



PROCÉDÉS DE DÉVELOPPEMENT

Procédés synthétiques (réputés agiles)

PR002
v231d
2025-01-21

Christina.Khnaisser@USherbrooke.ca
Luc.Lavoie@USherbrooke.ca

© 2018-2025, Μητις (<http://info.usherbrooke.ca/lavoie>)
CC BY-NC-SA 4.0 (<https://creativecommons.org/licenses/by-nc-sa/4.0/>)

chrono : 100 minutes

TABLE DES MATIÈRES

- Aperçu
- XP
- Scrum
- Kanban
- Synthèse
- Vocabulaire usuel
- Références
- À suivre

APERÇU

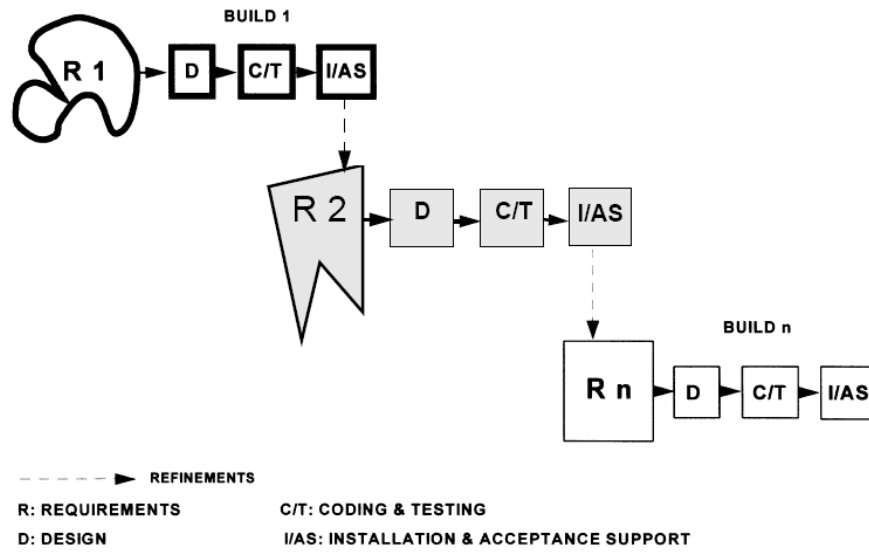
- Concepts
- Schéma général
- Commentaires

APERÇU

CONCEPTS (SYNTHÈSE)

- Les procédés synthétiques induisent la spécification des exigences plutôt qu'ils n'en découlent.
- L'exploration, l'analyse et la synthèse des exigences se font tout au long des itérations.
- La dernière itération doit voir à leur spécification.
- Tout au long du procédé, la pleine participation des représentants des parties intéressées est requise.
- Pour cela, ils sont réputés agiles (du moins lorsqu'ils sont pratiqués adéquatement).

APERÇU SCHÉMA GÉNÉRAL




APERÇU COMMENTAIRES

- C'est une approche empirique tout aussi scientifiquement valable que la méthode prédictive (spéculative): dans les deux cas, l'expérience tranchera.
- Il est facile de garantir le respect de l'échéancier et du budget.
- Plusieurs caractéristiques corolaires du procédé posent cependant de sérieux problèmes en pratique:
 - incertitude quant à la convergence (en fonctionnalité ou en qualité);
 - mobilisation importante, voire quasi exclusive, d'importantes ressources au sein des parties intéressées;
 - très grande **expertise** requise de l'ensemble des participants;
 - très grande **expérience** requise de l'ensemble des participants.

XP

- Présentation
- Itérations XP
- Caractéristiques
- Courte analyse
- Quand l'utiliser?



2025-01-21 J8002: Processus synthétiques (v231d) — L. Lavoie, C. Khattar
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

7

XP

PRÉSENTATION

- L'*Extreme Programming* (XP) aurait initialement été mis au point par Kent Beck, Ward Cunningham et Ron Jeffries lors du développement d'un logiciel de calcul des rémunérations chez Chrysler.
- La programmation extrême est une méthode agile constituée d'un ensemble de techniques et d'activités pratiquées par une équipe de **programmeurs** dans le but de produire des logiciels de **qualité**.

