Addressing Non-Functional Requirements with Agile Practices

Mario Cardinal
Software Architect

Version: Oct 29th

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
- Your start getting a bad reputation when you spend too much time 'Doing it'

Source: agile101.net

Who Am I?

- Independent senior consultant specializing in software architecture
- www.mariocardinal.com

Montréal .NET Community

Microsoft® Visual Studio® 2010


Visual Studio Talk Show:0126:J...
http://www.visualstudio...
File Edit View Favorites Tools Help
Page Safety Tools
VISUAL STUDIO TALK SHOW
depuis 2004
Un podcast "en français" sur l'architecture logicielle
Suivez les entrevues de Mario Cardinal et Guy Barrette avec les experts de la programmation Microsoft .Net
Dernières émissions :::
Date
Titre
12 octobre 2010
Joé Quenneville
Développement avec Sharepoint 2010
28 septembre 2010
Joel Quimper
Windows Phone 7
8 septembre 2010
Mathieu Szablowski
Visual Studio Scrum 1.0
13 août 2010
Laurent Bossavit
L'Institut Agile et le futur de la communauté Agile
14 juillet 2010
Andre Vachon
Les nouveautés du langage C++ dans Visual Studio 2010
21 juin 2010
Dominic Sevigny
Silverlight dans la grande entreprise
7 juin 2010
JP Duplessis
Visualisation et analyse de code dans Visual Studio 2010 Ultimate
21 mai 2010
Joel Quimper
Quand et dans quel contexte est-ce que «adéquat» est «adéquat»?
7 mai 2010
Erik Renaud
La séparation des responsabilités entre les commandes et les requêtes
Simon Ferquel et Thomas Lebrun
Microsoft Surface
12 octobre 2010 (Émission #0126) :::
Joé Quenneville: Développement avec Sharepoint 2010
Nous discutons avec Joé Quenneville des nouveautés de SharePoint 2010 pour les développeurs. Entre autres, nous discutons de la programmation de "Visual webpart", du déploiement de package WSP et de l'utilisation de LINQ avec SharePoint.
Joé Quenneville est un consultant indépendant spécialisé en architecture de solutions d'affaires électroniques et tout particulièrement en développement de portails, d'intranets et de sites Web avec les technologies SharePoint et .NET. Il compte plus de onze années d'expérience en informatique dans des entreprises de services. Au cours des dernières années, il a assumé les rôles d'architecte et de développeur principal pour des portails SharePoint chez des clients tels que la Caisse de dépôt et placement du Québec, Groupe Pages Jaunes, Molson et la Banque Laurentienne du Canada.
Télécharger l'émission
Si vous désirez un accès direct au fichier audio en format MP3, nous vous invitons à télécharger le fichier en utilisant un des boutons ci-dessous.
MP3 PODCAST
Si vous désirez utiliser le feed RSS pour télécharger l'émission, nous vous invitons à vous abonner en utilisant le
Futures émissions :::
Découvrez le détail des prochaines émissions
Sylvie Trudel
Jean-Baptiste Evain
Francois Beauregard
Question?
Soyez notifié à la suite de la publication d'une nouvelle émission
RSS 2.0
Commandités :::
Communauté .NET Montréal
.NET EXPERTISE
Consultants experts en technologies Microsoft
www.dotnet-expertise.com
SCRUM + TFS = urban TURTLE
Podcasts francophones :::

Agenda

1. Non-functional requirements
   - Quality expectations
2. Functional requirements and agile processes
   - User Story and scenario
3. Non-functional requirements and agile processes
   - Improving quality during construction
   - Improving quality during execution

Non-Functional Requirements

What are they?

- Specify "how well" the "what" must behave
- Impose constraints that typically cut across functional requirements
  - Constraint to be obeyed either during the implementation by the builders (internal quality) or at run time by the software (external quality).

