



IBM Rational software

Méthodologie de développement Agile

Dom Derrien
Concepteur de logiciels
Spécialiste en technologie Web 2.0

Avertissement

- **Mon message n'est pas celui d'IBM.**
- **Mon message n'est pas celui d'IBM Rational.**
- **Mon message s'appuie sur mon expérience, notamment de celle acquise chez IBM Rational.**
- **IBM Rational a une stratégie pour le développement « Agile » qui n'est pas toujours en phase avec mon message parce qu'elle vise plus les décideurs que les développeurs.**



Introduction

Introduction

- **Je vous veux donner le goût d'essayer la méthodologie Agile.**
- **Vous n'avez pas encore assez d'années d'expérience pour la refuser ;)**
- **Agile est un état d'esprit d'abord, puis une méthode dans les faits.**
- **Développer Agile est contraignant mais la qualité de la production est là !**

Agenda

- **Constat des méthodes traditionnelles**
- **Introduction de la méthode Agile**
- **Caractéristiques de la méthode :**
 - Petites étapes et *refactoring* constant ;
 - Tests unitaires et mentalité du « zéro défaut » ;
 - Intégration continue et déploiement automatique ;
 - Tests fonctionnels et validation par le client.
- **Perspectives de la méthode**

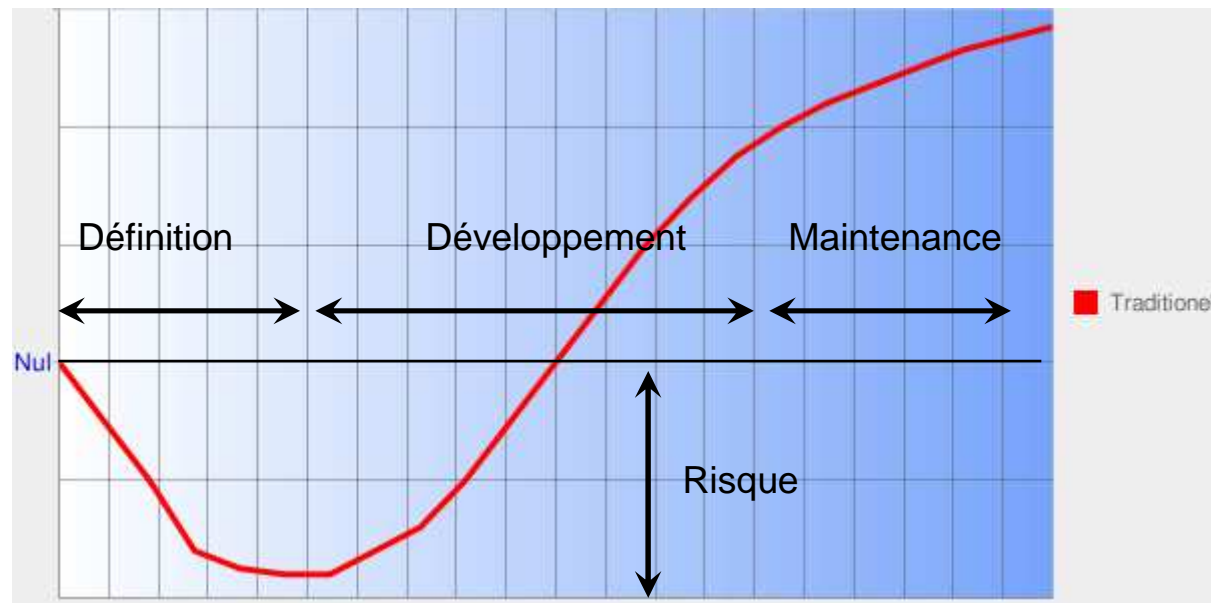
Méthodes traditionnelles

Méthodes traditionnelles

- **Structurées comme les processus industriels :**
 - Phase de définition de requis ;
 - Phase de définition de l'architecture ;
 - Phase d'implémentation ;
 - Phase de test ;
 - Phase de maintenance.
- **Ces phases sont séquentielles.**

