INTRODUCING DISCIPLINE TO XP: APPLYING PRINCE2 ON XP PROJECTS

Mouhib Alnoukari, Zaidoun Alzoabi
Arab Academy for Banking and Financial Sciences.
Damascus, Syria

Asim El Sheikh
Arab Academy for Banking and Financial Sciences
Amman, Jordan

ABSTRACT

Agile methods are increasingly gaining momentum, and more and more growing as the market standard for software development. However agile methods suffer from the lack of disciplined planning. We will try in this paper to see the effect of introducing disciplined project management methodology on the flexibility and agility with which agile methods are characterized. We will also see the effect of using a modified XP, the most well-known and oldest method of the agile spectrum, along with PRINCE2 one of the successful management methods. In doing so, we will resort to the findings of a real life project, where a team of eleven developers was able to deliver high quality software within budget and under strict time constraints. The result was a new method derived from both XP and PRINCE2. We call this method eXPeReINCE as it takes its principles from the two methodologies.

KEYWORDS

Project Management, Agile, XP, PRINCE2, Pair Development.

1. INTRODUCTION

Since the 1960's crisis and software practitioners sought more discipline to the field in order to control the software process start, end and deliverables. However, in the mid 1990's a revolutionary software approach was initiated by the introduction of eXtreme Programming by Kent Beck (Beck, 2000) as a software process that emphasizes on software engineers' most important deliverable i.e. code. This approach was seen as a major shift from heavyweight document-based software processes to a more customer-focused, code-centric approach. As stated in the Agile Manifesto (www.agilemanefisto.com), the agile development values individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan.

However, more agility and flexibility may badly affect project's main measures such as scope, time, cost and quality. Moreover the incremental nature of agile methods makes it quite difficult for planners (Nawrocki et al., 2005) to establish plans that satisfy customers. At the end of the day no customer would accept a project without a clear plan showing the major determinants of the intended project.

On the other side PRINCE2 (**PR**ojects **IN C**ontrolled **E**nvironments) is a popular management method known for its rigidness, strictness and document-centric nature. PRINCE2 has been a very useful method in planning and tracking project progress since its introduction in 1989 by CCTA (the Central Computer and Telecommunications Agency) (PRINCE2 manual, 2002). However, too much strictness kills creativity and initiative spirit in the project team (Nawrocki et al., 2005), (Nawrocki et al., 2006), (Noble et al., 2004).

Hence our main objective is to try to extract the best out of XP and PRINCE2. This will lead us to a method that makes advantage of XP's values: flexibility and adaptability, and of PRINCE2 main value: control.

Generally speaking, software practitioners have always avoided using an agile method along with a strict method such as PRINCE2, as the two may look contradicting, while experience has shown that the two could be complementary.

We have used the two successfully in implementing full-fledged software for a new tax system in Syria as a part of the EU-funded, Modernization of Ministry of Finance (MMoF) project. The team was formed from 11 developers mainly from the IT departments in the Ministry of Finance and Damascus Directorate after been trained on PRINCE2 concepts and coached on how to use XP as the software process. In the following sections we will see how this team was organized and coached in order to achieve the goals of the project in extremely strict time constraints.

The result was a modified version of XP that took into consideration the introduction of some management best practices in order to reduce "anarchy" that may face XP projects. The strength of this method is that most of its features were derived from a real case in very unpredictable circumstances and fuzzy environment. The case, under which this method was developed, was the computerization of a tax system passing through radical change. For example the system had to deal with 6 decrees issued within the span of the project causing lots of changes.

Moreover, the team that was allocated this task had no real experience with software processes. And although the team had very good programmers, these were merely able to write code without any concrete knowledge of systems analysis and design.

The following sections are organized as follows; first we will look onto previous work done in agile project management specially those who worked with PRINCE2. Then we will have an overview of XP software process and principles. Then we will have an overview of PRINCE2 and the aspects that will be adopted from both methods (PRINCE2 and XP) along with practical examples from the progress of the MMoF project. Finally, we will see some lessons learnt from the project.

2. PREVIOUS WORK

As stated earlier in this paper, agile software practitioners were not interested in PRINCE2 as a project management method possibly due to the following reasons:

- PRINCE2 heavily depends on paper work; almost every step in PRINCE2 is documented and filed in what is so-called management files, whereas agile methods, especially XP, rely on oral communications rather than documents (Beck, 2000), (Newkirk, 2002).
- PRINCE2 team structure drastically differs from XP structure. PRINCE2 depends on a 4-level hierarchical structure while there is no hierarchy as such in agile methods.
- PRINCE2 was seen as less flexible than any agile method can ever tolerate, as it requests a control method for almost every step in the project life cycle. In contrast, agile fans require maximum possible flexibility in adopting changes.

