Ghulam Rasool; Shabib Aftab; Shafiq Hussain; Detlef Streitferdt

eXRUP: a hybrid software development model for small to medium scale projects

Original published in:
Journal of Software Engineering and Applications, 6 (2013), 9, pp. 446-457.
ISSN (online): 1945-3124
DOI: 10.4236/jsea.2013.69055
URL: http://dx.doi.org/10.4236/jsea.2013.69055
[Visited: 2015-06-02]

Journal of Software Engineering and Applications, 2013, 6, 446-457
http://dx.doi.org/10.4236/jsea.2013.69055 Published Online September 2013 (http://www.scirp.org/journal/jsea)

eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects

Ghulam Rasool[1*], Shabib Aftab[1], Shafiq Hussain[2], Detlef Streitferdt[3]

[1]Department of Computer Science, COMSATS Institute of Information Technology, Lahore, Pakistan; [2]Department of Engineering, Computing and Technology, University of Sunderland, Sunderland, UK; [3]Software Architectures and Product Lines Group, The Ilmenau University of Technology, Ilmenau, Germany.
Email: *grasool@ciitlahore.edu.pk

Received June 12th, 2013; revised July 15th, 2013; accepted July 23rd, 2013

ABSTRACT

The conventional and agile software development process models are proposed and used nowadays in software industry to meet emergent requirements of the customers. Conventional software development models such as Waterfall, V model and RUP have been predominant in industry until mid 1990s, but these models are mainly focused on extensive planning, heavy documentation and team expertise which suit only to medium and large scale projects. The Rational Unified Process is one of the widely used conventional models. Agile process models got attention of the software industry in last decade due to limitations of conventional models such as slow adaptation to rapidly changing business requirements and they overcome problems of schedule and cost. Extreme Programming is one of the most useful agile methods that provide best engineering practices for a good quality product at small scale. XP follows the iterative and incremental approach, but its key focus is on programming, and reusability becomes arduous. In this paper, we present characteristics, strengths, and weaknesses of RUP and XP process models, and propose a new hybrid software development model eXRUP (eXtreme Programming and Rational Unified Process), which integrates the strengths of RUP and XP while suppressing their weaknesses. The proposed process model is validated through a controlled case study.

Keywords: Process Models; Agile Modeling; eXtreme Programming; Rational Unified Process; Process Evaluation

1. Introduction

With the gradual passage of time, the conventional software development process models have been in the process of replacement by lightweight agile software development methodologies. The conventional models are presented and discussed in various papers [1-4]. In the last decade, agile models got attention of the software industry due to their unique features, such as quick response to requirement changes, reduced/pragmatic documentation and agility [5]. Authors of studies [6,7] refer to agile models as lightweight compared to conventional heavyweight models. The important characteristics of agile models are the incremental development style, a cooperative development, straightforward and adaptive process steps. These characteristics can be measured by small releases, continuous feedback, collective ownership and small team size.

XP is the most popular agile model widely used in various organizations and software industry [8-10]. It is a lightweight and fast agile software development process model for simple and small scale projects. Its basic working disciplines/values are simplicity, feedback, communication and courage. XP is suitable for small teams of 2 to 10 people and its basic working metaphor is that "the whole team is working together on a single table" [11]. It is person centric rather than process oriented [12]. XP follows an iterative and incremental approach; heavily focusing on constant customer collaboration, delivering early release through small iterations, it provides low bug rates and frequent adaptation of changing business requirements [13,14]. XP also helps the developers to constantly identify and work on the highest priority artifacts of the software. It has the capability to manage and handle the frequently changing business requirements which are the main reason for XPs tendency to be within budget even with changing business requirements. Due to constant customer feedback, XP has a positive effect on the correspondence of the requirements to the application, the final application better fits to customer desires [15] and is of good quality due to lower software bugs rates.

*Corresponding author.

XP strengths also include fast development, cost saving, high satisfaction of client and test driven development. Results of these strengths include less errors and acceptance of changes at any stage with minimum cost. XP practices include Planning Game, Small Releases, Metaphor, Simple Design, Tests, Refactoring, Pair Programming, Collective Ownership, Continuous Integration, 40-hour Week On-site customer, and Coding Standards [16].

The Rational Unified Process (RUP) is an incremental, iterative and architecture centric framework, based on sound software engineering principles [17]. It is a well defined process model that provides step by step guidelines to develop object oriented software applications. RUP has been evolved in different areas in different situations. RUP provides a very structured and formalized process for software development through its deep planning, thorough analysis, best design practices, codified process and extensive documentation. The key features of RUP are case driven design using, tailoring and tool support processing [18]. RUP or any other document driven approach is very beneficial when it deals with large scale projects due to its straightforwardness. These approaches are also providing better predictability and high assurance due to its potential benefits for large scale projects [18]. However, RUP can be tailored according to requirements of customers for medium scale projects.

Both XP and RUP share common features; they are iterative, customer oriented and role based [19]. RUP comprises 100 artifacts while XP focuses only on code. Similarly, RUP has 40 roles while XP has only 5 roles. These similarities and differences motivated us to analyze and integrate best features of both models. The feature shows that the advantage of one model is unavailable or limited in the other model. eXRUP has all the advantages of both the models by reducing the limitations of both models to a minimum level. The important feature of eXRUP is that even after the integration, it is lightweight and easy to understand for the developers and other stakeholders. The rest of paper is organized as follows.

Section 2 discusses the related work. Section 3 discusses and compares the features of RUP and XP and maps key features with proposed model. The proposed eXRUP model is presented and discussed in Section 4. Section 5 discusses the case study and its evaluation. The validity of proposed model is discussed in Section 6. Finally, Section 7 presents conclusion and future work.

