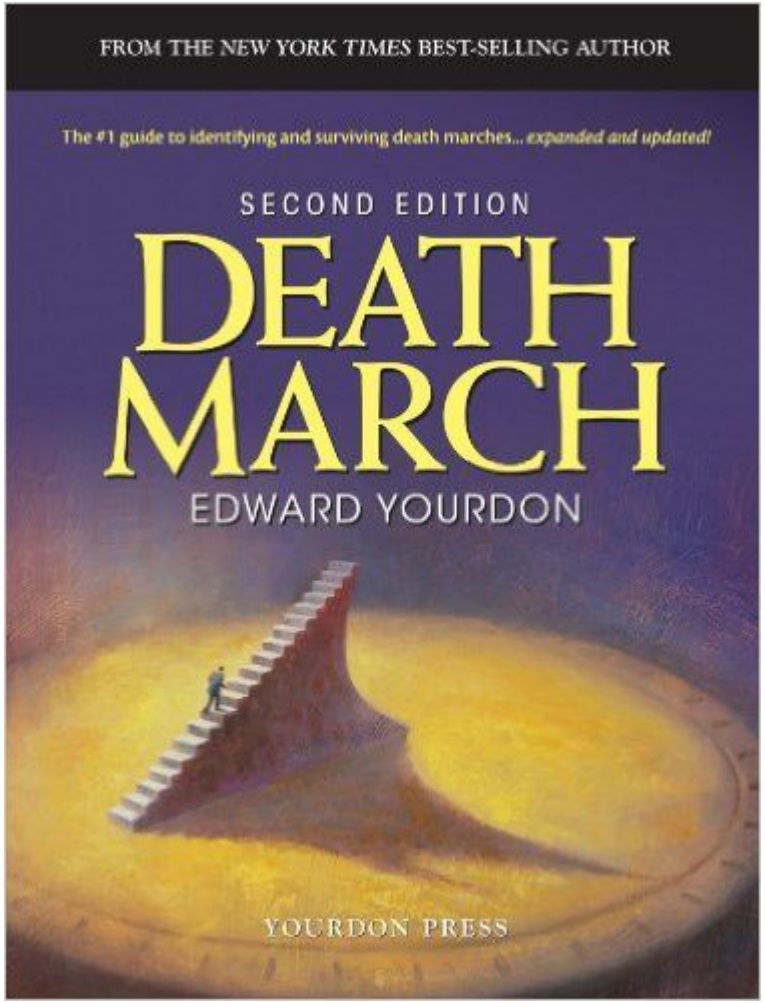
LOG8371 : Ingénierie de la qualité en logiciel

Qualité du Logiciel Agile

Hiver 2017

Fabio Petrillo
Chargé de Cours

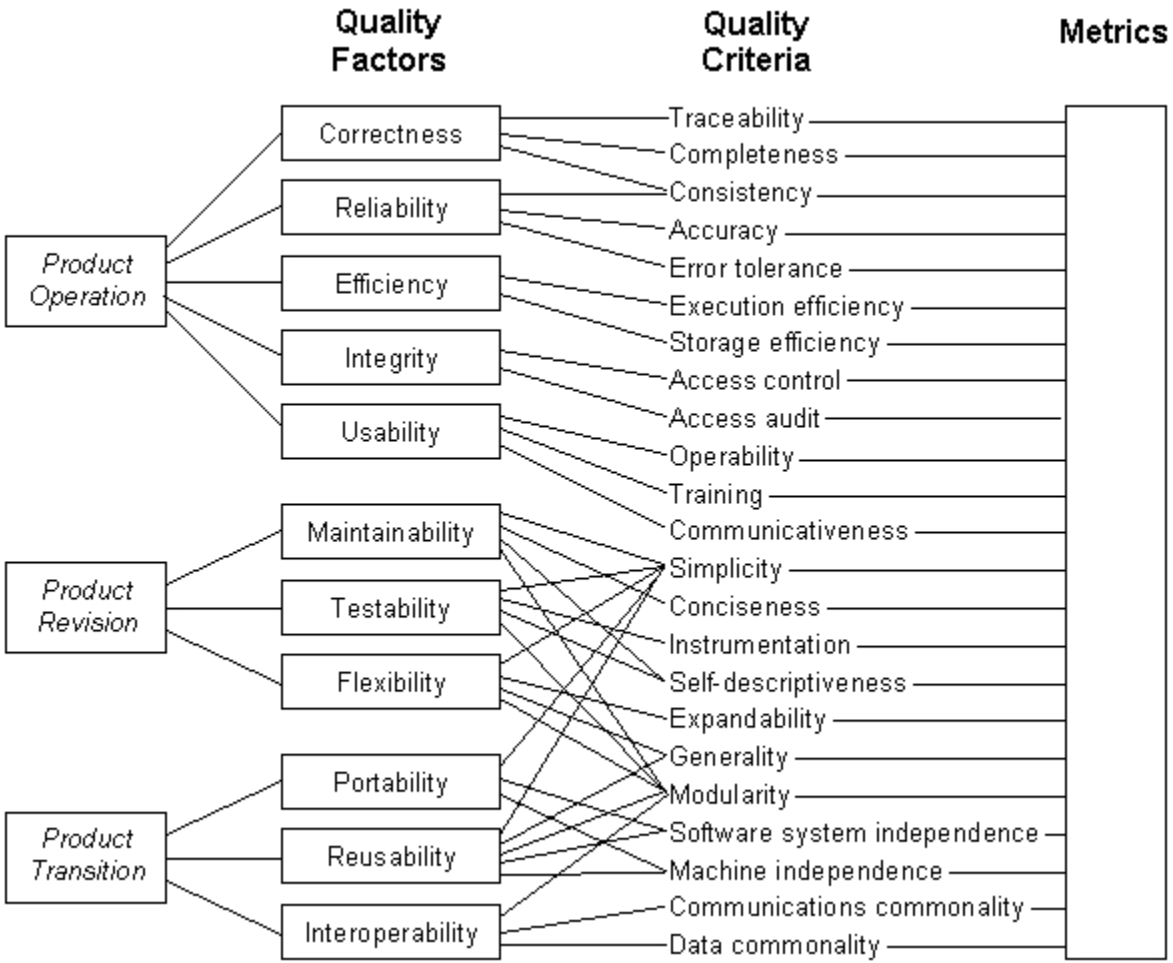
POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE


FROM THE NEW YORK TIMES BEST-SELLING AUTHOR
The #1 guide to identifying and surviving death marches... expanded and updated!
SECOND EDITION
DEATH MARCH
EDWARD YOURDON
YOURDON PRESS

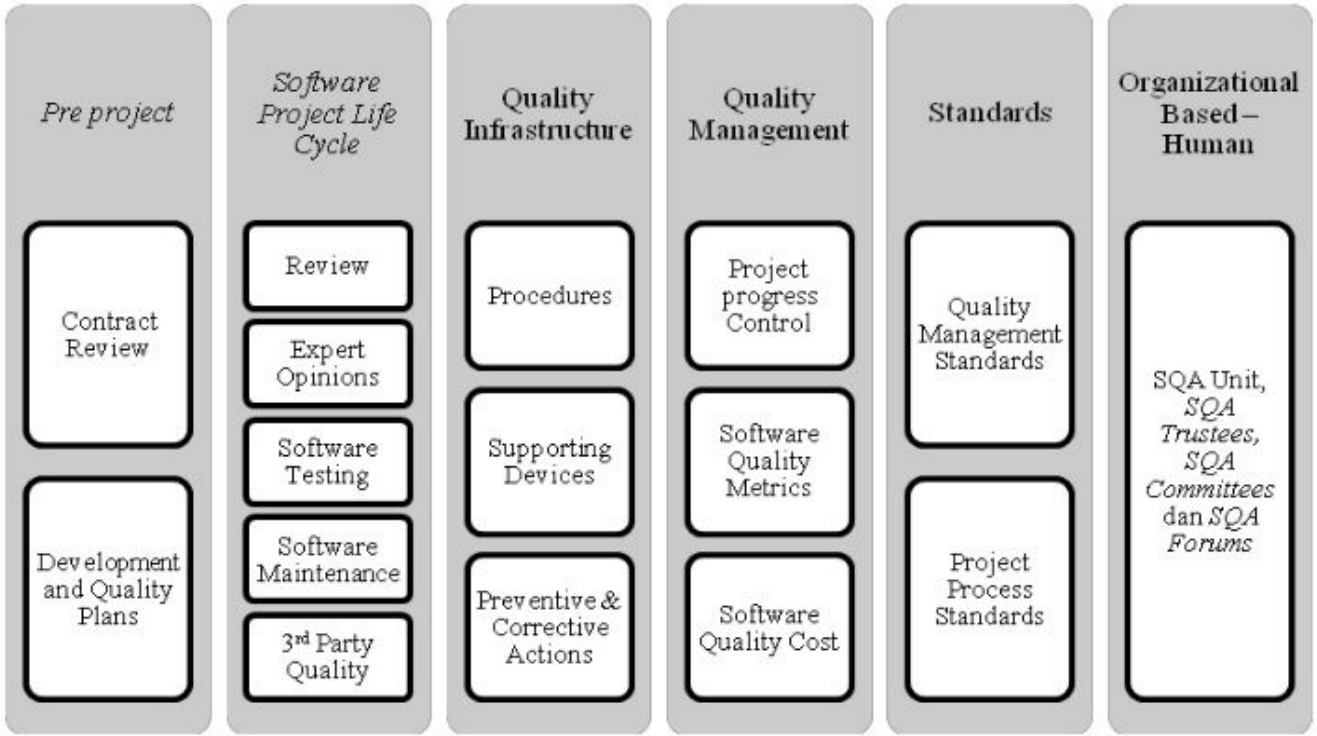
Il n'y a **aucun moyen de sauver** les projets de Death March!

Quittez aussi vite que possible, si c'est possible.

Classic Software Quality Assurance


Quality Factors
Quality Criteria
Metrics
Product Operation
Product Revision
Product Transition
Correctness
Reliability
Efficiency
Integrity
Usability
Maintainability
Testability
Flexibility
Portability
Reusability
Interoperability
Traceability
Completeness
Consistency
Accuracy
Error tolerance
Execution efficiency
Storage efficiency
Access control
Access audit
Operability
Training
Communicativeness
Simplicity
Conciseness
Instrumentation
Self-descriptiveness
Expandability
Generality
Modularity
Software system independence
Machine independence
Communications commonality
Data commonality

DANIEL GALIN'S QUALITY SHRINE



Alors, quel est le problème principal de l'approche **“classic”** Software Quality Assurance ?

Alors, quel est le problème principal de l'approche **“classic”** Software Quality Assurance ?

Pour utiliser la **même approche** pour tout les projet informatique !!!

