www.ietdl.org

Published in IET Software
Received on 22nd June 2011
Revised on 14th August 2011
doi: 10.1049/iet-sen.2011.0110

ISSN 1751-8806

# Agile software development methodology for medium and large projects

*M. Rizwan Jameel Qureshi*

*Faculty of Computing and Information Technology, King Abdul Aziz University, P.O. Box 80221, Jeddah 21589, Kingdom of Saudi Arabia*
*E-mail: anriz@hotmail.com*

**Abstract:** Extreme programming (XP) is one of the most widely used agile methodologies for software development. It intends to improve software quality and responsiveness to changing customer requirements. Despite the facts that the use of XP offers a number of benefits and it has been a widely used agile methodology, XP does not offer the same benefits when it comes to medium and large software projects. Some of the reasons for this are weak documentation, lack of strong architecture and ignorance to risk awareness during the software development. Owing to the ever-increasing demand of agile approaches, this study addresses the problem of XP's ability to handle medium and large projects. Most of the companies that employ XP as a development methodology for medium and large projects face this problem, which echoes the importance of this problem. To address this problem, in this study XP model is extended in such a way that it equally offers its benefits for medium- and large-scale projects. As an evaluation of the extended XP, three independent industrial case studies are conducted. The case studies are described and results are presented in the study. The results provide evidence that the extended XP can be beneficial for medium and large software development projects.

## 1 Introduction

Agile methods for software developed emerged in the mid-1990s [1, 2] and focus on agility for software development. In essence, agility means responding to changes quickly and efficiently. Possible changes required in software projects are in budget, schedule, resources, technology, requirements and team. These are the 'reacting' changes on which agile models focus by delivering first increment in a couple of weeks and complete software in couple of months. Twelve golden principles have been defined in an agile alliance meeting conducted in 2001 [1]. These principles provide support for development of only small software projects having small teams [3]. However, there is no guidance about how to customise agile process models for the development of medium and large software projects [4–6]. There are several criteria to classify projects such as size, complexity and mission criticality [7].

Generally, a software project is considered 'small' if line of code (LOC) of the project is between 10 000 and 40 000, 'medium' if LOC is between 40 000 and 100 000 and 'large' if LOC are typically over 100 000. In addition to this, 'size' of a project can also be measured in terms of human effort (e.g. number of person-months applied) [7]. This paper measures 'size' in terms of LOC to classify the projects as small, medium and large.

Key benefits of agile models are fast development and cost reduction. Fast development in some cases leads to poor software (SW) quality and carrying all disadvantages of rapid application development (RAD) and prototype models, such as weak documentation, difficult to upgrade, lack of reuse and excessive maintenance [8]. Extreme programming (XP) is a widely used among all agile models. A number of studies have been reported about effective implementation of XP for small projects [9, 10]. XP was proposed based on ideas and practices from previously proposed process models to achieve advantages such as time saving, cost reduction, refactoring and suitability for small projects with small teams [7, 11]. An example is the refactoring technique which improves a software quality in terms of design and code throughout software development. However, some drawbacks of XP that can be found in the literature are inappropriate for safety critical projects, limited support for outsourcing, inadequate assistance for distributed development environment, weak documentation and unsuitable for medium and large projects [12–14].

Medium and large projects have some characteristics similar to small projects such as time constraints, changing business situations and vague requirements [12]. Software industry has to bear an immense amount of stress to deliver products timely without sacrificing quality. Traditional methodologies are not fulfilling the need of software industry to achieve fast development without compromising quality whereas agile methodologies cannot be directly implemented for medium and large development projects because of inadequate documentation, weak architecture and lack of risk management. Therefore there is a need to adapt the agile methodologies for medium and large projects to achieve fast development with high quality to

handle software industrial problem. This paper focuses on extending XP model for medium and large projects in order to meet the industrial demand of an agile methodology.

The rest of the paper is organised as follows: Section 2 covers related work. Section 3 describes the research problem. Section 4 provides motivation for the new improved agile XP model to be proposed. An extended XP model is proposed in Section 5. Section 6 presents validation of the proposed extended XP model using three case studies.

## 2 Related work

A number of case studies have been reported for the success of agile process models for the development of small-scale projects [15–17]. Two of the case studies in these projects were reported to be completed in 3 months and remaining one was reported to be completed in 2 months. These projects used qualitative and quantitative techniques to estimate and analyse the results. The findings of the case studies are as follows.

