



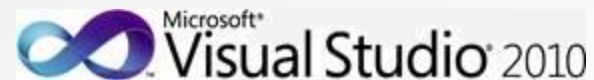
Addressing Non-Functional Requirements with Agile Practices

Mario Cardinal
Software Architect

Version: Dec 16th

Who Am I?

- Independent senior consultant
 - Software architecture
 - Agile coaching
 - ALM with Team Foundation Server
- www.mariocardinal.com



Agile is Like Teen Sex Because...

- Everyone wants to do it
- Many say they're doing it
- Everybody else seems to be doing more than you
- Very few of you/your friends are doing it correctly
- You start getting a bad reputation when you spend too much time 'Doing it'

Source: agile101.net

Agenda

1. Non-functional requirements
 - External and internal quality
2. Functional requirements and agile framework
 - User Story and scenario
3. Non-functional requirements and agile framework
 - Improve external quality using expectations
 - Ensure internal quality using sound engineering practices

Non-Functional Requirements

What are they?

- Specify "how well" the "what" must behave
 - Not about new features to deliver, but rather about desirable characteristics of existing features
 - Set constraints that typically cut across functional requirements
- Also known as "technical requirements", "quality attributes" or "quality of service requirements"

Non-Functional Requirements

It is all about quality

- Can be divided into two main categories:
 1. **External quality** such as performance, correctness, security and usability, which carry out the software's functions at run time, and as such, is not only visible by stakeholders but also highly desirable
 2. **Internal quality** such as maintainability, modifiability and testability, which is barely visible by stakeholders but simplify how to build the software

Non-Functional Requirements

Knowledge is not experience

- I do not intend to tell you how to satisfy the many non-functional requirements
 - It is a skill that one acquires with experience

Non-Functional Requirements

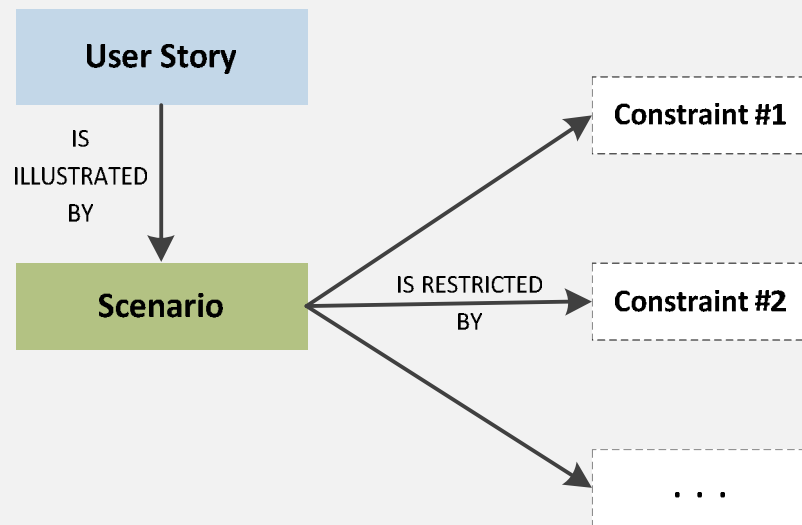
I aim for a simpler goal

- I will explain how to translate non-functional requirements into constraints
 - Constraints set a limit to comply with
 - Constraints guide your work
 - Constraints help determine whether you have satisfied the non-functional requirements

Non-Functional Requirements

Need to review functional requirements

- Constraints weave through the functional requirements



Functional Requirements

Express goals with user stories

- A user story is a short description written in everyday language that represents a discrete piece of demonstrable functionality
 - It is a desirable outcome by stakeholders
- Classic template
 - “As a < **role**>, I want <**goal**> so that <**benefit**>”

