(online) = ISSN 2285 – 3642
ISSN-L = 2285 – 3642
Journal of Economic Development, Environment and People
Volume 3, Issue 3, 2014

URL: http://jedep.spiruharet.ro
e-mail: office_jedep@spiruharet.ro

Management of Innovative Projects through Agile Technology

Rocsana Bucea-Manea-Țoniș[1], Radu Bucea-Manea-Țoniș[2]

[1] Spiru Haret University
[2] mysqlbrowser.codeplex.com

Abstract. *Because of the globalization and the evolution of internet and technologies, nowadays the innovation is associated with open collaboration conducted by a legal framework. The paper analyses the methods that allow a better management for innovative projects and focuses on agile projects within a technological network.*

Keywords: open innovation, Extreme Programming (XP), Scrum, agile programming, NIO package

JEL Codes: *M15*

1. Introduction

Firms' managers understood that without innovation they will abandon the road to success. The innovation results in cost-efficient actions and quality manner of programming and collaborating within a network. Although big companies and SMEs innovate in different ways, today trend is open innovation. Open Innovation refers to each stage of the product/service life cycle. Either we consider Extreme Programming (XP), Scrum, or some other agile method, all of them implies open communication between counterparts (firms, clients, policy institution, universities, ecological agencies, consultancy agencies, inventors and marketing agencies) and implements specific principles that have stood the test of time.

2. Agile programming

The key practices that distinguish an agile method from other methods are [Eeles, 2014]

- *Test-Driven Development (TDD):* This practice is about creating tests, which reflect what the code should do and unpunished an efficient rhythm for developers —test, code, refactor, test, code, refactor" [Beck, 2005].
- *Continuous Integration:* This practice talks about frequent integration and testing of programming changes.
- *Refactoring:* This practice is about rewriting an existing body of code as to obtain a better internal structure, updating the initial design of models created.

(online) = ISSN 2285 – 3642
ISSN-L = 2285 – 3642
Journal of Economic Development, Environment and People
Volume 3, Issue 3, 2014

URL: http://jedep.spiruharet.ro
e-mail: office_jedep@spiruharet.ro

- Whole Team: Modern programming mitigates the role of project managers and emphasizes the role of teams' members collaboration, even within a network. The team is flexible and dynamic. They self-organize and adapt accordingly to the project goals, bearing a collective ownership and responsibility.
- *User Story-Driven Development:* This practice describes functional and non-functional requirements such as to be understood by the relevant stakeholders throughout a project, too. The practice refers at many stages of the project: planning, development and testing.
- *Team Change Management:* This practice allows discovering defects and integration of new requirements, by any member of the team.

Additional practices for more complex projects may be:

- *Measured Performance:* Measuring the performance is important to check the project stage in every moment and for continuous adjusting of the methods, practices and key business decisions, for a successful project.
- *Formal Change Management:* The changes has to have a formal control when stakeholders outside have the decision, or when the contract terms are modified.
- *Concurrent Testing:* An external independent test team presents the user-acceptance within the iteration or release

Extreme Programming (XP), divide the project into several iterations, delivered into production in order to get the feedback from the client and increase the project's performance, delivering greater functionality, answering to any unexpected changes or correct problems arising from any misunderstandings about the client's requirements at any stage of the project. Active client involvement and pair programming and collective ownership are more efficient then reading the project documentation associated. The XP practices categories are foundations, customer planning, craftsmanship and XP team factor: [Wood, 2013].

Foundation refers to testing, pair programming and refactoring.

Customer planning refers to planning, customer access, short releases and stand-up meetings.

Craftsmanship refers to sustainable pace, simple design and use of metaphor.

XP team factor refers to continuous integration, coding standards and collective code ownership.

All these factors are related to project performance.

In their case study on 40 small-scale software development teams which used Extreme Programming (XP), the authors [Wood, 2013] found that there is a direct and positive relation between customer planning, the XP-specific team factor and performance while the relation between foundations and performance is negative. This negative relation is explained by the fact that ”The real benefits of foundations are revealed in advantages for the user, and these only materialize after delivery.”

(online) = ISSN 2285 – 3642
ISSN-L = 2285 – 3642
Journal of Economic Development, Environment and People
Volume 3, Issue 3, 2014

URL: http://jedep.spiruharet.ro
e-mail: office_jedep@spiruharet.ro

3. Development environment

In agile programming the changes must be adopted incrementally such that people can assimilate the new information, technology, standards, collaboration techniques, etc.

A platform for innovation based on architecture and agile principles has greater efficiency if the members of the teams implements well the project lifecycle, and adopt incremental changes. (fig. 1.)