2. Related Work

We discuss some important studies which focused on integration of various models and they are similar to the eXRUP model proposed in this paper. To the best of our knowledge, different authors worked on the integration of models but no work is reported on the integration of RUP and XP.

Permeation of RUP and XP on small to medium scale projects is presented in [20]. Authors presented theoretical study which focused on comparing features of RUP and XP. These discussed and compared activities of both models during phases of primary investigation, analysis and design, implementation and transition. Authors claim without any practical case study that the proposed model is more efficient than XP and RUP models as it exploits human experience during software development. Such claims need the evaluation of model by developing different software applications and especially feedback from the industry.

Authors in [19] presented Contrasts or Synonyms of RUP and XP and they concluded both models have some common character tics but they are quite different. The key focus of paper was to compare similarities and differences of both models based on a framework. They concluded that the selection of both models for different types of projects needs to be investigated empirically. They did not propose any new model based on the characteristics of both RUP and XP.

Reference [21] presented a model which integrates features from SCRUM and RUP. Authors evaluated the proposed model on a case study. They did not implement the same case study using SCRUM and RUP separately. The key objection which could be risen is that there is no need of integrating SCRUM and RUP according to the authors, they have used the RUP for project management activities of SCRUM but the RUP itself has a total of 9 workflows and three of them totally deal with the project management such as project Management, environment and configuration & change management. Moreover, the proposed model is still pretty much a traditional software development model.

A new process model by amalgamating best characteristics of XP, Scrum and RUP namely SPRUL is presented in [22]. Authors claim that proposed integrated model will be effective and efficient by satisfying customer and business needs. They conducted a controlled case study to validate their model but did not compare presented model with other process models. The usability and effectiveness cannot be measured without compareson and empirical evaluation.

In publication [23], the integration of XP with Scrum is presented which combined the advantages of both models and reduced their limitations. There is no doubt that XP in contrast with Scrum has amazing engineering practices and Scum in contrast with XP has extensive project management activities and these two basic features leads towards the proposed integration in this research. The main limitation of the presented model is the scope of its validation. The proposed model is validated through a controlled case study and a comparison with

published case studies but cannot be justified because these applications have different sizes, parameters and conditions for development. Furthermore the research is also silent on the question that the process model is good for which type and size of projects (small, medium, large)?

Reference [24] used the combination of XP and Scrum in the department of Software Engineering Services which is a part of Philips Research Organization in the Netherlands. The purpose was to add some critical success factors like Delivery on time, Quality and Scope of functionality in their projects to qualify for the certifications of both CMM Level 2 and ISO9001. Initially, team started using XP in daily practice which resulted in high customers and programmer's satisfaction. After working with only XP for a period of one year, the developers identified some issues regarding the use of XP model. They identified that XP did not help them in determining how to interact with the management as well as how to improve the way of working. Moreover, the customers provided ambiguous requirements which imposed difficulties for the developers to perform automated testing for all requirements using the XP model. They also felt the need of including non-functional requirements in the user stories along with the functional requirements. In order to address these issues, the author decided to merge XP and Scrum when he discovered the solutions of these issues using Scrum. He combined the engineering practices of XP with managerial and organizational aspects of Scrum. The use of a combined model XP@Scrum enabled the company to be certified according to ISO9001: 2000.

3. Characteristics of RUP and XP

It is evident from studies [25,26] that RUP and XP have common and varying characteristics. Here we discuss strengths and weaknesses of both models and map their characteristics with the proposed eXRUP model.

RUP is a complete framework supplemented with tool support and it can be customized for different projects according to customer requirements. The major strength of RUP is its structured and methodical approach which assures the process stability and a high quality of the developed products. RUP follows use case driven, architecture centric, incremental and iterative approach. However, RUP has following weaknesses which are highlighted by different other authors as well:

- RUP is a complex methodology and it is difficult to learn and apply it correctly on all type of projects [27, 28].
- While developing software according to RUP, an expert who has already developed such type of projects is necessary in the team to get high quality software [28].
- It is process oriented and does not focus on people at all [29].
- It works well with the large projects due to its complexity and heavy documentation [30,31].
- In RUP there is much focus on documentation, accommodating changes in the software is time consuming and difficult as first changes are implemented in use cases then in remaining diagrams and then in code [32].

Like other conventional software development methodologies, RUP also slowly adopts the frequent change in business requirements due to its complexity and heavy documentation. This is also the reason that the projects developed by RUP have the tendency to be over budgeted and behind the schedule [33].

XP is a lightweight methodology which has major stress on coding, communication, feedback, simplicity and problem solving [34]. It involves best engineering practices and accommodates rapidly changing requirements with quick feedback from customers However, we observed following weakness of XP:

- XP is suitable only for small scale projects and does not provide structured approach for medium and large scale projects [35-37].
- XP follows the code centered approach rather than design centered approach. Lack of design approach might go well with the small scale project, but when the scope of the project or team members grows then it is not suitable at all [22,38,39].
- XP model is a disciplined software development approach which is characterized by the continuous feedback, communication and courage [40].
- XP supports less or no documentation which makes it suitable only for small scale projects and this feature of XP make it difficult to get the benefits of reusability [22,38,39].
- XP document the project after coding which is itself a difficult and time consuming task. This feature of XP becomes almost impossible when the scope of the project grows [22,38,39].
- XP totally depends upon testing for its quality; however lack of structured reviews ultimately brings more time consumption in testing and lack of quality [22, 38,39].
- XP does not support global software development [22, 38,39].

Finally, we analyzed the characteristics of both models and map their activities, artifacts and roles to our proposed model as shown in **Table 1**. The list of characteristics in **Table 1** is indicative and not exhaustive.

4. Proposed eXRUP Model

The proposed eXRUP model integrates best RUP practices into XP phases. In this section, we discuss phases,

Table 1. Comparison of characteristics of RUP, XP and eXRUP.