Non-Functional Requirements

It is all about quality

- Also known as "technical requirements", “quality attributes” or "quality of service requirements”
- Can be divided into two main categories:
  1. **Internal qualities** such as maintainability, modifiability and testability, which are barely visible by stakeholders but simplify how to build the software
  2. **External qualities** such as performance, correctness, security and usability, which carry out the software's functions at run time, and as such, are not only visible by stakeholders but also highly desirable

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
- I will explain how to obtain the desired quality by imposing constraints

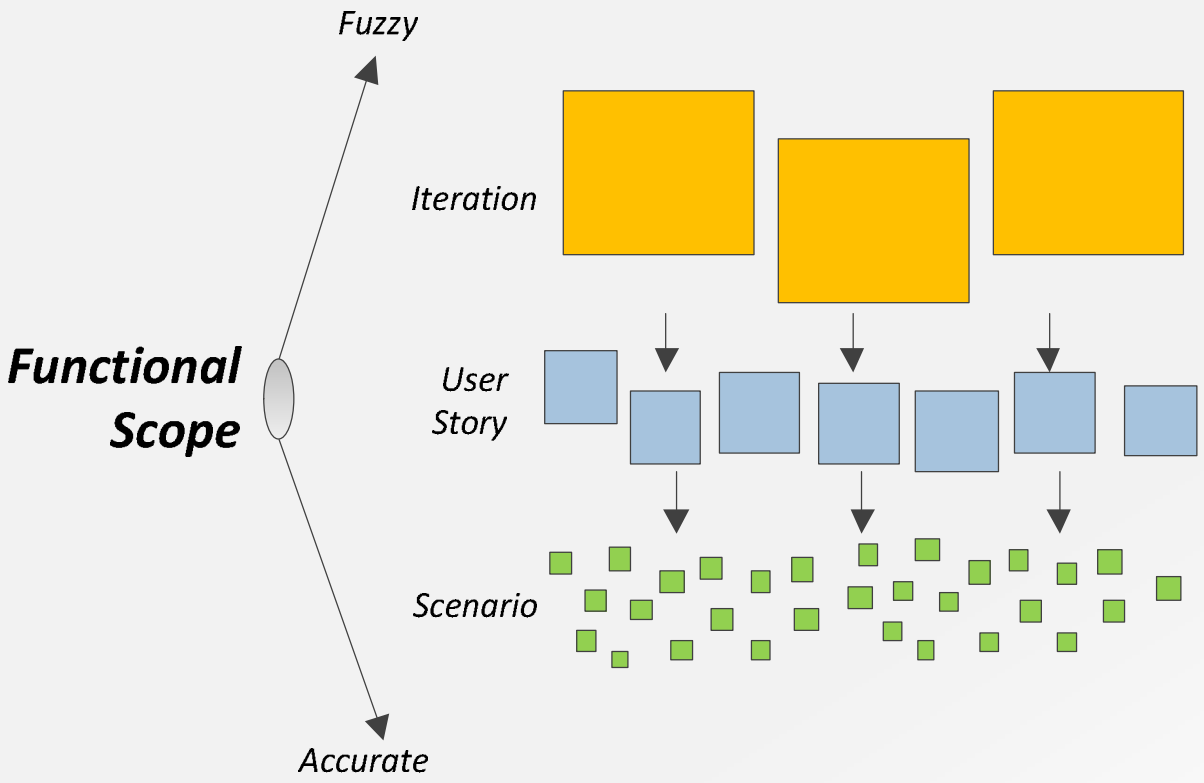
Non-Functional Requirements

Impose constraints to guide your work

- A constraint is a condition to make the requirements in line with quality expectations
- A constraint sets a limit to comply with
- It helps determine whether you have satisfied the non-functional requirements

