Documentation Agile : pratiques actuelles et défis

Asma HACHEMI
Département d'Informatique
USTHB
Alger, Algérie
asma_hachemi@yahoo.fr

Mohamed AHMED-NACER
Département d'Informatique
USTHB
Alger, Algérie
anacer@mail.cerist.dz

***Résumé*-La documentation une est partie prenante dans un projet logiciel. Elle pérennise des informations sur tous les aspects du logiciel, et joue un rôle important dans la qualité de ce dernier. Toutefois, les méthodes de documentation existantes ne conviennent pas aux projets Agiles, qui favorisent le changement et livrent très rapidement. Donc, une documentation convenable à ces projets, qui est la documentation Agile devient un impératif. Ainsi, nous donnons à travers cet article un aperçu sur les pratiques actuelles et les défis de la documentation Agile ; pour mettre l'accent sur cette dernière mais aussi sur les lacunes qui demeurent existantes dans ce domaine.**

Mot-clés : ***Documentation Agile ; documentation de logiciels ; méthodes Agiles ;***

I. INTRODUCTION

Le code source est le principal artefact du procédé de développement logiciel, mais ce code n'existe jamais seul. Il évolue dans un environnement constitué de différents autres artefacts, produits ou requis lors du développement, de l'exploitation ou de la maintenance du logiciel. Destinés à plusieurs types d'utilisation et offrants multiples intérêts, ces artefacts pérennisent des informations sur tous les aspects du logiciel, et représentent la documentation de ce dernier.

Le manque de documentation peut conduire à des problèmes tels que : dépenser beaucoup de temps à répondre aux mêmes questions, rencontrer les mêmes problèmes sans se rappeler la solution, perdre une importante connaissance avec le départ des membres de l'équipe [1] ; et peut conduire à la perte de la mémoire de l'entreprise [2].

La documentation de logiciels est donc une parti prenante de tout procédé logiciel, particulièrement des procédés Agiles qui sont en plein essor. En effet, ces procédés offrent une formidable voie pour gérer des cycles de développement rapides [3]. Ces procédés ont gagné beaucoup de partisans [4], et les projets utilisant les méthodes Agiles sont en train de rapporter des améliorations dans le temps et le coût, par rapport à ceux utilisant des méthodes classiques [5]. Le feedback en provenance de ces projets est généralement positif [6].

Les méthodes Agiles ont leur particularités, et une documentation qui convient aux projets Agiles est par conséquent particulière. Il s'agit de la documentation Agile, qui est une documentation souple et efficace. Ainsi, cet article donne un aperçu sur les pratiques actuelles et les défis de la documentation Agile. Il est structuré comme suit : il commence par définir la documentation de logiciels en section II. La section III présente les inconvénients de cette documentation. La documentation Agile est présentée, et son état actuel ainsi que ses challenges font l'objet de le section IV. Enfin, une conclusion est donnée en section V.

II. DOCUMENTATION DE LOGICIELS

La documentation de logiciels réfère tout artefact dont le but est de communiquer des informations sur le logiciel [7]. Elle joue un rôle essentiel dans la communication [8] ; permet de répéter les succès et d'éviter de refaire les erreurs ; permet aux compagnies de mesurer leurs performances courantes et initialiser des actions d'amélioration [9].

Depuis longtemps, des données empiriques ont montré que la documentation est un composant clé dans la qualité du logiciel [10] [11] [12]. Ces études révèlent qu'une documentation dépassée, incomplète ou de qualité médiocre est une cause majeure des erreurs dans le développement et la maintenance du logiciel [13]. Une solution à ces problèmes est l'amélioration du procédé de documentation [13]. Ainsi, l'idée d'intérêt du procédé de documentation malgré qu'elle soit ancienne, demeure un sujet d'actualité [14].

Dés lors, nombreux chercheurs se sont penchés sur le thème de la documentation, pour apporter des nouveautés, des solutions et des améliorations. Sont cités dans ce qui suit quelques travaux dans ce domaine :

- Des normes et des projets de normes ISO ont abordé la documentation de logiciels [15] [16] [17] [18], et continuent à émerger pour apporter un plus à cet élément critique des projets logiciels.

- Similairement, des guidages concernant la documentation de logiciels sont donnés par des standards et projets de standards de l'IEEE [19] [20] [21].