Functional Requirements

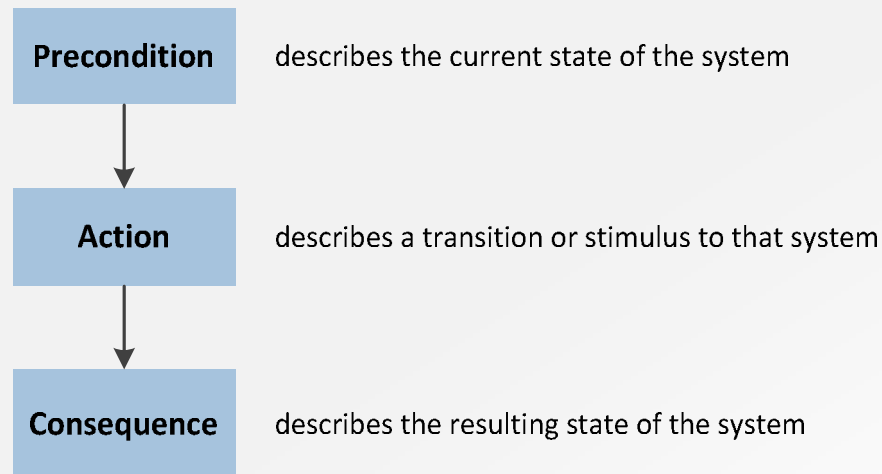
Example: User stories for a Transit Authority

- As a <**student**>, I want <**to buy a pass valid only on school days**> so that I can <**go to school**>
- As a <**worker**>, I want <**to buy a monthly pass**> so that I can <**go to work**>

Functional Requirements

Illustrate User Story with scenarios

- A scenario is a concrete example written in everyday language
- It describes a significant exercise that is required for the fulfillment of a user story



Functional Requirements

Confirm success criteria with scenarios

- Scenarios establish the conditions of acceptance
- Scenarios are concrete examples that says in the words of the stakeholders how they plan to verify the desirable outcome
- Scenarios enables the team to know when they are done
- Scenarios are a specification as important, if not more important, than stories

Functional Requirements

Express scenarios with formality

Given one precondition

And another precondition

And yet another precondition

When an action occurs

Then a consequence

And another consequence

Functional Requirements

Express scenarios with formality

Given an empty shopping cart is created

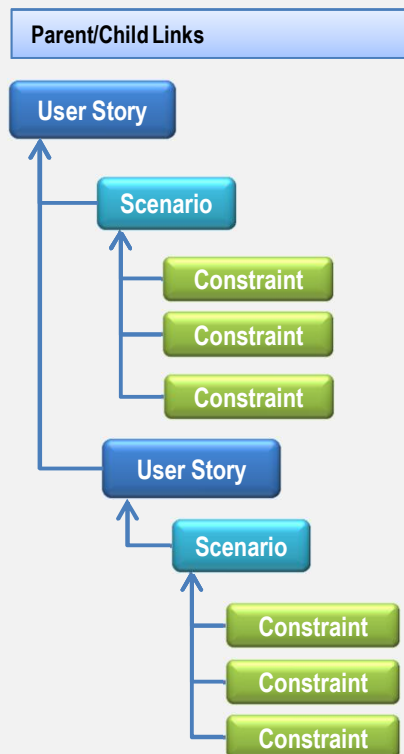
And a monthly student pass is added to shopping cart

When buyer checkout the shopping cart

Then a 76 dollars sale occurred

Functional Requirements

Store requirements in a database



Create a web
of
interconnected small pieces
of
requirements

Non-Functional Requirements

Two categories of constraint

- External quality
 - **Expectations** impose conditions that sets a limit to comply during software execution
- Internal quality
 - **Practices** ensure that the software construction is done correctly

External Quality

What is it?

Non-Functional Requirement	Definition
Correctness	Ability with which the software respects the specification.
Performance	Ease with which the software is doing the work it is supposed to do. Usually it is measured as a response time or a throughput.
Reliability	Ability with which the software performs its required functions under stated conditions for a specified period of time.
Robustness	Ability with which the software copes with errors during execution.
Scalability	Ability with which the software handles growing amounts of work in a graceful manner.
Security	Degree to which the software protects against threats.
Usability	Ease with which the software can be used by specified users to achieve measurable goals.