Non-Functional Requirements

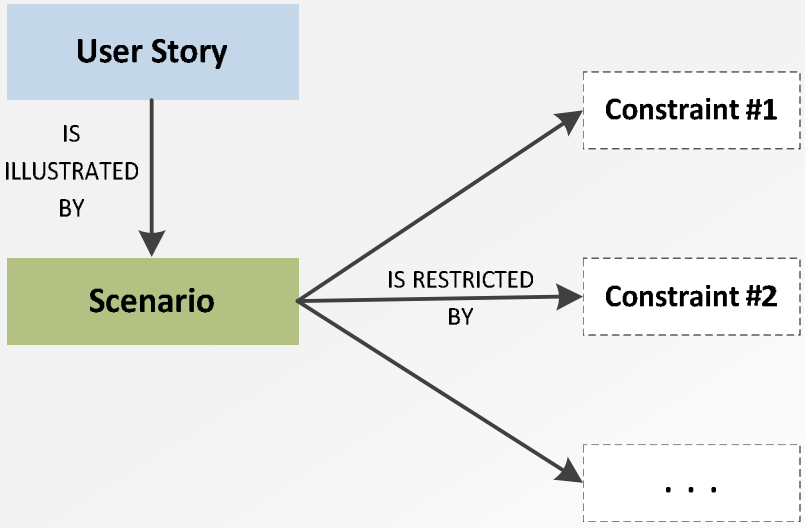
Constraints crosscut functional requirements



Non-Functional Requirements

Reduce the functional scope to a scenario

- A constraint should be satisfied in a finite period of time
- A constraint is addressed side by side with its linked functional scope



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 a stakeholder
- 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

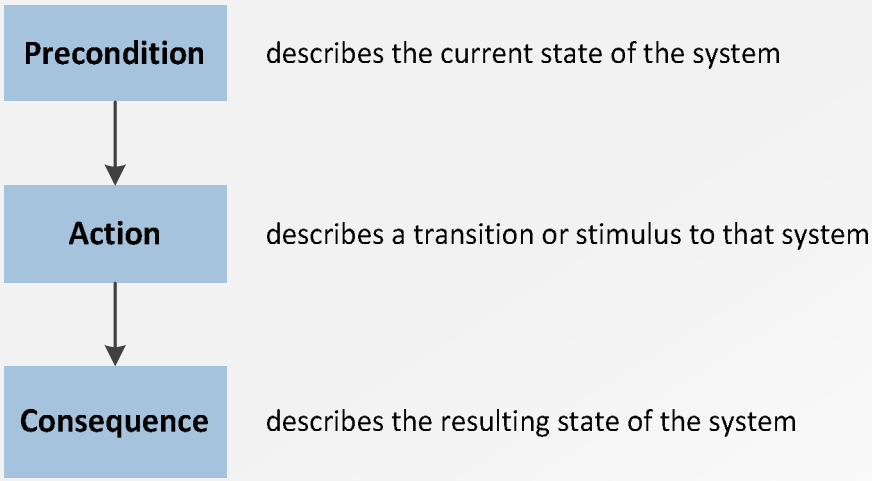
Confirm success criteria with scenarios

- While planning the iteration, the details of each story are confirmed with success criteria
  - Success criteria establish the conditions of acceptance
  - Success criteria are concrete examples
    - It says in the words of the stakeholders how they plan to verify the desirable outcome
  - Success criteria enables the team to know when they are done
- Express success criteria with scenarios

