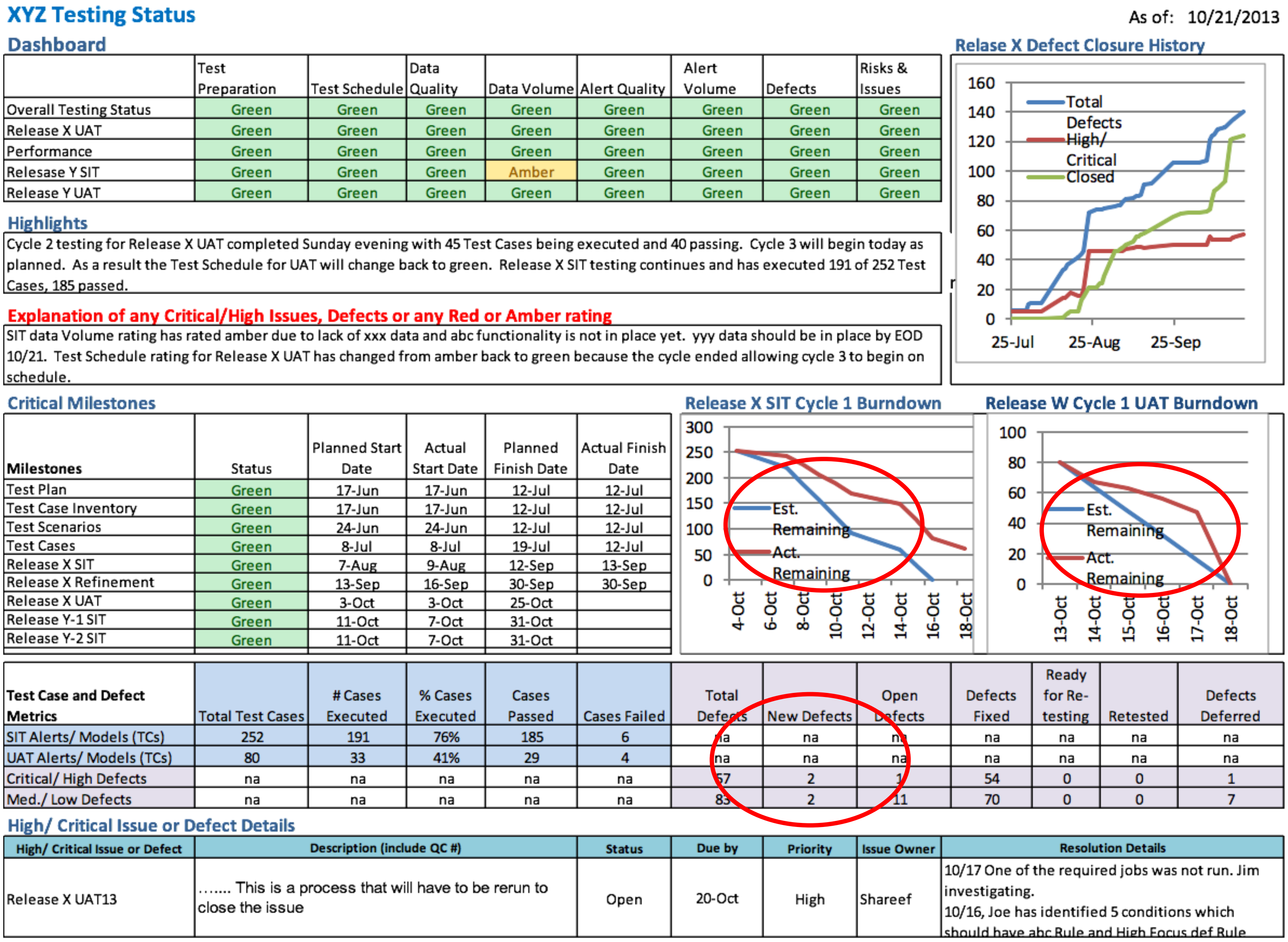
Purpose of User Acceptance Test	• To validate User visible function for the integrated platform
Entry Criteria	• Successful completion of theIntegration Tests
	• Defect Reporting and tracking system is implemented and leveraged
Major Testing Activities	• Execute all Test Cases and Scripts in the User Acceptance Test Detailed Test Plan
	• Issue Defects for all problems, issues or limitations found
	• Re-test Defects that have had fixes applied
Performing Role Feature or Function	• Users
	• Business Analysts
Exit Criteria	• All Test Cases and Scripts have been successfully executed, documented and maintained
	• All fixes have been re-tested

Measuring Testing Progress (1)