2025-01-21

PR002: Processus informatiques (V231d) — L. Lavoie, C. Knaissier
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

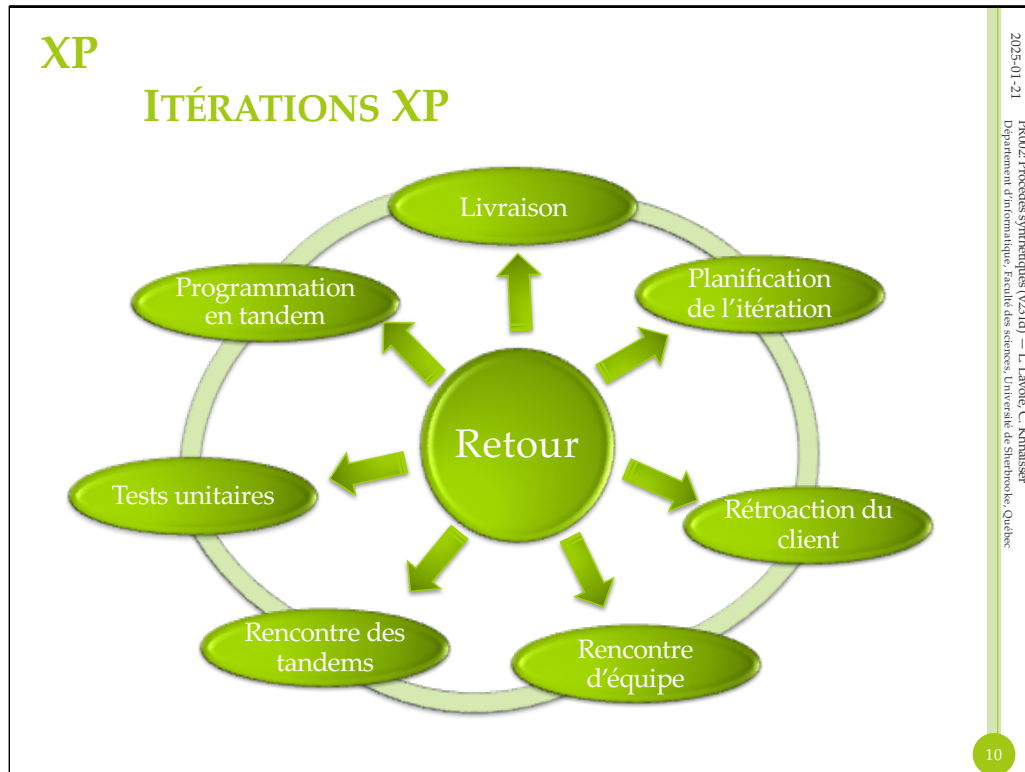
8

Ici, il faut entendre « programmeur » au sens de Boehm, Dijkstra, Hoare, Meyer et Wirth. Pas au sens d'IBM, Microsoft ou Google. Pour les distinguer, nous préférons utiliser le terme « informaticien » (au sens de Wirth) en opposition à « hacker » (au sens de Google).

XP

CARACTÉRISTIQUES

- Programmation en tandem.
- Utilisation des «histoires de cas» (à ne pas confondre avec les cas d'utilisation).
- Développement systématique des essais unitaires avant la conception.
- Primauté de la simplicité de conception sur tout autre critère (de conception).
- Systématisation de la refactorisation.
- Pilotage du projet par le client.



Inspiration Figure 4.1 (page 78) Pressman 2005

XP

COURTE ANALYSE

Caractéristique	Analyse
Tandem	Est-ce vraiment mieux que les autres méthodes ? Aucune étude concluante à ce jour.
Histoire de cas	Est-ce vraiment mieux que les cas d'utilisation ? Aucune étude concluante à ce jour.
Antériorité des tests unitaires	Reprise d'une bonne idée déjà largement pratiquée, mais non imposée par les autres procédés.
Primauté de la simplicité	Exclusion pratique de nombreux domaines d'application.
Refactorisation	Excellente idée rendue nécessaire par la primauté de la simplicité, mais qui a des limites : celles de l'architecture.
Pilotage par le client	Principe naturel ou abdication de responsabilité par l'informaticien ?

