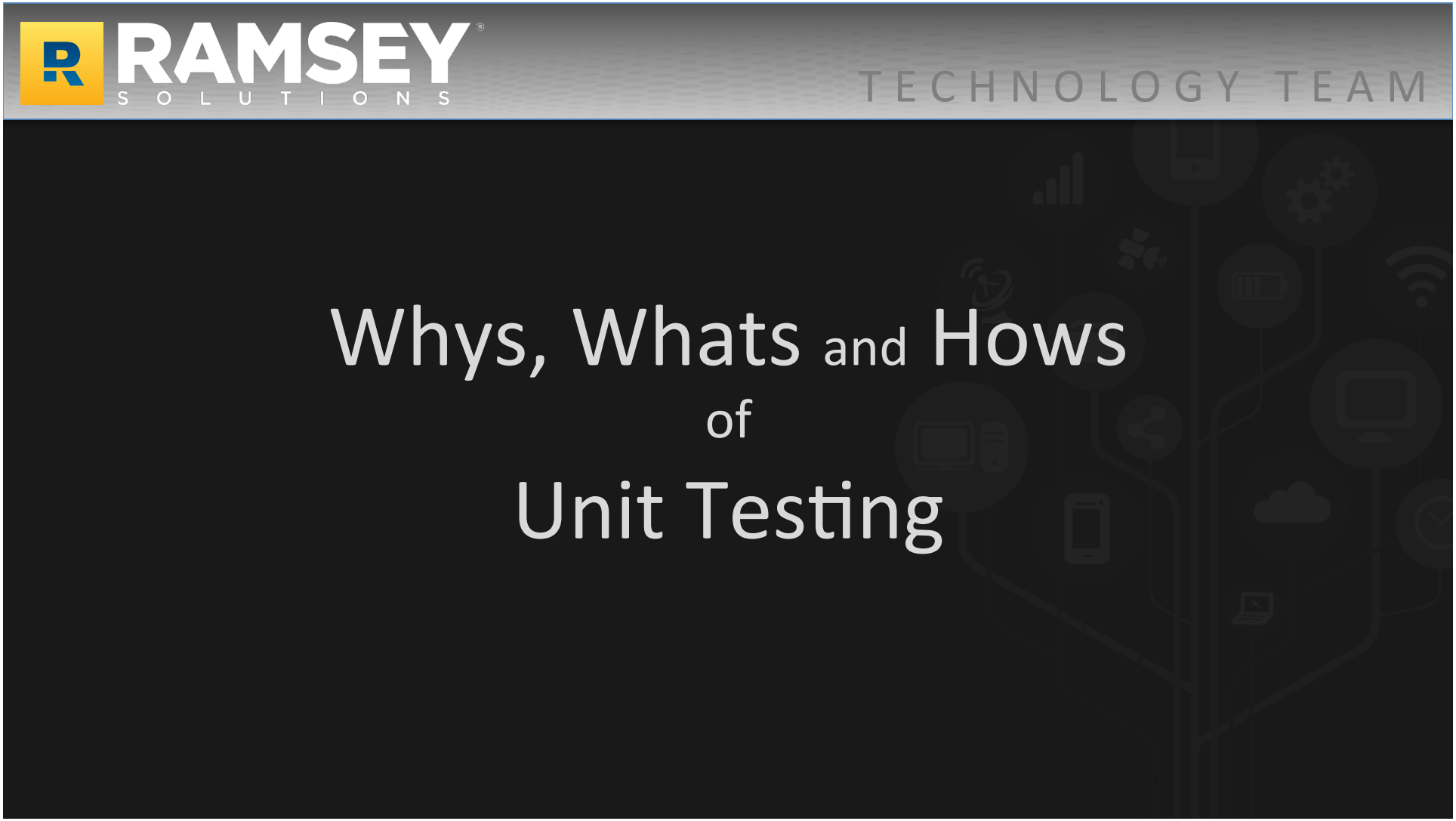