External Quality

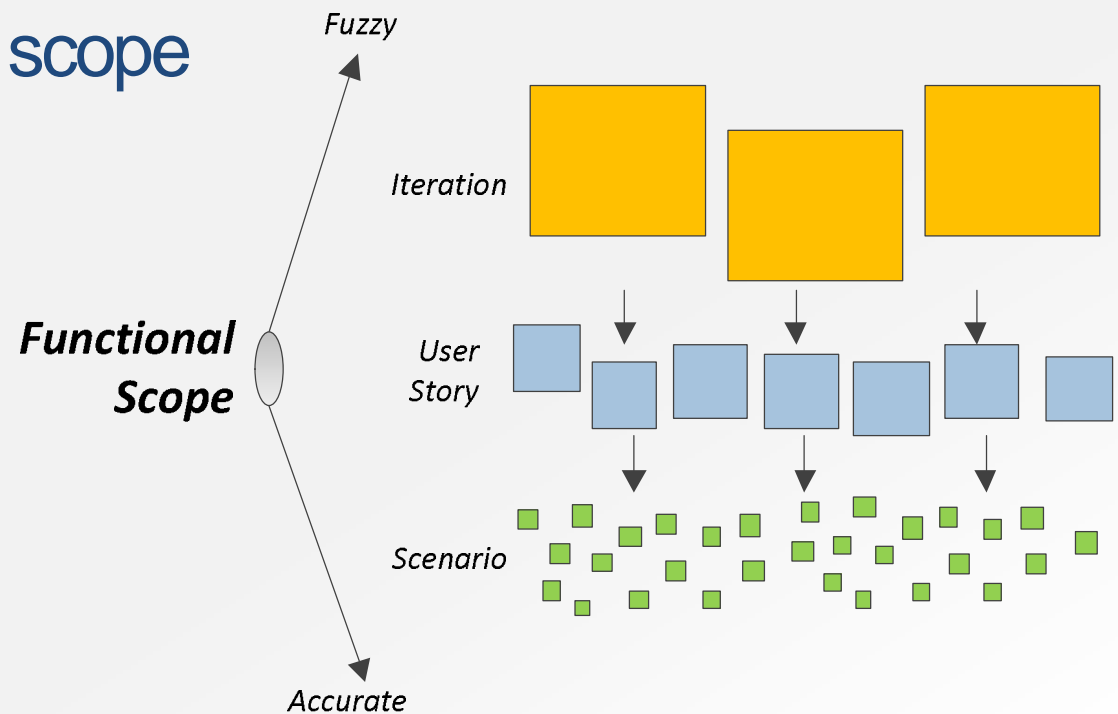
Expectations should be SMART

- **S**pecific
 - It should target a piece of functionality that is small, consistent and simple
- **M**easurable
 - It imposes a limit that is measurable, otherwise how would you know when you've addressed it
- **A**ttainable
 - It is recognized as achievable by the team
- **R**elevant
 - It is directly related, connected, and pertinent to the non-functional requirement
- **T**raceable
 - It is linked with a requirement and a target that justifies why it exists

Expectation

The most important element is the 'measure'

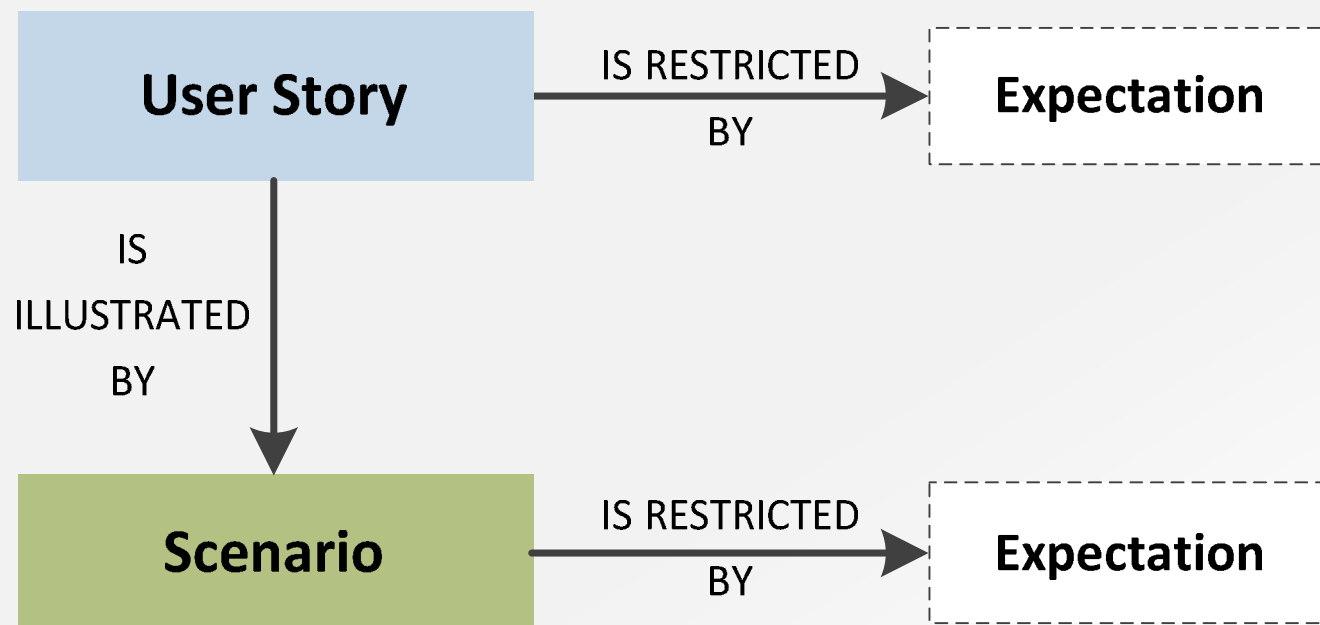
- Easier to express if you
 - Reduce the scale of what needs to be measured
 - Reduce functional scope