- D'autres travaux de recherche s'intéressent à des problèmes de plus en plus accrus des procédés de documentation, tels que [22], [23], [24], [25], [26].

- Certains chercheurs se penchent sur la structuration et catégorisation des documents, tels que [27], [28].

- D'autres se penchent sur la synchronisation de la documentation avec le code source [29] [30] [31], ou bien la synchronisation des différents documents entre [32], [33].

- La présentation du contenu des documents est abordée par des travaux tels que [34], [35], [36].

- Des recherches sur les supports de documentation continuent à se faire [37], [30]. Alors que l'exploitation de la documentation pour les phases Test [38] et Maintenance [39] [36], ainsi que la génération automatique de documents [40] [41] [27] font aussi l'objet de nombreux travaux.

III. LACUNES DE LA DOCUMENTATION DE LOGICIELS

Les méthodes de documentation classiques (non Agiles), telles que [42] qui montre de quoi est constituée cette documentation et le procédé pour sa production, sont complètes du point de vu couverture de tous les besoins en matière de documentation ; mais sont lourdes à manipuler et consommatrices de temps et d'effort. Les méthodes courantes de documentation souffrent de plusieurs lacunes :

- Dans la majorité des projets, une méthode de documentation nécessite une équipe dédiée pour pouvoir être mise en place. Ainsi, une grande proportion du cout du procédé de développement est induite par la production de documentation.

- Les documents générés par ces méthodes sont nombreux et volumineux. Les erreurs et les incohérences sont ainsi multiples.
- La mise à jour de la documentation en même temps que la mise à jour du logiciel est souvent négligée dans les méthodes de documentation.

- Une méthode de documentation appliquée lors d'un projet suggère la création d'un ensemble standard de documents ; sans vérifier pour chaque document son utilité par rapport au projet en cours.

- Certains documents sont exigés dans une méthode juste pour le besoin de communication ; alors que d'autre moyens doivent être privilégiés pour communiquer (les outils de travail collaboratif, les réunions face à face, les conférences vidéo ou téléphoniques…) [43].

- Enfin, ces méthodes de documentation ne conviennent pas aux procédés Agiles qui favorisent le changement et livrent très rapidement. En effet, ces méthodes livrent la documentation à chaque fin d'itération, cette documentation doit être synchronisée avec le logiciel. De plus, ces méthodes traitent de nombreux changements, qui doivent se refléter sur la documentation.

IV. DOCUMENTATION AGILE

A. Méthodes Agiles

Une méthode Agile est une approche de développement de logiciels itérative et incrémentale. « Elle génère un produit de haute qualité, tout en prenant en compte l'évolution des besoins du client » [44] [60] [61] [62] [63]. La notion de méthode Agile a été officialisée en 2001 par un manifeste [45], mais les méthodes Agiles étaient antérieures à ce manifeste qui n'est que la formalisation consensuelle de ces méthodes.

L'essor des méthodes Agiles souligne leur intérêt. Ces dernières ont gagné beaucoup de partisans et beaucoup d'entreprises ont commencé à les adopter [4]. Les projets utilisant les méthodes Agiles sont en train de rapporter des améliorations dans le temps et le coût, par rapport à ceux utilisant des méthodes classiques [5], et le feedback en provenance de ces projets est généralement positif [6].

La gestion de projets Agiles a pour objectif de développer le logiciel en se basant sur les exigences du client, de manière itérative, simple et basée sur l'expérience de l'équipe [46]. L'agilité c'est l'efficience au futur immédiat [47]. Les méthodes

Agiles offrent un retour sur investissement rapide, avec un investissement inférieur et un gain supérieur par rapport aux méthodes classiques [48]. Toutefois, un des principaux écueils qui peuvent être rencontrés avec ce type de méthodes, est la frontière ténue entre l'application d'une méthode Agile et le « n'importe quoi » qui s'installe quand on laisse travailler une équipe sans méthode prédictive et figée (une méthode traditionnelle). En effet, mettre en place une méthode Agile nécessite que tous les acteurs soient impliqués dans le processus, connaissent les avantages et acceptent de jouer le jeu.

B. Documentation Agile

1) Définition

La documentation Agile est comme son nom l'indique une documentation souple, qui convient aux méthodes Agiles. Produire une documentation Agile c'est documenter de façon intelligente, appropriée et précise en s'appuyant sur ce dont on a réellement besoin [49]. Ainsi, une méthode de documentation qui convient aux projets Agiles suggère de produire juste ce qu'il faut comme documents, au bon moment et pour une audience ciblée [50] ; ce qui résulte en une documentation Agile.