Date	**3/20/2013**	**Cycle#**	**1**	**Schedule**	**3/11/2013**	**3/29/2013**	**Day**	**7**	**Days Left**	**8**			
Summary							**To Do - No Runs & Open Defects**						
41% test cases execution completed							**App**	**High**	**Medium**	**Low**	**Total**	**To Fix**	**To Validate**
31 defects are validated & closed							AAA	**268**	**60**	**0**	**328**	**13**	**29**
							BBB	**183**	**19**	**19**	**221**	**1**	**7**
App	**Total**	**Executed**	**Execution %**	**Passed**	**Failed**	**Pass %**	**Defects**	**New**	**Open**	**Fixed**	**Closed**	**Rejected**	**Duplicate**
AAA	625	275	44%	245	30	89%	73	3	11	29	23	6	1
BBB	378	132	35%	120	11	91%	23	1	1	7	8	5	1
Total	**1003**	**407**	**41%**	**365**	**41**	**90%**	**96**	**4**	**12**	**36**	**31**	**11**	**2**
Issues							**Workarounds**						
All the available data fields have been activated…….							**Coordinating with Joe to resolve blocking test data issues**						
As abc option is defined as pro-rated across all the fields, even if we change the option from pro-rated to another option – it will get reflected only in the next monthly cycle													

Measuring Testing Progress (2)



XYZ Testing Status

As of: 10/21/2013

Dashboard

	Test Preparation	Test Schedule	Data Quality	Data Volume	Alert Quality	Alert Volume	Defects	Risks & Issues
Overall Testing Status	Green	Green	Green	Green	Green	Green	Green	Green
Release X UAT	Green	Green	Green	Green	Green	Green	Green	Green
Performance	Green	Green	Green	Green	Green	Green	Green	Green
Relesase Y SIT	Green	Green	Green	Amber	Green	Green	Green	Green
Release Y UAT	Green	Green	Green	Green	Green	Green	Green	Green

Relase X Defect Closure History

Highlights

Cycle 2 testing for Release X UAT completed Sunday evening with 45 Test Cases being executed and 40 passing. Cycle 3 will begin today as planned. As a result the Test Schedule for UAT will change back to green. Release X SIT testing continues and has executed 191 of 252 Test Cases, 185 passed.

Explanation of any Critical/High Issues, Defects or any Red or Amber rating

SIT data Volume rating has rated amber due to lack of xxx data and abc functionality is not in place yet. yyy data should be in place by EOD 10/21. Test Schedule rating for Release X UAT has changed from amber back to green because the cycle ended allowing cycle 3 to begin on schedule.

Critical Milestones

Milestones	Status	Planned Start Date	Actual Start Date	Planned Finish Date	Actual Finish Date
Test Plan	Green	17-Jun	17-Jun	12-Jul	12-Jul
Test Case Inventory	Green	17-Jun	17-Jun	12-Jul	12-Jul
Test Scenarios	Green	24-Jun	24-Jun	12-Jul	12-Jul
Test Cases	Green	8-Jul	8-Jul	19-Jul	12-Jul
Release X SIT	Green	7-Aug	9-Aug	12-Sep	13-Sep
Release X Refinement	Green	13-Sep	16-Sep	30-Sep	30-Sep
Release X UAT	Green	3-Oct	3-Oct	25-Oct	
Release Y-1 SIT	Green	11-Oct	7-Oct	31-Oct	
Release Y-2 SIT	Green	11-Oct	7-Oct	31-Oct	