Expectation

Reduce the functional scope to a scenario

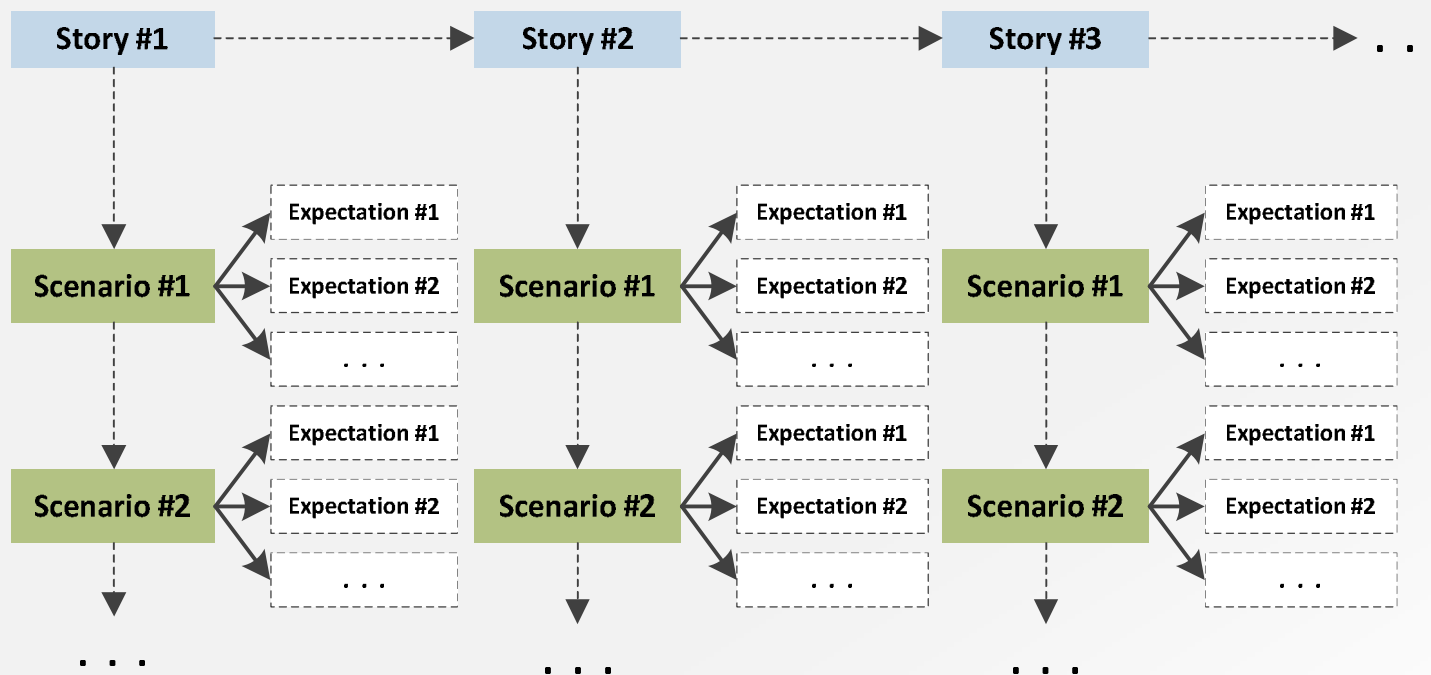
- An expectation is addressed side by side with its linked functional scope