Activities/Artifacts	RUP	XP	eXRUP
Requirement analysis and business modeling	Vision document Use-Case analysis	User Stories Communication Feedback On-site customer	User Stories
Analysis & Design	Preliminary architecture design, UML Diagrams (Class Diagram, Sequence Diagram, Collaboration Diagram, Activity Diagram)	Simple Design System Metaphor	Use case diagrams, Class, diagrams, Sequence diagrams
Implementation/Development	use-cases prototypes Architecture prototype	Small Releases Continual Integration Collective Ownership Refactoring Pair programming	Small Releases Pair programming Testing, Validation
Project Management	Project Schedule Defined Project Plan Status Assessment document	Story Estimates Iteration Plan	Project plan
Customer Team	Customer	System Analyst, Project manager	Project Manager
Project Size	Medium to large	Small	Small to medium
Configuration & Change Management	Yes	No	Yes
Integrated tool support	Yes	No	No
Focus on teams	Yes	Yes	Yes
Pair programming	No	Yes	Yes
Iterative software development	Yes	Yes	Yes
Tailoring	Yes	No	Yes

iteration cycle and practices of proposed model. The architecture of proposed model is presented in **Figure 1**.

4.1. eXRUP Phases

This is the first phase of eXRUP iteration which has following activities.

4.1.1. Initialization Phase

This is the first phase of our proposed model and it has 2 logical activities namely requirement gathering and project planning. We perform necessary tasks such as requirements gathering, project planning in this phase before starting the iterations because when the iteration is started then customer hardly has a chance to give feedback in this phase (during the iteration) until he/she wants to change the overall project plan or project scope. Furthermore in this phase, it is finally decided how many iterations are needed in current project.

1) Requirements Gathering

In this activity the customer/stakeholder elaborates all the features and requirements needed in the project. The project manager can assign this activity to any particular member/s of the team and can involve himself according to the nature of the project. All the requirements of customers are known as user stories in this process, which are written on story cards. Each story card clearly describes an individual feature, which should be in the project. These requirements are further categorized as functional and non functional requirements.

2) Project Planning

This is the key activity of the Initialization phase which keeps the functional and nonfunctional requirements on true direction towards the success of the project within limited time, budget and resources. This activity starts with the consensus of customer, project manager and development team on the project scope. Further, it includes budget estimation based on requirements, requirement prioritization, iteration time, software architecture diagram, effort estimation, resource estimation, risk identification and tool/technology selection.

4.1.2. Evolution Phase

This is the first phase of eXRUP iteration which has following activities:

1) Analysis

This activity starts with the risk monitoring plan which is optional and will only work when the risks are involved. The risks which were identified in the project planning are analyzed deeply at the start of this phase. Risk analysis explicitly involves the project manager. He plans the monitoring and controlling strategy for identified risks after analyzing the nature of the risks and may