Coût / bénéfice

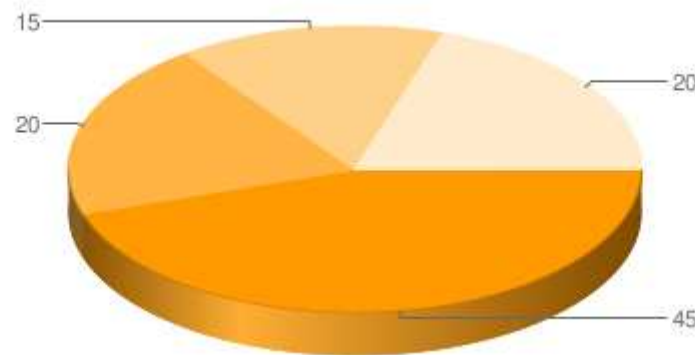
- Phase de définition et début d'implémentation sont à risque



Usage des fonctionnalités spécifiées

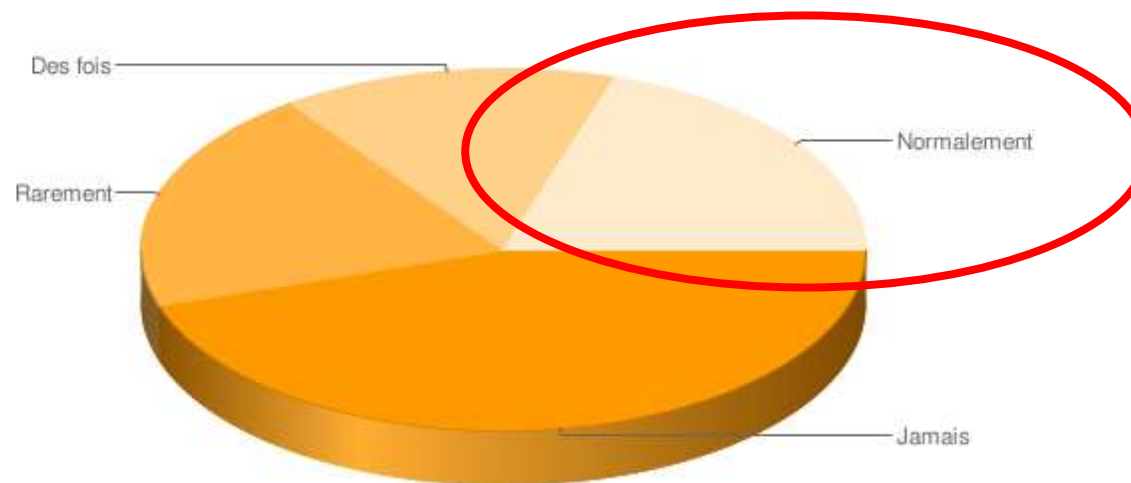
- Habituellement, le cahier des charges décrit toutes les fonctionnalités à implémenter.
- Question : quelle part représente les fonctionnalités essentielles ?

• Jamais	• 40 %
• Rarement	• 20 %
• Des fois	• 20 %
• Normalement	• 15 %



Usage des fonctionnalités spécifiées

- **Seulement 20% des fonctionnalités sont couramment utilisés !**
- **Pire : 40% ne sont jamais utilisées**



Coût / bénéfice

- **Le coût du développement est souvent contrôlé parce que le plan est à dates fixes.**
- **L'immuabilité de ces dates et l'importance des objectifs à rencontrer font que la qualité est souvent sacrifiée.**
- **Il y a des projets qui meurent parce que les coûts de maintenance dépassent les revenus.**



Méthode Agile

Méthode Agile

- **Manifeste créé en 2001 par 17 experts.**
- **4 valeurs fondamentales et 12 principes :**
 - L'**interaction avec les personnes** plutôt que les processus et les outils.
 - Une **production opérationnelle** plutôt qu'une documentation pléthorique.
 - La **négociation avec le client** plutôt que le respect d'un contrat.
 - La **collaboration au changement** plutôt que le suivi du plan.

Interaction avec les personnes

- **Les outils sont des moyens qui ne remplacent pas la discussion entre les intervenants.**
- **Réunions de synchronisation journalières (*SCRUM meetings*).**
- ***Poker Planning.***
- ***Code review* avant de livrer les fonctionnalités.**
- **Communication directe avec le client quand c'est nécessaire.**

Production opérationnelle

- **Les spécifications n'ont pas à être complètement écrites pour commencer le développement.**
- **Le code et le résultat des tests deviennent la documentation finale.**
- **La documentation « écrite » est restreinte :**
 - Aux stratégies abandonnées ;
 - Aux scénarios associés à l'étape de développement.