L'interrogation qui se pose autour de la documentation Agile concerne son intérêt, puisque le manifeste agile favorise un logiciel qui marche à la documentation [45]. En effet, il est plus efficace d'avoir un logiciel qui fonctionne pouvant être objectivement évalué, que d'avoir une documentation volumineuse qui le décrit [7]. Mais aujourd'hui, peu de projets peuvent s'en passer d'une documentation précise et adaptée [49], et les connaisseurs de l'industrie des logiciels savent que les arguments disant que la documentation est superflue sont faux.

2) Spécificités de la documentation Agile

La documentation dans les projets Agiles se base sur le principe « *Traveling Light* » [51] [52], qui signifie créer juste le minimum de documents que requiert le bon fonctionnement du projet et du logiciel. Ainsi, dans les projets Agiles les spécificités suivantes s'imposent sur la documentation [43] :

- Les documents Agiles sont assez simples et contiennent juste suffisamment d'information pour remplir leurs objectifs.
- Un document Agile traite un seul sujet bien précis. Si l'objectif du document n'est pas clair ou est contestable, alors il faudra repenser la création de ce document.
- Un document Agile décrit les *bonnes informations*, celles qui sont critiques ou non évidentes.
- Un document Agile a un lecteur ciblé. Ainsi, il faut œuvrer à capturer et communiquer l'information à ce lecteur en utilisant le meilleur moyen.
- Un document Agile n'a pas besoin d'être parfait, il a juste besoin d'être assez bien en étant précis et cohérent.
- Une règle empirique est de ne produire aucun document jusqu'a en avoir réellement besoin, et jusqu'à ce que l'information soit stable ; pour éviter de produire un document qui s'avèrera plus tard inutile ou qui devra être changé.
- Dans un projet Agile, ne documenter que si c'est le seul moyen de communiquer. La documentation doit rester le dernier recourt au lieu d'être le choix préféré.
- Une mise à jour de documents ne doit être opérée que si elle s'avère absolument nécessaire « Update only when it hurts ».
- Il faut faire collaborer le développeur et le rédacteur technique pour rédiger une documentation Agile, car les développeurs maitrisent le sujet rédigé alors que les rédacteurs ont la compétence de rédaction.
- Il faut faire un compromis sur le niveau d'expérience des développeurs impliqués dans la rédaction d'une documentation Agile ; car un développeur non expérimenté rédige des informations superflues, par contre un développeur très expérimenté rédige très peu d'informations.

C. Défis de la documentation Agile

Parmi les challenges de la documentation Agiles se trouve le fait que la quantité et les types de documents diffèrent d'un projet à un autre [53], et que le besoin en documentation n'est pas connu avec exactitude à priori. Aussi, le logiciel est livré à chaque fin d'itération avec toute la documentation qui l'accompagne. De plus, toute documentation rédigée doit évoluer dans le temps et être synchronisée avec le logiciel.

En réalité la plus grande difficulté de la documentation Agile, est qu'il n'existe pas un procédé concret pour le développement de documents dans les projets Agiles. Les auteurs des principales méthodes Agiles n'ont pas été très précis à propos de la documentation. Ils se sont

contentés de donner des idées atour du sujet, sans fournir de procédures concrètes pour la production de documentation Agile. Ces idées sont récapitulées dans ce qui suit :