Release X SIT Cycle 1 Burndown

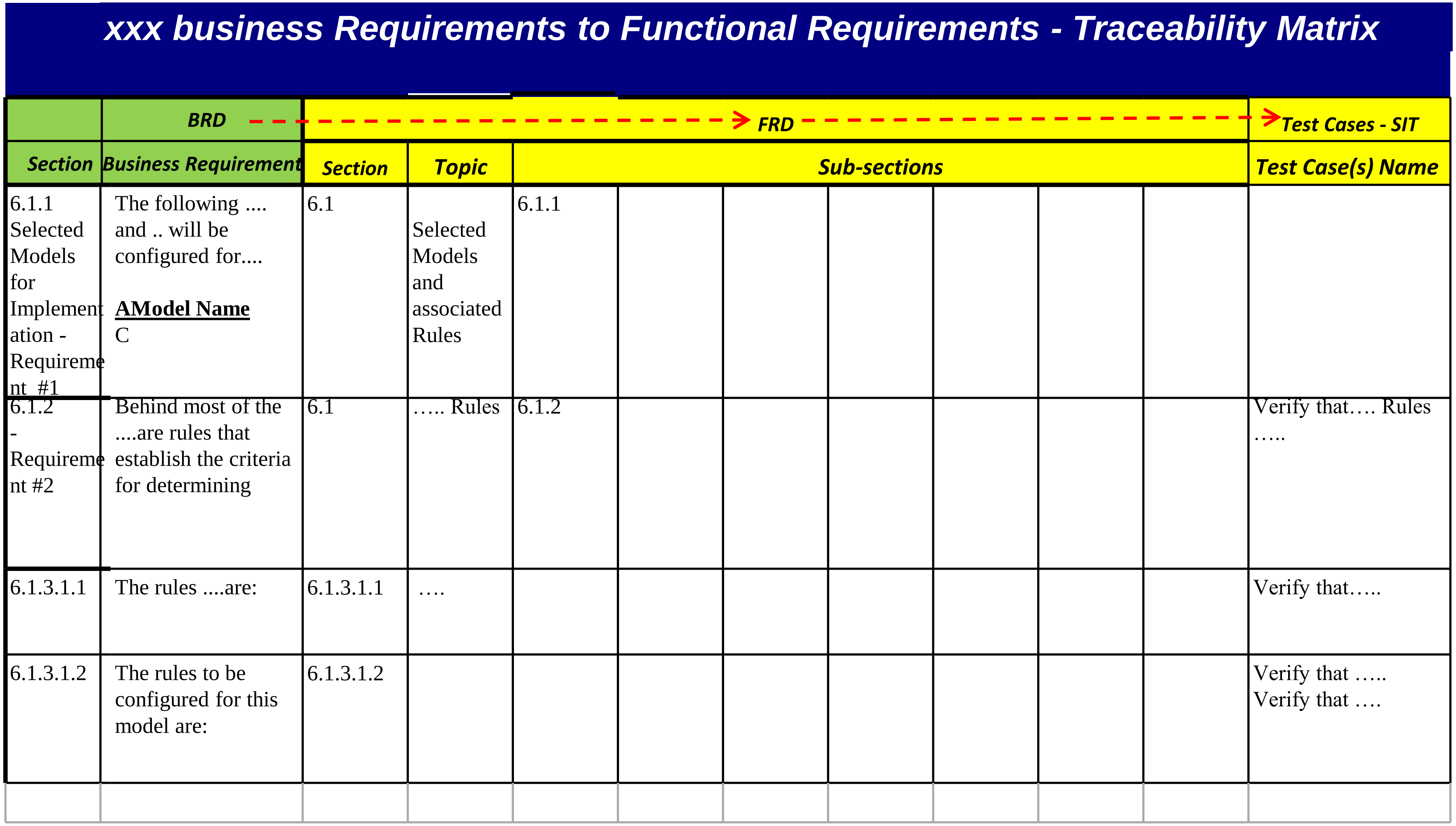
Release W Cycle 1 UAT Burndown

Test Case and Defect Metrics	Total Test Cases	# Cases Executed	% Cases Executed	Cases Passed	Cases Failed	Total Defects	New Defects	Open Defects	Defects Fixed	Ready for Re-testing	Retested	Defects Deferred
SIT Alerts/ Models (TCs)	252	191	76%	185	6	na	na	na	na	na	na	na
UAT Alerts/ Models (TCs)	80	33	41%	29	4	na	na	na	na	na	na	na
Critical/ High Defects	na	na	na	na	na	57	2	1	54	0	0	1
Med./ Low Defects	na	na	na	na	na	83	2	11	70	0	0	7

High/ Critical Issue or Defect Details

High/ Critical Issue or Defect	Description (include QC #)	Status	Due by	Priority	Issue Owner	Resolution Details
Release X UAT13	……. This is a process that will have to be rerun to close the issue	Open	20-Oct	High	Shareef	10/17 One of the required jobs was not run. Jim investigating. 10/16, Joe has identified 5 conditions which should have abc Rule and High Focus def Rule