Négociation avec le client

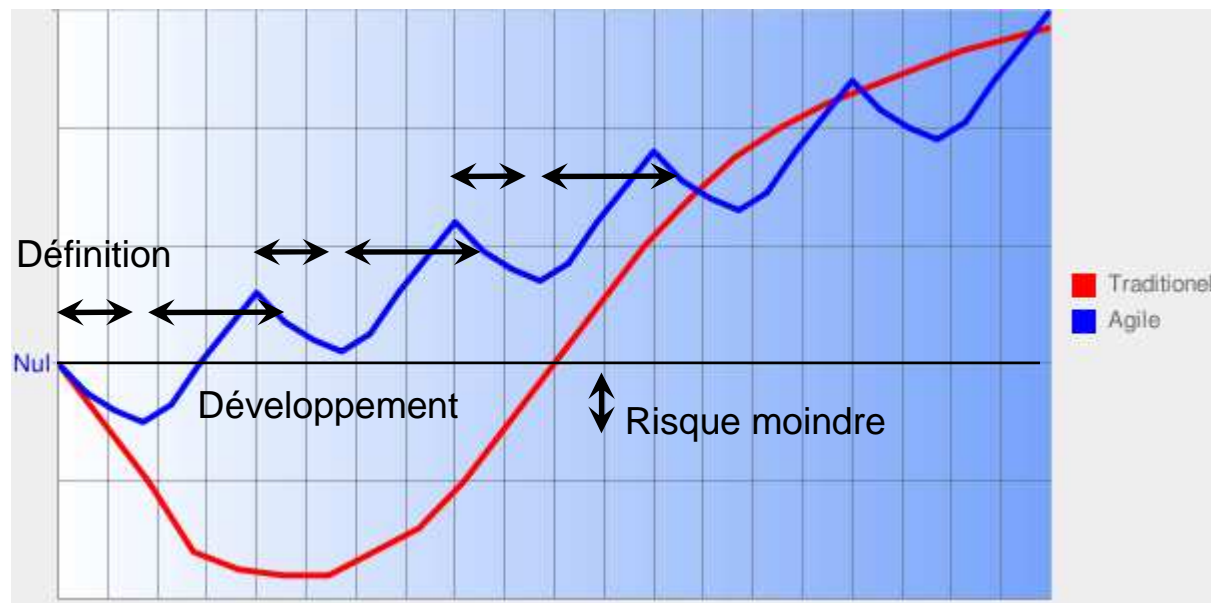
- **Le client doit faire partie du processus de développement :**
 - En tant décideur pour la gestion des priorités ;
 - En tant que critique du produit livré à chaque étape.
- **Sans doute l'élément le plus perturbateur car les attentes du client doivent être modérées...**

Collaboration au changement

- **Le processus du développement est incrémental et vise juste à remplir les objectifs de chaque étape, sans tenir compte des suivantes.**
- **Une étape peut changer :**
 - Parce que le client le demande ;
 - À cause de contraintes récemment découvertes ;
 - Par l'équipe de développement change.
- **Le changement (*refactoring*) doit être transparent.**
- **Accepter le *code review*.**

Coût / bénéfice comparés

- L'identification des priorités, la mise en œuvre du zéro défaut, etc., permettent la mise en marché plus rapide et diminuent les risques



Coût / bénéfice

- **La probabilité de profit n'est pas plus grande.**
- **Le risque et l'exposition au risque sont réduits.**
- **Les possibilités d'ajustement sont augmentées.**
- **Le coût de maintenance est grandement réduit.**
- **La distribution des coûts est plus floue.**
- **La dépendance au professionnalisme des acteurs est critique.**



Pratique de la méthode Agile

Pratique de la méthode Agile

- **Le rythme des étapes varie de 6 à 8 semaines.**
- **Pour chaque étape, les tâches sont déjà classées.**
- **Le déroulement du travail comprend :**
 - La phase de conception (1 semaine) ;
 - La phase de codage et test unitaire (4 à 6 semaines);
 - La mise à jour des documents, la consolidation des tests unitaires, les tests fonctionnels (1 semaine).