- Cockburn dans son ouvrage *Agile Software Development* [54] introduit la famille des méthodes Crystal, utilisées par des projets de différentes tailles et de différentes criticités. Les méthodes Crystal requièrent la création d'une documentation, mais laissent chaque projet décider de quoi sera constituée cette documentation.
- Highsmith dans son livre *Agile Software Developpement Ecosystems* [55] avertit de ne pas produire une documentation juste pour documenter, mais plus tôt de produire une documentation modérée qui facilite la communication, améliore le transfert de connaissances et préserve l'historique.
- L'idée véhiculée par la méthode XP [56] concernant la documentation est la suivante : « *agile models are paintings, not photographs* ». elle signifie : ne pas tout décrire et avec détails (comme dans une photographie), par contre ne parler que d'idées importantes avec juste ce qu'il faut comme informations (comme dans une peinture).
- La méthode FDD [57] contient cinq procédés, chacun est constitué de plusieurs taches obligatoires ou optionnelles. Certaines taches ont pour objectif de produire des documents, ce qui sous entend que la méthode préconise un ensemble de documents obligatoires ou optionnels à produire. Toutefois, les documents générés par la méthode se limitent à ceux requis pour l'avancement du projet en cours, et aucune règle n'est imposée pour les autres types de documents.
- Dans la méthode RAD [58] on préconise une multitude de dossiers à fournir lors d'un projet, ainsi que les documents qui les constituent ; mais sans préciser quels sont les documents obligatoires et ceux optionnels. Ceci rappelle les méthodes de documentation classiques qui recommandent un maximum de documents (pour couvrir tous les besoins du projet), mais qui sont difficiles à gérer et à exploiter.
- Scrum [59] offre quelques guidages sur la documentation. Les bases de cette documentation sont les dossiers de Produit, de Version et de Sprint. Mais aucun détail n'est fournit concernant la production ou la gestion de ces dossiers.

De plus, des problèmes récurrents restent posés dans le domaine de la documentation Agile, qui sont inhérents à la documentation de logiciels. Par exemple, le besoin d'accumuler toutes les informations concernant un thème, mais aussi de les structurer de manière adéquate dans un document ; ou la gestion des versions et l'archivage de la documentation, une activité qui doit absolument être gérée.

V. CONCLUSION

Un aspect important des projets logiciels est abordé à travers cet article, qui est la documentation de logiciels. Cette dernière permet de rendre l'information disponible et de pérenniser la connaissance pour des utilisations futures.

Cependant, les approches de documentation classiques sont complètes mais lourdes, et souvent des documents sont créés pour des causes non pertinentes. L'expérience a montré qu'un maximum de documents est aussi néfaste pour un projet qu'un manque de documents, vu que cette multitude de documentation influe négativement sur la qualité de cette dernière et sur la facilité de son exploitation : documents incohérents, difficilement mis à jour et difficilement exploitables.

Les méthodes Agiles ont prouvé leur intérêts et sont de plus en plus adoptées. La documentation Agile est par conséquent de plus en plus requise. Il s'agit d'une documentation qui convient aux projets Agiles, et qui est elle-même Agile. La valeur du manifeste Agile qui préconise « un logiciel fonctionnel plutôt qu'une documentation complète », ne signifie pas absence de documentation. C'est souvent une mauvaise interprétation de cette valeur qui motive certaines équipes à ne pas documenter, mais aussi un manque de volonté (voir de compétence) de certaines équipes soumises à un timing toujours plus serré.

Du fait, des moyens qui permettent de créer et gérer la documentation Agile s'avèrent nécessaires, et des solutions doivent venir en aide pour mettre efficacement en œuvre cette approche de documentation.

BIBLIOGRAPHIE

[1] H. Holz, F. Maurer, "Knowledge Management Support for Distributed Agile Software Processes," Advances in Learning Software Organizations, Lecture Notes in Computer Science, 2003, Volume 2640/2003, 60-80.

[2] D. Turk, R. France, and B. Rumpe, "Limitations of Agile Software Processes," Proceedings of the 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering, May 2002.

[3] M. Pikkarainen, U. Passoja, "An Approach for Assessing Suitability of Agile Solutions: A Case Study," 6th International conference of eXtreme Programming and

agile process in software engineering, 18-23 June, Sheffield University, UK, 2005.

[4] S. Nerur and V.l Balijepally, “Theoretical reflections on agile development methodologies,” Commun. ACM, 50(3):79– 83, 2007.

[5] J. Charvat, Project Management Methodologies—Selecting, Implementing, and Supporting Methodologies and Processes for Projects, John Wiley & Sons, Inc. 2003.

[6] L. Vijayasarathy, D. Turk, “Agile software development: a survey of early adopters,” Journal of Information Technology Management Volume XIX, Number 2, 2008.

[7] A. Forward, T. Lethbridge, “The Relevance of Software Documentation, Tools and Technologies: A Survey,” ACM symposium in document engineering, 2002.

[8] L. Garceau and E. Jancura, “Documentation and Training as a Systems Development Tool”, The CPA Journal, 60(11), 1990, pp. 84-89.

[9] G. Coleman and R. Verbruggen, “A Quality Software Process for Rapid Application Development”, Software Quality Journal, 7(2), 1998, pp. 107-122.