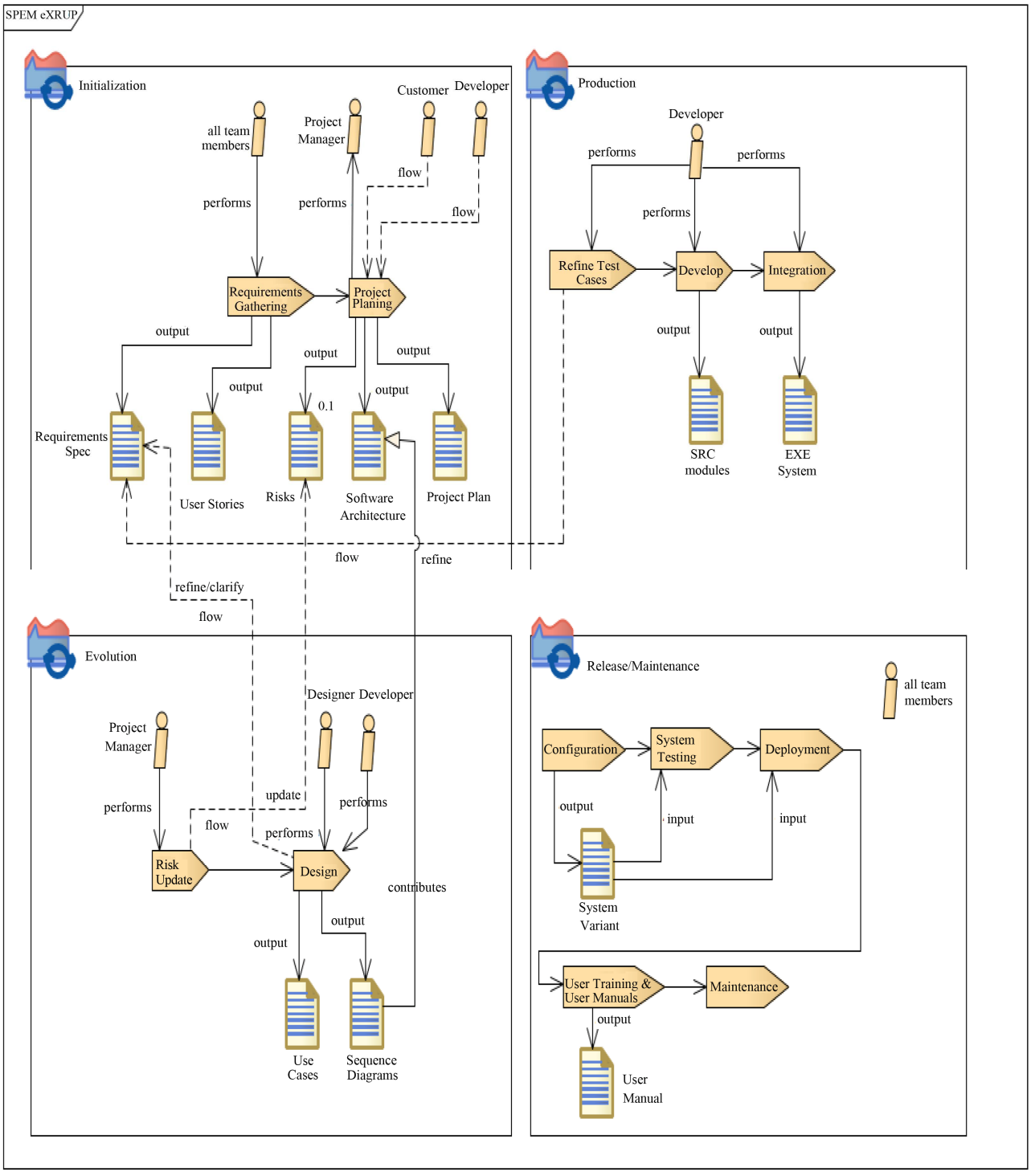


Figure 1. Architecture of proposed eXRUP model.

assign extra duties of other team members for the review and testing of risky part of the software. The risk identification and management is optional and depends upon the nature of the project.

2) Design

Design activity of this phase focuses on UML as it is used for the visualizing, constructing, specifying, and documenting the software. It is platform independent and has become the industry standard. We used only use-case diagrams, class diagrams and sequence diagrams. In eXRUP, our purpose is to keep track of changes, document the project and breaking the monopoly of develop-

ers. In eXRUP these diagrams will be used as the abstraction of the detailed software design. It tells the developers that how the system will work. The architecture is the most crucial aspect of the software which is used to control the iterative development of the project throughout its lifecycle.

4.1.3. Production Phase

It is the third phase of the iteration and consumes more time as compared to other two phases of iteration (initialization and maintenance). This phase ensures the development of test cases, development (coding) of modules/sub-systems according to user stories/requirements and validation of modules/sub-systems using validation techniques to make sure that there is no bug and error in that module.

4.1.4. Maintenance Phase

This is last step of eXRUP iteration cycle and consumes more resources than evolution phase and fewer resources than production phase. In this phase, team have to manage the system which has been released to the customer and also take care of the integration with the previously developed/released module as well as integration testing of the product.

4.1.5. Release Phase

This is the last phase of the process and work starts on it when iteration process is complete and developed product is error free. Release phase follows following activeties:

1) Deployment

In this phase, the completed software product is deployed on the customer site. Chances of errors are minimized as testing is conducted at the end of each iteration. Configuration management is also important and ensured during this phase.

2) User Training & User Manuals

User training is important in this phase as all the users have to be trained for their particular interface. Software may have many interfaces (such as data entry interface, admin interface) and particular user/group of users interact with the particular interface so each user group is trained according to his desired part of the software and then they can work better with the software in their environment. User manuals and documents are finalized in this phase for training of users. They can take help from the manuals at any later stage or at that time. When they hire any new person to operate the software then these manuals will be very helpful for the training of the new users.

3) Alpha System Testing

Alpha testing is performed in which group of customers/potential users test the software at developers site before the deployment of the software at customer site. Any error or bug can be reported back to maintenance phase because keeping the previous release/sub-system in working condition is the key responsibility of this phase.

4.2. eXRUP Iteration

eXRUP follows an iterative approach which helps the developers in understanding the problem gradually and provides them a way to solve that particular problem with incremental approach. The whole project in each iteration cycle is divided into different releases. The SQA related activities are performed on each iteration of the project and each iteration of eXRUP produces a subsystem/module of the whole product. The project manager controls the project velocity to complete the project within limited time frame. Finally, all the necessary decisions in project planning (in project initialization phase) such as prioritized list of requirements, risk analysis, iteration time analysis and cost & benefit analysis data is transferred to the business modeling part of this phase. In business modeling use case diagrams are created on the basis of selected requirements (functional and non-functional). These use cases are then transferred to the production phase where developers write test cases on the basis of use cases which are followed for the development activity.

Each eXRUP iteration cycle starts with the Evolution phase. In Evolution phase, all those selected requirements and the risks (optional) are analyzed. Monitoring and controlling strategies for the risks are included in the analysis. Project manager decides in consultation with team members whether serious risks shall be involved in the project or not? If they are involved then the risk analysis includes that how much they could be vulnerable for the project, for environment or for people. If there are no risks involved in the project then data is transferred to the business modeling part of this phase. The working of iteration cycle is presented in **Figure 2**.

With reviews and testing developers integrate the developed story in maintenance phase and perform integration testing to validate the input/output flow between different subsystems. Now if they feel any problem in the integration then can go back to production phase and then evolution phase, if problem is related to design. In iteration customer can interact in any phase and may present any change request.

4.3. eXRUP Practices

These are the practices and the key principles for the eXRUP model. Implementation of each activity/practice/ workflow can influence a software project positively. eXRUP practices are given below:

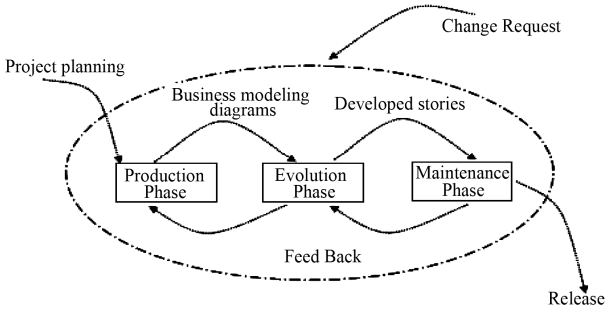


Figure 2. eXRUP iteration cycle.