Despite these obstacles, there have been two courageous attempts in integrating the two methodologies. The first was in introducing XPrince in Poznan University of Technology, Institute of Computing Science, Poland where XP was integrated with PRINCE2 (Nawrocki et al., 2006). The other was integrating DSDM with PRINCE2 (DSDM with PRINCE2 Task Group, 2000).

XPRINCE (eXtreme PRogramming IN Controlled Environment) integrates the team structure of XP and PRINCE2 to make the maximum benefit of the two. Figure 1 shows how the two structures were merged.

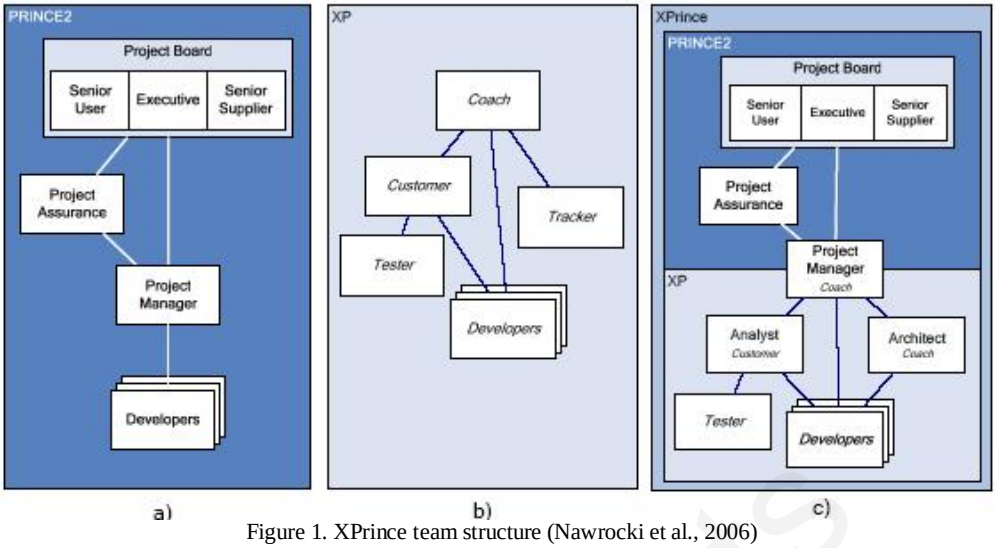

Figure 1. XPrince team structure (Nawrocki et al., 2006)

XPrince then demonstrates a project life cycle that fits XP incremental nature and XP planning requirements. It also encourages Side-by-Side programming explained by Cockburn rather than pair programming. It also defends other aspects of XP such as re-factoring and automated tests (Nawrocki et al., 2005).

The major weakness of XPrince is that it is still at experimental level in the university and it has not been tested at the industrial level.

The other trial was the integration of DSDM and PRINCE2, which in fact seemed to be a safer adventure as there are a number of concepts common between the two. Figure 2 shows these common areas.

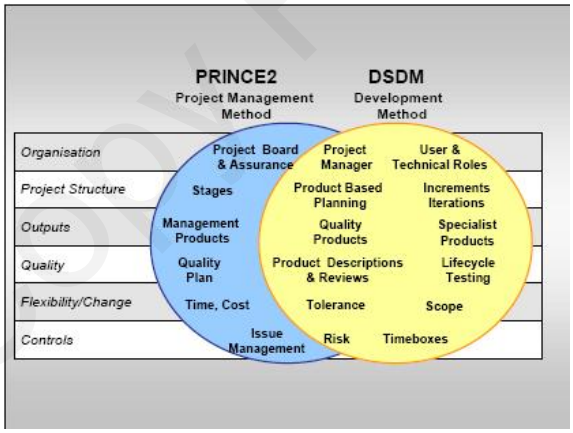

Figure 2. DSDM and PRINCE2 (DSDM with PRINCE2 Task Group, 2000)

The DSDM with PRINCE2 Task Group sees the two methodologies as complementary. PRINCE2 can provide control and DSDM can provide agility. Moreover the paper claims that DSDM developers had PRINCE2 in mind. This could be found in concepts like product-based planning, the involved partnership of users and developers, and the strong emphasis on underlying business case.