Pratique de la méthode Agile

- **Réunion de coordination en groupe chaque jour.**
- **Révision collective des documents, du code et des tests produits.**
- **Observance du « zéro défaut » : les erreurs sont corrigées avant de poursuivre le développement.**
- **L'objectif: à la fin de chaque étape, le produit doit être fonctionnel et évaluable par le client dans son environnement.**

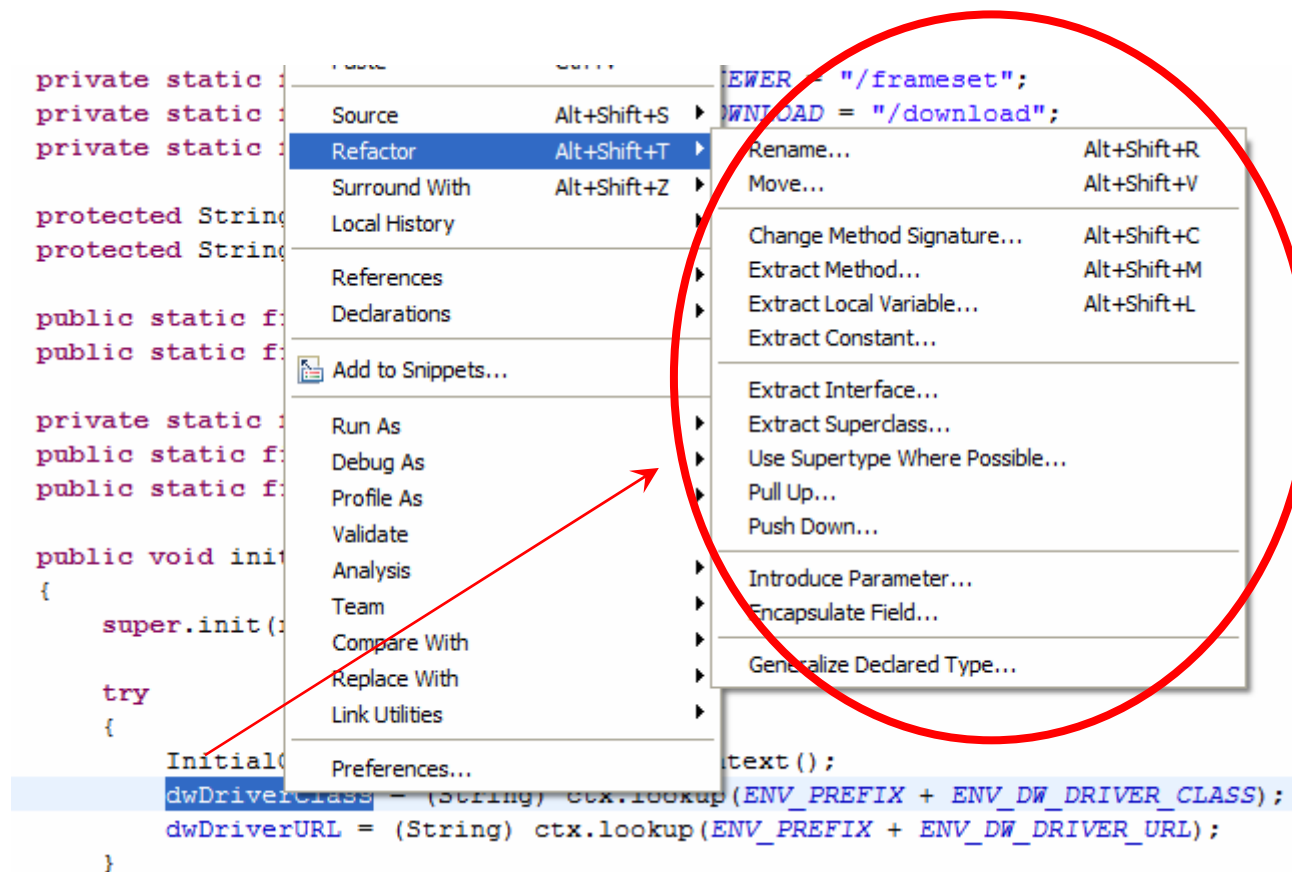
Pratique de la méthode Agile

- **L'outil est juste un moyen certes, mais avoir le bon outil est très important !**
 - *Refactoring*
 - Tests unitaires
 - Gestion des sources
 - Intégration continue