[10] N. D. Card, E. F. McGarry, and G. T. Page, "Evaluating software engineering technologies," IEEE Transactions on Software Engineering, Vol. Se-13,No. 7, 1987.

[11] B. Lientz, E. Burton Swanson, "Problems in applications software maintenance," Communications of the ACM, November 1981.

[12] H.D. Rombach and V.R. Basili, "Quantitative assessment of maintenance: an industrial case study," IEEE Proceedings of Conference on Software Maintenance, City Press, 1987

[13] C. Cook, M. Visconti, DOCUMENTATION IS IMPORTANT, crosstalk, Nov 1994.

[14] M.F.F. Nasution, H.R. Weistroffer, “Documentation in Systems Development: A Significant Criterion for Project Success,” IEEE CONFERENCES HICSS '09. 42nd Hawaii International Conference on System Sciences, 2009.

[15] Ingénierie du logiciel et des systèmes — Exigences pour les concepteurs et les développeurs de la documentation de l'utilisateur, ISO/IEC 26514:2008

[16] Ingénierie des systèmes et du logiciel -- Exigences pour testeurs et vérificateurs de documentation utilisateur, ISO/IEC 26513:2009

[17] Ingénierie du logiciel et des systèmes -- Exigences pour les managers de la documentation de l'utilisateur, ISO/IEC WD 26511, Current stage date: 2009-05-23

[18] Ingénierie du logiciel et des systèmes -- Exigences pour les acheteurs et les fournisseurs de la documentation de l'utilisateur, ISO/IEC FCD 26512, Current stage date: 2010-02-15

[19] IEEE Standard for Software and System Test Documentation, IEEE Std 829-2008

[20] IEEE Draft Standard: Systems and software engineering - Requirements for acquirers and suppliers of user documentation, Dec 2009

[21] IEEE Draft Standard XSoftware and systems engineering - - Content of life-cycle information products (documentation), Dec 2009

[22] Isaac Sánchez-Rosado, Pablo Rodríguez-Soria, Borja Martín-Herrera, Juan José Cuadrado-Gallego, Javier Martínez-Herráiz and Alfonso González, “Assessing the Documentation Development Effort in Software Projects,” Lecture Notes in Computer Science, Software Process and Product Measurement, Springer, Nov 2009.

[23] U. Zdun, “Guest Editor's Introduction: Capturing Design Knowledge,” Software IEEE Volume : 26 , Issue:2 March-April 2009.

[24] J. Holdaway, “Technical reviews: Framing the best of the best practices,” IEEE International Professional Communication Conference, 2009.

[25] A.L. Bartell, K.A. Brown, “Secrets to managing a large documentation project virtually—process, technology, and group ethos: Lessons learned,”; IEEE International Professional Communication Conference, 2009.

[26] R.L. Nord, P.C. Clements, D. Emery, R. Hilliard, “Reviewing architecture documents using question sets,” Joint Working IEEE/IFIP Conference on Software Architecture, European Conference on Software Architecture, 2009.

[27] M. Heinrich, A. Boehm-Peters, M. Knechtel, “A platform to automatically generate and incorporate documents into an ontology-based content repository,” DocEng '09: Proceedings of the 9th ACM symposium on Document engineering, 2009.

[28] J.K. Kokkoniemi, L. Harjumaa, “From Software Documents to Experience Knowledge Based Artifacts,” 42nd Hawaii International Conference on System Sciences, 2009.

[29] Y. Zhang, R. Witte, J. Rilling, V. Haarslev, “Ontological approach for the semantic recovery of traceability links between software artefacts,” IEEE Software, IET Volume: 2 , Issue: 3, 2008.

[30] Y. Ben-Chaim, E. Farchi, O. Raz, “An effective method for keeping design artifacts up-to-date,” IEEE CONFERENCES ICSE Workshop on Wikis for Software Engineering, 2009.

[31] O. Díaz, F. I. Anfurrutia, J. Kortabitarte, “Using DITA for documenting software product lines,” Proceedings of the 9th ACM symposium on Document engineering, September 2009.

[32] Collin McMillan, Denys Poshyvanyk, Meghan Revelle, “Combining textual and structural analysis of software artifacts for traceability link recovery,” Proceedings of the ICSE Workshop on Traceability in Emerging Forms of Software Engineering, Publisher: IEEE Computer Society, May 2009.