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

Express scenarios with formality

Given one precondition

And another precondition

And yet another precondition

When an action occurs

Then a consequence

And another consequence

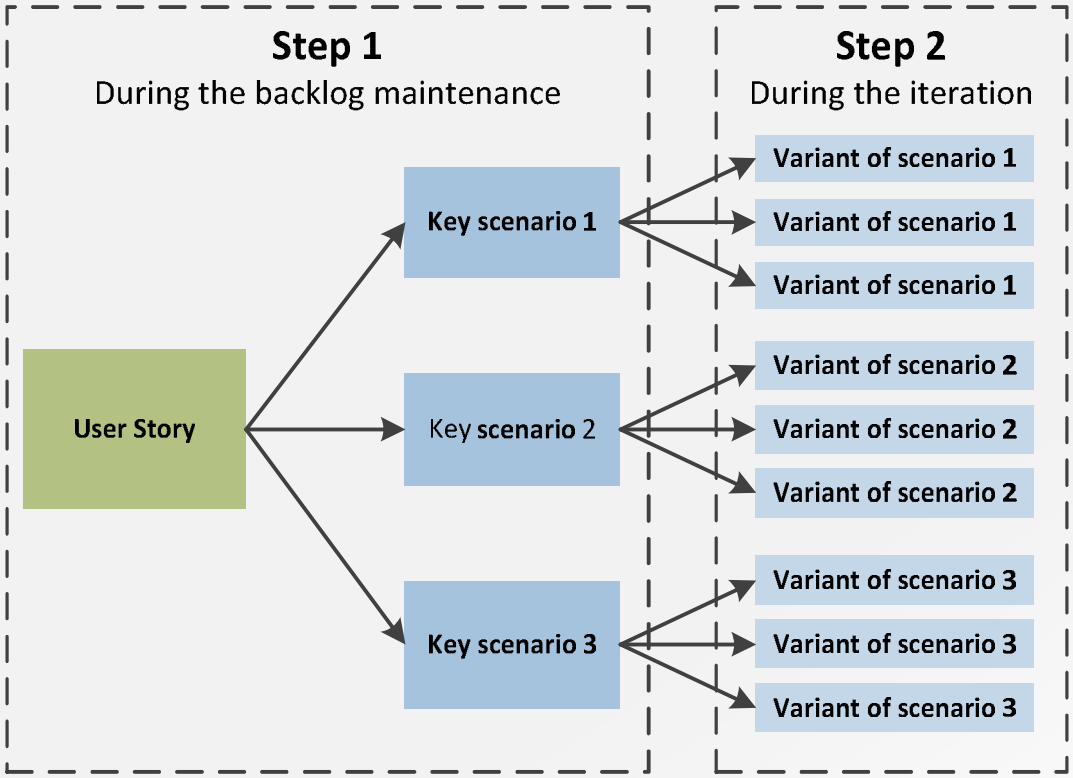
Functional Requirements

Express scenarios with formality

Given the shopping cart contains a monthly pass
When buyer checkout the shopping cart
Then the price is 146 dollars