- *Method:* The formally or informally method practiced by developers refers to their roles, the tasks and processes, the standards, the guidelines, the templates and examples and the components, the work products used to develop the software.
- *Tools:* Tools are used to automate a sequence of steps from the methods above. There are tools for configure agile task boards, sprint planning, to make velocity charts such as Planio. Continuous integration is more effective if the proper tools for the project are chosen.
- *Infrastructure:* Agile programming requires open collaboration, which implies working within a network, with a specific hardware and software environment. In Open Source they are frequently using Linux operating system, MySql database management system, BOUML to specify and generate code in C++, Java, Php, Python and to design UML diagrams, etc.
- *Enablement:* It refers to training and constant mentoring for developers, professionalization of the staff and adopting external standards, obtain certifications in order to understand and internalize the methods, how to use the tools and infrastructure for a successful project.
- *Organization:* Any project must be managed by a professional organization with specialist in different fields, such as developers, method experts, tool specialists, trainers, system/platform administrators, etc. It is important to know whom you ask if there is a new requirement, a change, a bug or to recommend it for a new project.
- *Adoption:* The firm may establish a center of excellence (CoE), which has the task to facilitate the adoption of work environment, plans, working methods and practices. The effectiveness of the environment has to be measured with specific metrics.
- *Cross Cutting Concerns:* Refers at functionality, qualities, and constraints of the development environment.

Agile puts the focus on functionality of the software, on answering the clients' requirements and on the value of the project/product, through the constant feedback from the clients that may test different improved releases before the final version.

The management of agile project is fundament on collaboration between the project and client team and the organisations concerned. This approach emphasizes the feeling valued of the team members. It also involves sharing knowledge with colleagues to achieve that next breakthrough. [Collins, 2014]

(online) = ISSN 2285 – 3642
ISSN-L = 2285 – 3642
Journal of Economic Development, Environment and People
Volume 3, Issue 3, 2014

URL: http://jedep.spiruharet.ro
e-mail: office_jedep@spiruharet.ro

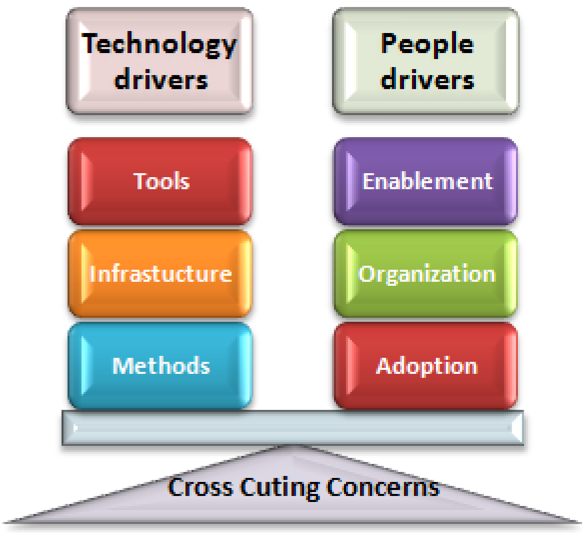

Fig.1. Key development environment elements

In our paper the focus is set on refactoring in agile projects. Refactoring the architecture is an iterative way for implementing variability. There are basically three criteria to decide if an environment is suitable for combining agility and variability: [Galster, 2014]

- Commonalities and differences between products justify systematic handling of variability and not implementing separate, custom-built products.
- The high dynamic of domain changes justify an agile, variability-handling approach
- The complexity or magnitude [McGregor, 2008] of products, including a big team involved in the project, the organization which develops a software system and the client organization.

Agile architecting is often used when the project is characterised of changeability that require flexibility and adaptability. The flexibility and adaptability refers to the ability to deal with changes. In the first case the changes can be either anticipated or planned, and in the second case the can neither be neither anticipated nor foreseen [Highsmith, 2009]. Thus, agile methodologies benefit of the advantage of variability mechanisms in order to flexibly adapt software architecture and to incrementally develop it together with a working product [Perez, 2010].

4. Case study:

Agile projects use iterative and cooperative source code development techniques, leveraging the client feed-back. Java is a favourite development platform for Agile projects because allows distributed software development (object-oriented, platform independent), code reengineering due to its robust and secure capabilities and resource balancing being a multithreaded programming language. Below is an example of optimizing alternative structures in accessing data on disk using NIO Java package.

Optimizing alternative structures and replacing them by organizing program code in subclasses, drive to the following advantages:

- efficiency in executing programs - substituting alternative structures with multiple calls to objects that implement the same abstract method, defined in a common base class;

(online) = ISSN 2285 – 3642
ISSN-L = 2285 – 3642
Journal of Economic Development, Environment and People
Volume 3, Issue 3, 2014

URL: http://jedep.spiruharet.ro
e-mail: office_jedep@spiruharet.ro

- source code maintainability - in case of new alternatives (conditions) are necessary it is sufficient to create subclasses that implement specific behaviour, calling these instances of the base class is done independently by their behaviour.