XP

VARIANTES


- Tandem (variantes diverses)
 - ange gardien en alternance
 - âme soeur
 - pour revue seulement (ce n'est plus du XP!)
- Cycles
 - variables
 - évolutifs (alternance court-moyen)
 - fixes (tendance Scrum)

XP

QUAND L'UTILISER?

- Équipe réduite
(maximum 12 personnes, plutôt 4 à 8)
- Durée réduite
(maximum 1 an, plutôt 4 à 8 mois)
- Besoins changeants
- Architecture fixée à l'avance

SCRUM



- Présentation
- Itérations du Scrum
- Caractéristiques
- Quand l'utiliser?

<http://learning2bdoctor.files.wordpress.com/2011/01/rugby-scrum.jpg>

2025-01-21

IR002: Processus synthétiques (V231d) — L. Lavoie, C. Knaissier
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

14

<http://learning2bdoctor.files.wordpress.com/2011/01/rugby-scrum.jpg>

SCRUM PRÉSENTATION

- Le terme *Scrum* emprunté au rugby signifie *mêlée*. Ce processus s'articule en effet autour d'une équipe soudée, qui cherche à atteindre un but... par tous les moyens!
- C'est un processus évolutif qui vise à livrer à chaque cycle
 - une version du produit atteignant (*au moins*) la **qualité minimale convenue**,
 - construite à partir des **seules ressources disponibles**,
 - dans une **durée déterminée**.

2025-01-21

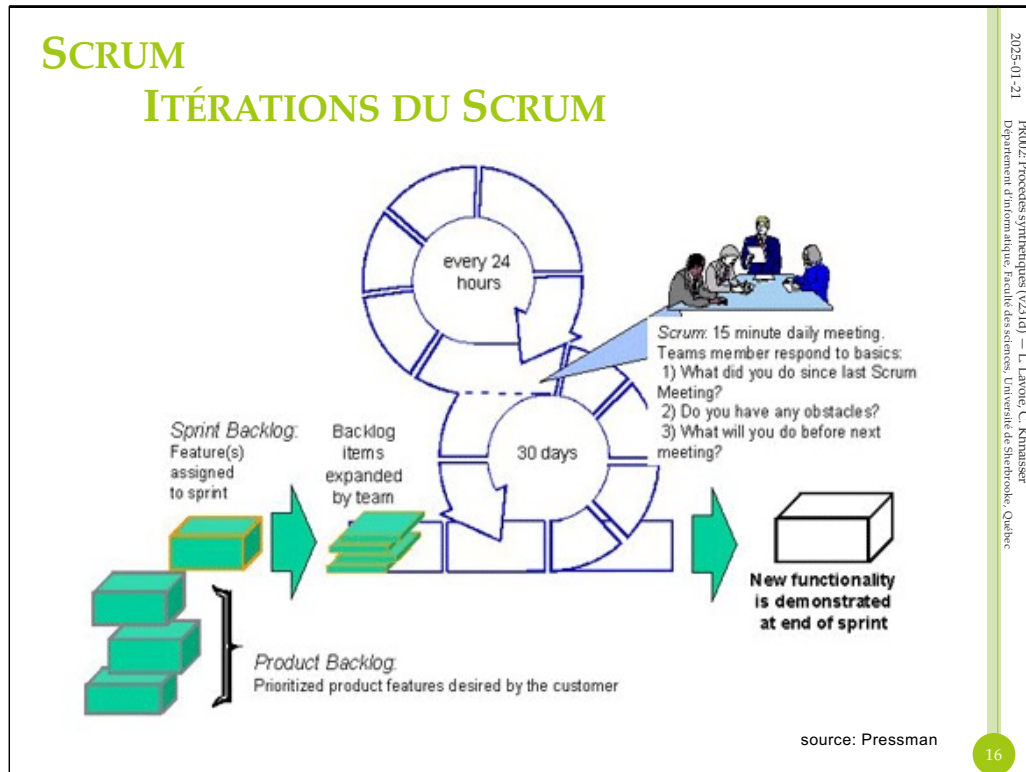
IR002: Processus informatiques (V231d) — L. Lavoie, C. Khailaie
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

15

En pratique, cette méthode a également conduit à certaines pratiques discutables en termes des lois du travail.

Ainsi, les employés sont payés au mois et l'employeur peut exiger des horaires de travail dépassant les limites légales au prétexte que les employés sont responsables des engagements qu'ils ont eux-mêmes pris.