Le logiciel est un problème monstrueux

*“"Un **problème monstrueux** est un problème **difficile** ou **impossible** à résoudre en raison de **besoins incomplètes, contradictoires et changeants** qui sont souvent difficiles à reconnaître." [Wikipedia, 2016]*

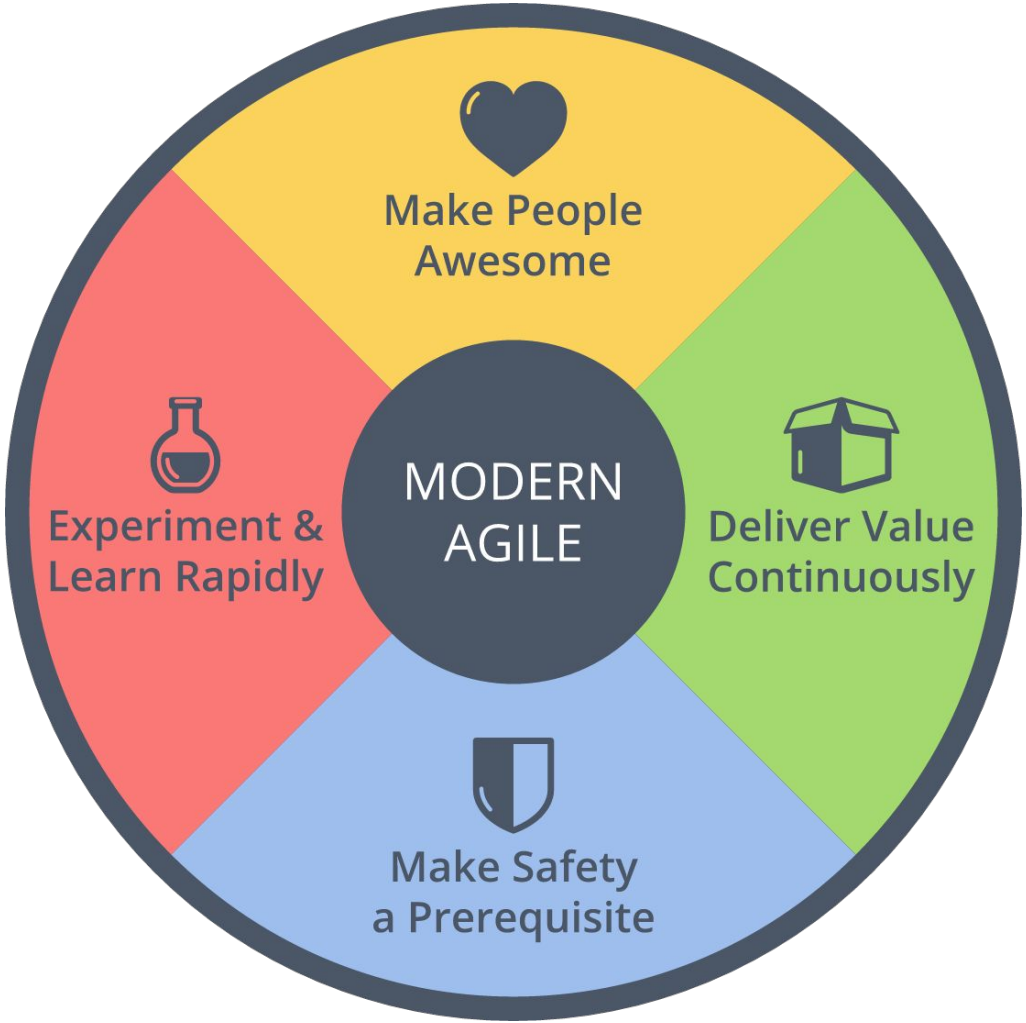
Interdépendances complexes !

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.


Make People
Awesome
Deliver Value
Continuously
Make Safety
a Prerequisite
Experiment &
Learn Rapidly
MODERN
AGILE

Développement du Logiciel Agile

- Offrir fréquemment de la valeur aux clients / utilisateurs
  - Satisfaire le client grâce à la livraison rapide et continue de logiciels précieux
  - Le logiciel de travail est la principale mesure des progrès
- Rythme soutenable
- Rétroaction constante
- Accueillir les changements
  - Monde incertain
  - Modernité liquide - Zygmunt Bauman
  - La fin de la certitude - Ilya Prigogine

Agile (ou Agility) est une approche évolutive!

Développement du Logiciel Agile

- Construire des projets autour de personnes motivées

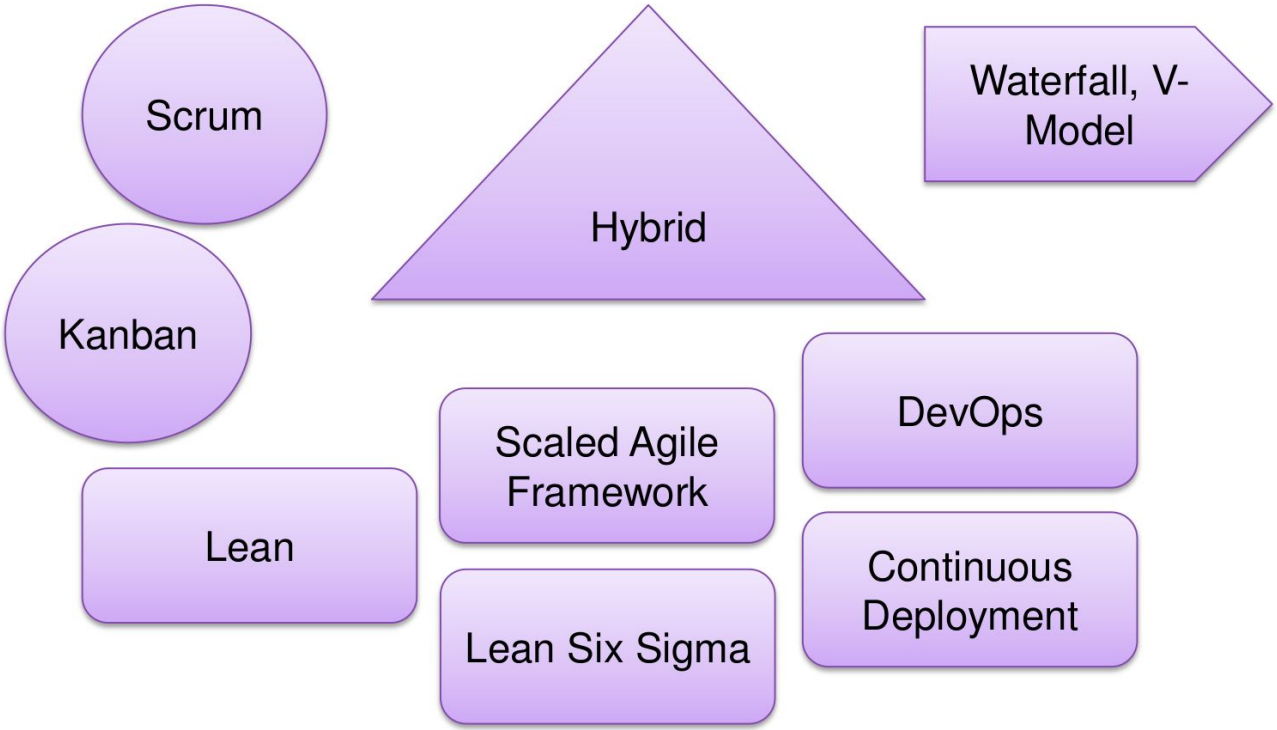
Développement du Logiciel Agile

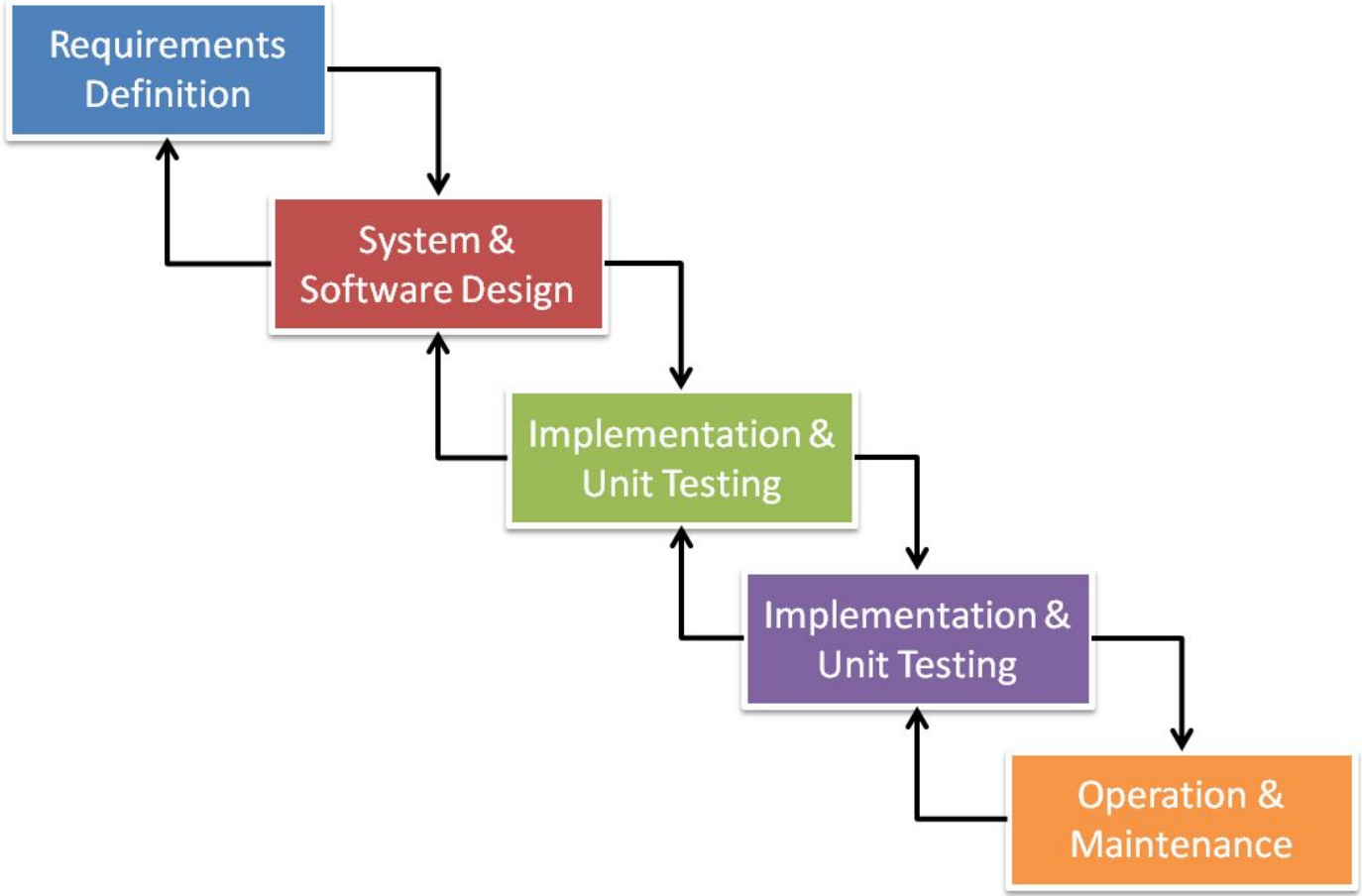
- **Construire des projets autour de personnes motivées ??**
  - **POURQUOI? Est-ce l'ingénierie?**

Que signifie être agile?