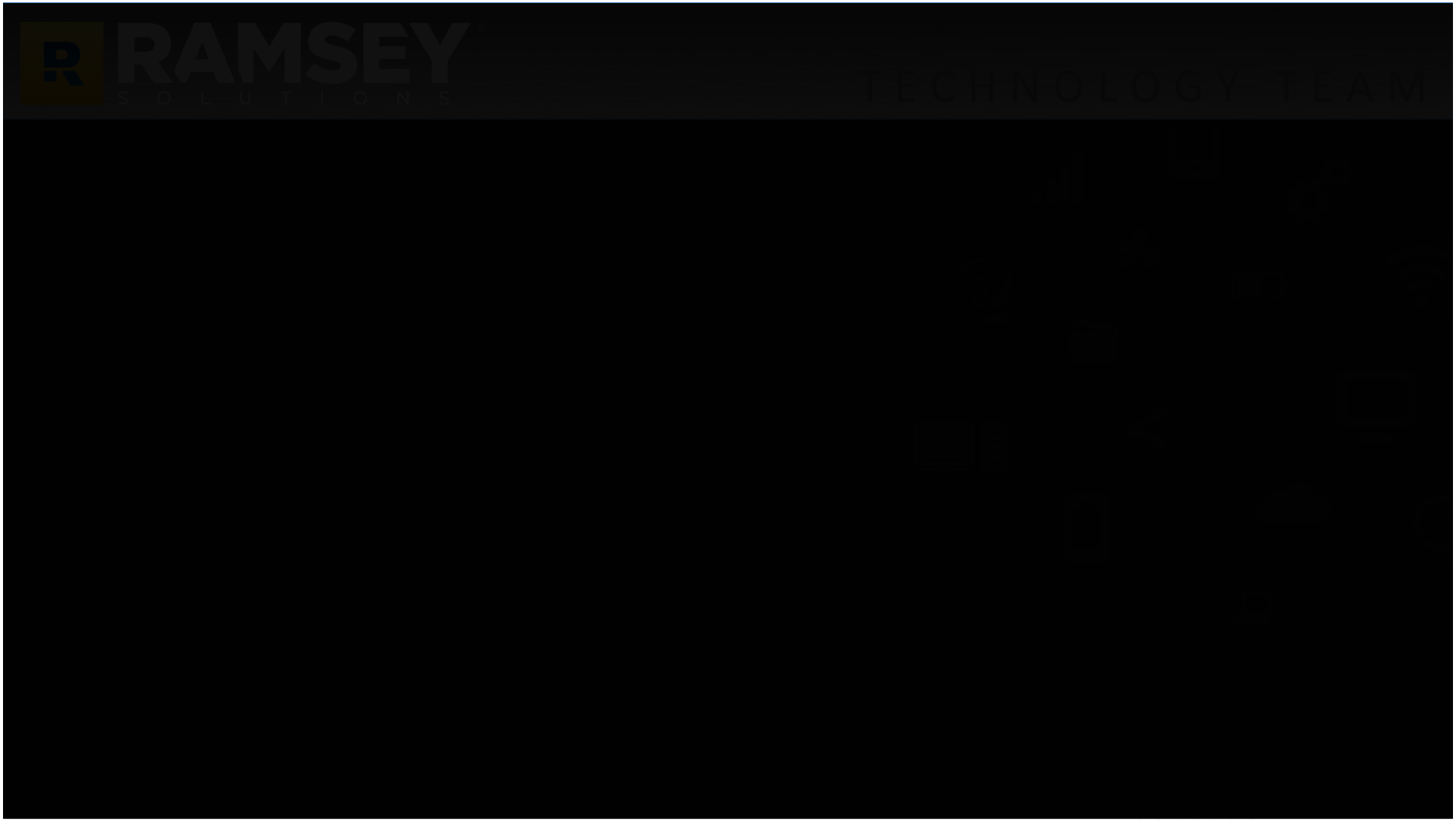

