Manifeste pour le développement Agile de logiciels (2001)

4 Valeurs du manifeste AGILE

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 produits opérationnels plus qu'un rapport exhaustif
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.

12 Principes du manifeste AGILE

1/ Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.

2/ Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.

3/ Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.

4/ Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

5/ Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.

6/ La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.

7/ Un logiciel opérationnel est la principale mesure d'avancement.

8/ Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.

9/ Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.

10/ La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.

11/ Les meilleures architectures, spécifications et conceptions émergent d'équipes auto organisées.

12/ À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

4 Valeurs du manifeste AGILE

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 product 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

In french : Manifeste pour le développement Agile de logiciels

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 produits opérationnels plus qu'un rapport exhaustif
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.

12 Principes du manifeste AGILE

1/ Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.

2/ Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.

3/ Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.

4/ Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

5/ Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.

6/ La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.

7/ Un logiciel opérationnel est la principale mesure d'avancement.

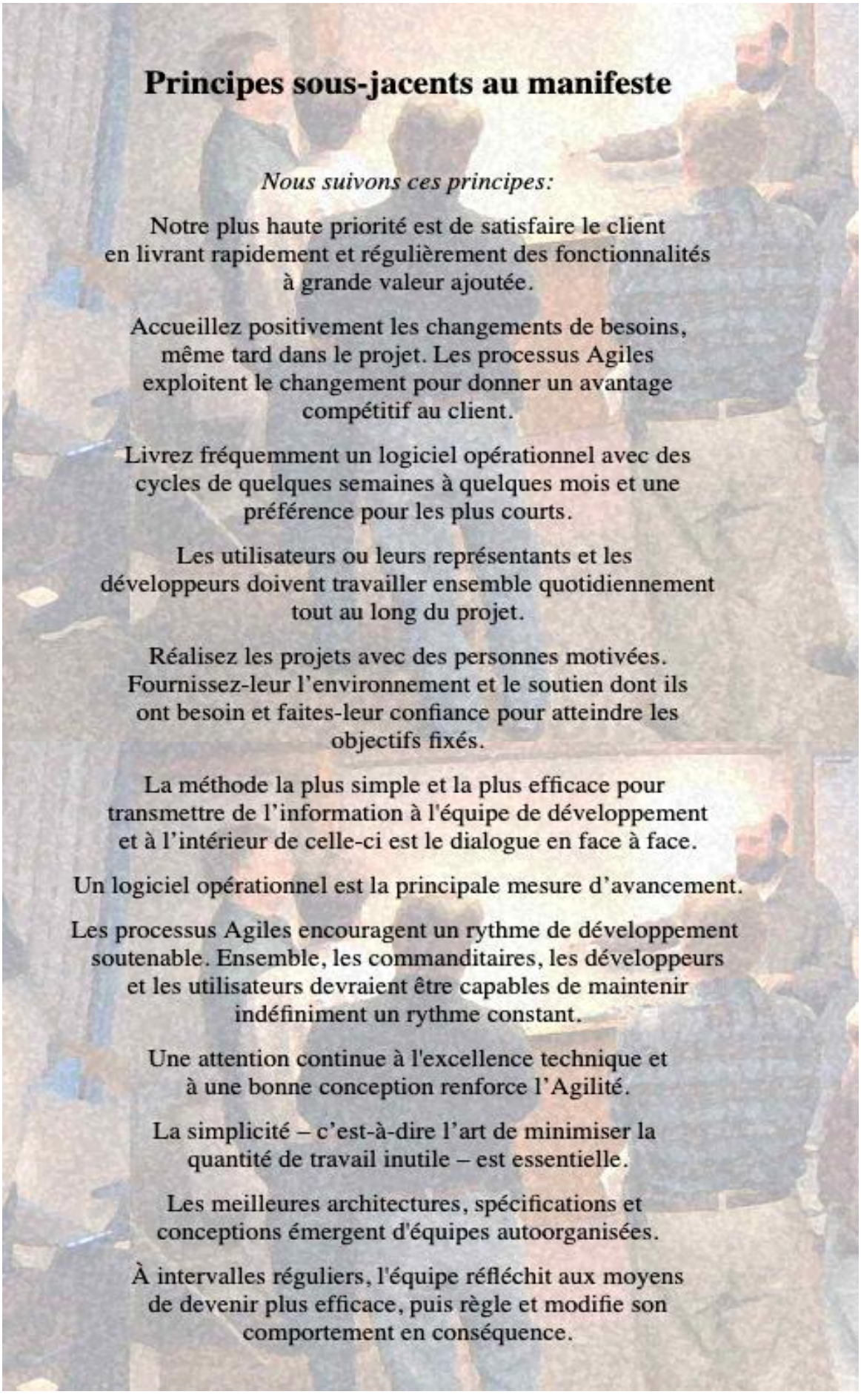
8/ Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.

9/ Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.

10/ La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.

11/ Les meilleures architectures, spécifications et conceptions émergent d'équipes auto organisées.

12/ À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.



Principes sous-jacents au manifeste

Nous suivons ces principes:

Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.

Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.

Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.

Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.

La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.

Un logiciel opérationnel est la principale mesure d'avancement.

Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.

Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.

La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.

Les meilleures architectures, spécifications et conceptions émergent d'équipes autoorganisées.