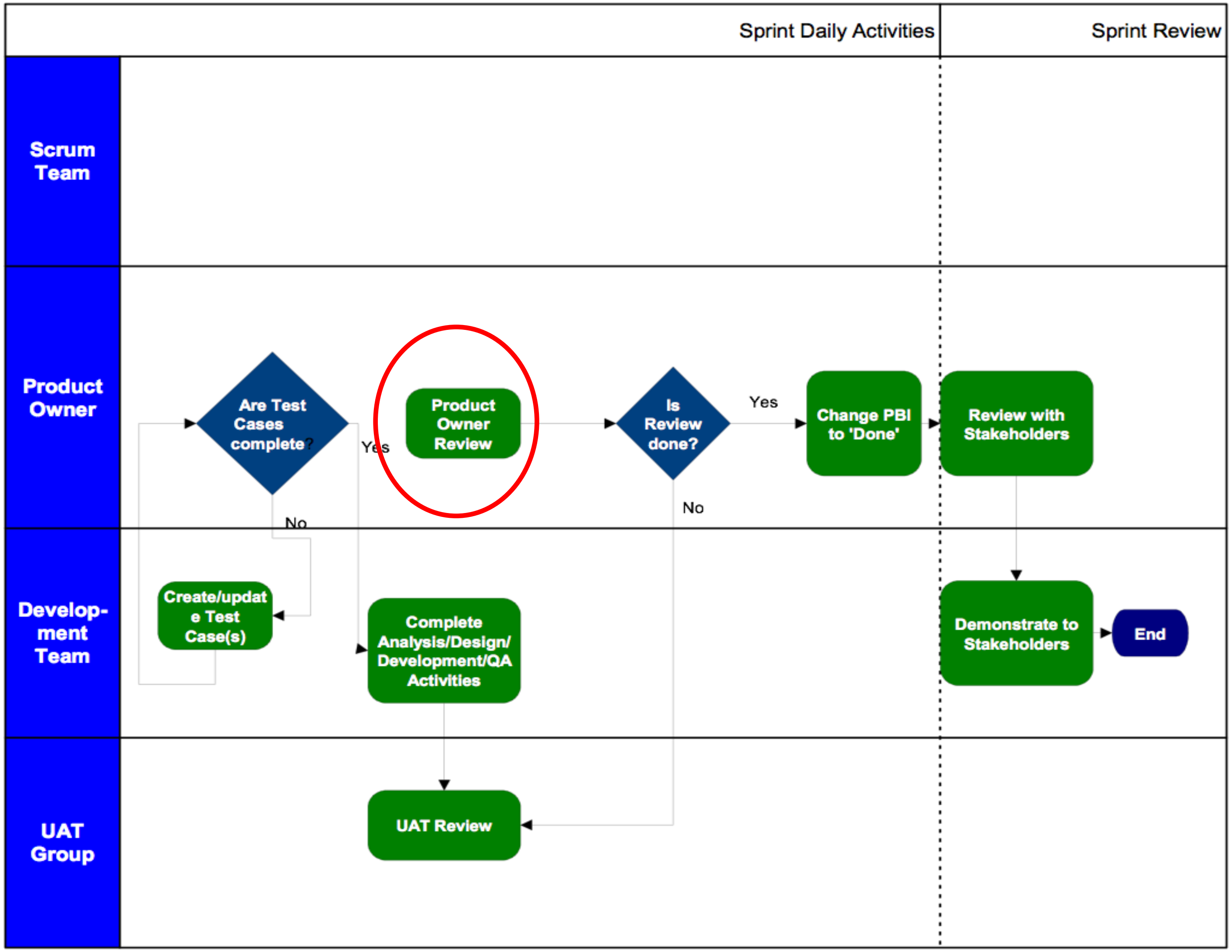
A Traceability Matrix



xxx business Requirements to Functional Requirements - Traceability Matrix

	BRD	FRD									Test Cases - SIT
Section	Business Requirement	Section	Topic	Sub-sections							Test Case(s) Name
6.1.1 Selected Models for Implementation - Requirement #1	The following .... and .. will be configured for.... **AModel Name** C	6.1	Selected Models and associated Rules	6.1.1							
6.1.2 - Requirement #2	Behind most of the ....are rules that establish the criteria for determining	6.1	….. Rules	6.1.2							Verify that…. Rules …..
6.1.3.1.1	The rules ....are:	6.1.3.1.1	….								Verify that…..
6.1.3.1.2	The rules to be configured for this model are:	6.1.3.1.2									Verify that ….. Verify that ….

Test Driven Development (TDD) in an *Agile* environment

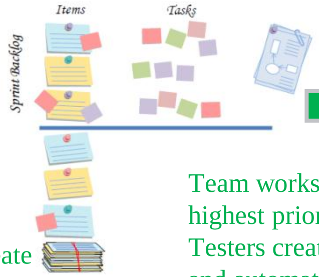


Do you think that this should also work in a Waterfall environment?

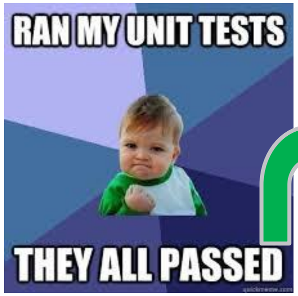
Test Driven Development (TDD) in practice

Sprint Planning

Estimate stories & create tasks in TFS

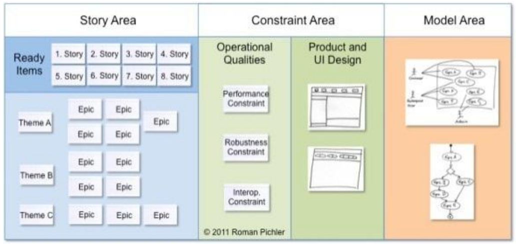


Team works on tasks of highest prioritized story
Testers create test cases and automate UI scripts