RAMSEY
SOLUTIONS
TECHNOLOGY TEAM
Whys, Whats and Hows
of
Unit Testing


RAMSEY
SOLUTIONS
TECHNOLOGY TEAM

1960's

NASA - Project Mercury

Objectives

- Get a human into outer space.
- Learn if humans could function in space.
- Could we get human and spacecraft back safely?

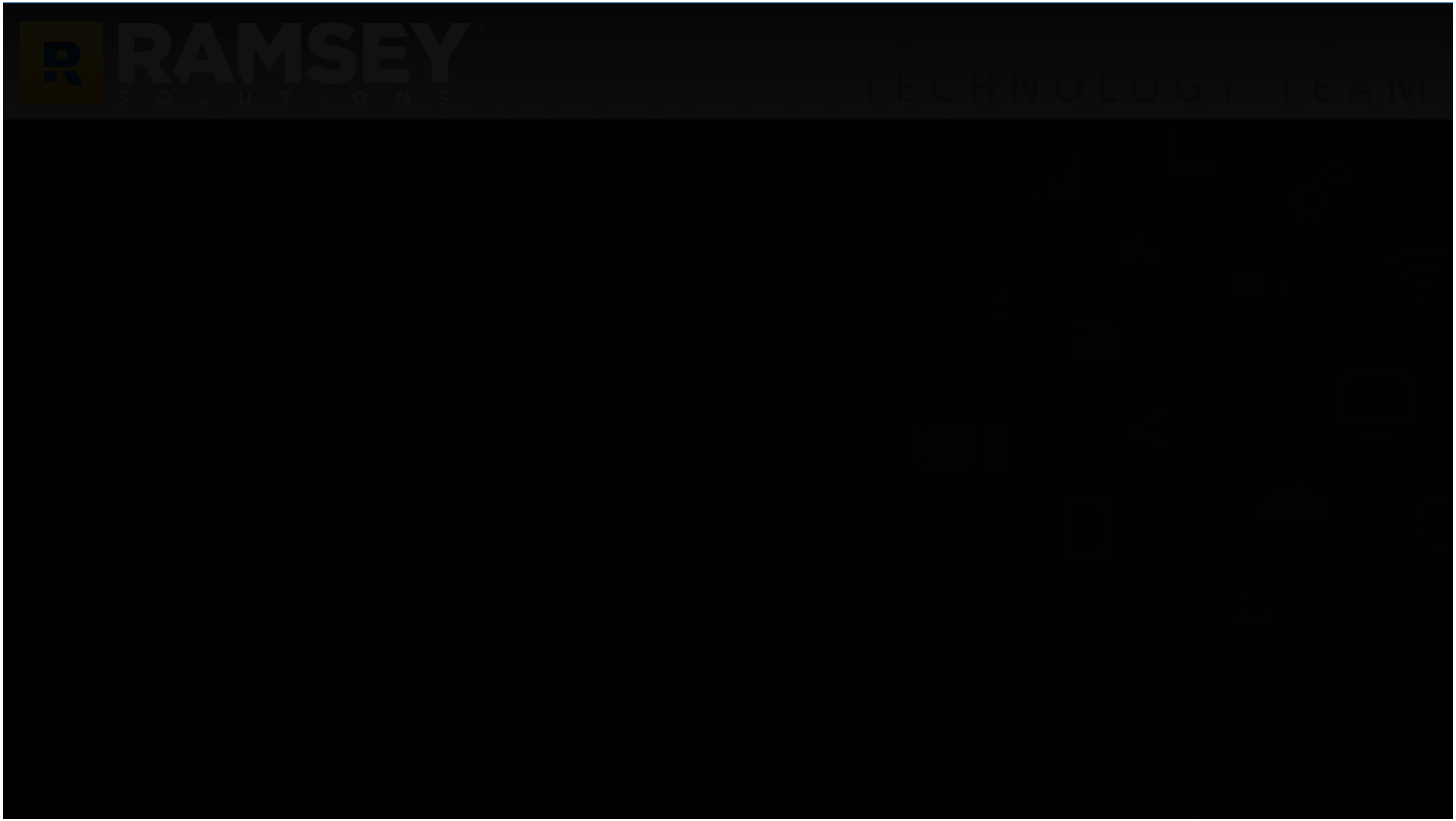
NASA Reduces Risk - Test Early - Test Often

Unit Testing Was Born

So what's Unit Testing?