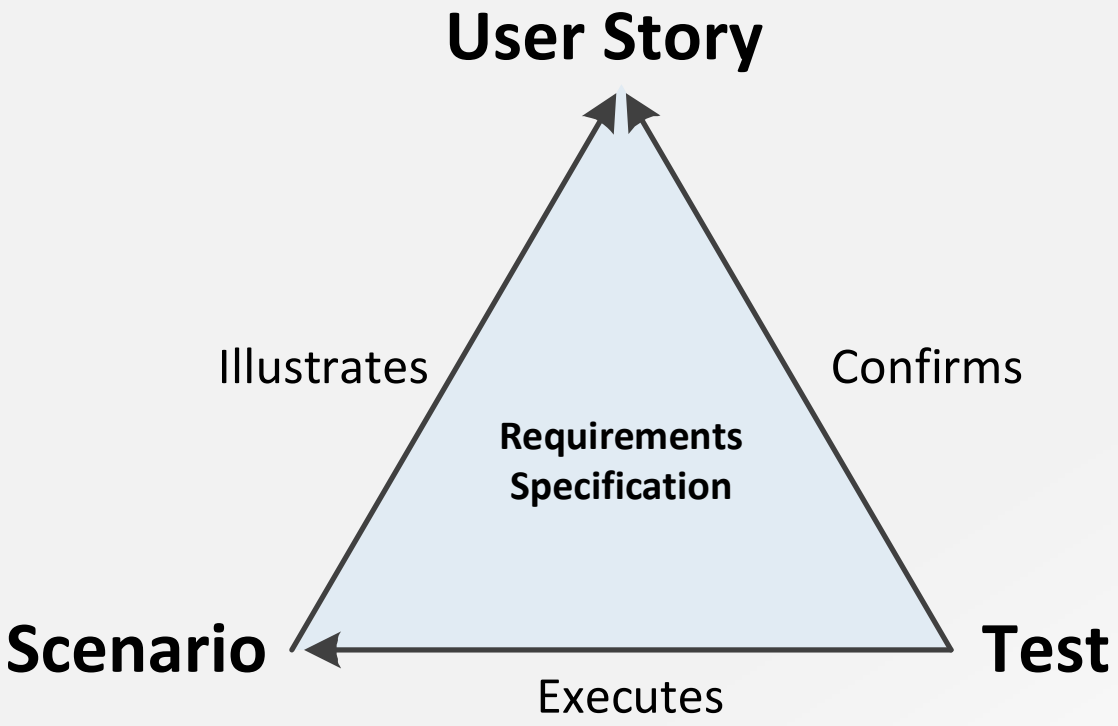
Functional Requirements

Illustrate collaboratively in a two-step process



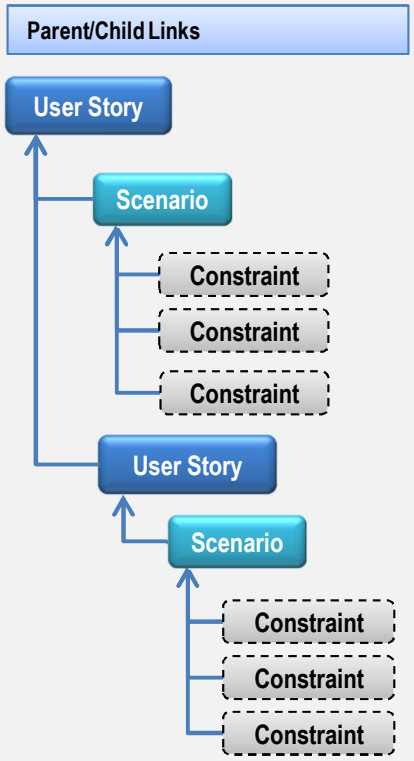
Functional Requirements

Automate confirmation with scenarios



Functional Requirements

Store requirements in a database



Functional Requirements

Use an Agile ALM platform

- Microsoft TFS, IBM Jazz, Jira, Rally, VersionOne, …
  - Work Items
    - User Stories, Tasks
    - Bugs, Test Cases
  - Version Control
    - Check-out/in
    - Label
    - Shelve
    - Branch/Merge
  - Automated Build
  - Reports & Metrics

Non-Functional Requirements

Should we express constraint with 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, the story belongs to the 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 types of constraint

- Internal quality
  - **Rule** is a “constraint” that sets a limit to comply during software construction
- External quality
  - **Restriction** is a “constraint” that sets a limit to comply during software execution

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 growth

Internal Quality

Impose constraints using “Rules”

- Rule is a constraint that sets how software construction is built
- Rules are global and applies to each scenario

Internal Quality

Set rules with explicit quality objectives

Non-Functional Requirement	Rule
Simplicity	**Naming convention**: Practices that ensure code is its own best documentation by allowing useful information, such as programming constructs, to be deduced from the names.