- Agile teams are supposed to monitor team performance and SW development procedures continuously for efficiency.
- Defect rates and team's overall productivity are improved by following agile practices.

These case studies are conducted to adapt agile process model for 2–3 months projects according to agile principles [15–17]. These projects do not provide any information about adaptation of agile XP model for medium and large projects (greater than 100 000 LOC).

Pekka *et al.* intended to classify, examine and formulate meaning in the scattered area of agile software development process models [18]. A comparative analysis was presented using the method's life-cycle reporting, project management assistance, type of practical guidance, fitness-for-use and empirical evidence as the analytical parameters. Pekka *et al.* concluded that agile models dealt with all main phases of system development life cycle (SDLC) but there were limited support for project management [18].

Cao in [19] works out a complete scene of agile development in order to improve the knowledge of software development community and to foretell about the agile process. The objective is to model the dynamics of agile software development process and explore the implementation and usefulness of agile methods [11]. Another purpose is to investigate the influence of agile process models on functioning of software such as quality, schedule, cost and customer's satisfaction [19].

Various attempts have been made from last many years to improve agile models [20, 21]. Jachi *et al.* [22] proposed a modified XP agile process model for developing diagnostic knowledge system and its main phases were system metaphor, planning, implementation and integration. The proposed model was a hybrid approach consisting of process model to develop expert systems (ESs) and XP model. The objective is to initiate agility in expert systems. As the diagnostic knowledge system is in its early stages of development, it is too difficult that this model will meet its objectives to develop large projects [22].

Lucas *et al.* [23, 24] presented two case studies using adapted XP model. One case study was conducted with IBM for 12 months and other one was conducted with Sabre Airline Solutions for 3 months. Lucas *et al.* [23, 24] proposed an extreme programming evaluation framework (XP-EF) to adapt XP model. The framework used feedback loop throughout the project for the evaluation of agile team and procedures involved. Framework needs further validation through more case studies and needs to be improved particularly with respect to the XP adherence metrics. The teams in both case studies were typically agile and had management support to deploy XP process model. Hence, the success stories of both teams cannot be taken as an example to fit for non-agile, large and distributed teams and for the teams have limited management support for the deployment of agile models.

### 2.1 Weaknesses of the XP model for the development of medium and large projects

Cao *et al.* [12] have argued that XP cannot be used for medium and large projects because of inadequate architectural planning, over-focusing on early results, weak documentation and low levels of test coverage. The XP approach emphases on coding/development and operational software instead of comprehensive documentation/architecture design. It supports to do additional effort to throw architectural features that do not assist current increment [13]. This practice delivers the desired results for small projects but throws important architectural features for medium and large projects where cost and effort of a change in the architectural design is very high [14].

## 3 Research problem

A number of facts have been put forward from a variety of industrial domains in support of agile software development methodologies. However, most of them come from small projects using controlled studies [15–17]. So, there is a need for extending/revising XP in order to acquire the benefits (offered to small projects) for medium and large projects in an industrial context. The research question therefore becomes:

How to adapt XP for the development of medium and large projects?

## 4 Motivation for the proposed extended XP process model

Classical XP methodology is not applicable to medium- and large-scale software development projects. This motivates three changes, including 'Project Planning' phase, 'Analysis and Risk Management' phase and merging 'Design' and 'Development' phases of existing XP process model. These changes also offer some additional improvement as a byproduct of the to-be-proposed model that is besides wider scope of the output, the quality of the proposed extended XP model will be significantly improved than the existing one because of risk awareness, better documentation, stable requirements and strong architecture. It can also be advocated that better documentation further helps a software engineering team to evolve and reengineer software owing to strong architectural design. The inclusion of 'Project Planning' phase, 'Analysis and Risk Management' phase and merger of 'Design' and 'Development' phases are likely to enable the proposed extended XP to be applicable to medium and large software projects.

## 5 Proposed agile methodology for medium and large projects

In this section, the proposed XP methodology is presented. It is an extended form of the classical XP methodology that is expected to be equally suitable for small, medium and large software projects unlike the existing XP model that is meant for only small projects. The main phases of classical XP are planning, design, coding and testing [7], whereas the main phases of the extended XP methodology (our proposed) are 'Project Planning', 'Analysis and Risk Management', 'Design and Development' and 'Testing'. The Testing phase of the extended XP model is executed in the same manner as of the existing XP model. Fig. 1 shows the proposed extended XP model.