[33] F. Correia, H. Ferreira, N. Flores, A. Aguiar, “Patterns for Consistent Software Documentation,” Pattern Languages of Programs Conference, Chicago, USA, 2009.

[34] S. Tilley, “ Documenting software systems with views VI: lessons learned from 15 years of research & practice,” Proceedings of the 27th ACM international conference on Design of communication, October 2009.

[35] W.J. Dzidek, E. Arisholm, L.C. Briand, “ A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance,” IEEE Transactions on Software Engineering, Volume: 34 , Issue: 3, 2008.

[36] M. Torchiano, F. Ricca, P. Tonella, “Empirical comparison of graphical and annotation-based re-documentation approaches,” IET Software, Volume: 4 , Issue: 1, 2010.

[37] Gotel Olly, Morris Stephen, “More than Just "Lost in Translation",” IEEE Software, Volume: 26 , Issue: 2, 2009.

[38] D. Connolly, F. Keenan, F. McCaffery, “ Developing acceptance tests from existing documentation using annotations: An experiment,” IEEE CONFERENCES ICSE Workshop on Automation of Software Test, 2009.

[39] D. Hyland-Wood, D. Carrington, S. Kaplan, “Towards a software maintenance methodology using Semantic Web techniques and paradigmatic documentation modelling,” IET IEEE Software, Volume: 2 , Issue: 4, 2008.

[40] L. R. Carter, A. Karatsolis, “Lessons from trying to develop a robust documentation exemplar,” Proceedings of the 27th ACM international conference on Design of communication, October 2009.

[41] W. Maalej, H. Happel, “ From work to word: How do software developers describe their work?,” 6th IEEE International Working Conference on Mining Software Repositories, 2009.

[42] I. Sommerville, Software Documentation, Lancaster University, UK, 2001.

[43] S. Ambler, Agile/Lean Documentation: Strategies for Agile Software Development, Ambysoft Inc., 2009. Available at http://www.agilemodeling.com/essays/agileDocumentation.htm

[44] V. M. Rota, Gestion de projet : Vers les méthodes agiles, EYROLLS, 11/2007.

[45] Agile Manifesto, 2001. available at http://agilemanifesto.org/

[46] J. Highsmith, Agile Project Management, Creating innovative products, Addison-Wesley, 2004.

[47] J.Vickoff, AGILE l'Entreprise et ses Projets, QI Editions, 2007.

[48] G. Oldani, Agilité, Hortis, 2006. available at http://social.hortis.ch/wpcontent/uploads/2006/10/Agilit%C3%A9_synth%C3%A8se_v2.doc.

[49] J. GROSJEAN, Documentation Agile : Juste ce qu'il faut, Qualitystreet, 2007. available at http://www.qualitystreet.fr/2007/08/28/documentation-agile-juste-ce-qu-il-faut/

[50] A. Aguiar, "Tutorial on Agile Documentation with Wikis," WikiSym09, Florida, U.S.A, 2009.

[51] J. A. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Dorset House, December 1999.

[52] S. W. Ambler, Agile Modeling – Effective Practices for eXtreme Programming and the Unified Process, John Wiley & Sons, 2002.

[53] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley, 2005.

[54] A. Cockburn, Agile Software Development, Addison-Wesley, 2001

[55] J. Highsmith, Agile Software Development Ecosystems, Addison-Wesley, 2002.

[56] D. Wells, Extreme Programming: A gentle introduction, last modified September 2009. Available at www.extremprogramming.org

[57] Feature Driven Development, Nebulon Pty. Ltd., Available at http://www.featuredrivendevelopment.com

[58] J.Vickoff ; Site historique de la méthode RAD ; Available at www.rad.fr

[59] K.Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice-Hall, Inc., 2002.

[60] B. Boehm, R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley Publishers, USA, 2003.

[61] T. Stober, U. Hansmann, Agile Software Development, Springer-Verlag Berlin Heidelberg, 2010.

[62] D. F. Rico, "WHAT IS THE ROI OF AGILE VS. TRADITIONAL METHODS? An analysis of XP, TDD, Pair Programming, and Scrum (Using Real Options)", TickIT International 2008, Issue: 4, Pages: 9-19.

[63] M. Ganis, "Agile Software Methods (fact or fiction)", TCF2010, New Jersey USA.