- The « naming convention » guide the team during software construction

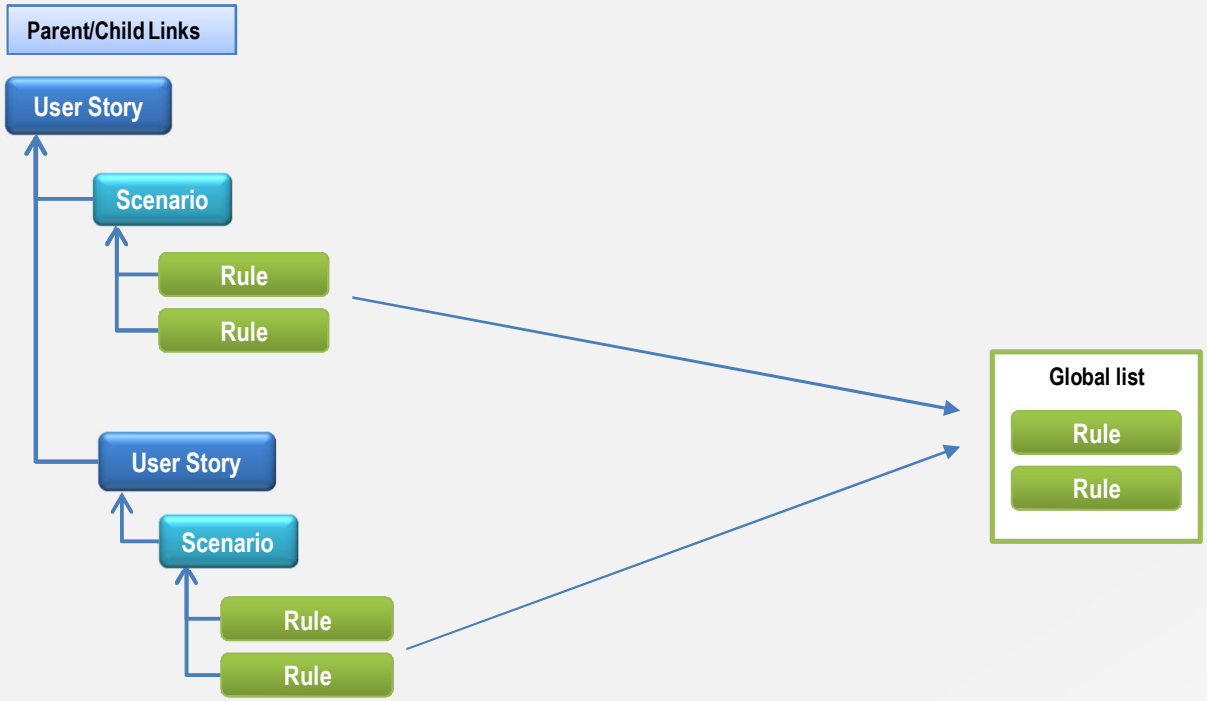
Internal Quality

Iteration 0: Produce a “Naming convention”

Element	Case	Sample	More information
Class name	Pascal	ClassName	Class names should express the abstraction the class is implementing.
Interface name	Pascal	IInterfaceName	Interface names should express the contract the interface is implementing. To make explicit that the contracts are interfaces, the names are prefixed with a “ I ”.
Method names	Pascal	MethodName	Method names should express the unit of functional cohesion the routine is implementing.
Member variables	Camel	memberVariable	Variable names should express whatever the variable represents. Inside a method, always use the instance ‘this.’ in front of the member variable to make explicit the class membership of the variable.
Parameter variables	Camel	parameterVariable	Variable names should express whatever the variable represents. If a parameter and a member variable express the same concept use the same name for both variables.
Local variables	Camel	localVariables	Variable names should express whatever the variable represents.

Internal Quality

Store rules in your Agile ALM platform



Internal Quality

Confirm rules 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 rules
      - Come up with ideas for improvements
  - The two teammates switch roles frequently

Internal Quality

Confirm rules with collaborative construction

- Peer review (aka formal inspection during construction)
  - Well-defined roles
    - Moderator, author, reviewers, scribe
  - Planning
    - Inspection is scheduled by moderator (from code analysis measure)
  - Preparation
    - Reviewer works alone to scrutinize the work product under review
    - Reviewer uses the rule definition 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 would replace with … », Emphasis is on improving knowledge
  - After inspection
    - Moderator is responsible for seeing that all rework is carried out promptly by the author

Internal Quality

Other examples of explicit quality objectives

Non-Functional Requirements	Rule (Global for each scenario)
Simplicity	**Code layout convention**: Practices that ensure code is its own best documentation by using code layout that shows the logical structure.
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 the integrated code compile on every platform.

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

Internal Quality
Rules and User Story

Non-Functional Requirements	Rule (Global 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 rule is confirmed

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 specified goals.