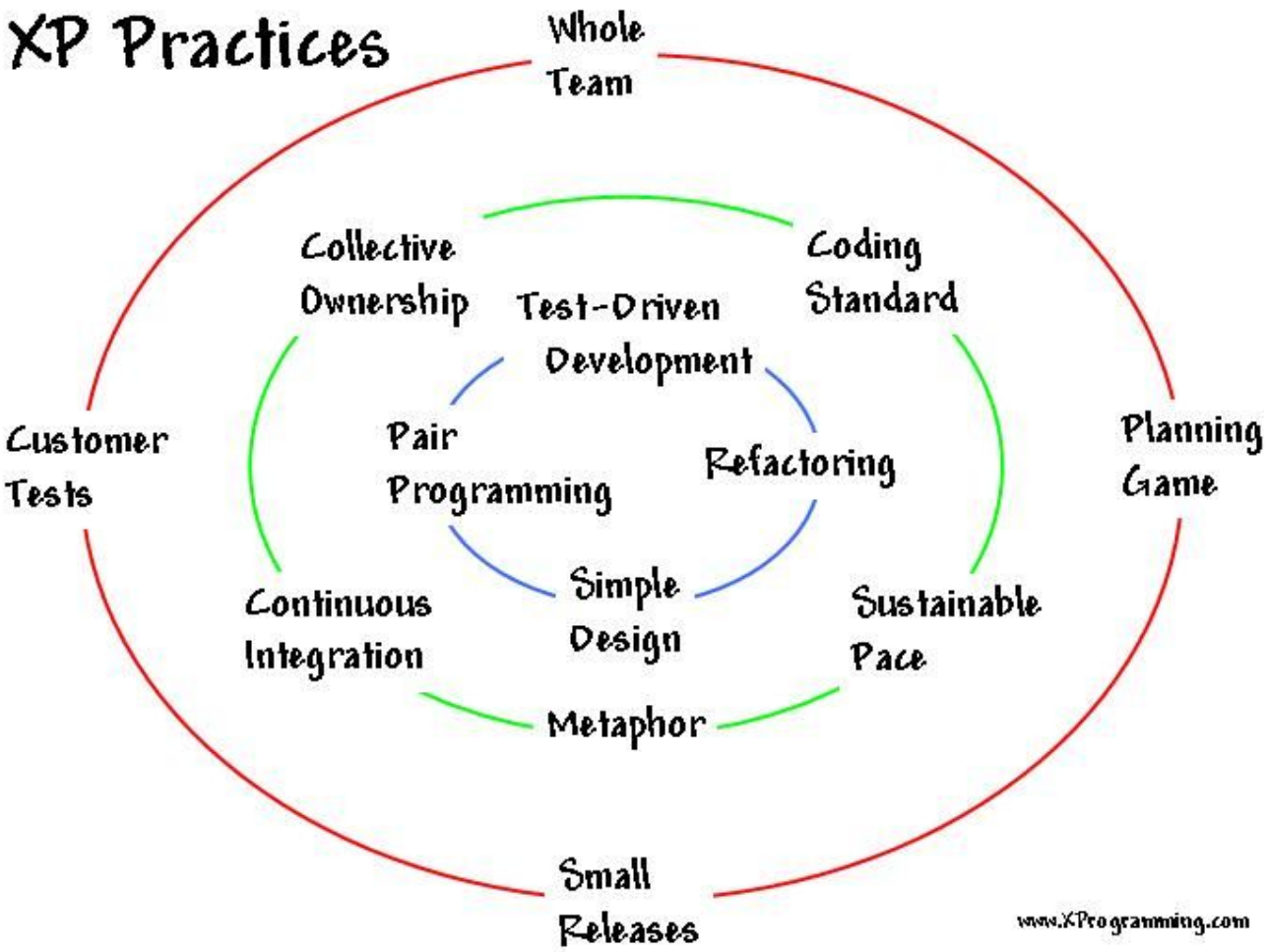
- **Construire des projets autour de personnes motivées ???**
  - **POURQUOI? Est-ce l'ingénierie?**
  - **Parce que le logiciel est un problème monstrueux!**
  - **Un processus régulier et «bien défini» ne suffit pas!**
- **Avertissement:** facile à parler, Difficile à mettre en œuvre

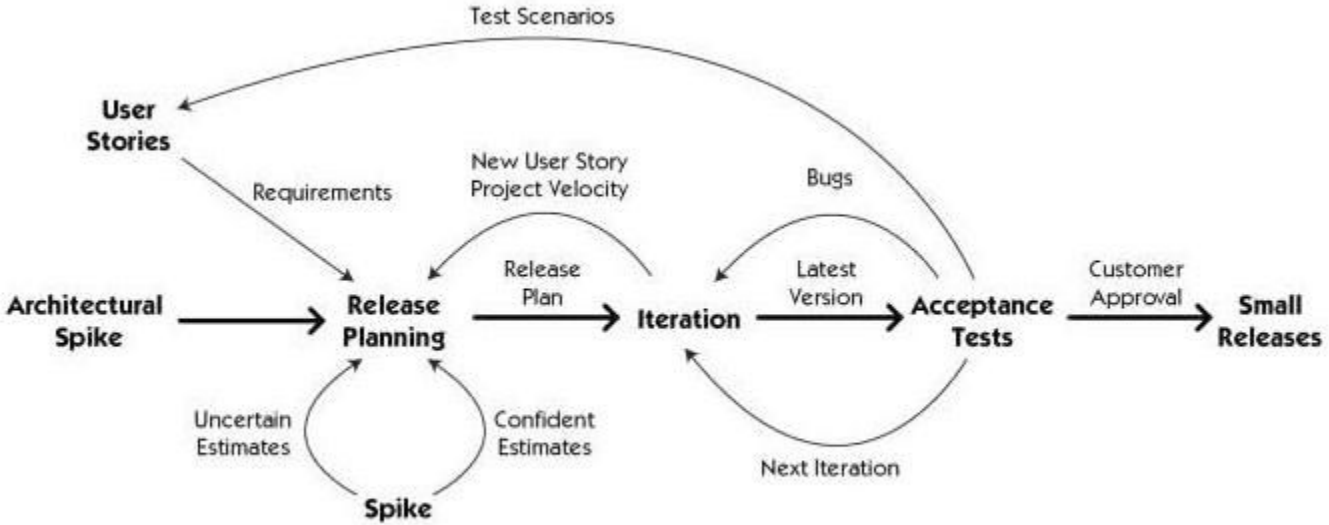
Agile développe ce qui convient





Waterfall Software Process Model


XP Practices
Whole Team
Collective Ownership
Coding Standard
Test-Driven Development
Pair Programming
Refactoring
Customer Tests
Planning Game
Continuous Integration
Simple Design
Sustainable Pace
Metaphor
Small Releases
www.XProgramming.com

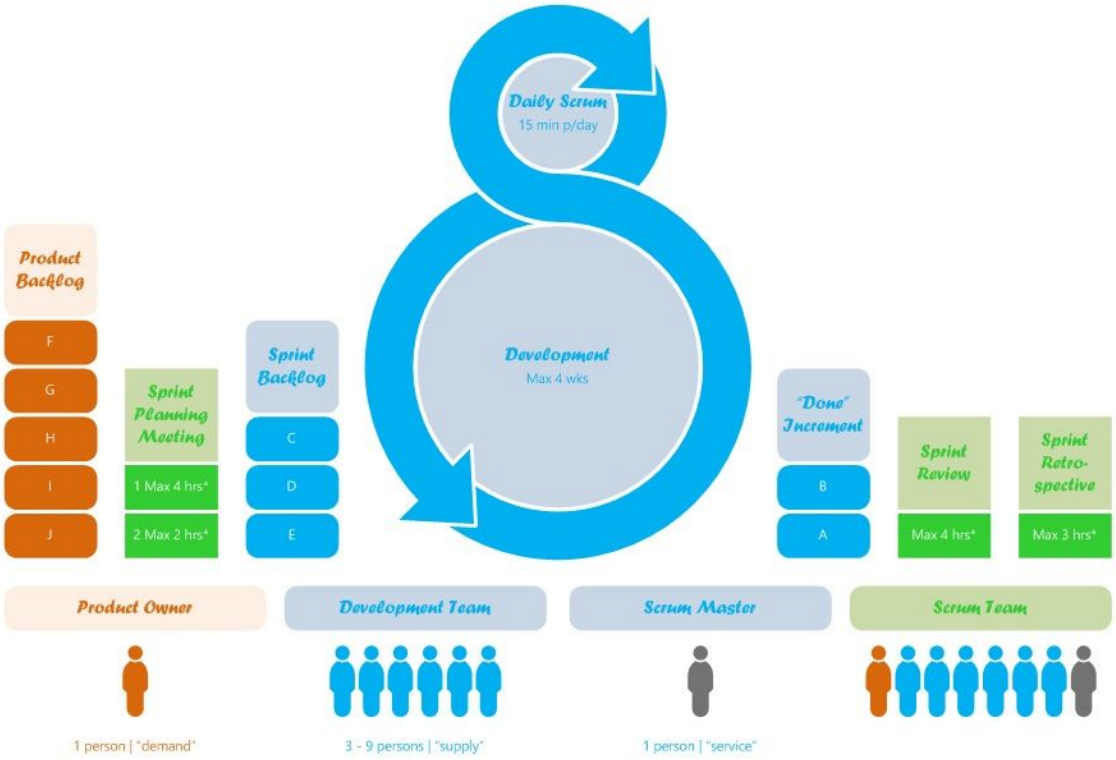


A visual process model for XP

Le logiciel qui travail est la principale mesure du progrès

- Histoires d'utilisateurs traduites en fonctionnalités
- Fonctionnalités livrées après chaque itération

Scrum Overview



* Duration of this event depends on the duration of the Sprint


Backlog
W.I.P
Done
Unprepared
For Kick Off
Story
To Do
In Progress
Out
Sprint:


Backlog
W.I.P
Done
Unprepared
For Kick Off
Story
To Do
In Progress
Out
Sprint:

When a Product Backlog item or an Increment is described as “Done”, everyone must understand what “Done” means.