The extended XP model is multidimensional in nature and it is equally suitable for incremental (like existing XP) and parallel development (like traditional process models). The proposed model supports the development of medium and large software projects. A detailed description of the phases of the extended XP model is as follows.

### 5.1 'Project Planning' phase

The first phase of existing XP is the 'Planning'. Its main elements are gathering user stories, communication feedback loop, story estimations, acceptance test criteria and iteration plan [7]. These activities are not enough to develop a medium- to large-scale project because more emphasis is laid on the process rather than production of a solid document that can become a foundation for the entire project. The information about the project is heavily relied on the project team members. Project planning is one of the project drivers and it provides a systematic or pragmatic approach to direct and complete a project. Project planning plays an important role for the success or failure of a project. A software project will deteriorate after few years if the project planning is not properly made and documented [7].

Therefore the 'Planning' phase of existing XP is replaced by the 'Project Planning' phase in the proposed extended XP model by keeping in mind the demands of medium and large projects. Hoffer *et al.* [8] described that project plans for medium and large projects can be hundred pages in length therefore types of activities can be performed must include the defining the project scope and its alternatives, feasibility assessments, dividing the project into manageable tasks, estimating resources, developing a schedule, identifying major risks and its assessments and use case modelling. The objective of 'Project Planning' phase is to concentrate on the major milestones of project instead of micro-milestones.

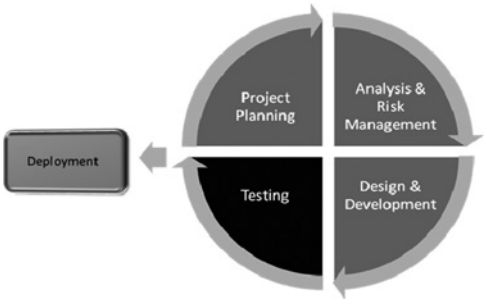


**Fig. 1** *Proposed extended XP process model*

### 5.2 'Analysis and Risk Management' phase

The existing XP model gathers the requirement/stories during the 'Planning' phase using an index card. It does not have any architectural design except the metaphor (i.e. an abstract design). According to Cao *et al.* [12], 'XP de-emphasizes up front design because it is claimed that everything is changing. Instead, a "metaphor" is used to describe the basic elements and relationships of the application'. XP approach gives more consideration to operational software than comprehensive consideration [13]. XP throws away valuable architectural support of the previous increments if it is not relevant to the new increment that is essential while dealing with medium and large projects. XP works well for small projects having few stories to be implemented within few weeks. Analysis phase improves quality of software through proper documentation. 'Analysis and Risk Management' phase in the proposed XP model has many benefits for example, stable requirements, strong architecture and risk management plan. Stable requirements facilitate a development team to achieve strong architectural design by aiding the factor of reuse and easy to evolve a software. Risk management plan helps a team to cater potential risks regarding the failure of a project. It has been reported that a large number of projects failed because of lack of risk management when XP was initially used in projects [25].

This is the phase where an analyst gathers detailed requirements/user stories. Analysis phase results in more comprehensive user requirements/user stories and strong architecture. Strong architecture means a flexible architecture that allows using design patterns. Design pattern decreases the time and cost for the implementation of new requirements/user stories in contrast to write them from the scratch. The client is requested to prioritise the user stories on his need basis and provides an index value similar to existing XP model. High index value indicates high priority. Client can: (i) change the order of user stories at any time and (ii) provide new user stories at any time. Planning poker technique is implemented to achieve a manageable value (in development weeks) of user stories to be completed in a release. Modelling of the user stories is kept simple to avoid complicacies for the software engineering team. Project velocity is measured after successful deployment of first release and then in all subsequent releases throughout the development of project. The objective is to adjust the delivery dates of remaining user stories to be developed.

### 5.3 Design and Development phase

Design and coding phases of existing XP model are merged to incorporate agility in the new adaptive XP process model. XP design produces only metaphor and spike solution. Spike solution is an operational prototype. The proposed XP process model uses prototype/demos cycles to verify the design and user stories. Software is developed in small releases/increments as the customer approves prototypes. Merging of design and development phases also improves efficiency of software development. Refactoring technique is implemented during designing and coding of a release. Interface specification document is designed for the user stories to be developed in first release. The task of coding is assigned to programmers following pair programming. Interface specification document of user stories of the second release is designed simultaneously by the time programmers coded the user stories of first release. This process of designing and coding is cyclic for the remaining




doi: 10.1049/iet-sen.2011.0110

releases till the whole software is developed. The programmers continuously integrate the code as they complete it for a particular release.