External Quality

Differences with internal quality rules

- The constraints set a limit to comply during software execution
- The constraint is a restriction
  - It is not a rule

External Quality

Differences with internal quality rules

- Restriction is specific for one scenario
- Restriction has a measurable quality objective
- Restriction has recovery action(s)
- Restriction is confirmed by test(s)

External Quality

Restrictions 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

External Quality
Specific for one scenario

Scenario

Given buyer is logged has 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 10 Single tickets	15
		15

Restriction (Performance)

Measure: Response time must be smaller than 10 seconds
Recovery: Log issue and notify the user that the query cannot be completed

Restriction (Security)

Measure: User must be authenticated
Recovery: Log issue and redirect to login page

- Set restrictions with measurable quality objectives

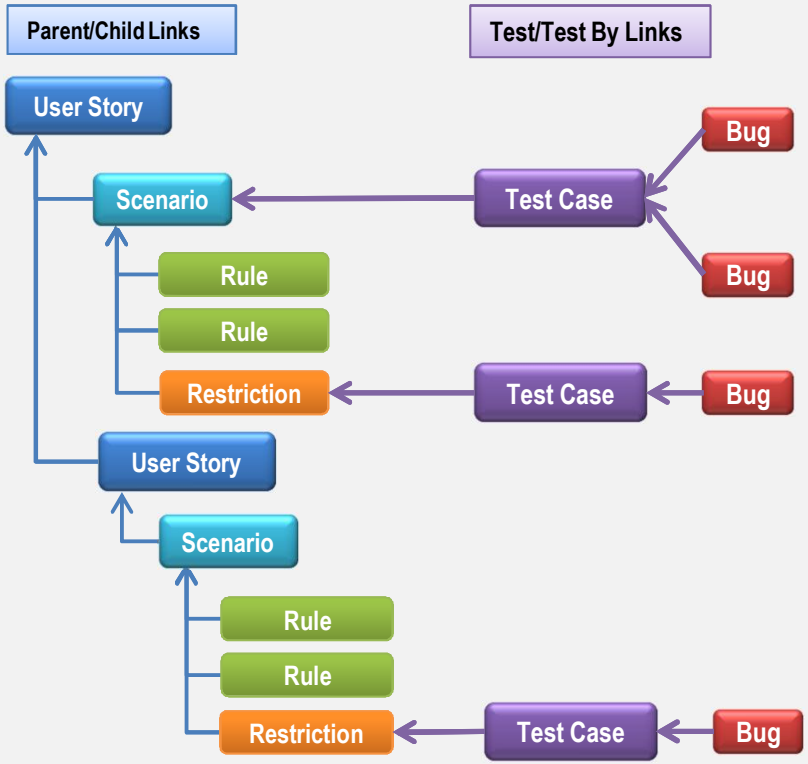
External Quality

Confirm restrictions with tests

- **Correctness**: Acceptance testing
- **Performance**: Response time or throughput testing
- **Reliability**: Testing over a period of time (memory leaks ???)
- **Robustness**: Simulate broken component
- **Scalability**: Load testing (with growing amounts of work)
- **Security**: Intrusion testing
- **Usability**: Testing with real users

External Quality

Confirm restrictions with tests



External Quality

Less is more, testing restriction is costly

- Negotiate with stakeholders to reduce number of restrictions
  - 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
  - Benefit: Handle the concern with a user story

Next Steps

- Add ‘Scenario’, ‘Rule’ and ‘Restriction’ Work Item Type to your Agile ALM platform
- Improve internal quality using rules
  - Create global rules for scenario and user story
  - Enforce rules with collaborative construction
- Improve external quality using restrictions
  - Create specific restrictions for each scenario
  - Enforce restrictions with tests

Resources

Book

- **Title**: Agile Specification
- **Author**: Mario Cardinal
- **Publisher**: Addison-Wesley
- **Publication Date**: Spring 2012

Shameless plug

Training

- Adopting Agile with Microsoft Tooling
  - Learn how to customize your Team Foundation Server to effectively adopt an agile process
  - Learn how to efficiently mix Scrum, Kanban and executable specification
- **Ottawa**: Nov 2nd
- **Quebec**: December 14th (en francais)
- http://mariocardinal.com

Q & A