- Scrum Guide

Définition de Terminé “Done”

Deux aspects de la définition de “Terminé“

- Critères d'achèvement
- Critères d'acceptation

- https://www.agilealliance.org/definition-done-user-stories/
- https://www.agilealliance.org/glossary/definition-of-done/

User Story Done Criteria

Completion Criteria

- "Code Complete" (definition TBD)
- "Unit Tested" (definition TBD)
- "Peer Reviewed" (definition TBD)
- "QA Complete" (definition TBD)
- Documented (As needed; determined by the **Scrum Team*** through tasking at beginning of Sprint)

The **Scrum Team** determines when the Completion Criteria have been met with the guidance of the **Scrum Master***.

Acceptance Criteria

- The list of expectations for a specific User Story as defined by the **Product Owner*** prior to the beginning of a Sprint
- The Product Owner may define these alone or with the help of the Scrum Team and/or Scrum Master
- For cases where Acceptance Criteria are not clear, a Spike User Story will be used to define the problem and Acceptance Criteria for a User Story to be completed in a future Sprint
- The Scrum Team must agree to these Acceptance Criteria at the Sprint Planning meeting
- Minor changes to the Acceptance Criteria once the Sprint is underway as long as there is formal agreement between the Scrum Team, Scrum Master, and Product Owner
- When the Scrum Team believes these Acceptance Criteria have been met, the User Story is ready for a Product Owner review (Demo), which occurs throughout the Sprint
- The Review (Demo) of each User Story should NOT be left until the very end of the Sprint

The **Product Owner** determines when these Acceptance Criteria have been met. At that point, the User Story is considered "accepted"; i.e. "Done"

*These are the three core roles prescribed by Scrum: Product Owner, Scrum Master, and Scrum Team. Together, the three roles make up the Project/Product Team.

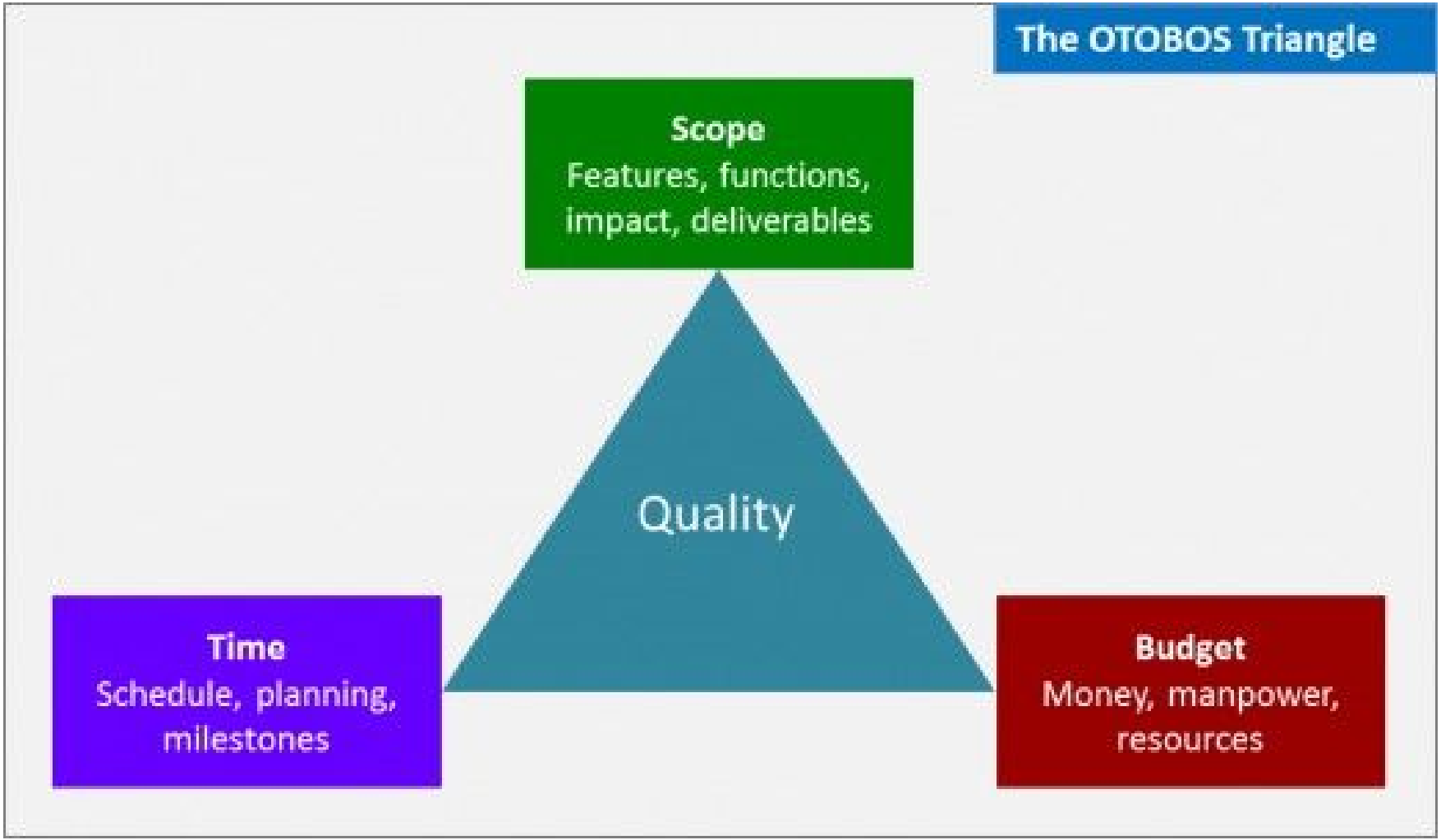
Perspective Agile de l'assurance de la qualité

McBreen (2003) définit l'assurance de la qualité agile comme le développement d'un logiciel capable de **réagir au changement**, car le client le demande de changer. Cela implique que la **livraison fréquente** de logiciels testés, **fonctionnels et approuvés par le client** à la fin de chaque itération.

Ambler (2005) considère la qualité agile comme le résultat de pratiques telles que le **travail collaboratif** efficace, le développement **progressif** et **itératif** mis en œuvre par des techniques telles que le **refactoring, le développement piloté par des tests, la modélisation** et des techniques de **communication** efficaces.

La qualité est-elle une variable dans un contexte Agile?

The OTOBOS Triangle

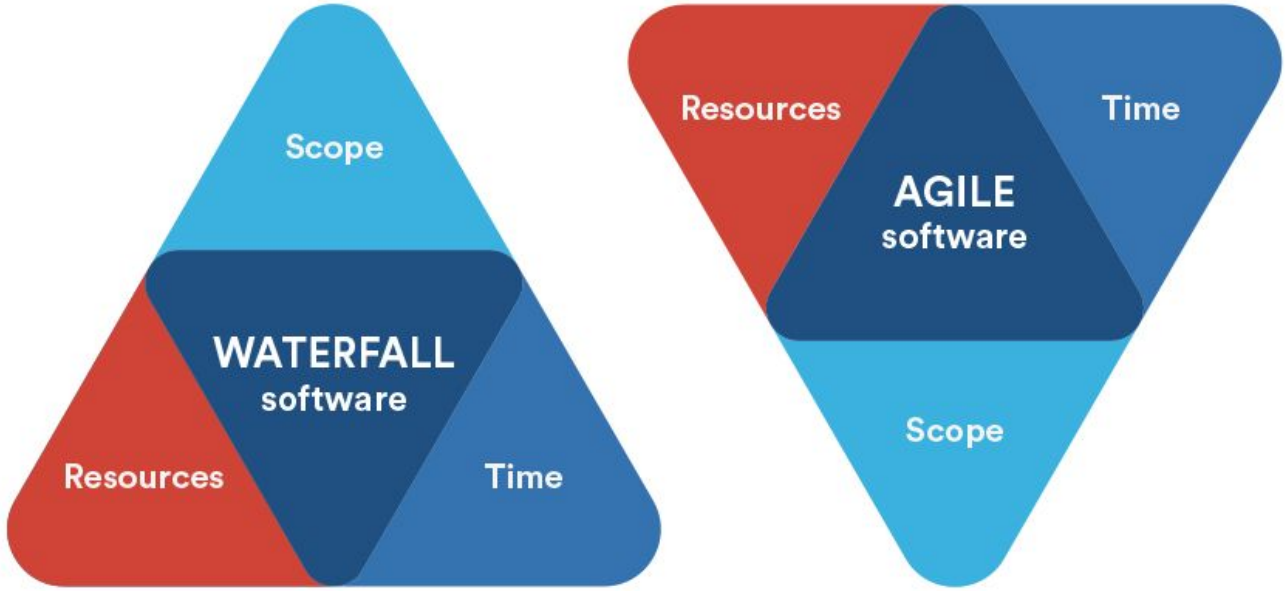


La qualité est-elle une variable?

La qualité n'est pas une variable dans le contexte Agile!

La portée, le temps et les ressources sont des variables.

Fixed



Estimated

Questions QA Agile

- QA fait-elle partie de l'équipe de développement?
- Pouvons-nous intégrer QA dans la même itération que le développement?
- Qui fait QA?
- QA coûte-t-il plus cher dans Agile que le produit semble passer du sprint au sprint?
- Comment pouvons-nous évaluer le QA Agile?
- Avons-nous besoin de «plan de test»?
- Qui définit les cas de test?
- Les tests d'acceptation des histoires sont-ils suffisants?
- Quand savons-nous que les tests sont effectués?
- Avons-nous besoin de suivre les bugs?

Test Agile

- Les tests sont essentiels!
- S'adapte aux changements très rapides
- Non planifié, mais vit avec le projet
- Les tests sont effectués dès le début du projet, pas seulement à la fin de celui-ci
- Met l'accent sur l'équipe, l'humanité / l'humain-centré et la coopération
- Tests itératifs pièce par pièce
- Le client est impliqué même dans toutes les phases de test
- Intégration continue
- Automatisation des essais au moins dans les essais unitaires

Il est plus important de trouver des bugs que d'écrire des documents complets

Réinventer les tests pour Agile

- Les testeurs restent essentiels à la qualité
- Un long délai avant l'essai actif n'est pas réalisable
- Comment concilier la planification des tests avec l'exécution initiale?