Once again, the main emphasis is the project team structure, but with a real realization of customer-on-site principle through the use of ambassador user and advisor users. The ambassador user ensures effective communication between users and developers. And the existence of advisor users will bring customer-on-site principle into reality.

Another issue of similarity is the product-based planning, where the specifying of quality criteria of products to be built is done before it is built. In fact, the descriptions of the DSDM standard products in the manual are generic PRINCE2 specialist product definitions.

Again there seems to be little or no industrial application of this combination.

3. XP PROGRAMMING

eXtreme Programming as an identifiable methodology is distinguished by twelve main practices, along with a number of secondary practices (Beck, 2000), (Newkirk, 2002). These practices are similar to the activities or techniques of conventional methodologies, in that they are particular things that programmers actually do to produce software. XP has four core values that are used to guide the practices that are employed. These values are:

- Communication - This includes communication between all team members, customers, programmers and managers.
- Simplicity - "What is the simplest thing that could possibly work?" The message is very clear. Given the requirements for today design and write your software. Do not try to anticipate the future, let the future unfold.
- Feedback - XP practices are designed to illicit feedback early and often. The practices such as short releases, continuous integration, testing provide very clear feedback.
- Courage – XP changes the position of software engineers from defense to offense. It takes courage to say I have done enough design for now and I'll let the future happen.

In order to realize these values XP put under one umbrella 12 practices that programmers have developed over decades, integrated them, and tried to make sure they are practiced well. These practices are:

- Planning- Determine the scope of the next iteration by working with customers who provide business priorities and with programmers who provide technical estimates.
- Small Releases- Get the system into production quickly. This is a key factor in getting feedback on the actual software.
- Metaphor - Understand how the whole system works. This is important for both developers and customers
- Simple design - One of the key values is simplicity. The system should be designed for the features that are implemented today, and add features gradually.
- Testing - Tests include unit tests, which programmers write and acceptance tests, which customers write. Tests are the indicator of completion.
- Refactoring -Programmers are responsible for improving the design of existing software without changing its behavior.
- Pair Programming - Working with a partner is a requirement when writing production code.
- Collective ownership - Anyone on the team can change any part of the system.
- Continuous integration- Programmers integrate and build the software many times a day.
- 40-hour week – XP encourages working for 5days X 8 hours.
- On - site customer-The customer is on the team, available to answer questions full-time.
- Coding standards - Communication is a key value. Adopting coding standards improves communication.

Next we will have an overview of PRINCE2 and then see how we could successfully put XP in PRINCE2-managed environment.

4. PRINCE2

PRINCE2 is a structured method for effective project management. The method was first established in 1989 by CCTA (the Central Computer and Telecommunications Agency). PRINCE was developed from PROMPTII, a project management method created by Simpact Systems Ltd in 1975. PROMPTII was adopted by CCTA in 1979 as the standard to be used for all government information system projects mainly

in UK but spreading across whole Europe. PRINCE superseded PROMPTII in 1989 within government projects. CCTA (now the Office of Government Commerce) continued to develop the method, and PRINCE2 was launched in 1996 (PRINCE2 manual, 2002).

PRINCE2 defines a structure for authority, delegation, and decision-making. It also defines the communication channels between different stakeholders. A key principle in PRINCE2 is that management of the project is by exception. There is automatic management control for deviations from the agreed project plan.

PRINCE2 projects are measured by tangible results, so the reliance here is on products and outcomes rather than tasks required to produce them. Hence, the Product Breakdown Structure (PBS) is used in defining products and measures.

PRINCE2 is criticized for its strictness and heavy use of documents. It uses a document for almost every step in the project, which caused software engineers to avoid using it as the management methodology in software projects. However this strictness may have been the main cause behind PRINCE2 projects' successes. Rationalizing this strictness could be quite helpful in controlling IT projects, and in tying discipline with flexibility in order to strike a balance between the two.

It could be quite useful to have a look on the PRINCE2 processes. PRINCE2 has mainly eight processes: starting up, initiating, planning, directing, managing a stage, managing stage boundary, managing product delivery, and closing the project.

These processes are further broken down into sub-processes, each with specific output. We will discuss the PRINCE2 processes, especially those related to production further in details later in this paper.

5. XP IN PRINCE2 CONTEXT

There is no de facto management methodology that fits seamlessly with XP. Instead XP uses its own management techniques such as: planning game. This has an advantage and a disadvantage.