4.1. Browse recursive directory structures (browse):

We called static method walkFileTree of File class accepting two parameters: one is Path type and the other is SimpleFileVisitor type <Path>:

```
Path srcPath = src.toPath();
ListDirVisitor listDir = new ListDirVisitor(srcPath);
Files.walkFileTree(srcPath, listDir);
out = listDir.sb.toString();
```

We ensure that the object inherits the SimpleFileVisitor abstract class implements visitFile method (Path, FileAttr) so as to obtain the file name encountered:

```
class ListDirVisitor extends SimpleFileVisitor
{
public String sb;//variabila care va strange numele fisierelor
public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException
{
    sb=file.getFileName().toString();
    return FileVisitResult.CONTINUE;
}
}
```

4.2. Copying directory structure recursively (copy dir)

We called static method walkFileTree of File class accepting two parameters: one has the Path type and the other one is a SimpleFileVisitor<Path> derived object:

```
File src = new File(ResPath);
File dest = new File(DestPath);
Path srcPath = src.toPath();
Path destPath = dest.toPath();
Files.walkFileTree(srcPath, new CopyDirVisitor(srcPath, destPath, ResName,
StandardCopyOption.REPLACEEXISTING));
```

We ensure that the object inherits the SimpleFileVisitor abstract class implements visitFile method (Path, FileAttr) so as to copy every file encountered:

```
public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException
```

```
{
if (file.toString().matches("(.*)" + FName))
    Files.copy(file, toPath.resolve(fromPath.relativize(file)), copyOption);
    return FileVisitResult.CONTINUE;
}
```

Note that walkFileTree static method - called twice by the same object - differs only by the type of the second parameter, instantiated inline on the second polymorphic call, see Fig.2.

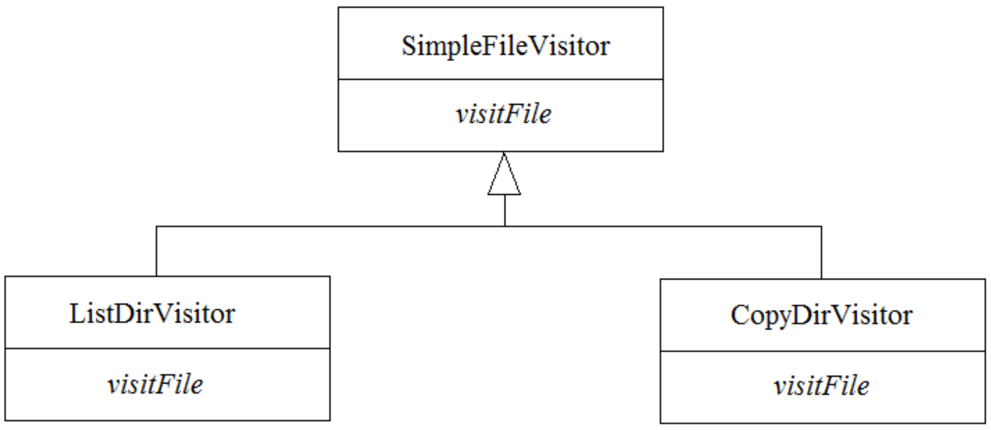

Fig.2. Inheritance relationship evidenced by a report of specialization

5. Conclusion

Innovation in agile programming is possible in refactoring, by optimizing alternative structures and replacing them with subclasses. This will result in increasing efficiency and source code maintainability. Agile programming is a solution for answering the clients' requirements in a shorter time and a more efficient way.

6. Acknowledgement

This work is supported by the Romania's Operational Program for Human Resource Development (POSDRU), financed from the European Social Fund and the Romanian Government under the contract number POSDRU ID 134393 (cempdi.pub.ro/knowledge).

7. References

[1] S. Wood, G. Michaelides , C. Thomson, Successful extreme programming: Fidelity to the methodology or good teamworking?, *Information and Software Technology,* 2013, **55:** 660–672, Published by Elsevier B.V

[2] P. Eeles, Agile Software Architecture, Chapter 13: Building a Platform for Innovation: Architecture and Agile as Key

Enablers, 2014, pp. 315-333, IBM, London, UK.

[3] Beck K. *Extreme programming explained*. Boston: Addison-Wesley; 2005

[4] Graham Collins, Agile Project Management, Project Management, Planning and Control (Sixth Edition), 2014, pp. 523-538

[5] M. Galster, P. Avgeriou, Agile Software Architecture, Chapter 6 - Supporting Variability Through Agility to Achieve Adaptable Architectures, 2014, pp. 139-159

[6] Jennifer Pérez, Jessica Díaz, Juan Garbajosa, Agustín Yagüe, Agile Software Architecture,Chapter 9 - Bridging User Stories and Software Architecture: A Tailored Scrum for Agile Architecting, 2014, pp. 215-241

[7] McGregor JD. Agile software product lines, deconstructed, *The Journal of Object Technology* 2008, **7**:7–19.

[8] Pe´rez J, Dı´az J, Garbajosa J, Alarco´n PP. Flexible working architectures: agile architecting using PPCs. In: 4th European conference on software architecture (ECSA). LNCS, vol. 6285; 2010

[9] Highsmith J. Agile project management: creating innovative products. 2nd ed. Boston, MA: Addison-Wesley Professional, 2009.