Refactoring

- **Outils : Eclipse, IntelliJ, Visual Studio.**
- **Options de *refactoring* :**
 - Renommer et déplacer ;
 - Extraire l'interface (hiérarchisation) ;
 - Insérer des paramètres, etc.
- **L'outil doit être couplé avec le système de gestion des sources pour permettre des *refactorings* faciles.**

Refactoring



Gestion des sources

- **Utilisation du système de gestion des sources très fréquente :**
 - Plus facile de partager les progrès ;
 - Plus facile de défaire plusieurs petits changements qu'un seul gros ;
 - Plus facile à porter dans une autre branche ;
 - Plus de retour du système d'intégration continue.
- **Outils : Subversion (CVS), ClearCase, GIT.**

Intégration continue

- **Si tout développement peut commencer sans documentation complète, le système d'intégration prime pour :**
 - Extraire le code du système de gestion des sources ;
 - Compiler et rouler les tests unitaires et fonctionnels ;
 - Produire les statistiques de couverture ;
 - Déployer l'application ;
 - Produire la documentation.
- **Outils : Ant, CruiseControl, BuildForge.**

Tests unitaires

- **Quand le développeur écrit les tests unitaires :**
 - C'est le premier consommateur du code ;
 - Il peut fixer les problèmes avant de livrer le code ;
 - Il s'assure que des *refactorings* subséquents ne briseront pas son code.
- **Quand le développeur écrit les tests AVANT le code :**
 - Il se concentre sur la fonctionnalité (l'*API*) ;
 - Il produit du code plus « réutilisable ».

Processus de développement

- **Petites tâches.**
- **Souvent poussées dans le système de gestion des sources.**
- **Toujours testées (100% de succès dans les tests, ~100% de couverture).**
- **Vite déployées (pas plus de 10 min. pour informer d'un problème).**
- **Pilote accessible à n'importe qui avec une méthode facile pour rapporter les erreurs.**

Tests unitaires (Java)

- Outil : JUnit

[Home](#)

Packages

[com.ibm.rpm.build](#)

[com.ibm.rpm.i18n](#)

[com.ibm.rpm.rest](#)

[com.ibm.rpm.restdrivers](#)

Classes

[TestAjaxResponse](#)

[TestDefaultMetadataFac](#)

[TestDefaultResponseFac](#)

[TestDocumentDriver](#)

[TestGenericDrivers](#)

[TestLabelConstants](#)

[TestLabelExtractor](#)

[TestLocaleController](#)

[TestModel](#)

Unit Test Results

Designed for use with [JUnit](#) and [Ant](#).

Summary

Tests	Failures	Errors	Success rate	Time
226	0	0	100.00%	15.780




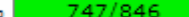



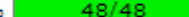

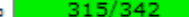

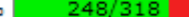
Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Name	Tests	Errors	Failures	Time(s)
com.ibm.rpm.build	63	0	0	0.656
com.ibm.rpm.i18n	42	0	0	0.734
com.ibm.rpm.rest	71	0	0	8.939
com.ibm.rpm.restdrivers	50	0	0	5.451

Couverture des tests

- Outil : Cobertura

Packages		Coverage Report - All Packages				
All						
com.ibm.rpm.build						
com.ibm.rpm.i18n						
com.ibm.rpm.rest						
com.ibm.rpm.restdriver						
						
All Packages						
Classes						
Package 	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	31	97%	1392/1440 	88%	747/846 	1.91
com.ibm.rpm.build	2	97%	292/301 	99%	136/138 	0
com.ibm.rpm.i18n	3	100%	90/90 	100%	48/48 	3.167
com.ibm.rpm.rest	30	97%	688/706 	92%	315/342 	1.69
com.ibm.rpm.restdrivers	13	94%	322/343 	78%	248/318 	2.068
Report generated by Cobertura 1.9 on 3/27/08 4:39 PM.						