Typically

- Part of the SDLC
- Testing smallest parts of an application (Methods, Functions)
- Assertion (Validation) of proper operation (including exception handling)

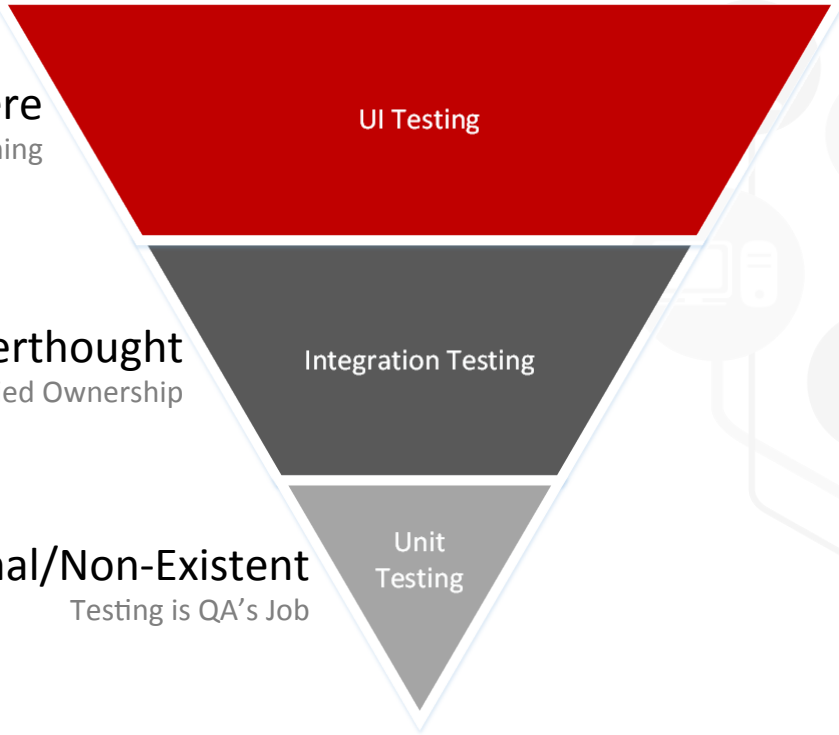

RAMSEY
SOLUTIONS
TECHNOLOGY TEAM

Traditional Testing Approach

Manual Testing Here
Ad Hoc / Test Everything

Typically an Afterthought
Unidentified Ownership

Minimal/Non-Existent
Testing is QA's Job




RAMSEY
SOLUTIONS
TECHNOLOGY TEAM

RAMSEY SOLUTIONS

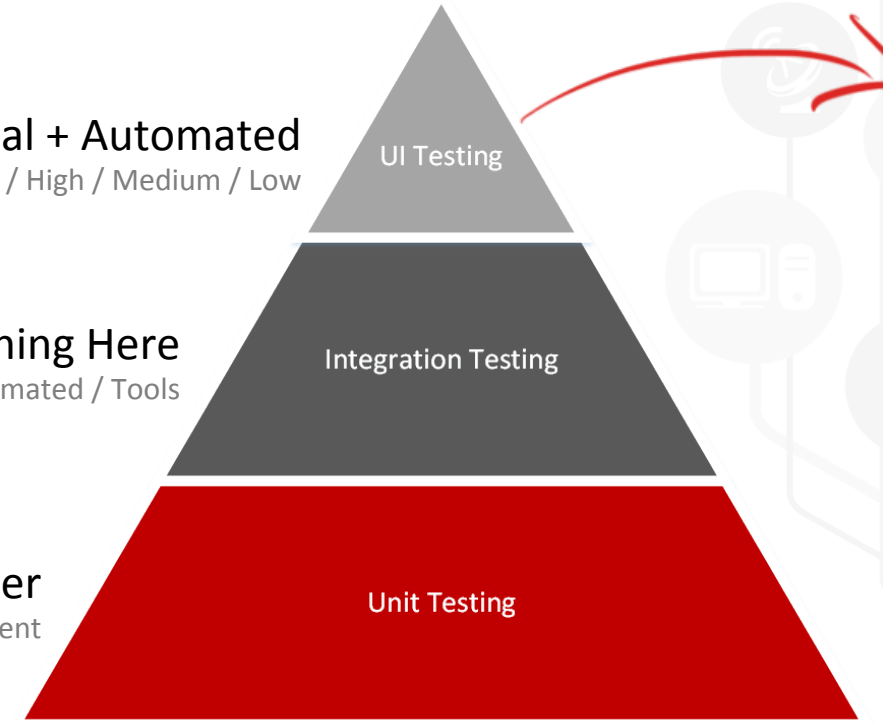
TECHNOLOGY TEAM

Revisiting the Idea of Testing

Manual + Automated
METS / Critical / High / Medium / Low

Learning Here
Environment Validation / Automated / Tools

Strategic Testing Earlier
Built by Development



Learn more about METS
Minimal Essential Testing Strategy
METSTesting.com

METS — **Physical Test Grid**

Category	Critical	High	Medium	Low
Pages	Page completes loading	Loading completes in a reasonable amount of time	Page loading consistent between browser versions	Page loading time reasonable under load
	Graphics, text and other elements seem to be in correct locations	Page has logical flow	Color contrast issues	
Graphics	Graphics loading completely	Graphics completed loading within same time frame	Consistently loading every time	
	Scaling, cropping or image quality problems	Highly artifact (distorted) image quality	Unpredictable color rendering at various bit depths	Non-browser safe color pallet used
	Rollover graphics displaying correctly	Graphic rollover state providing correct transition illusion	Preloaders working correctly for quick screen redraw	
	Graphical text within graphic is legible	Correctly spelled text within graphic		
Forms	Dropdown menus are functional	Dropdown menu contains all desired options	Submitted form contains dropdown menu selection(s)	