Expectation

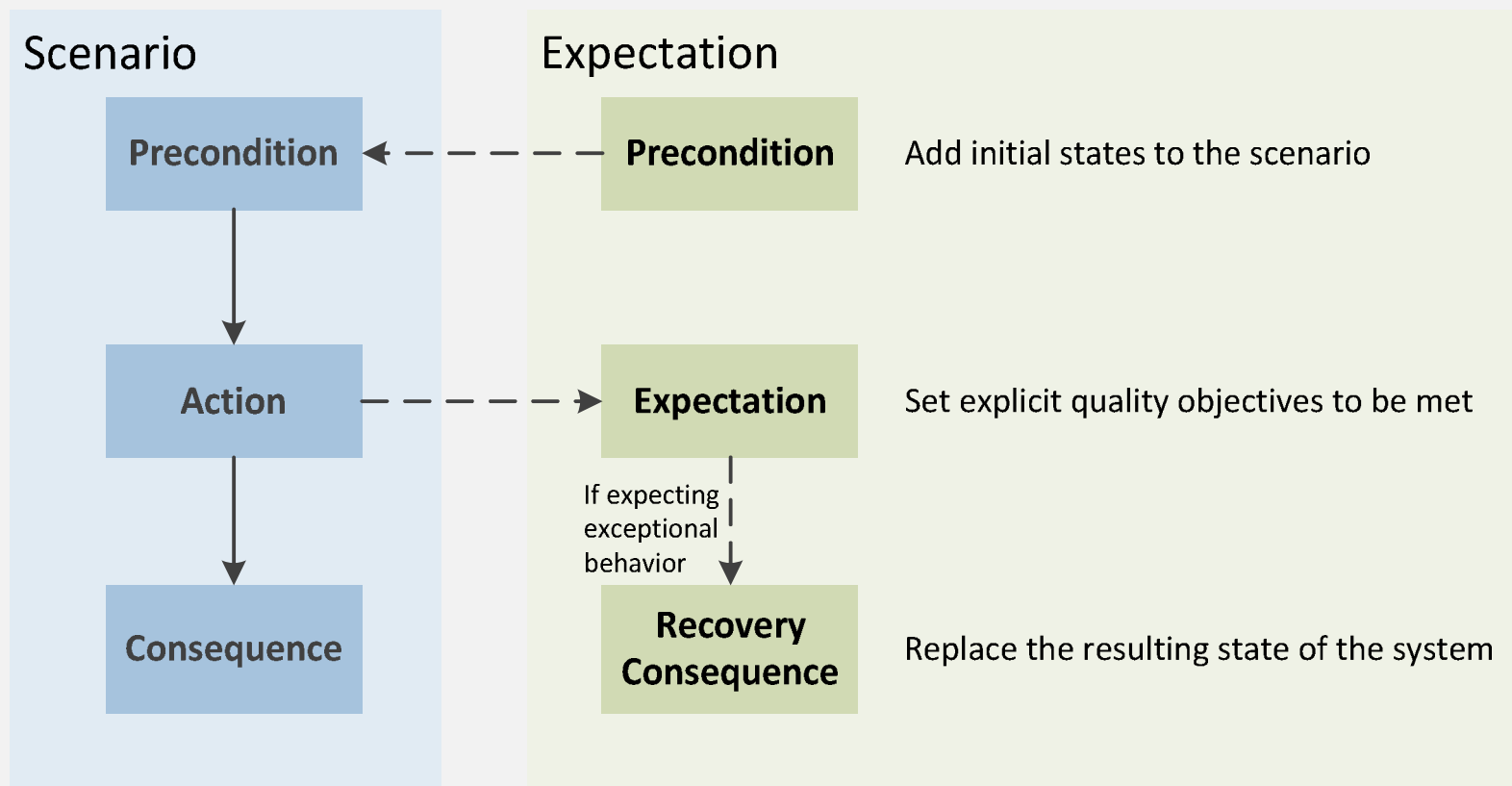
Reduce the functional scope to a scenario

- Linking expectations with scenarios is a processed repeated story after story



Expectation

Set Explicit Quality Objectives



Expectation

Set expectations with formality

Given one precondition

And another precondition

And yet another precondition

Expect a quality objective

Then a consequence

And another consequence

Expectation

Set expectations with formality

Scenario			
Given buyer is a student			
When buyer request the list of transit fares			
Then	Id	Name	Price
	002	Student Monthly Pass	76
	100	One Day Pass	6
	202	Student Booklet of	15
		10 Single tickets	15

Expect response time less than 5 seconds

Expectation

Set positive expectations (Happy path)