### 5.4 Testing phase

Test cases are prepared for before coding of a release following test-driven development (TDD) environment. Each release is tested on unit basis. Integration test is then conducted to check integration among modules. System test is the next phase to validate the whole increment as one unit. Acceptance testing is the last test to verify a release from the customer. Tested release is maintained and deployed. Testing cycle and learning and post learning cycles of the software development team are continued throughout the software development. The main objective of cyclic strategy is to improve and change the development activities by the reorganisation of extra actions required during the research. The proposed extended XP process model is cyclic and evolutionary till software development is completed. Main activities of deployment are installation, training and security.

## 6 Evaluation of the proposed extended XP model

As presented in the preceding section, XP is adjusted to make it equally beneficial for all three types (small, medium and large) projects. In order to evaluate the usefulness of the proposed methodology, case study is used as a research method. Three case studies were conducted for small, medium and large projects to evaluate the proposed XP (one for each).

### 6.1 Three case studies

The case studies are based on three projects of three independent software companies who volunteered for participation. An intensive 2-week training was arranged for all teams involved in the case studies. The aim of the training was to present the concepts of our proposed methodology and explain the XP practices and strategy to implement the extended XP model for small, medium and large projects. The main practices of XP were explained include planning poker technique, small releases, simple design, test-driven development environment, refactoring, pair programming, collective code ownership, continuous integration, onsite customer and coding standards [26]. The training also covered development and technical environment of the case studies/projects.

From a study of existing approach [15, 17, 23], the author observes that two to six releases are considered sufficient for presenting results and claiming some findings. In line with that, in this study the author deemed it appropriate to conclude the results of case study based on first four releases of the three case studies. Specifications of case studies conducted for evaluation of the proposed artefact are shown in Table 1 (as cases 1–3, respectively). This is followed by a detailed design and description of each case study.

*6.1.1 Case study 1:* Case study 1 is a small project conducted for a software company to develop a logistic information system (LIS) for an invoice management company dealing in USA on the premises of software company. The software company is working with two offices that are situated at Pakistan and USA. The software company was dealing in office management systems since 2002 and its main product was Recruitment Management System. The average duration to complete previously completed projects was 1 year. The software company was using rational unified process (RUP) model to develop software previously. The software company had no experience of working with XP process and its practices.

A team of 12 members was selected for this pilot project from the software company from 20 employees. Team formation was suggested by the author to implement new adaptive XP model for small-scale projects in a distributed development environment. The criteria for the selection of team were qualification and experience of object-oriented analysis and design. The experience level of team members was ranging from 1 to 8 years in software industry. Project manager had high experience of 8 years in software industry. Team had high experience of ASP.NET tool with a medium to high experience of the domain. Only those programmers were selected for this project which had database development experience.

*6.1.2 Case study 2:* Case study 2 is a medium project performed for a software company to develop an academy management system for an army cadet college on the premises of Software Company at Pakistan. The software company offices are located at five countries Pakistan, UK, Australia, China and USA. Case study 2 has been completed for a software company that is dealing internationally. The software company has been dealing in financial software solutions since 1995 and well experienced in developing software projects. Case study 2 was the first development experience of the software company using existing agile XP process and practices. The company was using formerly RUP model to develop software. Estimated schedule for completion of whole project was 18 months. Project is broken down into modules based on initial communication with the army officials. The main modules of ERP project are mess, army cadets training and academics.