		Dropdown items are spelled correctly		
	Radio Buttons are functional	Radio button effect, turning off related radio buttons is working	Submitted form contains radio button selection	
		Radio buttons are spelled correctly		
	Checkboxes are functional	Selection of multiple checkboxes is possible	Submitted form contains checkbox selection(s)	
		Checkbox text is spelled correctly		
	Text fields and boxes are functional	Text field and boxes have correctly spelled default text	Text fields allow enough room for a typical data entry	
			Submitted form contains text field and text box information	
	Buttons are functional	Button submitting or resetting form correctly		
		Buttons are spelled correctly		
	Forms submitting correctly	Hitting Return/Enter submits form		
	Form data being received	Data from submitted form validated and correct		
	Form validation working correctly			
Links	Hyperlinks working	Hyperlinks going to correct destinations	Link to page is not an error page	
		Hyperlinks spelled correctly		
	Image links working.	Image links going to correct destination		
	Email links working.	Email links launching email client	Email link addressing mail client correctly	Email links going to correct recipient

RAMSEY SOLUTIONS

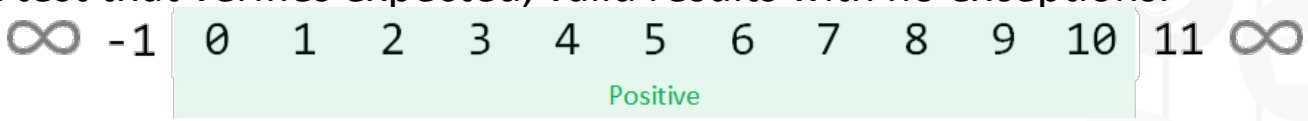
TECHNOLOGY TEAM

3 Simple Tests

Foundational Testing Concepts

Positive Test

A test that verifies expected, valid results with no exceptions.



Foundational Testing Concepts

Positive Test

A test that verifies expected, valid results with no exceptions.

Negative Test

A test that verifies expected, invalid results and exceptions.

Foundational Testing Concepts

Positive Test

A test that verifies expected, valid results with no exceptions.

Negative Test

A test that verifies expected, invalid results and exceptions.