1) Software Quality Assurance (SQA)

Our key emphasis is that final product should fulfill the requirements of the customer in terms of reliability and functionality. The parameters of quality like application performance and system performance must be ensured by proper testing and reviews while keeping the environment in mind where the software is going to work. In eXRUP quality assessment and assurance is performed with the following sub activities:

Project and Environment management.

Configuration and Change Management.

Testing.

2) Iterative Development

eXRUP provides an iterative approach of software development so the testing is an ongoing process throughout the development cycle of software. This approach addresses change requests and new requirements at any stage and also reduces the overall cost to detect the defects at early stages. eXRUP also ensures the quality product because of its qualitative testing efforts (functional testing, integration testing, alpha testing, beta testing), as quality is ensured in four perspectives reliability, functionally (functional and non-functional requirements) and performance of application and system (environment for which the product is going to built).

3) Continuous Integration

This practice ensures the continuous integration of the code. This practice of eXRUP eliminates the problems which occur due to late integration, as continuous integration is less difficult than integrating the system in later stages. The cost of fixing bugs, which occurs in early and continuous integration, is far less than the integration at once in later stages. This practice can be easily implemented by dedicating a machine for testing. The pair with completed work can sit on that system, integrate their part of code, runs the test and fix the bugs.

4) Deployment

The purpose of this practice is to deliver the system to the end user/stakeholder. It includes the deployment strategies, supporting materials/user manuals, performing alpha and beta tests, installation of the complete software, migration to the new system and database, and training of the end users in their respective environment.

5) Pair Programming

Through this practice two developers work together on the same computer. One developer writes test cases and develops the software while other one continuously reviews the test cases and the coding. This exercise not only brings the two brains together on the same table to work simultaneously on the same part of code but also reduces the bug rate. Pairing exercise is dynamic which means that the two developers work together at one time may work with other individuals in second time. So any developer got the task for which he does not have much experience may have a partner/pair who has, and then produce a qualitative product by working with partner.

6) Collective Code Ownership

This practice ensures that the whole code belongs to every member of the team. This practice differs from other two ownership strategies: individual ownership and no ownership. In individual ownership the code belongs to the single person and if any other team member wants to change the code due to any reason then he/she has to submit the request to the owner of that code. On the other hand in no-ownership strategy any one could change the code according to his need and this could result in many problems. In such scenario code reflects the change and grows quickly but also vulnerable to bring down the system as change code may had a relationship with any other code and due to the change that integration may have got down. Collective ownership reduced problems in our proposed model as according to this practice every team member is equally responsible for the ownership and the improvement of code is easy in pair programming.

7) Coding Standards

This practice ensures that the proper coding standards should be followed by each developer. If we want to get the advantages from "collective ownership" and "pair programming" then to follow "coding standards" is essential because following the same standards by all developers will not only boost the quality development but also understand the code by any other developer would also be easy. Furthermore for developers, this practice also gives the advantage of consistency in naming conventions of programming elements such as modules, packages, classes and functions.

8) Business Modeling

All the previous practices improved quality of developed product but a gap was observed between the business engineering process and the software development process. This gap could lead to a project which may not fully respond to the stakeholder's requirements and may need further working by the development team results in the product which is overall behind the schedule. The proposed eXRUP reduces this gap by using business modeling diagrams (use case diagrams, class diagrams and sequence diagrams).

9) Management of Requirements

The purpose of this discipline is to elaborate that what exactly the system would do? Then stakeholders and the development team both agreed upon the description of functionality of the system. For this purpose, requirement elicitation and organization is performed by the development team. These requirements are documented after categorized as functional and non functional requirements (constraints).

10) Component based development

eXRUP provides the component based architecture in which gradually components/increments are built and integrated to a full product through iterations. Component based development provides the feature of reusability, which means once developed component, can be further integrated in any other relevant project. As eXRUP focuses on iterative development, in which each iteration gives us a component. Each component provides a specific functionality.

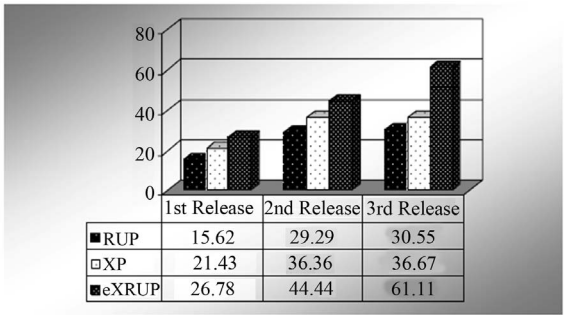
5. Case Study and Evaluation

The proposed eXRUP model is validated through a controlled case study. The primary intention of conducting this case study was to develop same application by three different teams of students under the supervision of one project manager in the same environment. A Portal for Real Estate was developed using PHP, Macromedia Dream viewer 8, Net beans, MS Visio and Apache Wamp Server for a client. The source code of developed application cannot be publicized due to licensing issues. A training session was conducted before the start of each iteration. The total numbers of iterations is same in all the releases. The project team consists of three team members and total duration for each development was one month. The second author of this paper was a MSCS student and he played the role of project manager in each team using three models and other two members were programmers. The programmers were final year students of BSCS at Comsats Institute of IT, Lahore.

5.1. Evaluation Parameters

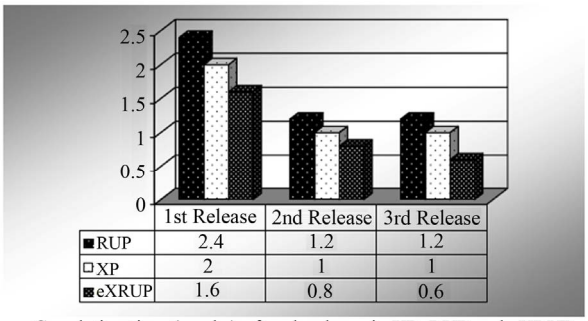
Software process models are evaluated for their usability and effectiveness on the basis of different parameters. Reference [23] presented 22 parameters which are important for evaluation of different process models. **Table 2** depicts results of three releases of our case study developed using RUP, XP and eXRUP models. We filtered some parameters which were redundant such as lines of codes and KLOC.