Intégration des tests dans le développement agile

- Il n'y a pas une seule prescription
- Les méthodologies agiles favorisent les cycles de libération courts
- Le modèle de test classique ne fonctionne pas sur un contexte d'intégration continue
- Une nouvelle approche pour tester
- Les testeurs doivent se réinventer et leur métier

Intégration des tests dans le développement agile

- Itérer sur la planification
  - Respecter le développement
- Itérer sur les tests
  - Gérer le risque afin que les caractéristiques et les actions les plus importantes soient bien testées
- Itérer sur la qualité
  - Mesurer votre qualité
  - Améliorer dans la prochaine itération
- Participation précoce des testeurs
  - Les testeurs doivent travailler directement avec les propriétaires de produits
  - Doit comprendre les besoins des utilisateurs
  - Obtenir une compréhension approfondie des histoires d'utilisateurs

Intégration des tests dans le développement agile

- Focaliser sur les fonctionnalités qui font la différence pour les utilisateurs
  - Pas d'exigences pour analyser
  - Pas de temps pour des plans de test complets
- Découvrez ce dont les utilisateurs ont besoin
  - Assurez-vous que ces fonctions fonctionnent
- S'assurer que les testeurs prennent la tête
  - Assumer la responsabilité de la qualité de l'application
  - Devenir une interface pour la communauté des utilisateurs
  - Travailler avec les développeurs en tant que partenaires égaux
- Permettre aux testeurs de travailler côte à côte avec les développeurs et les clients
  - Les testeurs doivent connaître les décisions de développement tactique
  - Ceci est particulièrement important dans les projets Agile sans exigences formelles

Intégration des tests dans le développement agile

- S'assurer que les testeurs prennent les devants sur le bâtiment et l'exécution des tests fonctionnels, de régression et d'acceptation
- L'indépendance vis-à-vis du développement permet d'assurer une évaluation objective
- Permettre aux testeurs et aux développeurs de prendre ensemble des décisions de codage
- **Aide à hiérarchiser les tests**
- Guide de développement dans la prise de décision

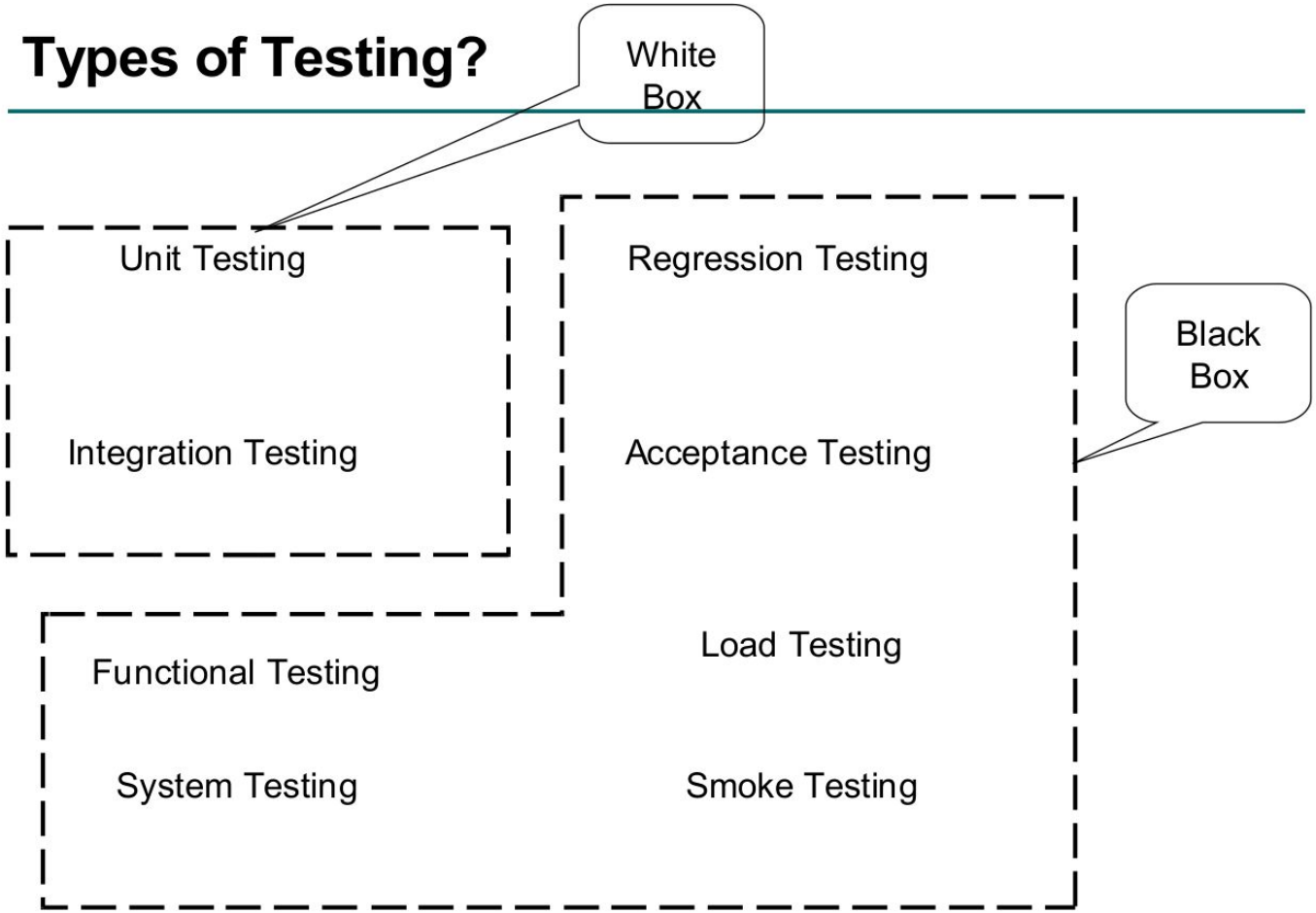
Intégration des tests dans le développement agile

- Faire de l'automatisation une partie intégrante des tests
  - Agilité, automatisation, main, main
  - Accélérer les tests pour montrer l'agilité
- Collecte et analyse des données essentielles
  - Carnet de commandes
  - Échec / Gestion des problèmes
  - Gestion technique de la dette
  - Enregistrement
  - Métrique
  - Commencer les commentaires
  - Suivi du temps, ...

Intégration des tests dans le développement agile

- Les outils sont essentiels pour la vitesse et la flexibilité
- Les outils accélèrent les processus manuels répétitifs
- Les frais généraux des outils ne peuvent pas dépasser les avantages
  - Facile à apprendre et à utiliser
  - Fournir des informations faciles à digérer
- Automatisation des tests
  - Gestion des tests
  - Tests fonctionnels et de régression automatisés
  - Tests de charge automatisés - précieux pour l'évaluation des pratiques de conception et de codage

Types of Testing?



Niveaux d'essai

- Test unitaire: Test Driven Development
- Tests d'intégration: intégration continue
- Tests de systèmes: principalement des rôles
- Essais d'acceptation:
  - Le client teste les fonctionnalités uniques dès qu'elles sont disponibles
  - Toutes les fonctionnalités incluses dans le sprint sont testées au cours des derniers jours du sprint
  - Acceptation au sprint
- Pas le moyen de travailler phase par phase, mais de donner la responsabilité

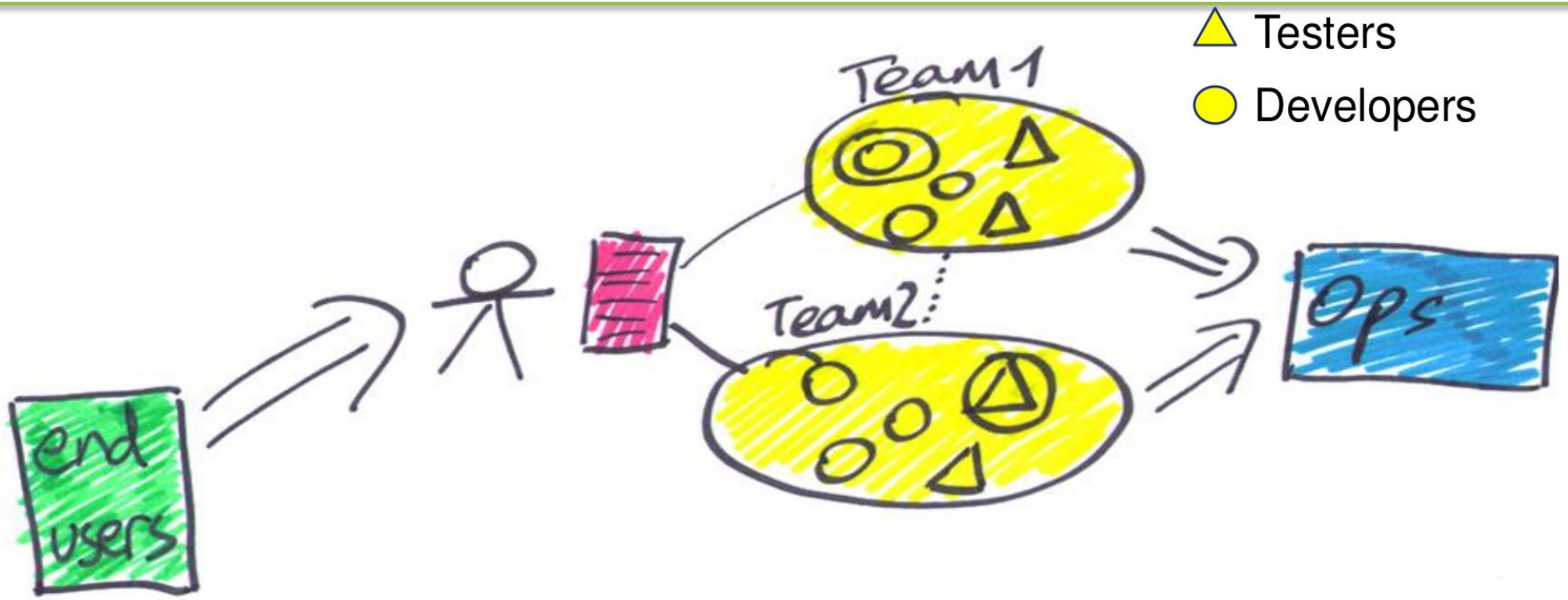
Who Performs What?