The advantage is that XP team will be self-managed team allowing more freedom in decision making, adopting change, and giving courage to try various options.

However, and here is the disadvantage, having more freedom than required, can lead to some sort of anarchy, leading to unknown outcomes of the project, contradicting targets, and distracted efforts. The coach role in XP resembles the role of a project manager, but it is of technical nature rather than managerial.

The coach's role is to ensure people are communicating (Beck, 2000) but who will be responsible for talking to the senior customer, will convince the project owner with changes, will control conflicting interests, will be responsible for controlling cost, time, and scope, will mange risks and control them etc.

PRINCE2 can be helpful in setting the pace for XP projects in order to control the extent of flexibility to ensure that flexibility does not drag the project beyond expectations.

In order to understand the potential PRINCE2 gives to XP we will group the XP's 12 practices into 4 groups, and see how these will be affected with PRINCE2. These groups are:

- Structural: pair programming, 40-hours a week, on-site-customer, and collective ownership.
- Planning: planning, and small releases.
- Production: simple design, testing, and continuous integration,
- Practices: metaphor, coding standards and refactoring

5.1 Pair Programming, 40-Hours, and On-Site-Customer

One of the great contributions of XP is pair programming. A pair is a small team of two programmers (analysts, designers, and testers) (Beck, 2000), who will work on the same part of the system. One of the two will think strategically and the other tactically. Pair programming proves that 1+1>=2.

Another concept is Side-by-Side programming (Nawrocki et al., 2005). In this approach a single task is given to two programmers; each one of them is working on his/her workstation.

In our project we took pair programming one step further. The pair was chosen such as that one of them has strong analysis skills with good programming skills, and the other has strong programming skills. The two formed what we called pair development. The analyst was heavily involved in the discussions with the customer and had good experience in the business area under development (i.e. tax administration).

Because we had eleven developers in the team, one had to work alone, and to overcome this we chose the single programmer to be one of the strongest. This developer had the hardest time and her module was last integrated with the system despite her very good skills and previous knowledge of the domain.

Of course it is not easy to separate analysts from programmers. But putting this issue in mind before the team structure has been formed will make it possible to get the best results. This restricts the option in forming the pairs. This seemed to be of great value, because giving the freedom for the team to form pairs ended up, in many occasions, having imbalanced pairs: a strong pair Vs weak pair.

An issue that was raised in (Beck, 2000) is the difference in skills level between the two members of a pair. Using pair development will make it easier, as the weak programmer can be more involved in the analysis and in discussions with the customer. The knowledge the analyst is developing (or had beforehand) will help him/her in thinking strategically, leaving the programmer to think of loops and conditions.

Another benefit of the pair-development (programming), which is not mentioned by the pair-programming fans is that pair development (programming), drastically reduces human-related risks. After our system was broken down according to PRINCE2 into smaller products, every pair was assigned one module to look after. This meant that, in case anyone of the pair had to leave the project – temporarily or permanently-, the project was still safe. For instance, Syria saw a period of flu in the mid of the project left one-third of the team in bed for almost a week, but still we were on time.

Another issue that played role in fault tolerance is something that XP hates and PRINCE2 is in love with: documentation. One of the authors, who played the role of coach, made daily, very short meetings taking feedback from all pairs, discussing short reports- of one page only- reporting everything happened during the previous day. This kept every step within "sight" minimizing risks drastically.

In order to have the maximum value of the pair development concept, and to apply the on-site-customer principle, an end-user with good knowledge of concerned business area was assigned to the team. One of the obstacles that faced us was finding the right customer representative to join the team. Skilled people had no time and others did not have enough experience. So the best way is to assign one customer representative to more than one pair. This is efficient as it is helpful to have the customer representative working with the pair on 8 hours per day and 5 days a week (the 40 hours principle). This is highly emphasized in PRINCE2. Moreover the team structure of the PRINCE2 involves senior user(s) at the project board level. The presence of such a group is very important for resolving hanging issues and answering questions that could not be answered by the customer representative.

In our project, two expert users where assigned to the team. This couple had to give support to the pair developers and of course to the single programmer almost on full time basis. In addition, an experience committee was formed with whom the team met on half monthly basis, unless there was something urgent. This committee was formed from owners and suppliers, whose tasks were: defining acceptance criteria, approving plan, resolving business-related problems, approving changes as they occur.