Ces pratiques sont plus particulièrement fréquentes dans l'informatique du divertissement (dont le développement de ce qu'il est convenu d'appeler des jeux vidéos).



Pressman

SCRUM

PRINCIPE DE BASE

- Le principe de base de Scrum est de focaliser l'équipe de façon itérative sur un ensemble de fonctionnalités (*features*) à réaliser, dans des itérations de durée fixe d'une à quatre semaines, appelées **sprints**.
- Chaque sprint possède un **but** à atteindre, défini par le *directeur de produit*, à partir duquel sont choisies les fonctionnalités à implémenter dans ce sprint.
- Un sprint aboutit toujours sur la livraison d'un produit partiel fonctionnel.
- Pendant ce temps, le *ScrumMaster* a la charge de réduire au maximum les perturbations extérieures et de résoudre les problèmes non techniques de l'équipe.

Librement inspiré de plusieurs sources, dont [https://fr.wikipedia.org/wiki/Scrum_\(développement\)](https://fr.wikipedia.org/wiki/Scrum_(développement)) et <https://martinfowler.com>

SCRUM

VOCABULAIRE

Backlog

C'est une liste des exigences et des fonctions que le client désire avoir. De nouvelles données peuvent être ajoutées à la liste en tout moment.

Product Backlog

Liste des fonctionnalités priorisées par le client.

Sprints

Unité de travail nécessaire pour accomplir une exigence définie dans le Backlog dans une durée déterminée. Durant un sprint, les exigences développées, ne sont jamais modifiées.

Scrum meeting

Les rencontres quotidiennes sont très courtes (15 minutes, debout). Elles consistent principalement à répondre aux questions suivantes (pour chaque participant):

- Quels sont les travaux accomplis depuis la dernière rencontre?
- Quels sont les problèmes rencontrés?
- Que doit-on accomplir avant la prochaine rencontre?

Demos

Cette activité commence par livrer l'incrément pour que les fonctionnalités soient approuvées et évaluées par le client.

Librement inspiré de plusieurs sources, dont [https://fr.wikipedia.org/wiki/Scrum_\(développement\)](https://fr.wikipedia.org/wiki/Scrum_(développement)) et <https://martinfowler.com>

SCRUM

CARACTÉRISTIQUES

- L'équipe est dirigée par le client à temps complet, le client étant aussi un expert du domaine d'application.
- L'équipe est vue comme une extension du client qui prend toutes les décisions.
- Pas de planification.
- L'intégration se fait par accumulation.
- Un objectif est déterminé à chaque itération, et un seul.
- Compilation quotidienne (voire continue) et livraison quotidienne.

2025-01-21 IRO02: Processus symphoniques (V231d) — L. Lavoie, C. Khatissar
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

19

Analogie de l'Orchestre symphonique et de son chef... en intégrant le fait que le chef n'a pas de formation musicale 😊

SCRUM

COROLLAIRES

- Environnement de développement sophistiqué, intégrant vérifications automatisées, tests unitaires, taux de couverture, etc.:
«*si ça « build », c'est bon*» !!!
- L'intégration doit pouvoir se faire par simple accumulation (l'architecture **doit** donc être **pré-établie** et **très** simple).
- Les programmeurs sont des *Code warriors* (on oublie les définitions de Wirth, Dijkstra, Knuth, Boehm et coll.)

Code warrior : autodidacte introverti travaillant à bas salaire

SCRUM

QUAND L'UTILISER?

- Petite équipe (max. 8, plutôt 4 à 6).
- Courte durée (max. 1 an, plutôt 3 à 6 mois).
- À considérer lors du développement d'applications éphémères reposant sur une expertise pointue.
 - Développement de jeux.
 - Développement de certains types de sites et d'applications web.

SCRUM

EN CONCLUSION?