Figure 3 graphically presents comparison of important parameters of model used for implementation of case study. The other parameters of eXRUP can be seen in **Table 2**. It's very clear from **Table 2** that our proposed model has significant improvement in all parameters



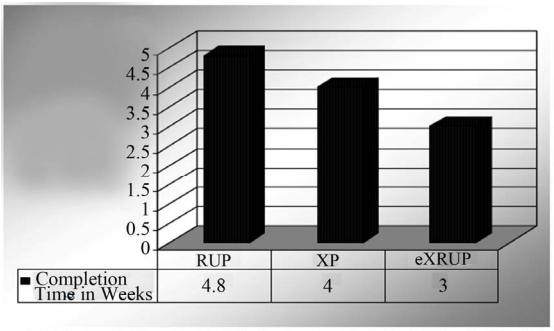
Productivity of each release in RUP, XP and eXRUP

(a)



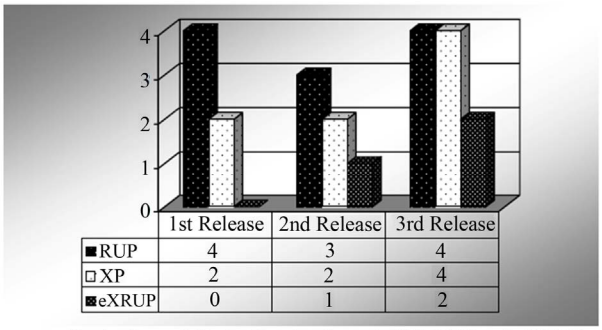
Completion time (weeks) of each release in XP, RUP and eXRUP

(b)



Total completion time (weeks) of project in XP, RUP and eXRUP

(c)



Post release defects of each release in XP, RUP and eXRUP

(d)

Figure 3. Productivity, completion time in each release, total completion time and post release defect.

Table 2. Evaluation results using three models.

ID	Parameters	RUP				XP				eXRUP			
		R1	R2	R3	Tot	R1	R2	R3	Tot	R1	R2	R3	Tot
1	Completion time duration (weeks)	2.4	1.2	1.2	4.8	2	1	1	4	1.6	0.8	0.6	3
2	Number of Modules	2	1	1	4	2	1	1	4	2	1	1	4
3	No of user stories	17	13	11	41	17	13	11	41	17	13	11	41
4	Total budgeted work effort (h)	288	144	144	576	240	120	120	480	192	96	72	360
5	Total actual work effort (h)	252	108	108	468	210	90	90	390	168	72	54	254
6	Number of User Interfaces	2	1	1	4	2	1	1	4	2	1	1	4
7	Number of Design Classes	45	33	30	108	46	34	30	110	45	33	30	108
8	Total KLOC	4.5	3.2	3.3	11	4.5	3.2	3.3	11	4.5	3.2	3.3	11
9	Post release change requests	1	0	0	1	1	0	0	1	1	0	0	1
10	Number of code Integrations	5	2	1	8	20	12	12	44	6	4	3	13
11	Post Release Defects	4	3	4	11	2	2	4	8	0	1	2	3
12	Post Release defects/KLOC	0.88	0.93	1.21	1	0.44	0.25	1.21	0.72	0	.31	0.60	0.27
13	Productivity (=line of code/ actual time spent)	15.6	29.2	30.5	23.2	21.4	35.3	36.6	28.2	26.7	44.4	61.1	37.4
14	Team Size	3	3	3	3	3	3	3	3	3	3	3	3
15	No. of pre-release change request	3	2	2	7	3	2	2	7	3	2	2	7
16	Total change requests/KLOC	0.66	0.63	0.61	0.64	0.66	0.62	0.60	0.63	0.66	0.62	0.60	0.63
17	Time duration to manage total change requests (h)	7	5	2	14	4	3	1	8	3	2.5	3.7	9.2
18	Pair Programming %	NA	NA	NA	NA	100	100	100	100	100	100	100	100
19	Total allocated actual spent time in %	87.5	75	75	81.2	87.5	75	75	81.6	87.5	75	75	81.6
20	Customer participation in %	10	10	10	10	80	85	85	83.3	30	30	30	30

R1: Release 1; R2: Release 2; R3: Release 3; Tot: Total.

which are used to evaluate the developed case study. We do not claim same percentage of improvement for other type of systems until the proposed model is evaluated by the academia and industry.

It is very difficult to compare results of presented model with other models because standard benchmark systems are not available for comparison. Different authors proposed integration of agile and conventional models and they evaluated performance of their models on different case studies. The nature of case studies vary in size, complexity, team structure, team size, team expertise, programming languages, tool support, customer requirements and various other factors. Due to these varying factors we developed same case study using three models one by one and results presented in **Table 2** reflect that our proposed model has significant improvement in productivity, it take less time to manage change requests per hour and same system was developed with less time as compared with XP and RUP.

5.2. Validity

Validity is the key challenge for researchers and practioners in conducting empirical research work. Reference [41] states that for empirical research to be acceptable as a contribution to scientific knowledge, the researcher needs to convince related academia and industry conclusions drawn from an empirical study are valid. Threats to validity of our results can be classified as:

5.2.1. Internal Validity

Internal validity is addressed more frequently in experimental studies. It is concerned with the consistency of the measurements, appropriate use of tools, and methods [42]. The implementation of real case study using RUP, XP and eXRUP validated proposed model. Internal validity is also affected by experimental bias. The same application was developed by three different teams in the same environment to reduce this threat as already pub-

lished case studies may have been developed with different intentions and environment conditions.