Finally, it was noticed that the 40-hours is realistic if it is taken as an average number. People tended to work less than 40 hours a week at the beginning of the project (, or a stage of a project) and more than that on delivery time. No matter how the project management emphasizes on balanced time distribution, people will be more relaxed at the beginning and under the student syndrome pressure at the time just before the deadline. The project management role is to reduce as much as possible this student syndrome but no way to eliminate.

5.2 Planning: Simple Design, and Small Releases

PRINCE2 asks the project manager to develop numerous plans: quality plan, communication plan, stage plan, risk management plan, project plan, etc. On the other hand XP emphasizes quick, incremental planning, with no specific plans for communication, quality, risk management, budget, schedule, etc. To make PRINCE2 a good management method for XP, we have identified the core plans that will be needed anyway.

The first is the communication plan, which is a straight forward plan, yet a very important one as it affects almost every aspect in the project such as:

- Customer involvement: the communication plan will identify on-site users, form of the project board, communication with the board, and tools of communication.

- Change management: change will have to be communicated to, and approved by the main stakeholders. Although XP insists on accepting changes and quick adaptations, we still found that formal ways of adopting changes is very important.

Feedback: having planned, regular meetings with project board will keep the feedback channels active.

The other plan is quality plan. This is because XP emphasizes on continuous adaptations and on embracing change. This will require agreed standards in order to keep the project on the right track. Moreover, XP allows anyone in the team to change anything in the software, so if these changes have to be compliant with the quality plan, the software might end up at a mess. One important issue that the quality plan should emphasize is the coding standards that we will discuss below.

Plans however, should not be followed blindly. One good description of plans is that planning is essential, plans are useless. Plans will never stay unchanged or at least we haven't seen any so far. But ignoring planning or giving it little attention will make success quite difficult if not impossible.

Also plans should not be detailed. A one page communication plan that describes frequency of communication, reporting line, methods of communication, and reports' format would be sufficient.

Planning XP using PRINCE2 method has a special advantage in that both of them have an important similarity. Both of them emphasize planning for the next horizon (Beck, 2000), (PRINCE2 manual, 2002), this horizon is described as release in XP and as stage in PRINCE2. Both ask for detailed stage (release) plan and a general outlined long-range plan. Another similarity between the two is the principle of product-based planning, rather than work-based planning. PRINCE2 identifies the products of the project and plans accordingly. Similarly, XP starts with user stories, which will talk about the required products and puts the plan on the basis of user stories which would be naturally in the form of required products.

XP uses a simple life cycle that is composed of four phases: planning, design, coding and testing. Every increment has to go through these four phases. One or more increments compose a release.

On the other hand PRINCE2 uses a 4-phase project life cycle: start, initiation, production, and closure. The production phase consists of 3 components: stage control, managing product delivery, and managing stage boundary. If we consider a stage is an increment then all four phases of XP are considered within PRINCE2 production phase. However, having 3 components to manage production is extremely inefficient and bureaucratic and could waste time in writing reports and documentation. A one phase of controlling the stage (release in XP terms) would be very efficient.

Planning is considered as an umbrella activity in PRINCE2, and could be removed from the XP lifecycle. Instead, an analysis phase proved very crucial in reducing design, coding, and testing time. This is of greater importance in complicated environments, where users have conflicting stories (Figure 3).

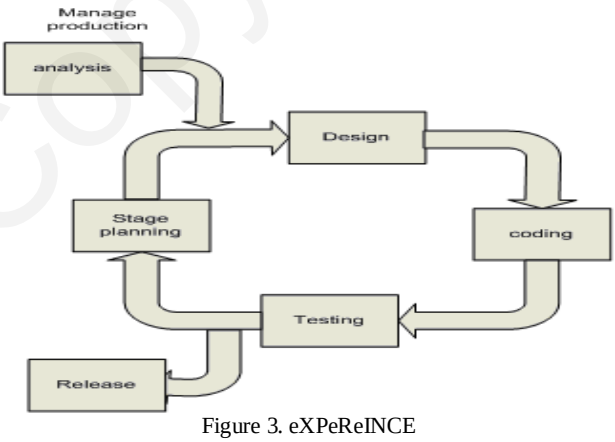

Figure 3. eXPeReINCE

5.3 Production: Simple Design, Testing, Continuous Integration, and Refactoring

The prescribed structure emphasizes on simple design in order to make on-site customer more effective as simple design will help in communication between the team and users. Moreover, simple design will help in planning as simple designs are easier to schedule and budget.