What?	Who?	When?	Automation?
Unit Testing	Developer	Coding	Always
Integration Testing	Developer	Coding	Always
System Testing	Tester	Test	Possible
Regression Testing	Developer/Tester	Build/Test	Possible
Acceptance Testing	Client/Users	Deployment/ Delivery	Possible
Smoke Testing	Tester/Support Engineer	Deployment	Possible
Load Testing	Performance Engineer	Deployment	Always

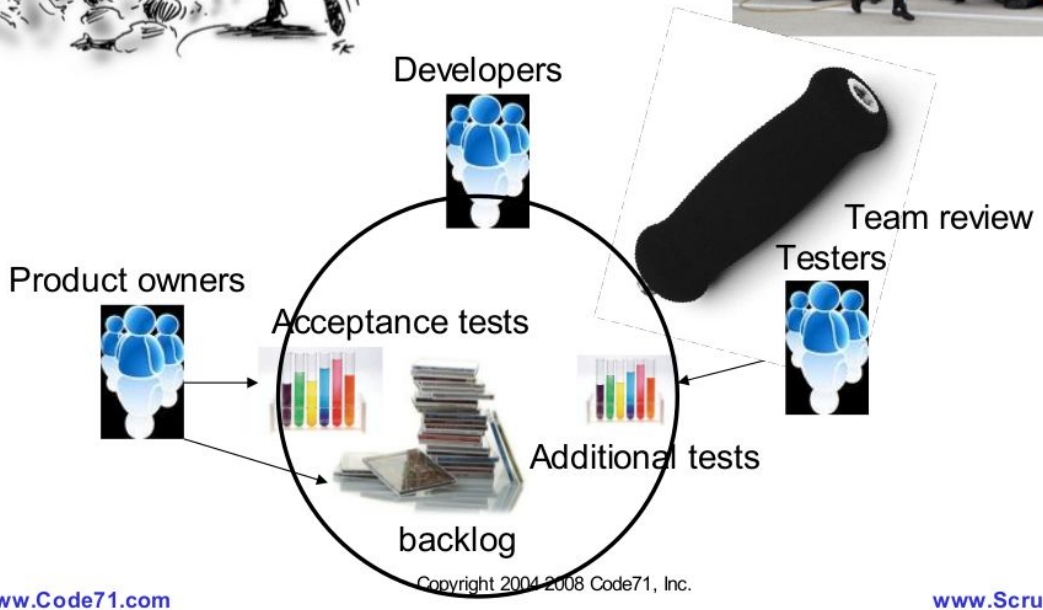
Right tools for right tests?

Test	Tool
Unit Testing	NUnit, JUnit, Mock, DBUnit
Integration Testing	Unit test tools, HttpUnit, SoapUI, RESTClient
System Testing	Selenium, Fit, WET, Watir, WatiN
Regression Testing	Unit test tools, System test tools
Acceptance Testing	FIT, FitNesse
Smoke Testing	Regression test tools
Load Testing	JMeter, Httperf

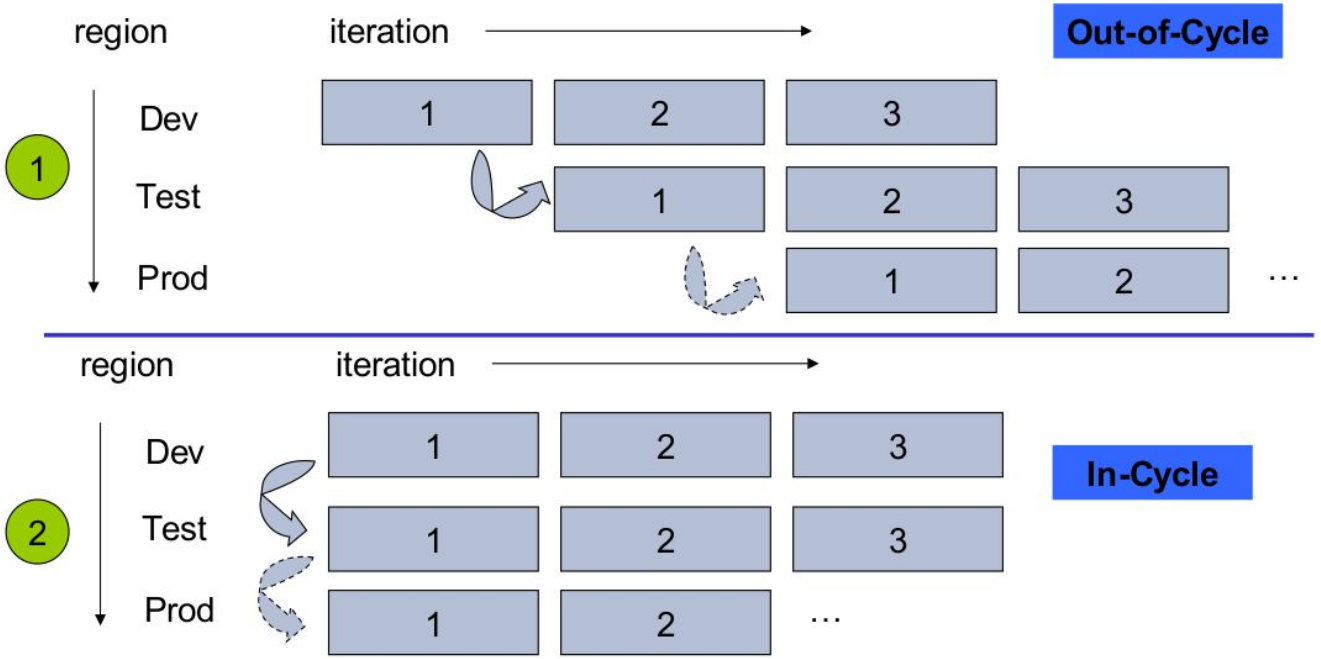
Équipes



Team Dynamics



Team Composition?



We will primarily focus on single-team model for our discussion

Testeurs à Scrum

- Réunion quotidienne Scrum
- Se tenir au courant des progrès réalisés par les développeurs
- Essais (comprend la conception de test parallèle et le test)
- Prise en charge de la couverture des tests
- Combler les défauts et les rapports
- Maintenance des tests et automatisation des tests
- Éviter le manque de tests (dette de qualité)
    - Respectez le rythme des développeurs
    - Évitez l'attitude:: "le reste des tests peut être fait plus tard"

TESTING PROCESS

Product backlog, sprint

Requirements, functionalities & user stories

List of test cases

Functional testing and exploratory testing

Testers

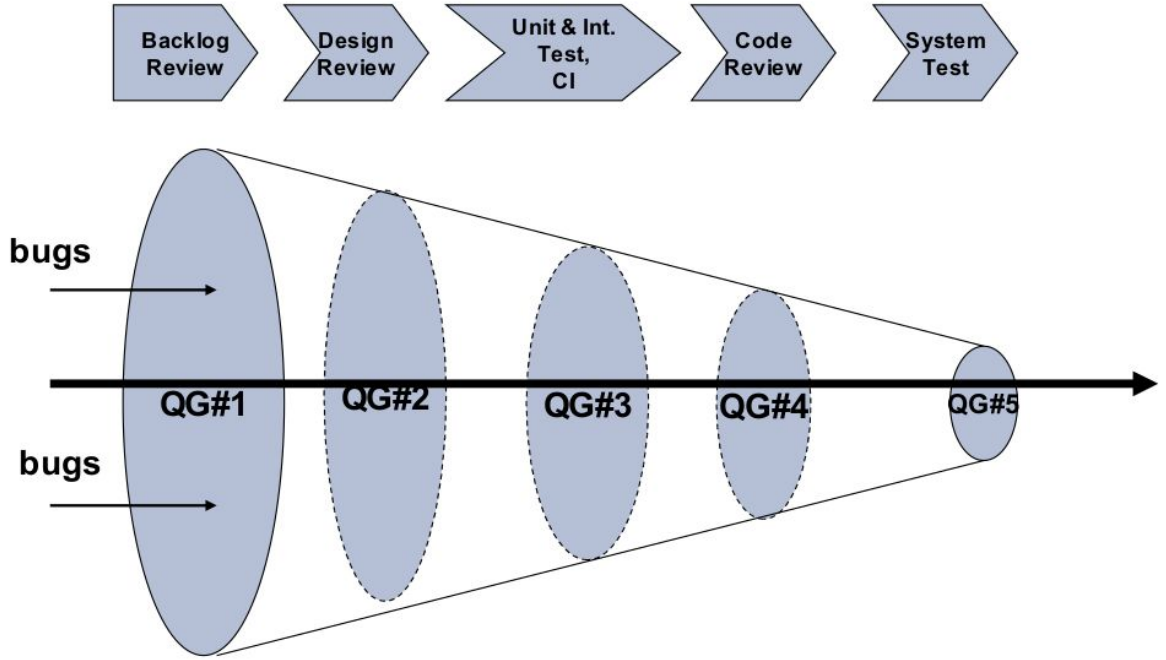
Developers

Day1 Start

Day 2-17 Implement functionality

Day 18-20 Fix defects

Quality Funnel



Planification des tests agiles

- Commencez par une légère planification d'essai globale
- Choisissez les outils appropriés
- Créer un environnement de test
- Donner la priorité aux choses à tester dans chaque itération (sprint)
- Créer des cas de test pour la première itération sur le niveau de précision (léger) convenu
- Établir les contours des prochaines itérations
- Automatiser les tests

Qu'est-ce test agile n'est pas?

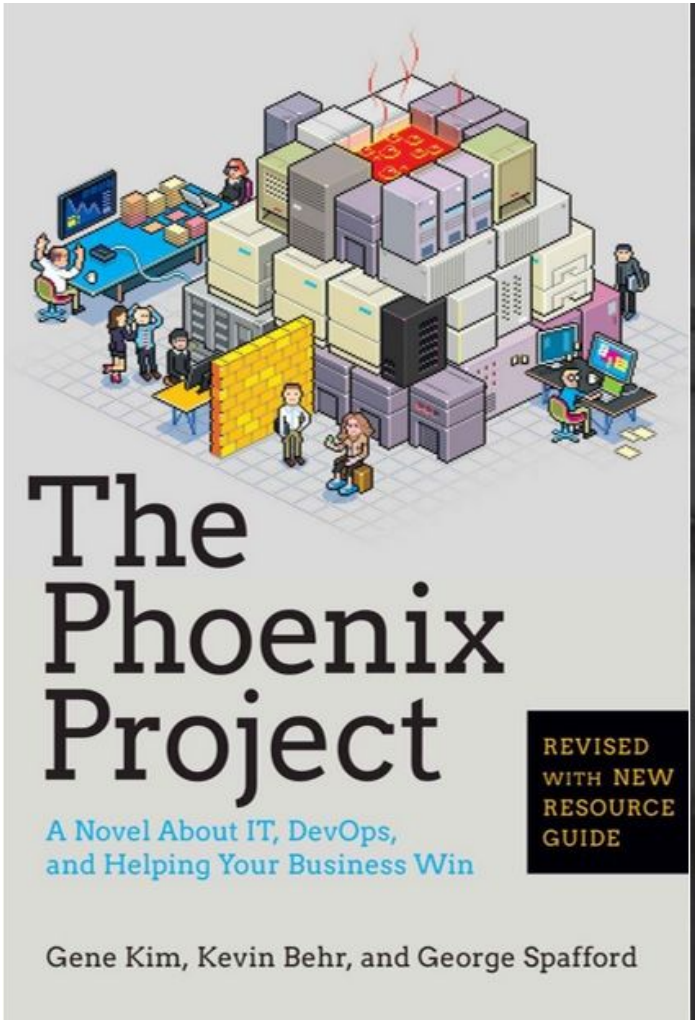
- Une phase distincte à la fin du projet de développement logiciel
- Peu systématique
- Documentation gratuite
- au hasard
- Incontrôlé
- Fonctionnement simple sans rétroaction