5.2.2. External Validity

Key concern while conducting external validity is whether the findings of the study can be generalized beyond the sample for which they were derived. To avoid this threat, the case study, and other tasks planned to conduct case study are designed keeping in view the schedule and knowledge level of students. The proposed model is an amalgamation of best features of RUP and XP which is validated through case study, but we cannot generalize the improvement in productivity and other attributes until it should be evaluated on different types of systems by academia and industry. Currently, we are working on different other projects using our proposed model which will evaluate threats to external validity.

5.2.3. Criterion Validity

Reference [43] discussed two types of validity criteria: Concurrent and Predictive validity. Concurrent validity uses an already existing and well accepted measure against which the performance of the new measure can be compared while predictive validity assesses the degree to which a measure can predict a future event of interest. The presented model demonstrates concurrent validity as results of eXRUP are compared results of RUP and XP. eXRUP exhibits predictive validity as the results of the case study can predict improvement in productivity and various other evaluation parameters of the delivered product when they will be compared with other conventional and agile models in future.

5.2.4. Construct Validity

Construct validity involve the relation between theory and observation. The proposed model can be customized according to nature of software and customers' requirements. The construct validity is established when proposed model is related to both conventional (RUP) and agile model (XP).

5.2.5. Reliability Validity

This reliability affects the replicability of our results. The proposed model is evaluated using standards parameters as used by other case studies. However, we cannot publish the implemented systems due to licensing issues.

6. Conclusion and Future Work

We present a hybrid software development process model named eXRUP which integrates best characteristics of XP and RUP models by suppressing their weaknesses. The objective of presented model is to develop high-quality, small-to-medium scale applications within budget and time constraints. The proposed model is validated through a controlled case study which is developed by three different teams by using three models. The results of case study proved that presented model has improved the productivity, performance, completion time and various other attributes as shown in **Table 2**, only with the exception of time duration to manage change requests. The improvement in productivity is 15% and completion time is reduced to 20% which is significant improvement. The proposed model will be evaluated on other types of medium size projects using different programming languages. We also plan to validate presented model from software industry. We strongly recommend the application of proposed model for web based applications where time to market is critical measure for success, but its effectiveness for other types of software is under experiments. One key question about proposed model is its generalization for large and complex systems which need to be empirically investigated. Second drawback of presented model is minimal interaction of developers with customers and higher management. Finally, time duration to manage total change requests is a little higher in eXRUP as compared with simple XP model.

REFERENCES

[1] N. M. A. Munassar and A. Govardhan, "A Comparison between Five Models of Software Engineering," *International Journal of Computer Science Issues* (*IJCSI*), Vol. 7, No. 5, 2010, pp. 94-101.

[2] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, "Agile Software Development Methods-Review and Analysis," VTT Publications 478, 2002.

[3] D. Truex, R. Baskerville and J. Travis, "A Methodical Systems Development: The Deferred Meaning of Systems Development Methods," *Accounting Management and Information Technologies*, Vol. 10, No. 1, 2000, pp. 53-79. doi:10.1016/S0959-8022(99)00009-0

[4] S. Cronholm, "Using Agile Methods?—Expected Effects," *Proceedings of* 17*th International Conference on Information Systems Development* (*ISD*2008), Paphos, 25-27 August 2008, pp. 913-924.

[5] N. Ganesh and S. Thangasamy, "Issues Identified in the Software Process Due to Barriers found during Eliciting Requirements on Agile Software Projects: Insights from India," *International Journal of Computer Applications*, Vol. 16, No. 5, 2011, p. 7.

[6] B. Boehm, "Get Ready for the Agile Methods, with Care," *Computer*, Vol. 35, No. 1, 2002, pp. 64-69. doi:10.1109/2.976920

[7] R. F. Roggio, "Process Driven Software Development: An Approach for the Capstone Sequence," *Proceedings of Information Systems Education Conference* (*ISECON*), Pittsburgh, 1-4 November 2007, pp. 234-242.

[8] J. Newkirk, "Introducing to Agile Processes and Extreme Programming," *Proceedings of* 24*th International Conference on Software Engineering*, Orlando, 25 May 2002,

pp. 695-696.

[9] P. Abrahamsson, "Extreme Programming: First Results from a Controlled Case Study," *Proceedings of* 29*th Euromicro Conference* (*EUROMICRO*'03), Antalya, 1-6 September 2003, pp. 259-266.

[10] L. Lindstrom and R. Jeffries, "Extreme Programming and Agile Software Development Methodologies," *Information Systems Management*, Vol. 21, No. 3, 2004, pp. 41-52. doi:10.1201/1078/44432.21.3.20040601/82476.7

[11] M. Grant, "Introduction to Extreme Programming". http://www.xprogramming.com

[12] A. Sillitti and G. Succi, "The Role of Plan-Based Approaches in Organizing Agile Companies," *Cutter IT Journal*, Vol. 19, No. 2, 2006, pp. 14-19.

[13] "Extreme Programming Official Website". http://www.extremeprogramming.org/map/project.html

[14] J. Osorio, M. Chaudron and W. Heijstek, "Moving from Waterfall to Iterative Development: An Empirical Evaluation of Advantages, Disadvantages and Risks of RUP," *Proceedings of* 37*th EUROMICRO Conference of Software Engineering and Advanced Applications*, Oulu, 30 August-2 September 2011, pp. 453-440

[15] A. Paul and P. A. Beavers, "Managing a Large 'Agile' Software Engineering Organization," *Proceedings of Agile Conference*, Washington DC, 13-17 August 2007, pp. 296-303.