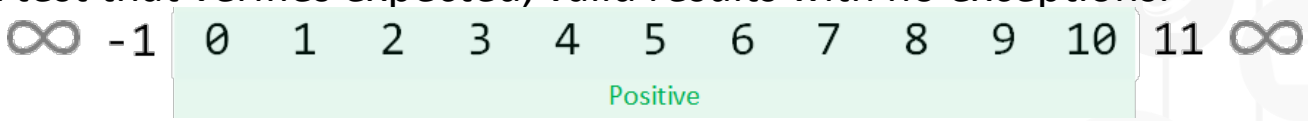
Boundary Test

A test that verifies expected, valid and invalid results at transition points.

Foundational Testing Concepts

Positive Test

A test that verifies expected, valid results with no exceptions.



Negative Test

A test that verifies expected, invalid results and exceptions.

Boundary Test

A test that verifies expected, valid and invalid results at transition points.

RAMSEY SOLUTIONS

TECHNOLOGY TEAM

Unit Test

Code Examples

Sample Method

Add()

```ruby
# Class providing simple addition functionality
class Adder

  # Method adding two numbers and returning result.
  # Throw exception for numbers < 0 and > 10
  def add(first_num, second_num)
    result = first_num + second_num
    raise UnderLimitResultException if result < 0
    raise OverLimitResultException if result > 10
    result
  end

end
```

add() method with two inputs, ***first_num*** and ***second_num***

add() sums inputs and stores value in ***result***.

result must be between 0 and 10.

add() method throws exception if ***result*** falls outside 0 - 10 range.

Unit Test Example

Positive Test

A test that verifies expected, valid results with no exceptions.

Test Data

TC #	first_num	second_num	Expected Result
1	3	4	7

```
1  # A Positive test of Add method.
2  it 'Test Positive Returns Expected Sum' do
3    expect(Adder.new.add(3,4)).to eq(7)
4  end
```

Unit Test Example

Negative Test

A test that verifies expected, invalid results and exceptions.

Test Data

TC #	first_num	second_num	Expected Result
1	4	-8	UnderLimitResultException
2	4	8	OverLimitResultException

```
1  # A Negative exception test of Add method.
2  it 'Test Negative Result Under 0 Throws UnderLimitResultException' do
3    expect { Adder.new.add(4,-8) }.to raise_error(UnderLimitResultException)
4  end
```

Unit Test Example

Negative Test

A test that verifies expected, invalid results and exceptions.

Test Data

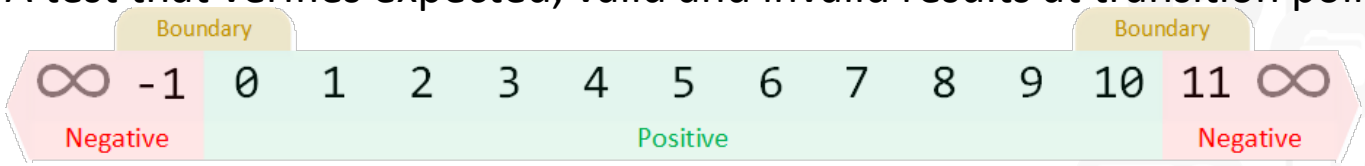
TC #	first_num	second_num	Expected Result
1	4	-8	UnderLimitResultException
2	4	8	OverLimitResultException

```
1  # A Negative exception test of Add method.
2  it 'Test Negative Result Over 10 Throws OverLimitResultException' do
3    expect { Adder.new.add(4,8) }.to raise_error(OverLimitResultException)
4  end
```

Unit Test Example

Boundary Test

A test that verifies expected, valid and invalid results at transition points.