Developers

Automated builds and automated unit test runs with each build



Product backlog

Create tasks for defects found within the sprint

Manually test stories in current sprint

Typical 2 week sprint

Run automated regression tests –
Integration & User Acceptance levels

Create bug records for defects found in production and add to product backlog

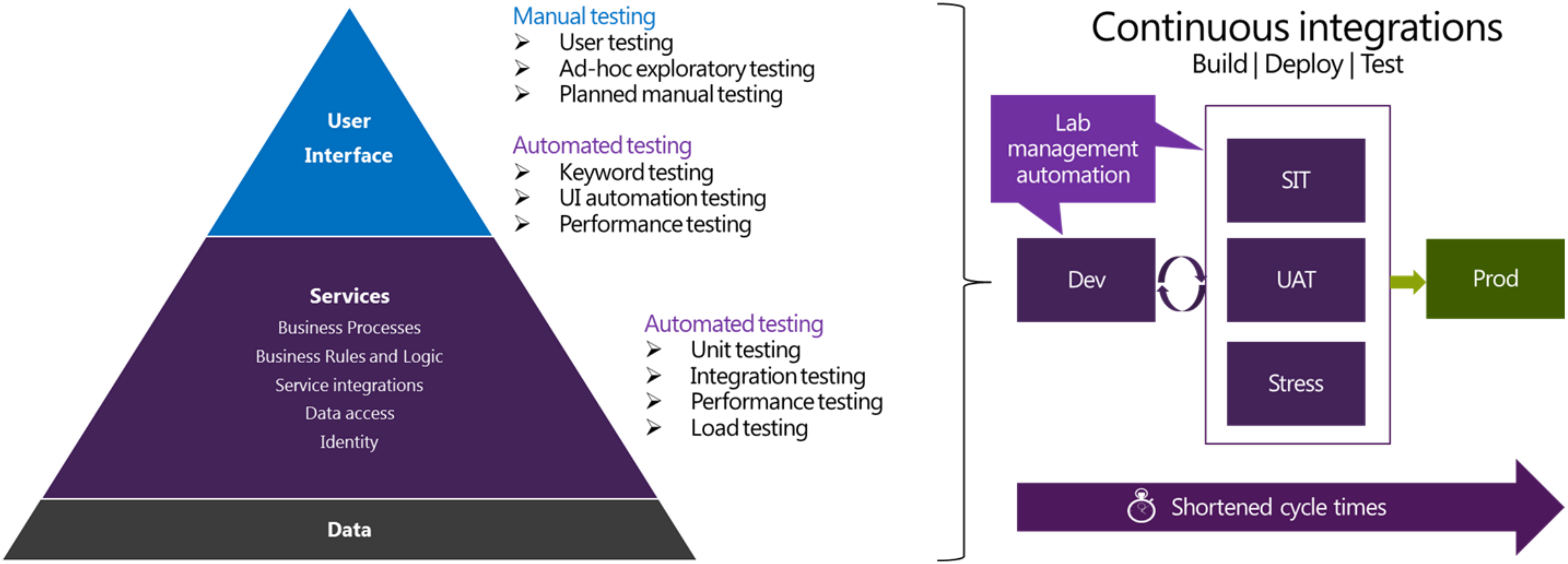
Refine the backlog and release plan

Backlog refinement

Create tasks for defects found within the sprint

new
UAT testers test ^ overall functionality before release

Continuous acceptance testing

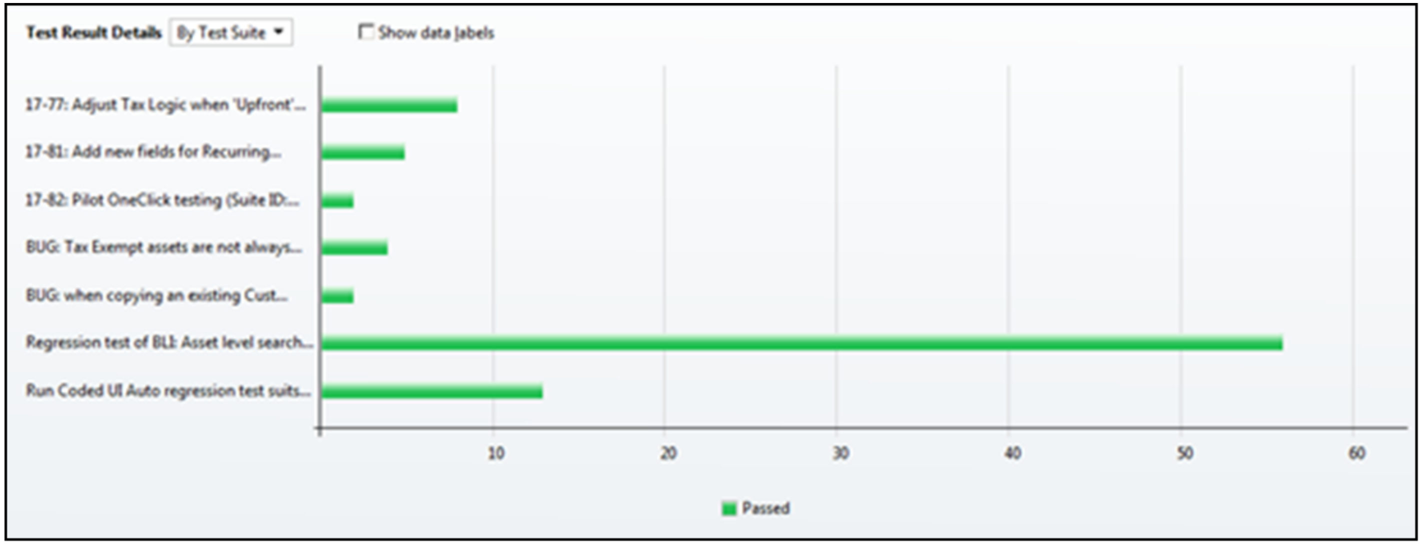


Manual testing
- User testing
- Ad-hoc exploratory testing
- Planned manual testing

Automated testing
- Keyword testing
- UI automation testing
- Performance testing

Automated testing
- Unit testing
- Integration testing
- Performance testing
- Load testing