A team of 15 members was selected for this project from the software company from 200 employees. Team

**Table 1** Criteria for the conduction of three case studies

| Criteria | Case 1 | Case 2 | Case 3 |
|---|---|---|---|
| team size | 12 | 15 | 20 |
| project size | small | medium | large |
| nature of project | logistic system | academic system | property estate business system |
| experience of object oriented development | high | high | high |
| no of releases compared | 4 | 4 | 4 |
| size, KLOC | 8 | 47 | 63 |

formation was suggested by the author to implement new adaptive XP model for medium-scale projects. The criteria for the selection of team were qualification and experience of object-oriented analysis and design. The experience level of team members was ranging from 3 to 10 years in software industry. Project manager had high experience of 8 years in software industry. Domain expertise of the team was low owing to specific nature of ERP applications for army. The team members are located at one site on the premises of the software company at Pakistani office. Two weeks deadline was decided to analyse, design, code, test and deploy a release and all subsequent releases. Product specialist approved the release before beta testing. The client provided feedback within 3 days of third week. The project team changed the release as per needs of client within third week. The postmortem analysis was performed before starting work on the next release.

*6.1.3 Case study 3:* This case study is a large project conducted for a software company that has two offices; one is located at Pakistan and second is located at USA. The case study is to develop software for a leading company of USA dealing in property estate business on the premises of the software company at both offices. There was no experience of the software company in developing projects. Case study 3 was the first development experience of the software company. Six years were estimated for completion of this project namely GREAT.

Four teams were hired for this large project by the software company to implement the extended XP model in a distributed development environment. This team formation was suggested by the researcher for the adaptation of XP model for large-scale projects. The criteria to select the teams for this project were qualification and experience of object-oriented analysis and design. The domain expertise of the team was medium to high and language expertise was high. Minimum experience of a team member was 3 years in software industry was a criterion for selection. The project manager had 10 years of experience in software industry. Each team consisted of two product specialists (each product specialist handles two teams) and one project manager (shared among four teams). Each team would be working on a particular release, meaning four releases would be designed, developed, tested and deployed at the same time.

The project was broken down into several sub-projects. Each sub-project was a complete project itself ranging from 1 5, 4 and 6 months. Product specialists and project manager decided that each project would be divided into releases. Each release was an increment based on the number of stories completed in it. Product specialist approved the release before deployment.

### 6.2 Results of three case studies (cases 1–3)

The extended XP model is validated using three case studies. The results are concluded based on the postmortem analysis and number of defects reported of the three case studies. The time (in hours) and effort (in %), spent per release to conduct postmortem analysis of the three case studies, are shown in Tables 2 and 3.

Many (existing) XP case studies reported that a minimum time of postmortem analysis was between 2 and 4 h and the effort consumption was around 4.7% on lightweight postmortem reviews [15, 27, 28]. In the three case study projects in Table 2, the average time is 1.6 h in Case 1, 2.8 h in Case 2 and 3.7 h in Case 3. In Table 3, the average effort is 4% in Case 1, 7% in Case 2 and 9% in Case 3. The extended XP model shows its adaptation for Case 2/medium and Case 3/large projects by completing the postmortem analysis within the minimum duration as prescribed by the existing XP for small [15, 27, 28]. There is another evidence of improvement in quality of the extended XP as the effort in Case 1/small project is 4% that is less than 4.7% of the recommended effort of existing XP for small projects.

**Table 2** Time of postmortem analysis

| Releases | Time, h | | |
|---|---|---|---|
| | Case 1 | Case 2 | Case 3 |
| 1 | 2.77 | 3.18 | 4.36 |
| 2 | 1.75 | 3.35 | 3.70 |
| 3 | 1.05 | 2.63 | 3.25 |
| 4 | 0.95 | 2.13 | 3.5 |
| average | 1.6 | 2.8 | 3.7 |

**Table 3** Effort of postmortem analysis

| Releases | Effort, % | | |
|---|---|---|---|
| | Case 1 | Case 2 | Case 3 |
| 1 | 5.67 | 11.35 | 15.56 |
| 2 | 3.67 | 8.49 | 8.65 |
| 3 | 3.63 | 4.83 | 6.44 |
| 4 | 3.36 | 3.5 | 5.9 |
| average | 4 | 7 | 9 |

Tables 2 and 3 show that there is an average increase in time and effort from Case 1 (small) to Case 2 (medium) and Case 3 (large-scale) projects because of increase in the size of the releases that is 8 kilo line of code (KLOC) in Case 1, 47 KLOC in Case 2 and 63 KLOC in Case 3. It can also be observed from Tables 2 and 3 that there is gradual decrease of time and effort from release to release in the three projects. The main reason, to decrease the time and effort from release to release, is because of the learning of the researcher and teams about the experiences of postmortem analysis as they progressed from release to release.