Test Data

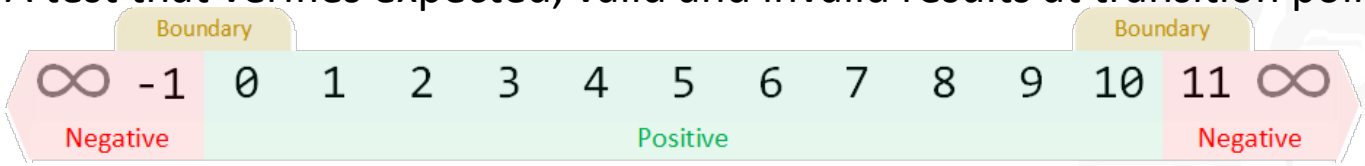
TC #	first_num	second_num	Expected Result
1	0	-1	UnderLimitResultException
2	0	0	0
3	9	1	10
4	9	2	OverLimitResultException

```
1  # Boundary test of Add method.
2  it 'Test Negative Result Under 0 Throws UnderLimitResultException' do
3    expect { Adder.new.add(0,-1) }.to raise_error(UnderLimitResultException)
4  end
```

Unit Test Example

Boundary Test

A test that verifies expected, valid and invalid results at transition points.



Test Data

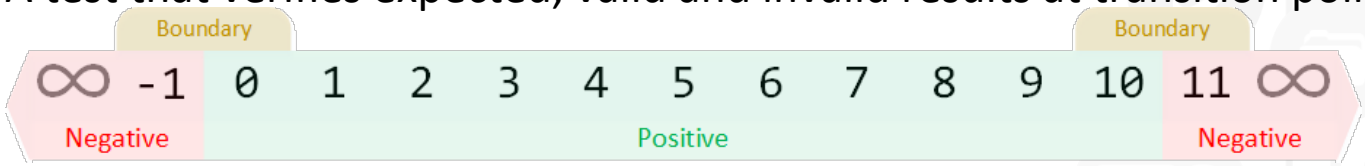
TC #	first_num	second_num	Expected Result
1	0	-1	UnderLimitResultException
2	0	0	0
3	9	1	10
4	9	2	OverLimitResultException

```
1  # Boundary test of Add method.
2  it 'Test Positive Normal Addition' do
3    expect(Adder.new.add(0,0)).to eq(0)
4  end
```

Unit Test Example

Boundary Test

A test that verifies expected, valid and invalid results at transition points.



Test Data

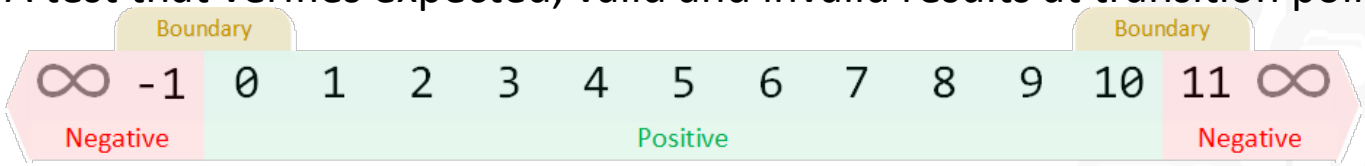
TC #	first_num	second_num	Expected Result
1	0	-1	UnderLimitResultException
2	0	0	0
3	9	1	10
4	9	2	OverLimitResultException

```
1  # Boundary test of Add method.
2  it 'Test Positive Normal Addition' do
3    expect(Adder.new.add(9,1)).to eq(10)
4  end
```

Unit Test Example

Boundary Test

A test that verifies expected, valid and invalid results at transition points.



Test Data

TC #	first_num	second_num	Expected Result
1	0	-1	UnderLimitResultException
2	0	0	0
3	9	1	10
4	9	2	OverLimitResultException

```
1  # Boundary test of Add method.
2  it 'Test Negative Result Over 10 Throws OverLimitResultException' do
3    expect { Adder.new.add(9,2) }.to raise_error(OverLimitResultException)
4  end
```

RAMSEY SOLUTIONS

TECHNOLOGY TEAM

Unit Testing

Advanced Concepts

Triangulation

Triangulation Test

A test that verifies return result is not hard coded.