- Cohérent avec l'approche Unix, telle que mise de l'avant par Kernighan et Plauger
 - *Try brute force, think later.*
 - *If brute force don't work, you're not using enough.*
 - *If it still don't work, call Chuck Norris.*



<http://www.gargarismes-ergonomiques.com/allblogs/public/article/melee-scrum.jpg>

22

<http://www.gargarismes-ergonomiques.com/allblogs/public/article/melee-scrum.jpg>

KA(N?)BAN

En développement

- Origines
- Adaptation au développement logiciel
- Commentaires



KANBAN ORIGINES

- Procédé d'origine industriel visant à optimiser la **production en chaîne**.
- Inspirée par l'approche en flux tendu et la minimisation de l'en-cours (*just-in-time*).
- Construit autour du concept de lot.
- Mettant de l'avant la constitution de petites unités de production autonomes.
- Utilisant le modèle de qualité TQM plutôt que le modèle ISO (très important).
- Expérimenté (avec succès) par Toyota.

L'intégration des méthodes d'assurance de qualité prévues par le modèle TQM est un préalable nécessaire à la mise en place et l'exploitation d'une processus Kanban.

Rappelons que ce modèle «Total Quality Methodology » (soutenu par le Japon et le Canada) n'a pas été retenu par l'ISO qui l'estimait trop exigeant pour les entreprises.

Un modèles plus laxiste, soutenu par les États-unis et l'Union européenne, a prévalu et est à l'origine des présentes normes ISO.

KANBAN PRINCIPES ET PRATIQUES	
4 Principes	6 pratiques
<ul style="list-style-type: none">○ Partir avec l'existant○ Opérer uniquement des changements évolutifs et un seul à la fois○ Respecter intégralement le processus, les rôles et les pratiques courantes○ Encourager le leadership à tous les niveaux	<ul style="list-style-type: none">○ Visualiser○ Limiter l'en-cours○ Gérer les flux○ Rendre explicites toutes les règles○ Intégrer la rétroaction dans chaque activité○ Améliorer collectivement, évoluer expérimentalement et seulement sur une base scientifique

25

Start with existing process

The Kanban method does not prescribe a specific set of roles or process steps. The Kanban method starts with existing roles and processes and stimulates continuous, incremental and evolutionary changes to the system. The Kanban method is a change management method.

Agree to pursue incremental, evolutionary change

The organization (or team) must agree that continuous, incremental and evolutionary change is the way to make system improvements and make them stick. Sweeping changes may seem more effective but have a higher failure rate due to resistance and fear in the organization. The Kanban method encourages continuous small incremental and evolutionary changes to your current system.

Respect the current process, roles, responsibilities and titles

It is likely that the organization currently has some elements that work acceptably and are worth preserving. The Kanban method seeks to drive out fear in order to facilitate future change. It attempts to eliminate initial fears by agreeing to respect current roles, responsibilities and job titles with the goal of gaining broader support.

Leadership at all levels

Acts of leadership at all levels in the organization, from individual contributors to senior management, are encouraged.

KANBAN

COMMENT ADAPTER KANBAN AU LOGICIEL

- L'adaptation de Kanban au logiciel semble relativement facile dans les cas de configuration de logiciels existants
 - par exemple, la configuration d'un site de commerce électronique à partir d'un système de référence.
- Elle est nettement moins évidente dans les cas de logiciels innovants ou spécifiques, puisque la base existante est réduite
 - voir le premier principe.

SYNTHÈSE

Et le gagnant est...

- Processus dérivé de la spirale
- XP
- Scrum
- Kanban
- ?

Comment réagissez-vous à ceci?				SYNTHÈSE
Facteur	SP	XP	SC	
Gestion				SP: Spirale XP: Programmation extrême SC: Scrum --
Simplicité	A	B	A+	
Expérience requise	G	M	M+	
Implication du client	DC	PR	CO	A: Excellent B: Très bien C: Bien D: Passable E: Insuffisant --
Suivi contractuel	A+	A	B	
Projet				
Taille	M-G	P-M	P	
Durée totale optimale	B	B+	A+	PI: ponctuelle, initiale mais intensive
Variabilité de la durée totale	B	B+	A	
Coût global optimal	C	B	A-	PR: ponctuelle, répétée mais pas toujours intensive
Variabilité du cout total	B+	C	A	DR: discontinue, répétée et d'intensité variable
Tolérance aux variations				DC: discontinue, répétée et d'intensité variable
Exigences	A+	B-	A	CO: continue
Ressources humaines	A-	B+	A	
Risques technologiques	B	B+	A+	
Produit fini				
Modifiabilité	A+	A-	A	--
Qualité architecturale	A	C+	B	M: moyen G: grand P: petit
Qualité globale	A	A-	B+	• : toutes tailles
Correctitude	B+	A	C	--
Validité	A-	A+	A	? : pas de consensus
Adéquation	A-	B	A+	