Fournir des logiciels de travail fréquemment

- Une semaine à un mois
- Le calendrier continue de raccourcir
- **But -> Livraison continue**

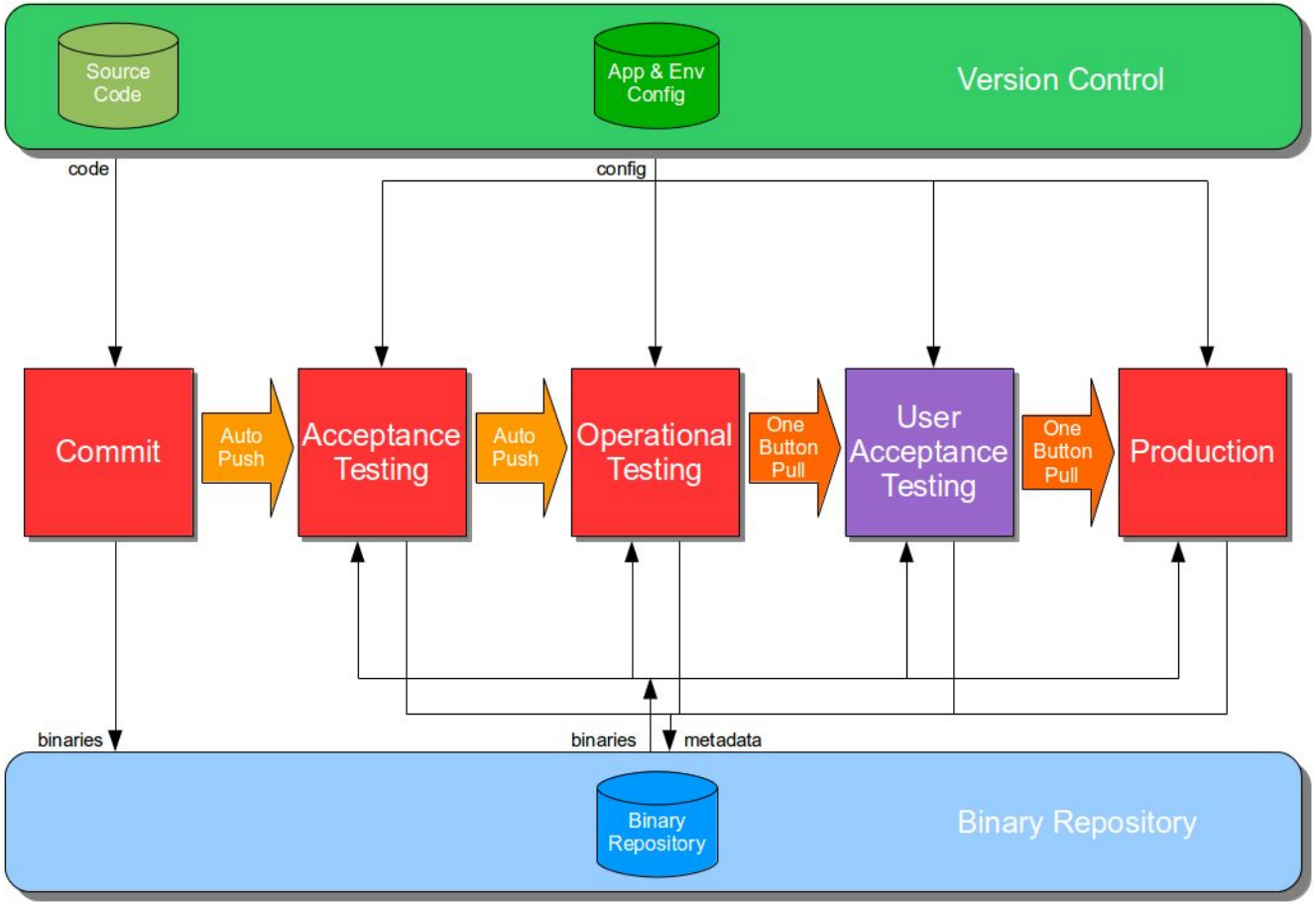
Test de valeur

- Donner le droit, c'est encore plus important que de le faire bien
- Adapter l'intention vraie et implicite du client à une spécification
- Ce n'est pas parce qu'un client a dit quelque chose que nous devons faire
- Ce n'est pas parce qu'un client n'a pas dit quelque chose que nous ne devons pas faire


The Phoenix Project
REVISED WITH NEW RESOURCE GUIDE
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford
Copyrighted Material
The Addison-Wesley Signature Series
A MARTIN FOWLER SIGNATURE BOOK
CONTINUOUS DELIVERY
RELIABLE SOFTWARE RELEASES THROUGH BUILD, TEST, AND DEPLOYMENT AUTOMATION
JEZ HUMBLE
DAVID FARLEY
Foreword by Martin Fowler
Copyrighted Material

Livraison précoce et continue des logiciels

- Exigences moins formelles - user stories
- Cycle de développement itératif
- Une semaine à un mois
- Logiciel de travail à la fin de chaque

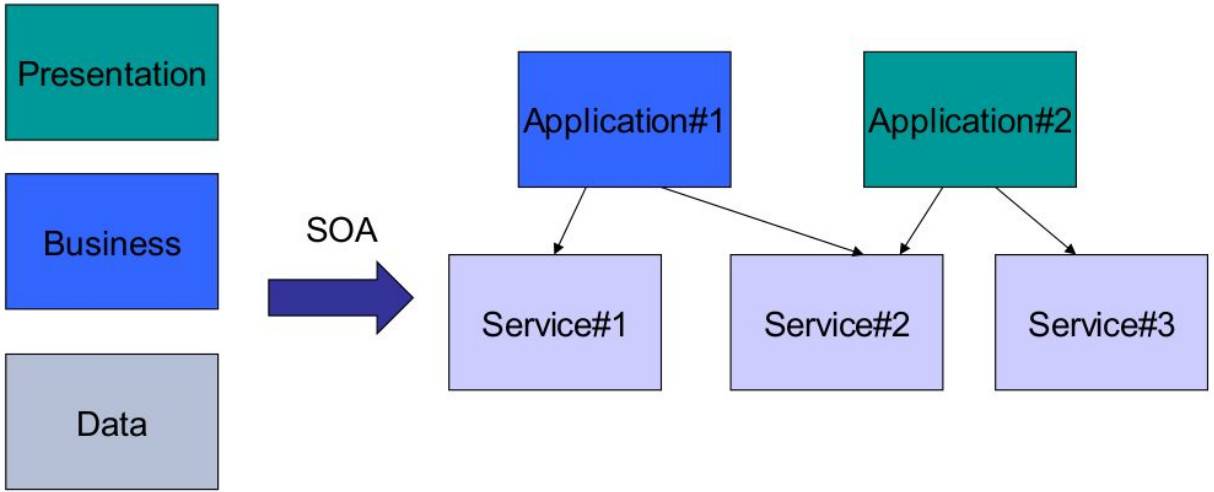


http://www.alwaysagileconsulting.com/training/continuous-delivery-for-practitioners/

Le dilemme de la faute zéro (défaut) sur le développement logiciel:

Si je sais qu'un système de logiciel sur lequel je travaille **a plusieurs défauts**, pourrais-je accepter les défauts allant au **déploiement en production**?

How to Scale?



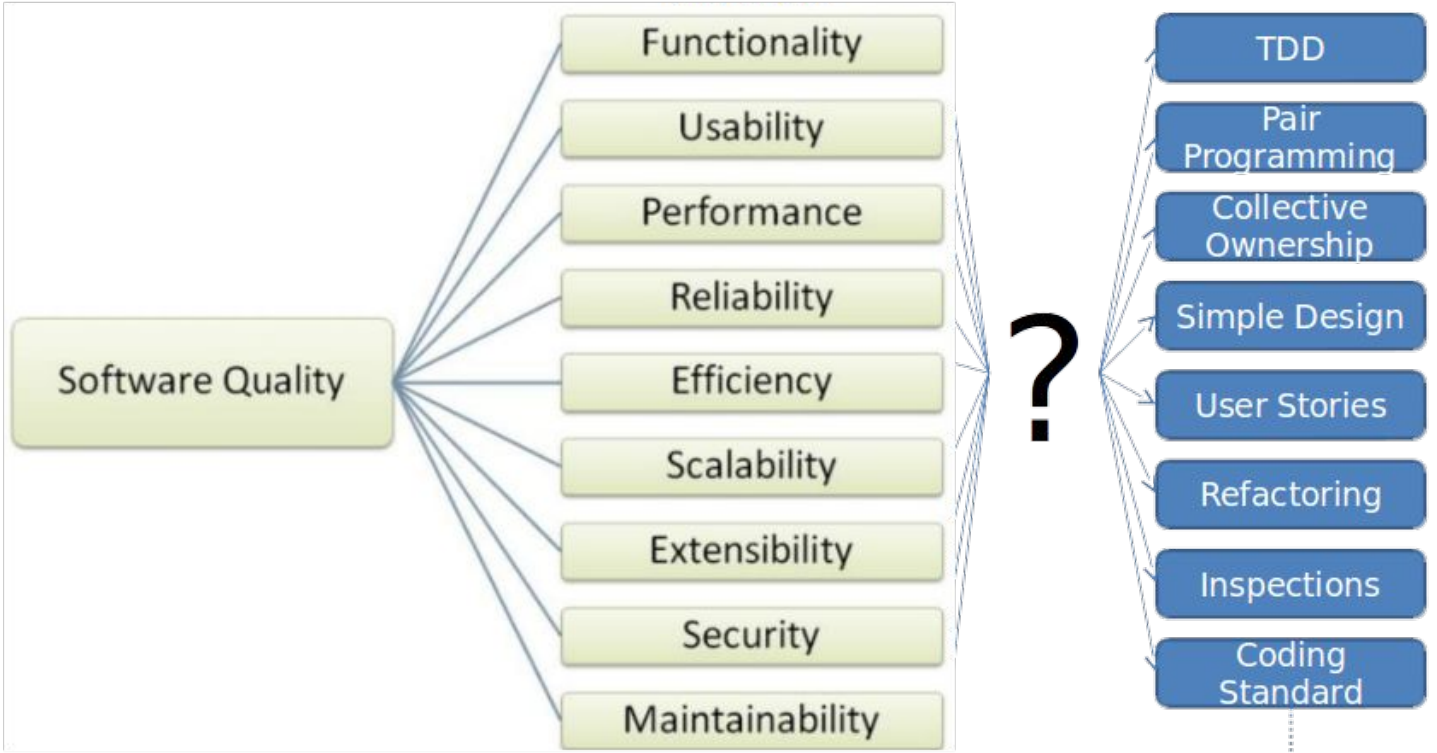
Contraintes typiques agiles

- Pas beaucoup de testeurs
  - Les développeurs ont la responsabilité principale de tester
- Le rôle du testeur consiste à:
  - Testez le système dans son ensemble à partir de nombreuses perspectives
  - Assurez-vous que chaque développeur prend soin de sa propre zone
- Risque
  - Tous les essais sont laissés aux quelques testeurs