Measuring Testing Progress (3) – from an Agile project via TFS

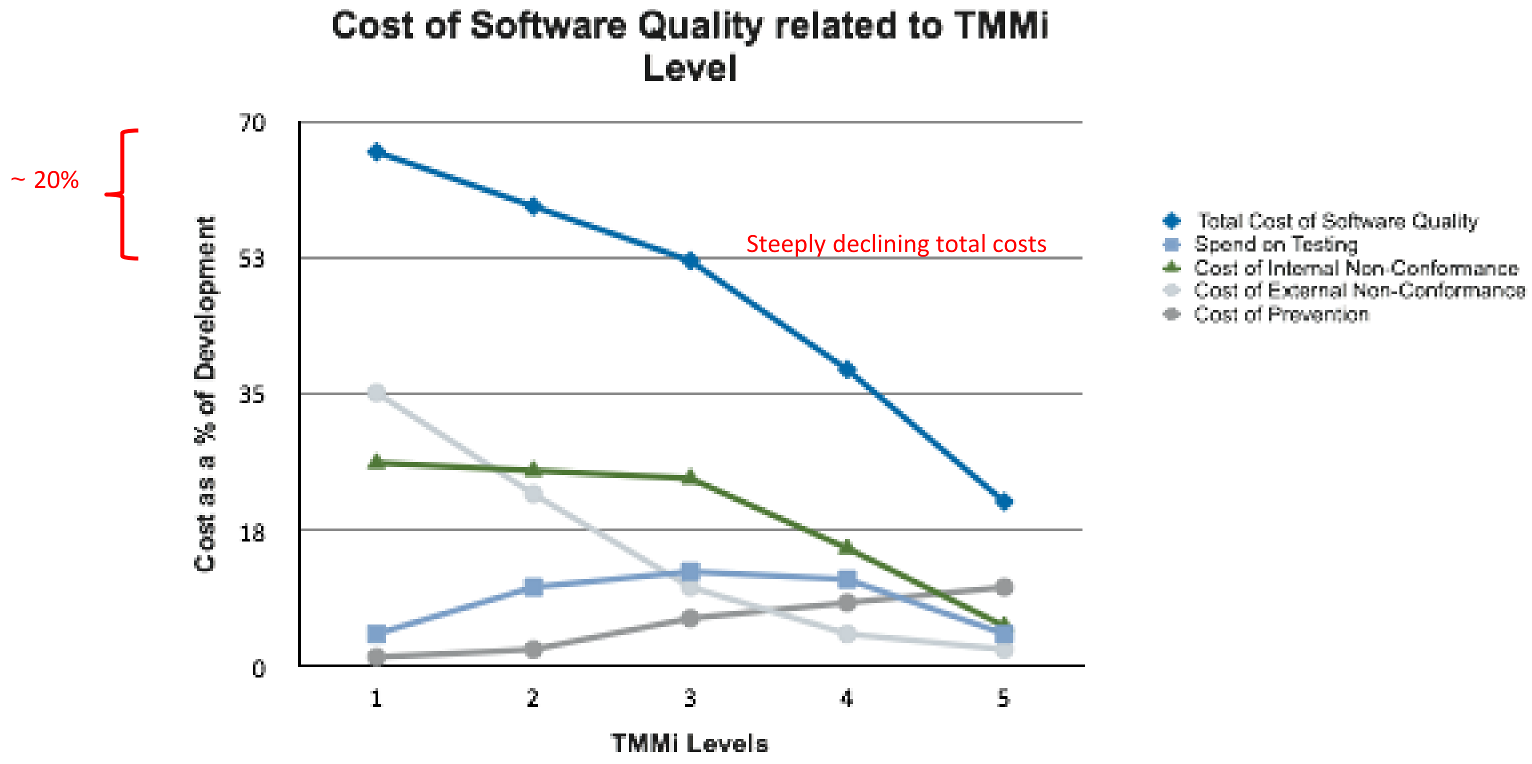

Test Result Details
By Test Suite
□ Show data labels
17-77: Adjust Tax Logic when 'Upfront'...
17-81: Add new fields for Recurring...
17-82: Pilot OneClick testing (Suite ID:...
BUG: Tax Exempt assets are not always...
BUG: when copying an existing Cust...
Regression test of BLI: Asset level search...
Run Coded UI Auto regression test suits...
10
20
30
40
50
60
🟩 Passed

The automated testing results chart shows the # and state (green = passed) of automated testing done in this sprint;
Rows = each story in this Sprint; Columns = # automated test cases

A TCOE can significantly reduce your Cost of Quality

- As investment in Testing & Prevention techniques increase, cost of quality decreases

- If you targeted Level 3 (TCOE established; testing well integrated into SDLC; good testing training program; test standards controlled and monitored), a **20% reduction in costs** should be realized