C : en cascade

CI : cascade itératif

C : en V

R : RUP

P : ponctuelle, initiale mais intensive

PR : ponctuelle, répétée mais pas toujours intensive

DC : discontinue, répétée et d'intensité variable

Et à ceci ? Faites votre propre grille!				SYNTHÈSE
Facteur	SP	XP	SC	
Gestion				SP: Spirale XP: Programmation extrême SC: Scrum --
Simplicité	B	B	A+	
Expérience requise	M+	G	G	
Implication du client	DC	PR	CO	A: Excellent B: Très bien C: Bien D: Passable E: Insuffisant --
Suivi contractuel	A+	A	B	
Projet				
Taille	M-G	P-M	P	
Durée totale optimale	B	B+	NA	PI: ponctuelle, initiale mais intensive
Variabilité de la durée totale	B	B+	NA	
Coût global optimal	B+	B	NA	PR: ponctuelle, répétée mais pas toujours intensive
Variabilité du cout total	B+	C	NA	DR: discontinue, répétée et d'intensité variable
Tolérance aux variations				DC: discontinue, répétée et d'intensité variable
Exigences	A	A-	A	CO: continue
Ressources humaines	A-	A+	C	--
Risques technologiques	A+	B+	A+	M: moyen G: grand P: petit • : toutes tailles -- ?: pas de consensus
Produit fini				
Modifiabilité	A+	B	C-	
Qualité architecturale	A+	B	C	
Qualité globale	A	B	C	
Correctitude	A	A	C	
Validité	A	B+	B	
Adéquation	A	A	A	

C : en cascade

CI : cascade itératif

C : en V

R : RUP

P : ponctuelle, initiale mais intensive

PR : ponctuelle, répétée mais pas toujours intensive

DC : discontinue, répétée et d'intensité variable

SYNTHÈSE XP VS SCRUM

XP

- Travail en tandem
- Cas d'utilisation au démarrage
- Planification KISS
- Itération typique de 2 semaines
- Développement des essais avant la conception
- Conception par prototypage
- Codage des tests unitaires avant celui des modules

Scrum

- Travail en symbiose
- Un objectif fonctionnel à la fois
- Peu de planification
- 5 à 20 itérations de 24 heures par cycle de «stabilisation»
- Livraison interne à la fin de chaque itération
- Livraison externe à la fin de chaque cycle
- «Scrum» tous les matins (15 à 30 minutes)
- Développement par effet tunnel

AGILITÉ CONCEPTS

- L'agilité n'est pas tant un procédé qu'une façon de l'appliquer.
- Un processus synthétique est «agile» lorsque l'équipe qui le réalise adhère au manifeste agile et en pratique les 12 adages.
- Le texte des deux diapositives suivantes est tiré de <http://agilemanifesto.org/iso/fr/manifesto.html>. (sauf les annotations en orange)

AGILITÉ MANIFESTE

«Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser:

- Les individus et leurs interactions plus que les processus et les outils.
- Des logiciels opérationnels plus qu'une documentation exhaustive.
- La collaboration avec les clients plus que la négociation contractuelle.
- L'adaptation au changement plus que le suivi d'un plan.

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.»

AGILITÉ

COMMENTAIRES

«Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser:

- Les individus et leurs interactions plus que les processus et les outils. *Les processus agiles sont pourtant ceux qui requièrent et consomment le plus d'outillage.*
- Des logiciels opérationnels plus qu'une documentation exhaustive. *Comment maintenir l'adéquation, la responsabilité et l'imputabilité?*
- La collaboration avec les clients plus que la négociation contractuelle. *Oui, tant que tout le monde, il est beau et gentil.*
- L'adaptation au changement plus que le suivi d'un plan. *Plus communément, n'appelle-t-on pas cela la fuite en avant?*

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.»

Outillage

Non seulement cet outillage nécessite-t-il qu'on lui consacre énormément de ressources tant pour le maintien, l'évolution que la formation, mais

- * il peut induire des contraintes contraires aux exigences ;
- * il peut procurer un sentiment de «confiance indue» et entraîner un laisser-aller, une déresponsabilités, une perte de motivation.