Scenario			
Given buyer is a student			
When buyer request the list of transit fares			
Then	Id	Name	Price
	002	Student Monthly Pass	76
	100	One Day Pass	6
	202	Student Booklet of	15
		10 Single tickets	15

Given buyer is logged in

Expect buyer to be authenticated positively

Expectation

Set negative expectations

Scenario			
Given buyer is a student			
When buyer request the list of transit fares			
Then	Id	Name	Price
	002	Student Monthly Pass	76
	100	One Day Pass	6
	202	Student Booklet of	15
		10 Single tickets	15

Given buyer is not logged in

Expect buyer to be authenticated negatively

Then event is saved in security database and user is redirected to “Login” page

Expectation

Omit implicit expectations

Scenario			
Given buyer is a student			
When buyer request the list of transit fares			
Then	Id	Name	Price
	002	Student Monthly Pass	76
	100	One Day Pass	6
	202	Student Booklet of	15
		10 Single tickets	15

Given the server is down

Expect the query to return 0 transit fare

Then user is redirected to “Server unavailable. Please try later” page

Expectation

Specific for one scenario

Set expectations with measurable quality objectives

Given 10 different users accomplished the scenario

Expect 8 users to complete the scenario with success

Expectation

Specific for one user story

“As a user, I want to log in so that I can do transaction”

Given 10,000 buyers are logged in

And new user is not logged in

Expect new user to be unable to complete the story

Then new user is redirected to “Server unavailable.
Please try later” page

External Quality

Test Expectations with Proven Practices

- **Accessibility:** Verify visual impairments, mobility difficulty, hearing inability and cognitive disabilities
- **Correctness:** Determine if the software respects the specification (Acceptance testing)
- **Performance:** Measure response time and inspect throughput
- **Reliability:** Seek for extraordinary resource consumption over a specified period of time (memory, CPU, disk space)

External Quality

Test Expectations with Proven Practices

- **Robustness:** Determine ability of the software to function correctly in the presence of invalid inputs or stressful environmental conditions
- **Scalability:** Verify software behavior under both normal and anticipated peak load conditions (Load testing)
- **Security:** Perform intrusion detection and vulnerability scanning
- **Usability:** Conduct heuristic evaluation, consistency inspection and activity analysis to verify if users achieve specified goals

External Quality

Less is more

- Negotiate with stakeholders to reduce number of expectations
 - Is it « really, really » a desirable outcome?
- Try to target a specific iteration for testing a non-functional requirement
 - Benefit: Transform from a recurrent concern to a one-time concern

Non-Functional Requirements

What about User Story?

- Cannot be satisfied in a finite period of time
 - The “what” that needs to be restricted is not concrete enough
 - The functional scope is fuzzy because it is an iteration
- Can easily induce technical debt
 - Once the story is completed, you must put it back in the backlog to make it available again for a future iteration
 - Complicates the management of the backlog unduly

Non-Functional Requirements

Two categories of constraint

- External quality
 - **Expectation** imposes conditions that sets a limit to comply during software execution
- Internal quality
 - **Practice** that ensure the software construction is done correctly

Internal Quality

What is it?

Non-Functional Requirement	Definition
Simplicity	Ease to understand or explain
Maintainability	Ease to change and evolve with minimal effort
Testability	Ease to confirm conformance by observing a reproducible behavior
Portability	Ease to reuse for multiple platforms
Extensibility	Ease to takes into consideration future changes

Internal Quality

Ensure quality using sound practices

- Practices define how the software construction is done
 - It preserves the sustainability of the source code for future developments

Internal Quality

Explicit Engineering Practices

Non-Functional Requirement	Practices (to be applied for each scenario)
Simplicity	Self-documenting code: Practices that ensure code is its own best documentation by allowing useful information, such as programming constructs and logical structure, to be deduced from the naming convention and code layout convention.