Test Data

TC #	first_num	second_num	Expected Result
1	3	4	7
2	3	5	8

```
1  # A Triangulation test of Add method.
2  it 'Test Positive Returns Expected Sum' do
3    expect(Adder.new.add(first_num,second_num)).to eq(expected_result)
4  end
```

Measuring Maturity

Typical steps of software maturity

* Based upon the Capability Maturity Model

RAMSEY SOLUTIONS

TECHNOLOGY TEAM

Unit Test - Maturity Model

Ver. 02

CMM	Unit Test Level	Details	Reference
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 reusage.	
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.	https://en.wikipedia.org/wiki/Single_responsibility_principle
	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.	https://en.wikipedia.org/wiki/Code_coverage
	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.	

Final Thoughts

RAMSEY SOLUTIONS

TECHNOLOGY TEAM

Q&A

RAMSEY SOLUTIONS

TECHNOLOGY TEAM

Contact Information

Greg Paskal

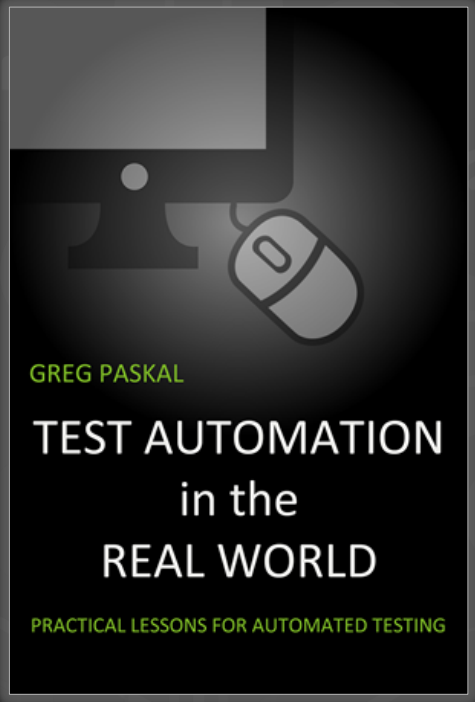
I would enjoy hearing how you're using the teachings from this presentation.

METS Minimal Essential Testing Strategy — **Physical Test Grid**

Category	Critical	High	Medium	Low
Pages	Page completes loading	Loading completes in a reasonable amount of time	Page loading consistent between browser versions	Page loading time reasonable under load
	Graphics, text and other elements seem to be in correct locations	Page has logical flow	Color contrast issues	
Graphics	Graphics loading completely	Graphics completed loading within same time frame	Consistently loading every time	
	Scaling, cropping or image quality problems	Highly artifact (distorted) image quality	Unpredictable color rendering at various bit depths	Non-browser safe color pallet used
	Rollover graphics displaying correctly	Graphic rollover state providing correct transition illusion	Preloaders working correctly for quick screen redraw	
	Graphical text within graphic is legible	Correctly spelled text within graphic		
Forms	Dropdown menus are functional	Dropdown menu contains all desired options	Submitted form contains dropdown menu selection(s)	
		Dropdown items are spelled correctly		
	Radio Buttons are functional	Radio button effect, turning off related radio buttons is working	Submitted form contains radio button selection	
		Radio buttons are spelled correctly		
	Checkboxes are functional	Selection of multiple checkboxes is possible	Submitted form contains checkbox selection(s)	
		Checkbox text is spelled correctly		
	Text fields and boxes are functional	Text field and boxes have correctly spelled default text	Text fields allow enough room for a typical data entry	
			Submitted form contains text field and text box information	
	Buttons are functional	Button submitting or resetting form correctly		
		Buttons are spelled correctly		
	Forms submitting correctly	Hitting Return/Enter submits form		
	Form data being received	Data from submitted form validated and correct		
	Form validation working correctly			
Links	Hyperlinks working	Hyperlinks going to correct destinations	Link to page is not an error page	
		Hyperlinks spelled correctly		
	Image links working.	Image links going to correct destination		
	Email links working.	Email links launching email client	Email link addressing mail client correctly	Email links going to correct recipient

Need a better Manual and Automated Test Strategy? Learn more about Greg's Minimal Essential Testing Strategy.

METSTesting.com



Interested in Test Automation? Greg shares from 30 years of experience in "Test Automation in the Real World".

RealWorldTestAutomation.com