Documentation

Rappel sur l'adéquation :

- * caractéristiques absolues
 - cohérence,
 - validité,
 - efficacité ;
- * caractéristiques relatives
 - complétude,
 - efficience,
 - évolutivité ;
- * méta-caractéristiques
 - réfutabilité,
 - acceptabilité éthique.

AGILITÉ

ADAGES

- **1. Objectif :**
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- **2a. Règle :**
Welcome changing requirements, even late in development.
- **2b. Affirmation (non démontrée) :**
Agile processes harness change for the customer's competitive advantage.
- **3. Règle :**
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- **4. Règle :**
Business people and developers must work together daily throughout the project.
- **5. Règle :**
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- **6. Affirmation (non démontrée) :**
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- **7. Axiome :**
Working software is the primary measure of progress.
- **8. Objectif :**
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- **9. Affirmation (non démontrée) :**
Continuous attention to technical excellence and good design enhances agility.
- **10. Affirmation (non démontrée) :**
Simplicity – the art of maximizing the amount of work not done – is essential.
- **11. Affirmation (non démontrée) :**
The best architectures, requirements, and designs emerge from self-organizing teams.
- **12. Règle :**
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

AGILITÉ

ADAGES (ORDONNÉS ET TRANSPOSÉS)

<ul style="list-style-type: none"> ○ 0. Objectif : Satisfy the customer through early and continuous delivery of valuable software. ○ 1. Sous-objectif : Achieve sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. ○ 2. Axiome : Working software is the primary measure of progress. ○ 3. Affirmation (non démontrée) : Agile processes harness change for the customer's competitive advantage. ○ 4. Affirmation (non démontrée) : The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. ○ 5. Affirmation (non démontrée) : Continuous attention to technical excellence and good design enhances agility. ○ 6. Affirmation (non démontrée) : Simplicity – the art of maximizing the amount of work not done – is essential. ○ 7. Affirmation (non démontrée) : The best architectures, requirements, and designs emerge from self-organizing teams. 	<ul style="list-style-type: none"> ○ 8 Règle : Welcome changing requirements, even late in development. ○ 9. Règle : Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. ○ 10. Règle : Business people and developers must work together daily throughout the project. ○ 11. Règle : Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. ○ 12. Règle : At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
--	---

2025-01-21
 PR002: Processus agiles (2310) – L. Lavoie, C. Kinnaird
 Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

36

0. C'est un objectif louable, mais ce n'est pas une panacée (il est généralement insuffisant). Ce n'est peut-être même pas toujours souhaitable (d'autres considérations peuvent être nécessaires et empêcher l'atteinte de cet objectif).

1. Nul n'est contre la vertu, malheureusement les lois de la physique prévalent.

2. Cet axiome n'est-il pas contredit en pratique ? Encore faudrait-il définir « working software ». Au fait, Turing n'a-t-il pas démontré que c'était incalculable ?

3. Qu'en est-il des logiciels hors du champ commercial (domaine des services en général), sans « avantage compétitif » ?

4. Pourquoi une grande partie des membres ne sont-ils pas d'accord ? Qui a démontré cela ?

5. Plausible. Encore faudrait-il définir « agilité ». Ces commentaires et ces adages ne sont pas une définition.

6. La définition de simplicité est très étonnante, à croire qu'elle a été « inventée » pour ce manifeste en contradiction avec les définitions usuelles.

7. Affirmation gratuite, non soutenue par l'expérience et la littérature. Par exemple OSI et TCP/IP

8. Une attitude responsable prenant en compte les conséquences est requise... pas la pensée magique.

9. Quel manque de considération pour les utilisateurs et les usagers qui en subissent les conséquences !

10. Encore une fois, en faisant référence explicitement à des « business people », les logiciels de service et les logiciels critiques sont totalement ignorés.

11. Encore la pensée magique !

12. C'est souhaitable, voire nécessaire, mais ce n'est pas suffisant.

VOCABULAIRE USUEL



- Risques
- Essais d'acceptation
- Essais et tests d'intégration
- Essais et tests de non-régression
- Essais et tests unitaires

RÉFÉRENCES



- Pressman, chapitre 3 et 4
- Leffingwell, chapitre 3