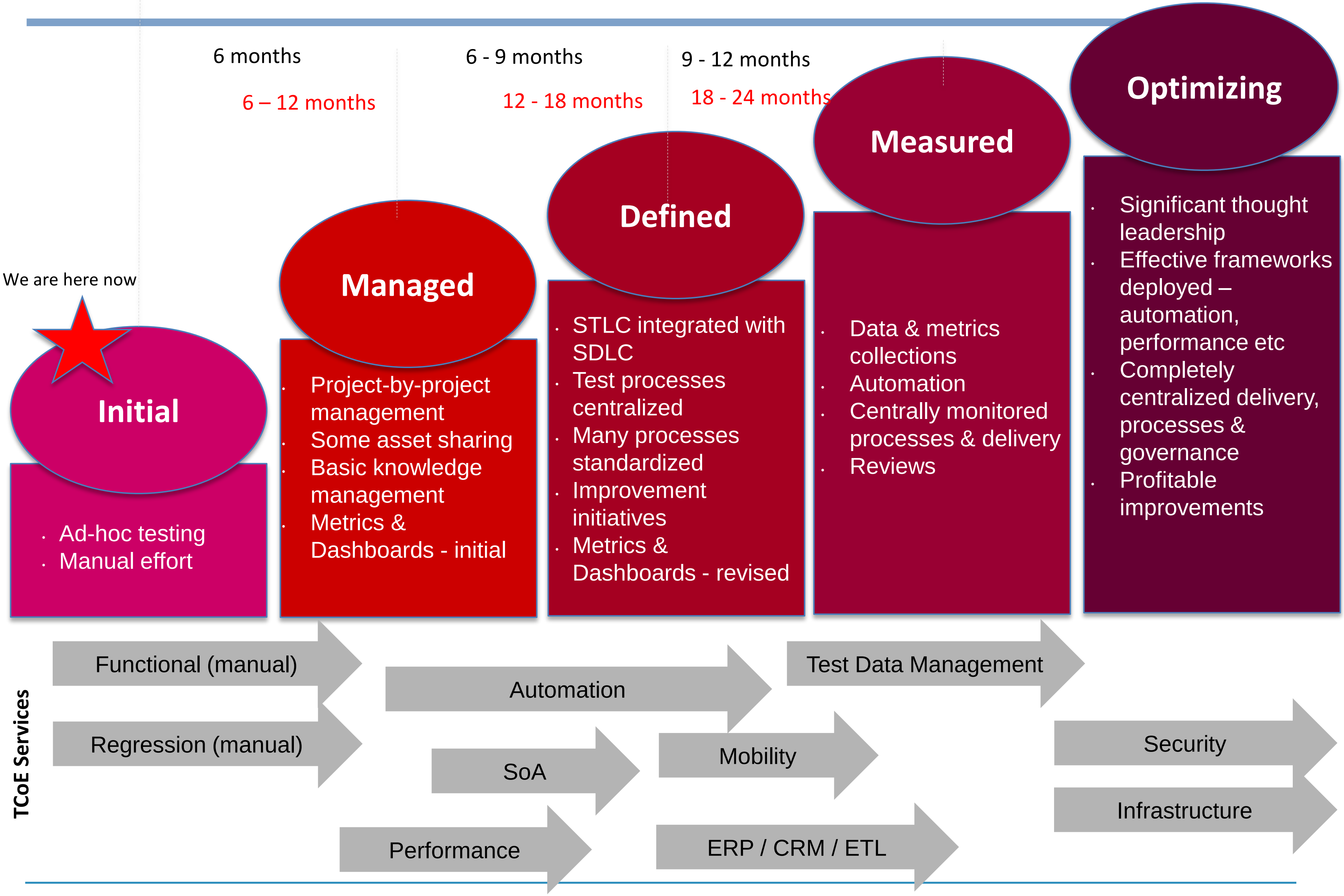


~ 20%

Steeply declining total costs

Acknowledgement & apology to Herb Krasner, U of Texas

A sample CoE Maturity Roadmap



6 months

6 – 12 months

6 - 9 months

12 - 18 months

9 - 12 months

18 - 24 months

We are here now

Initial

- Ad-hoc testing
- Manual effort

Managed

- Project-by-project management
- Some asset sharing
- Basic knowledge management
- Metrics & Dashboards - initial

Defined

- STLC integrated with SDLC
- Test processes centralized
- Many processes standardized
- Improvement initiatives
- Metrics & Dashboards - revised

Measured

- Data & metrics collections
- Automation
- Centrally monitored processes & delivery
- Reviews

Optimizing

- Significant thought leadership
- Effective frameworks deployed – automation, performance etc
- Completely centralized delivery, processes & governance
- Profitable improvements

TCoE Services

Functional (manual)

Regression (manual)