- The naming and code layout convention guide the team during software construction

Internal Quality

Confirm practice with collaborative construction

- Pair programming
 - Two teammates work together at one workstation
 - Driver
 - Type at the keyboard
 - Focus his attention on the task at hand
 - Use the observer as a safety net and guide
 - Observer
 - Look at what is produced by driver
 - Consider the constraints imposed by the practices
 - Offer ideas for improvements
 - The two teammates switch roles frequently

Internal Quality

Confirm practice with collaborative construction

- Peer review (aka formal inspection during construction)
 - Well-defined roles
 - Moderator, author, reviewers, scribe
 - Planning
 - Inspection is scheduled by moderator (according to predefined selection criteria)
 - Preparation
 - Reviewer works alone to scrutinize the work product under review
 - Reviewer uses a checklist to stimulate their examination
 - Inspection
 - Moderator chooses someone other than the author to present the work product
 - Author is a « fly on the wall » and scribe records reworks as they are detected
 - Constructive feedbacks, « I propose to replace with ... »,
 - After inspection
 - Moderator ensure that all rework is carried out promptly by the author

Internal Quality

Other examples of engineering practices

Non-Functional Requirements	Practices (to be applied for each scenario)
Maintainability	Continuous Integration: Practices of applying quality control for each new checked in code by verifying if it integrate with success in the development branch.
Testability	Red-Green-Refactor: Practice that promotes the notion of writing test first when programming a piece of code and that relies on the repetition of a very short development cycle divided into three stages (the red, the green and the refactor stage).
Portability	Multi-target compiling: Practices that verifies compilation on every platform.

- Each scenario is not « Done » until each practice is confirmed

Internal Quality

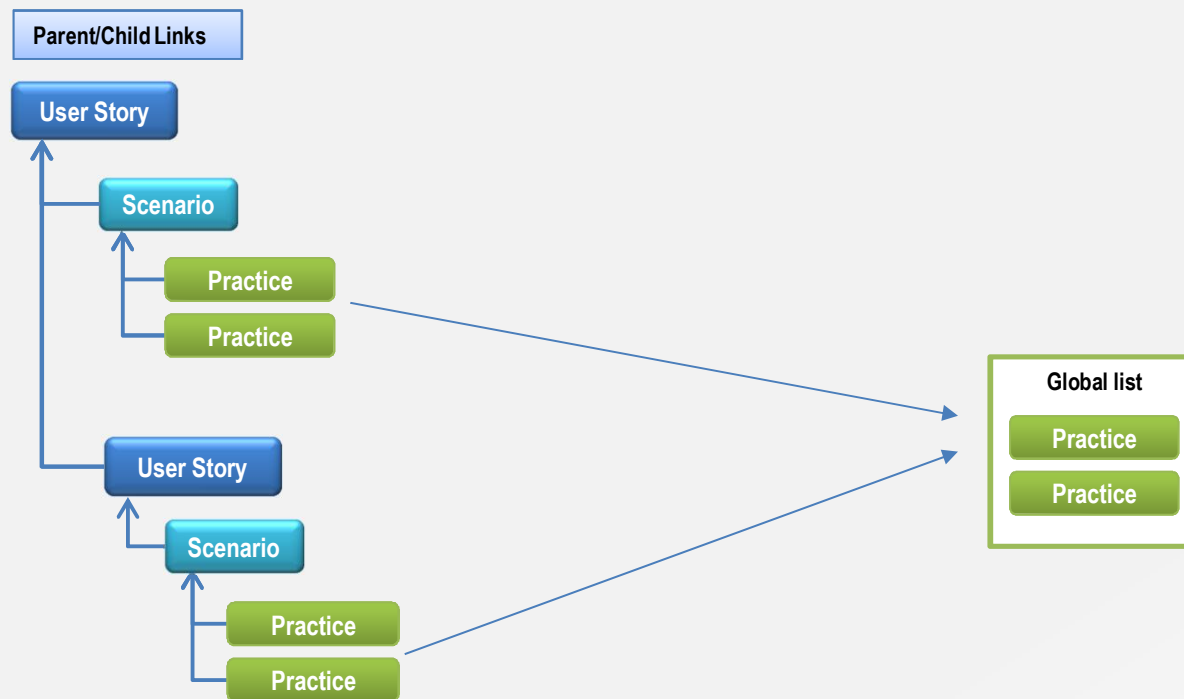
Engineering practices and user story

Non-Functional Requirements	Practices (to be applied for each user story)
Maintainability	Branching and merging : Practices to merge with the main branch (and tagged appropriately for traceability) source code from the development branch.
Portability	Multi-target deploying : Practices that verifies the automated build can deploy on every platform.

- Each user story is not « Done » until each practice is confirmed

Internal Quality

How to store practices description in TFS



Resources



- My website
 - <http://mariocardinal.com>
- Book
 - **Title:** Agile Specification
 - **Author:** Mario Cardinal
 - **Publisher:** Addison-Wesley
 - **Publication Date:** Spring 2012

Q & A