[16] J. Newkirk, "Introduction to Agile Processes and Extreme Programming," *Proceedings of 24th International Conference of Software Engineering*, Orlando, 19-25 May 2002, pp. 695-696.

[17] P. Kroll and P. Kruchten, "Rational Unified Process Made Easy: A Practitioner's Guide to the RUP," Addison Wesley, Boston, 2003.

[18] P. Kruchten, "The Rational Unified Process—An Introduction," 2nd Edition, Addison-Wesley, 2000.

[19] Y. Dubinskyl, O. Hazzanz and A. Keren, "Introducing Extreme Programming into a Software Project at the Israeli Air Force," *Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering*, Sheffield, 18-23 June 2005, pp. 19-27.

[20] K. Fertalk, N. Hlupic and D. Kalpic, "Permeation of RUP and XP on Small and Middle-Size Projects," *Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics*, Tenerife, 16-18 December 2006, pp. 98-104.

[21] W. C. de Souza Carvalho, P. F. Rosa and M. L D. S. Soares, "A Hybrid Approach to Integrate Agile and Traditional Software Development Processes," *Proceedings of Jornadas Chilenas de Computación*, Curico, September 2011, pp.

[22] S. U. Nisa and M. R. J. Qureshi, "Empirical Estimation of Hybrid Model: A Controlled Case Study," *I.J. Information Technology and Computer Science*, Vol. 4, No. 8, 2012, pp. 43-50. doi:10.5815/ijitcs.2012.08.05

[23] M. R. J. Qureshi, "Empirical Evaluation of the Proposed eXSCRUM Model: Results of a Case Study," *International Journal of Computer Science Issues*, Vol. 8, No. 3, 2011, pp. 150-157.

[24] C. Vriens, "Certifying for CMM Level 2 and ISO 9001 with XP@Scrum," *Proceedings of Agile Development Conference* (*ADC*'03), Salt Lake City, 25-28 June 2003, pp. 120-124.

[25] P. Runeson and P. Greberg, "Extreme Programming and Rational Unified Process—Contrasts or Synonyms?" *Proceedings European Software Process Improvement Conference* (*EuroSPI*), Budapest, 9-11 November 2005.

[26] http://www-106.ibm.com/developerworks/rational/library/4156.html

[27] W. Hesse, "Dinosaur Meets Archaeopteryx? Seven Theses on Rational's Unified Process (RUP)," *Proceedings of 6th International Workshop on Evaluation of Modeling Methods in System Analysis and Design*, Marburg, 4-5 June 2001, 9 Pages.

[28] "The Advantages and Disadvantages/Best Practices of RUP Software Development". http://www.my-project-management-expert.com/the-advantaes anddisadvantaes-of-rup-software-development.html

[29] N. Shahid, O. A. Khan, S. K. Anwar and U. T. Pirzada, "Rational Unified Process," Online Notes on RUP. http://ovais.khan.tripod.com/papers/Rational_Unified_Process.pdf

[30] M. Hirsch, "Making RUP Agile," *Proceedings for Conference of Object-Oriented Programming*, *Systems*, *Languages & Applications*, New York, 4-8 November 2002, p. 44.

[31] E. G. Sally and K. T. Rudahl, "Software Process in the Classroom: A Comparative Study," *Proceedings of 9th International Symposium on Communications and Information Technology*, Icheon, 28-30 September 2009, pp. 427-431.

[32] T. Massoni, A. Sampaio and P. Borba1, "A RUP-Based Software Process Supporting Progressive Implementation," In: *UML and the Unified Process*, Chapter 3, IGI Publishing, Hershey, 2003, pp. 375-387.

[33] J. Cho. "A Hybrid Software Development Method for Large-Scale Projects: Rational Unified Process with Scrum," *Journal of Issues in Information Systems*, Vol. 5, No. 2, 2009, pp. 340-348.

[34] K. Beck, "Extreme Programming Explained Embrace Change," Addison-Wesley, Boston, 2000.

[35] J. Smith, "A Comparison of RUP and XP," White Paper, Rational Software.TP167, 5/01.

[36] A. Farrel, "Selecting a Software Development Methodology based on Organizational Characteristics," An Essay Submitted in Partial Fulfillment of the Requirements for the Degree of "Master of Science in Information Systems", Athabasca University, Athabasca, 2007.

[37] A. Sillitti, M. Ceschi, B. Russo and G. Succi, "Managing Uncertainty in Requirements: a Survey in Documentation-Driven and Agile Companies," *Proceedings of* 11*th IEEE International Software Metrics Symposium*, Como, 19-22 September 2005, pp. 10-17.

[38] P. Emery, "The Dangers of Extreme Programing," Term Paper, 2002. http://members.cox.net/cobbler/XPDangers.htm#_Toc530

042781

[39] P. Mattis, A. Trafford and A. Sakalaspur, "Extreme Programming". http:// csis.pace.edu/~ctappert/cs616-02/pres-xp.ppt

[40] A. Ullah, G. Rasool and R. J. Qureshi, "IXPRUM—A Novel Agile Model for Software Development," *AWER Procedia Information Technology and Computer Science*, Vol. 1, No. 1, 2012, pp. 1314-1320.

[41] S. Easterbrook, J. Singer, M. A. Storey and D. Damian, "Selecting Empirical Methods for Software Engineering Research," Springer, London, 2008.

[42] M. Voka, "Defect Frequency and Design Patterns: An Empirical Study of Industrial Code," *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, 2004, pp. 904-917. doi:10.1109/TSE.2004.99

[43] M. J. Neale and J. M. R. Liebert, "Science and Behavior: An Introduction to Methods of Research," Prentice-Hall, Upper Saddle River, 1986.