À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

History: The Agile Manifesto

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground—and of course, to eat. What emerged was the Agile 'Software Development' Manifesto. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

Now, a bigger gathering of organizational anarchists would be hard to find, so what emerged from this meeting was symbolic—a Manifesto for Agile Software Development—signed by all participants. The only concern with the term agile came from Martin Fowler (a Brit for those who don't know him) who allowed that most Americans didn't know how to pronounce the word 'agile'.

Alistair Cockburn's initial concerns reflected the early thoughts of many participants. "I personally didn't expect that this particular group of agilites to ever agree on anything substantive." But his post-meeting feelings were also shared, "Speaking for myself, I am delighted by the final phrasing [of the Manifesto]. I was surprised that the others appeared equally delighted by the final phrasing. So we did agree on something substantive."

Naming ourselves "The Agile Alliance," this group of independent thinkers about software development, and sometimes competitors to each other, agreed on the Manifesto for Agile Software Development displayed on the title page of this web site.

But while the Manifesto provides some specific ideas, there is a deeper theme that drives many, but not all, to be sure, members of the alliance. At the close of the two-day meeting, Bob Martin joked that he was about to make a "mushy" statement. But while tinged with humor, few disagreed with Bob's sentiments—that we all felt privileged to work with a group of people who held a set of compatible values, a set of values based on trust and respect for each other and promoting organizational models based on people, collaboration, and building the types of organizational communities in which we would want to work. At the core, I believe Agile Methodologists are really about "mushy" stuff—about delivering good products to customers by operating in an environment that does more than talk about "people as our most important asset" but actually "acts" as if people were the most important, and lose the word "asset". So in the final analysis, the meteoric rise of interest in—and sometimes tremendous criticism of—Agile Methodologies is about the mushy stuff of values and culture.

For example, I think that ultimately, Extreme Programming has mushroomed in use and interest, not because of pair-programming or refactoring, but because, taken as a whole, the practices define a developer community freed from the baggage of Dilbertesque corporations. Kent Beck tells the story of an early job in which he estimated a programming effort of six weeks for two people. After his manager reassigned the other programmer at the beginning of the project, he completed the project in twelve weeks—and felt terrible about himself! The boss—of course—harangued Kent about how slow he was throughout the second six weeks. Kent, somewhat despondent because he was such a "failure" as a programmer, finally realized that his original estimate of 6 weeks was extremely accurate—for 2 people—and that his "failure" was really the manager's failure, indeed, the failure of the standard "fixed" process mindset that so frequently plagues our industry.

This type of situation goes on every day—marketing, or management, or external customers, internal customers, and, yes, even developers—don't want to make hard trade-off decisions, so they impose irrational demands through the imposition of corporate power structures. This isn't merely a software development problem, it runs throughout Dilbertesque organizations.

In order to succeed in the new economy, to move aggressively into the era of e-business, e-commerce, and the web, companies have to rid themselves of their Dilbert manifestations of make-work and arcane policies. This freedom from the inanities of corporate life attracts proponents of Agile Methodologies, and scares the begeebers (you can't use the word 'shit' in a professional paper) out of traditionalists. Quite frankly, the Agile approaches scare corporate bureaucrats— at least those that are happy pushing process for process' sake versus trying to do the best for the "customer" and deliver something timely and tangible and "as promised"—because they run out of places to hide.

The Agile movement is not anti-methodology, in fact, many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker.

The meeting at Snowbird was incubated at an earlier get together of Extreme Programming proponents, and a few "outsiders," organized by Kent Beck at the Rogue River Lodge in Oregon in the spring of 2000. At the Rogue River meeting attendees voiced support for a variety of "Light" methodologies, but nothing formal occurred. During 2000 a number of articles were written that referenced the category of "Light" or "Lightweight" processes. A number these articles referred to "Light methodologies, such as Extreme Programming, Adaptive Software Development, Crystal, and SCRUM". In conversations, no one really liked the moniker "Light", but it seemed to stick for the time being.

In September 2000, Bob Martin from Object Mentor in Chicago, started the next meeting ball rolling with an email; "I'd like to convene a small (two day) conference in the January to February 2001 timeframe here in Chicago. The purpose of this conference is to get all the lightweight method leaders in one room. All of you are invited; and I'd be interested to know who else I should approach." Bob set up a Wiki site and the discussions raged.

Early on, Alistair Cockburn weighed in with an epistle that identified the general disgruntlement with the word 'Light': "I don't mind the methodology being called light in weight, but I'm not sure I want to be referred to as a lightweight attending a lightweight methodologists meeting. It somehow sounds like a bunch of skinny, feebleminded lightweight people trying to remember what day it is."

The fiercest debate was over location! There was serious concern about Chicago in wintertime—cold and nothing fun to do; Snowbird, Utah—cold, but fun things to do, at least for those who ski on their heads like Martin Fowler tried on day one; and Anguilla in the Caribbean—warm and fun, but time consuming to get to. In the end, Snowbird and skiing won out; however, a few people—like Ron Jeffries—want a warmer place next time.

We hope that our work together as the Agile Alliance helps others in our profession to think about software development, methodologies, and organizations, in new– more agile – ways. If so, we've accomplished our goals.

Jim Highsmith, for the Agile Alliance