In Table 4, the three extended XP projects are compared with an existing XP project by taking the relative ratios of quality assurance (QA) to prove the point that quality of extended XP projects is better or less than quality of existing XP model. Sajid and Jongmoon [29] described that it is possible to compare the projects by taking the relative ratios for QA to provide evidence of quality. Fault rate per KLOC (the four releases) for 'analysis and risk management', 'design and development' and testing as a complete for each release of Case study 1, Case study 2 and Case study 3 so to calculate the average rate. This is accomplished by comparing the fault rate per KLOC for the

**Table 4** Fault rate/KLOC

| | 1 | 2 | 3 | 4 | Average |
|---|---|---|---|---|---|
| Case 1 | 2.84 | 2.44 | 2.41 | 1.97 | 2.41 |
| Case 2 | 3.64 | 3.36 | 3.22 | 2.16 | 3.09 |
| Case 3 | 4.06 | 3.93 | 3.85 | 3.04 | 3.72 |
| existing XP case study [30] | 2.19 | 2.10 | 2.04 | 8.70 | 3.75 |




doi: 10.1049/iet-sen.2011.0110

four releases of three case studies with four releases of a case study of the existing XP model in Table 4. The details about the existing XP case study can be found in [30]. The average rate shows a gradual increase in fault rate/KLOC as the size of the project increases for the three case studies.

In the three case studies the average rate for fault rate per KLOC is 2.41 in Case 1, 3.09 in Case 2 and 3.72 in Case 3 whereas it is 3.75 fault rate per KLOC in an existing XP case study. The average fault rate/KLOC for the three case studies is less than what is addressed by existing XP model indicating that quality of the extended XP model is better than the existing XP. The improvement in quality of the extended XP model is because of the proposed changes in the life cycle phases incorporated into existing XP model justifying its proposal.

## 7 Conclusions and future work

The agile models are a desirable replacement of traditional heavyweight models for the software development companies at the current time. XP is most widely used, documented and accepted model among all agile models. It was proposed for simple and small-scale projects. Software industry has to deal with medium and large projects as well. XP model has many success stories and there is a need for its modification for medium and large projects. Many efforts are made to extend existing XP model but still detailed methods are required in support of XP model to select, tailor and deploy it to meet the requirements of software industry. In this paper an extended XP model is proposed to address the limitations of classical XP methodology that is suitability of medium and large projects. Suitability of the proposed extended XP model is evaluated by small, medium and large projects by applying it to the three separate case studies (one for each). Empirical data are gathered using participant observation and direct observation based on the four releases of the three case studies.

The results are concluded mainly based on the postmortem analysis and fault rate per KLOC. The extended XP model indicates its adaptation for medium and large projects by completing the postmortem analysis within the 2–4 h (as recommended for small projects). An average increase in time and effort for the postmortem analysis is noticed because of increase in KLOC of projects whereas there is an average decrease in time and effort from release to release owing to learning of the author and development teams. The extended XP projects are compared with an existing XP project by taking the relative rations of QA to prove the point of quality that extended XP model is better, equal or less than the existing XP model. It is observed that quality of the extended XP model is better than the existing XP model because of less number of fault rate per KLOC for the three case studies. From the evaluation, the author has found evidences that the proposal of extended XP model is suitable for development of medium and large projects. However, the need for the statistical validation of extended XP model proposal will be addressed in days to come by comparing with future releases/versions.