Couverture de tests

```

155
156      /**
157       * Report an error code to the application instanciator
158       * in standalone mode.
159       *
160       */
161     protected void stopProcess()
162     {
163         processStopped = true;
164         if (runStandalone)
165         {
166             // To make the standalone program
167             // failing the build process
168             System.exit(1);
169         }
170     }

```

Coverage Report - com.ibm.rpm.i18n

Package ^	# Classes	Line Coverage		Branch Coverage		Complexity
com.ibm.rpm.i18n	3	100%	90/90	100%	48/48	3.167
Classes in this Package ^		Line Coverage		Branch Coverage		Complexity
LabelConstants		100%	1/1	N/A	N/A	0
LabelExtractor		100%	38/38	100%	18/18	0
LocaleController		100%	51/51	100%	30/30	3.167

Report generated by [Cobertura](#) 1.9 on 3/27/08 4:39 PM.

Tests unitaires (JavaScript)

- Outils : JUnit et JSCoverage

JUnit Test Results

Summary

Test	Browser	Test suite	Success rate	Time
47	D:\Temp\ff\firefox.exe	test/WebContent/JUnitSuite.html	100%	8.594

Note: success rate always equals 100% because the source report has been generated correctly (use the browser command "view source" to see the original report).

Test cases

Suite	Test name	Time
test/WebContent/tm/util/rtf/TestCodePageTables.html	testGetUnicodeIV	0.891
test/WebContent/tm/util/rtf/TestCodePageTables.html	testGetUnicodeIII	0.0
test/WebContent/tm/util/rtf/TestCodePageTables.html	testGetUnicodeII	0.0
test/WebContent/tm/util/rtf/TestCodePageTables.html	testGetUnicodeI	0.0
test/WebContent/tm/util/rtf/TestCodePageTables.html	testInitCPTableIII	0.016
test/WebContent/tm/util/rtf/TestCodePageTables.html	testInitCPTableII	0.0
test/WebContent/tm/util/rtf/TestCodePageTables.html	testInitCPTableI	0.0

Tests unitaires

```
public void testGetIV() {  
    // Verify the label is correctly extracted from the resource bundle  
    final String input = "test";  
    ResourceBundle rb = new ResourceBundle() {  
        @Override  
        public Enumeration<String> getKeys() {  
            return null;  
        }  
        @Override  
        protected Object handleGetObject(String key) {  
            throw new MissingResourceException("Done in purpose", "class", "method");  
        }  
    };  
    LabelExtractor.resourceBundles.put(Locale.US.toString(), rb);  
  
    String output = LabelExtractor.get(input, Locale.US);  
  
    assertEquals(input, output);  
}
```

*Mock object
(IOC Pattern)*



Perspectives

Perspectives

- **Agile se résume en :**
 - Faire simple et sûrement (*KISS*) ;
 - Avoir un retour rapide du client ;
 - Avancer ainsi là où est la priorité.

Perspectives

- **Tous les finissants se ressemblent.**
- **Ce qui les distingue, c'est leur expérience.**
- **Agile est une méthode qui permet de sortir du lot.**

- **Agile est une méthode relativement inconfortable dans la mesure où tout est ouvert et tout peut être remis en cause.**

Perspectives

- **Les petites entreprises pratiquent souvent Agile mais s'en servent comme un excuse.**
- **Les grosses entreprises vont vers Agile (elles cherchent toujours à réduire les coûts et les risques du développement logiciel) mais leur « penchant administratif » brime la méthode.**
- **Les initiatives qui ont du succès groupent peu de développeurs autour de fonctions indépendantes qui évoluent alors selon Agile.**



Conclusion

Conclusion

- **Vous avez vu une introduction de Agile depuis l'esprit jusque la méthode.**
- **Vous devez chercher et étayer. Dans le travail du développement, 20% du temps devrait être consacré aux travaux de recherche d'information.**
- **Vous devez essayer pour connaître vos limites à Agile et mesurer votre efficacité.**

Conclusion

- **Des questions ?**

Références

- **Martin Fowler**
(martinfowler.com)
- **Manifeste Agile sur Wikipedia**
(fr.wikipedia.org/wiki/Manifeste_agile)
- **Groupe « Agile Montréal »**
(www.pyxis-tech.com/agilemontreal/fr)
- **Initiative IBM « Agile@IBM »**
(www.ibm.com/software/rational/agile/resources)
- **Livre « JUnit in Action » (surtout le chapitre 7)**
(www.manning.com/massol)