Automation

SoA

Performance

Test Data Management

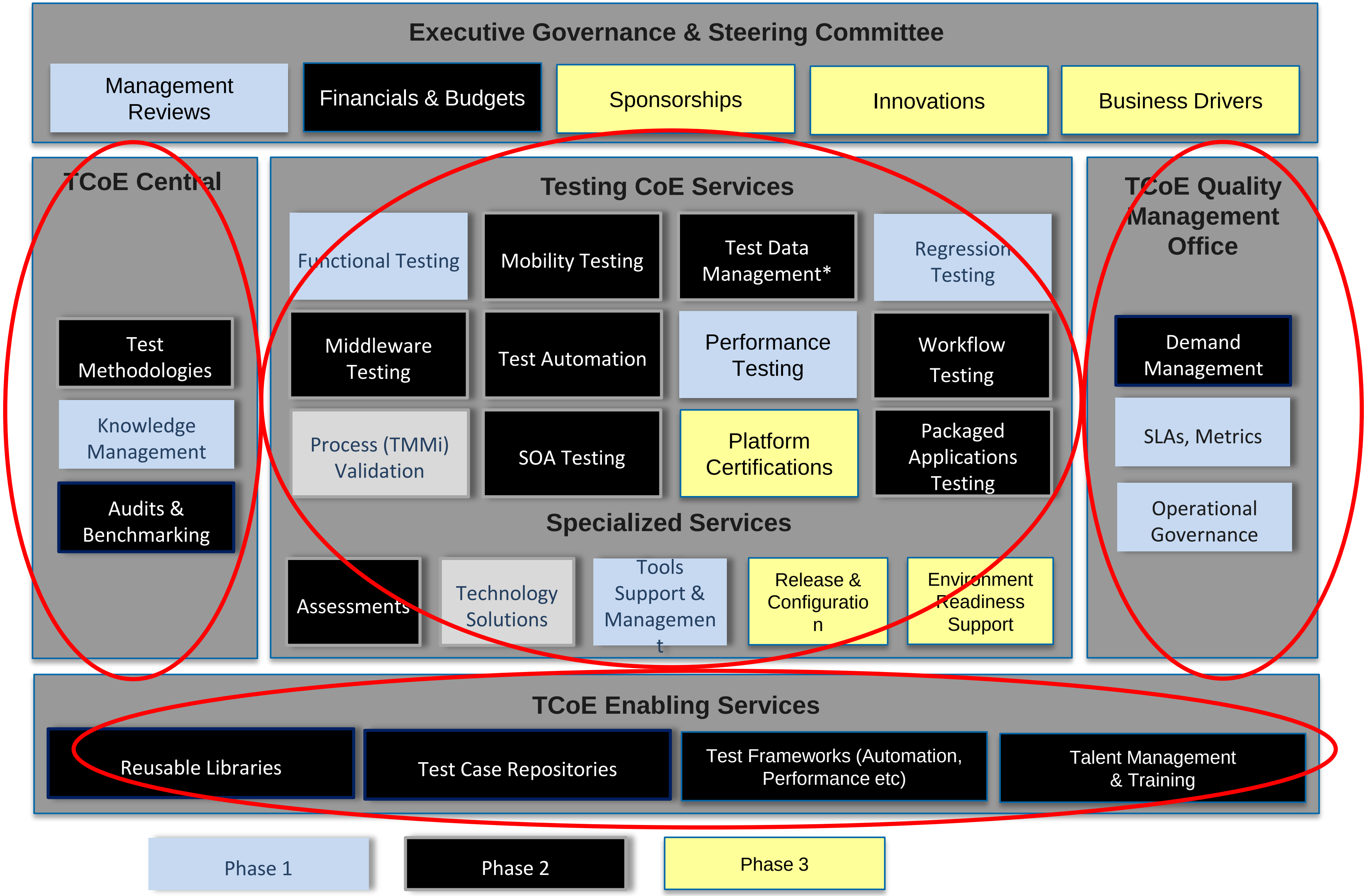
Mobility

ERP / CRM / ETL

Security

Infrastructure

A sample TCoE – "To-Be" Operating Model



Executive Governance & Steering Committee

- Management Reviews
- Financials & Budgets
- Sponsorships
- Innovations
- Business Drivers

TCoE Central

- Test Methodologies
- Knowledge Management
- Audits & Benchmarking

Testing CoE Services

- Functional Testing
- Mobility Testing
- Test Data Management*
- Regression Testing
- Middleware Testing
- Test Automation
- Performance Testing
- Workflow Testing
- Process (TMMi) Validation
- SOA Testing
- Platform Certifications
- Packaged Applications Testing

Specialized Services

- Assessments
- Technology Solutions
- Tools Support & Management
- Release & Configuration
- Environment Readiness Support

TCoE Quality Management Office

- Demand Management
- SLAs, Metrics
- Operational Governance

TCoE Enabling Services

- Reusable Libraries
- Test Case Repositories
- Test Frameworks (Automation, Performance etc)
- Talent Management & Training

Phase 1

Phase 2

Phase 3

Some references, terms (1)

“The Art of Unit Testing, Second Edition: with examples in C#“ by Roy Osherove

http://www.tmmi.org/ - The TMMi Foundation

“User Stories Applied – for Agile software development” by Mike Cohn

“Perfect Software: And other illusions about testing” by Gerald Weinberg

“The Art of Software Testing” by Glenford Myers

Some references, terms (2)

Glossary of Testing Types:

- (user) acceptance testing
- alpha testing
- beta testing
- black-box testing
- component /unit testing
- dress rehearsal testing
- dynamic testing
- exhaustive testing
- exploratory testing
- functional testing
- inspection
- (system) integration testing
- load testing
- non-functional testing
- performance testing
- regression testing
- security testing
- smoke test
- stress (volume) testing
- test automation
- Usability testing
- use case testing
- walkthrough
- white-box testing

Questions?