## 8 References


1 Stapleton, J.: 'DSDM: business focused development' (Addison Wesley, London, 2003)
2 Highsmith, J.: 'Adaptive software development: a collaborative approach to managing complex systems' (Dorset House Publishing, New York, 2000)
3 Williams, L., Cockburn, A.: 'Agile software development: it's about feedback and change', *Computer*, 2003, **36**, (6), pp. 39–43
4 Boehm, B., Turner, R.: 'Management challenges to implementing agile processes in traditional development organizations', *IEEE Softw.*, 2005, **22**, (5), pp. 30–39
5 Eckstein, J.: 'Agile software development in large-diving into the deep' (Dorset House Publishing, New York, 2004)
6 Lindvall, M., Muthig, D., Dagnino, A., *et al.*: 'Agile software development in large organizations', *IEEE Comput.*, 2004, **37**, (12), pp. 26–34
7 Pressman, R.S.: 'Software engineering' (McGraw Hill, 2009)
8 Hoffer, J.A., George, J.F., Valacich, J.S.: 'Modern system analysis & design' (Pearson Education, 2002)
9 Murru, O., Deias, R., Mugheddu, G.: 'Assessing XP at a European internet company', *IEEE Softw.*, 2003, **20**, (3), pp. 37–43
10 Rumpe, B., Schroder, A.: 'Quantitative survey on extreme programming projects'. Proc. Third Int. Conf. on Extreme Programming and Flexible Processes in Software Engineering (XP2002), Alghero, Italy, May 2002, pp. 95–100
11 Beck, K.: 'Embracing change with extreme programming', *IEEE Comput.*, 1999, **32**, (10), pp. 70–77
12 Cao, L., Mohan, K., Peng, X., Ramesh, B.: 'How extreme does extreme programming have to be adapting XP practices to large scale projects'. Proc. 37th Annual Hawaii Int. Conf. on System Sciences, Hawaii, 2004, p. 30083c
13 Boehm, B.: 'Get ready for Agile methods with care', *Computer*, 2002, **35**, (1), pp. 64–69
14 Turk, D.E., France, R.B., Rumpe, B.: 'Assumptions underlying agile software-development processes', *J. Database Manage.*, 2005, **16**, (4), pp. 62–87
15 Outi, S.: 'Improving software process in agile software development projects: results from two XP case studies'. Proc. 30th EUROMICRO Conf., France, 2004, pp. 310–317
16 Outi, S., Kari, K., Pekka, K., Jani, L., Sanna, S., Abrahamsson, P.: 'Self-adaptability of agile software processes: a case study on post-iteration workshops'. Proc. Fifth Int. Conf. on Extreme Programming and Agile Processes in Software Engineering, Germany, 2004, pp. 184–193
17 Outi, S., Minna, P., Jari, S.: 'Deploying agile practices in organizations: a case study'. Proc. European Conf. on Software Process Improvement (EuroSPI 2005), Hungary, 2005, pp. 16–27
18 Abrahamsson, P., Juhani, W., Mikko, T.S., Jussi, R.: 'New directions on agile methods: a comparative analysis'. Proc. 25th Int. Conf. on Software Engineering, Portland, OR, 2003, pp. 244–254
19 Cao, L.: 'Modeling dynamics of agile software development'. Companion to the 19th Annual ACM SIGPLAN Conf. on Object-Oriented Programming Systems, Vancouver, BC, Canada, 2004, pp. 46–47
20 Schwaber, K.: 'Agile project management with scrum' (Microsoft Press, 2004)
21 Schwaber, K., Beedle, M.: 'Agile software development with scrum' (Prentice Hall, 2002)
22 Jachi, B., Frank, P., Dietmar, S.: 'An agile process for developing diagnostic knowledge systems', *KI J.*, 2004, **18**, (3), pp. 12–16
23 Lucas, L., Laurie, W., William, K., Annie, I.A.: 'Toward a framework for evaluating extreme programming'. Proc. Eighth Int. Conf. on Empirical Assessment in Software Engineering, Edinburgh, Scotland, 2004, pp. 11–20
24 Lucas, L., Laurie, W., Lynn, C.: 'Motivations and measurements in an agile case study', *J. Syst. Archit.*, 2006, **52**, (11), pp. 654–667
25 Beck, K.: 'Extreme programming explained: embrace change' (Addison Wesley, 2000)
26 Kuppuswami, S., Vivekanandan, K., Ramaswamy, P., Rodrigues, P.: 'The effects of individual XP practices on software development effort', *ACM SIGSOFT Softw. Eng. Notes*, 2003, **28**, (6), pp. 1–6
27 Dingsøyr, T., Hanssen, G.K.: 'Extending Agile methods: postmortem reviews as extended feedback'. Fourth Int. Workshop on Learning Software Organizations, Chicago, Illinois, USA, 2002, pp. 4–12
28 Cockburn, A.: 'Agile software development' (Addison-Wesley, Boston, 2002)
29 Sajid, I.H., Jongmoon, B.: 'Software quality assurance in XP and spiral – a comparative study'. Proc. Fifth Int. Conf. Computational Science and its Applications, Malaysia, 2007, pp. 367–374
30 Abrahamsson, P., Juha, K.: 'Extreme programming: a survey of empirical data from a controlled case study'. Proc. Int. Symp. on Empirical Software Engineering, USA, 2004, pp. 73–82