In addition to that, the presence of an analysis phase will be beneficial in making more realistic planning and in getting simpler and more accurate design of the system. Analysis is required as one analysis phase per one release, and should not be repeated for every increment

Continuous integration and testing are very important XP principles as they lower the risk probability and impact. Continuous integration requires small pieces of code, and within short period of time (hours not days) (Beck, 2000). This means that the impact of the risk of going in the wrong direction is pretty small. Continuous (daily) testing, on the other hand, will lower the probability of risks.

PRINCE2 has here some hiccups as it requires very controlled procedures for testing and integration. PRINCE2 requires that reviews are scheduled, names of the participants are documented, and place of the review and duration identified. Such an approach would be again bureaucratic and time consuming.

Instead, project assurance and manager can, and should, have greater focus on the quality plan at the beginning of the project and then verify production against plan continuously through effective communication. One approach we used is that testing is carried out by a different team than the one was involved in the development. This is very beneficial as one can see others mistakes more than his/hers.

Another issue is standardization. Concentrating on standard coding structure and naming conventions is extremely helpful in controlling quality in the absence of frequent Formal Technical Reviews. Project manager who could play the role of project assurance should be very strict in imposing coding standards on developers. No one is allowed to change the interface format, naming convention, and even indentation, before it is agreed by everyone in the project. This change will become a new standard that should be documented and used afterwards.

5.4 Practices: Metaphor, Coding Standards, and Refactoring

Following we'll talk about three XP practices: the metaphor, coding standards, and refactoring. These three practices can be preserved as they are without the need for any amendments in order to fit the PRINCE2 environment. However coding standards practice is very important in order to fill the gap created by less documentation, as mentioned in the previous section. The coach of our project was very strict on coding standards, through documenting and circulating any standard, testing every piece of code against these standards, and testing any new suggestion for a new method collectively with the team.

Finally, let us have a word about refactoring. Refactoring is, from our experience, something that is inherent in the nature of programmers. Any statement from the coach regarding the efficiency of the coach was always faced with the answer:" we'll give you the most efficient code but let us first get the right functionality."

6. CONCLUSION

Agility and discipline are not mutually exclusive. Rather, they can be complementary. Agility can contribute to creativeness and improve customer relationship, and discipline will keep the project on track and within budget, time, and quality constraints. Of course, there is no wrong answer for which management method to use in XP. One can use PMI, PRINCE2, or any other management method, except that PRINCE2 is preferred in unstructured environments.

We in Damascus finance directorate have tried that combination successfully providing systems on time and within budget. And here are some of the lessons learnt from our project:

- Use always balanced pairs. Try to make the pair look like pair developers rather than pair programmers.
- Use rationalized documentation that keeps project within the control of the project manager or coach.
- Perform one analysis phase per one release. This eases the job of the on-site-customer.
- During the analysis phase, use the most expressive models. We have found that DFD's and ERD's are very powerful in conveying a clear, yet very precise message.
- Do not use too many plans. But there are two that are inevitable: communication plan and quality plan. In addition to that, a nice Gantt chart drawn using MS Project will be sufficient.

- Hold daily, very short meetings. Do not make these meetings longer than 15 minutes.
- Make sure pairs test each other's modules; others see what you can't see.
- Emphasize on coding standards. Have no mercy in this regard. Coach has to be democratic almost in every aspect of the project except this.
- Emphasize on on-site-costumer. Have at least one expert user on full time basis, supporting all pairs.
- Always ask for a committee from the customer side to resolve complex issues.

REFERENCES

Beck, K., 2000. *Extreme Programming Explained*. Addison-Wesley, Reading MA.

DSDM with PRINCE2 Task Group, 2000. *Using DSDM with PRINCE2*. DSDM consortium.

Nawrocki J., et al, 2006. *Rapid Integration of Software Engineering Techniques*, Springer, Berlin / Heidelberg.

Nawrocki, J., et al, 2005. Pair Programming vs. Side-by-Side Programming. *Proceedings of the European Software Process Improvement and Innovation Conference*.

Newkirk J., 2002. *Introduction to Agile Processes and Extreme Programming*, Addison-Wesley, Reading MA.

Noble J., et al., 2004. Less Extreme Programming. Available at http://crpit.com/confpapers/CRPITV30Noble.pdf

PRINCE2 manual, Crown Copyright 2002.

www.agilemanefisto.com