Carnet de commandes du produit et carnet de commandes du Sprint

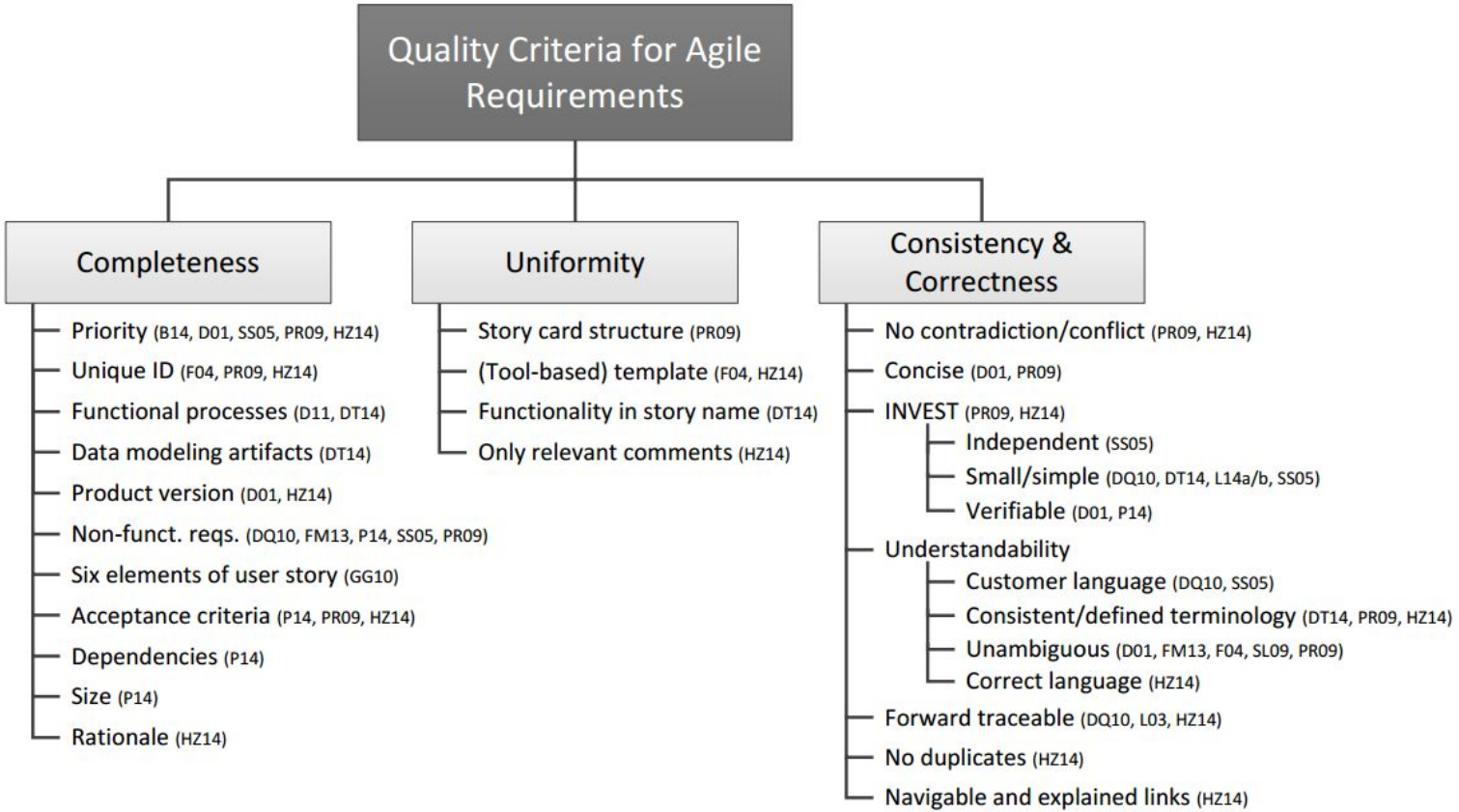
- Carnet de commandes du produit
  - Suivi général des tâches de développement, contient PASS / FAIL du point de vue des tests
  - L'arrière-plan du produit fonctionne comme un système interne de test
- Carnet de commandes du Sprint est une partie prévue du carnet de commandes du produit, sa propre vue
  - Les tâches de développement incluant également les tests unitaires
  - Le test du système est une tâche distincte pour chaque fonctionnalité
  - Suivi au niveau de la tâche des tests

ISO 9126



What quality characteristics are improved with each practice?

(Mnkandla, E., & Dwolatzky, B. (2006). Defining Agile Software Quality Assurance.)



Petra Heck · Andy Zaidman, 2016. A Systematic Literature Review on Quality Criteria for Agile Requirements Specifications. http://www.st.ewi.tudelft.nl/~zaidman/publications/HeckSQJ2016.pdf

Mettre l'accent sur la construction du bon produit pour le client
(Au lieu de le construire correctement)

(Timperi, O. (2004). An Overview of Quality Assurance Practices in Agile Methodologies.)

Le test est laissé obligatoire mais vague parce que son niveau et sa couverture ne sont pas indiqués.

(Timperi, O. (2004). An Overview of Quality Assurance Practices in Agile Methodologies.)

Le nouveau rôle du QA

- Aide à la définition d'entreprise
- Aide à la mesure du risque
- Connaître les besoins des utilisateurs
- Architecture sous-jacente

(Mnkandla, E., & Dwolatzky, B. (2006). Defining Agile Software Quality Assurance.)

Agile QA - Défis

- Les intervenants se concentrent sur la fonctionnalité du logiciel
- Les exigences fonctionnelles essentielles sont probablement mal définies et négligées
- Les développeurs habitués à la cascade supposent que nFR sont résolus par des architectes et des concepteurs.
- Une nouvelle approche est nécessaire pour améliorer les aspects non fonctionnels par des attributs de qualité.

Um, T., Kim, N., Lee, D., & In, H. P. (2011). A Quality Attributes Evaluation Method for an Agile Approach.
(Rachel Davies, Agile Experience Ltd, UK, Non-Functional Requirements: Do User Stories Really Help?)

Example:

As a blogger, I want to post an entry quickly and efficiently to easily update the blog

Test that ...
- A blog post should be posted in less than 3 clicks.

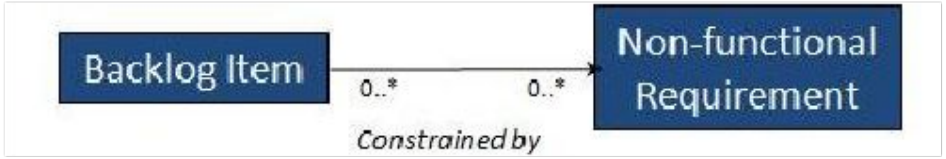
- A blog post should be posted in less than 5 seconds.

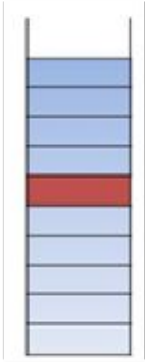
The problem is that:

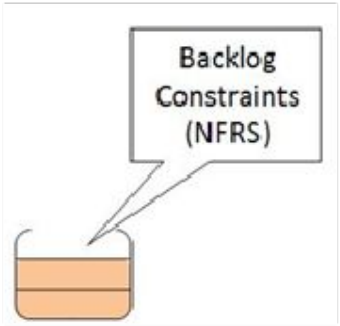
- The stories are designed to be completed

- Tunnel vision: when the team focuses on a story is easy to forget the RnF.

Comment traiter toutes les exigences non fonctionnelles couvrant toutes les histoires?







Involving the customer is essential, explaining the importance of quality attributes.

(Leffingwell D. (2009). Scaling Software Agility. Agile Requirements Information Model. *Online*.)

Reminders

As a blogger, I want to post an entry to update the blog

User friendly

nFR filters

Efficient

As a blogger, I want to post an entry quickly and efficiently to easily update the blog

Assign responsibles for each quality attr.

General Issues

New role, responsible for QA to understand and measure risk and assist in QA activities.

nFR ill-defined

Non-functional requirements

Involving the customer is essential, explaining the importance of quality attributes.

Proposed solutions

Stories quality filters.

Reminders

Assign responsibles for each quality attr.

Question de recherche:
Les projets Open Source / Free Software sont-ils agiles?

Les projets Open Source / Free Software sont-ils Agiles?

Mon hypothèse: OSS est un processus "hyper" agile.

Question de recherche:
Quelle approche dois-je utiliser?
Classique ou Agile?

Quelle approche dois-je utiliser? Classique ou Agile?

Mon expérience personnelle montre que cela dépend de la mentalité du client. Certains clients et entreprises ne sont pas compatibles avec l'approche agile.

Sources de référence importantes

- http://asq.org/qic/display-item/index.html?item=34129
- http://aetos.it.teithe.gr/~sfetsos/Publications.html
- http://ieeexplore.ieee.org/document/5654783/
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.482.2115&rep=rep1&type=pdf
- http://www.st.ewi.tudelft.nl/~zaidman/publications/HeckSQJ2016.pdf
- http://dl.acm.org/citation.cfm?id=1919947

TP 1 - Analyse et Diagnostic

- **Objectif:** analyser et rapporter un diagnostic d'un processus d'assurance de la qualité du logiciel en utilisant ces 4 dimensions / perspectives:
  - 1) Modèle de facteurs de qualité logiciel de McCall
  - 2) Composants du système SQA de Galin
  - 3) Agile SQA
  - 4) ISO 25000
  - Ou suivez ISO 15504-5: 2006 (Norme d'évaluation de processus) - Facultatif
- Trouvez et choisissez un projet open source (ou une entreprise qui vous soutient)
- Analyser, évaluer et rapporter le projet “Software Quality Assurance” en 4 dimensions
- La note est proportionnelle à la complexité, à l'exhaustivité, à la concision
- Groupes de 3 participants (idéal, max. 4). **Un rapport par groupe en Module!**
- Date limite: **Fèvrier, 17 pour les 2 GROUPES (B1 ET B2)!**

Prochain cours

CMMI and ISO 25000