A Risk-Driven Method for eXtreme Programming Release Planning

Mingshu Li [1,2], Meng Huang [1,3], Fengdi Shu [1], and Juan Li [1,3]
[1]Lab for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences
[2]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[3]Graduate University of Chinese Academy of Sciences
Beijing 100080, P.R. China

{mingshu, hm, fdshu, lijuan}@itechs.iscas.ac.cn

ABSTRACT
XP (eXtreme Programming) has become popular for IID (Iteration and Increment Development). It is suitable for small teams, lightweight projects and vague/volatile requirements. However, some challenges are left to developers when they desire to practise XP. A critical one of them is constructing the release plan and negotiating it with customers. In this paper, we propose a risk-driven method for XP release planning. It has been applied in a case study and the results show the method is feasible and effective. XP practicers can follow it to decide a suitable release plan and control the development process.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management –Cost estimation, Life cycle, Productivity.

General Terms
Management, Measurement, Economics, Experimentation.

Keywords
XP, release planning, risk, negotiation, decision

1. INTRODUCTION
Recently, eXtreme Programming (XP) is a popular Iteration and Increment Development (IID) practice and successful in software development [1,2,3]. It is based around the development and delivery of small increments of functionality, customer involvement in the process, constant code improvement and egoless programming [4].

Release planning is the activity that stakeholders decide requirements implementation and delivery scheme in XP practice. It is the key point, where developers consider how coding can contribute to software system's goals, since there are no detailed requirements and architecture design before system implementation. There are four steps during XP release planning: firstly, developers inquire customers about the business value of each story that describes the customers' needs; secondly, developers estimate the needed efforts for implementing those stories; thirdly, they analyze the technology risks in stories; and lastly, developers negotiate with customers about the release plan for the next iteration [5].

The vague XP release planning techniques often muddle practicers [6]. As a result, it is difficult for developers to make a suitable release plan in XP practice. They are often puzzled by many problems, such as how to assess stories' business value, how to analyze the technology risks, how to get consensus by negotiation, etc. Some researchers reported that release planning had become the response to customers' requests entirely and been uncontrollable because of the absence of integrated system's viewpoint, inadequate negotiation among stakeholders and so on. Thus, many XP processes are full of reworks and wrong problem solutions [3,7,8,9,10,11].

In the paper, we put forward a risk-driven method for XP release planning which aims at three main vague areas in XP practice. Firstly, the vague techniques of XP release planning often lead stakeholders to a poor decision when they do not consider multiple possible release plans for the next iteration. Secondly, stakeholders want to balance development risks and productivity, but it is difficult for them to analyze the risks because of the vague techniques in risks analysis. Lastly, it is a very important work to balance development risks and productivity in process, but there are not enough guides or techniques for stakeholders to reach an agreement about a release plan.

The remainder of the paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the risk-driven method for XP release planning. Section 4 presents a risk-driven XP process. Section 5 reports a case study of the method. We conclude the paper in Section 6 with a summary and directions for future work.

2. RELATED WORK
Release planning is one of the essential tasks in IID [6]. There are two kinds of release planning method for IID: predictive planning and adaptive planning [12]. Predictive planning means that developers make a detailed plan covering the whole software life cycle. On the contrary, according to the adaptive planning methodology, developers only make the detailed plan for a short time and keep a rough long-time plan. The detail plan covers a few release millstones or even only one iteration. Adaptive planning is more suitable for IID [12,13]. XP release planning is a typical adaptive planning. XP practicers plan release only for the

ICSE'06, May 20-28, 2006, Shanghai, China.

next iteration generally. The long-time plan keeps rough for easy adjustment.

In the early research of release planning, researchers focused on the method of assessing requirements' value and estimating cost [14,15,16]. Based on quantitive value evaluation and cost estimation of requirements, optimizing methods are used to select suitable combinations from all requirements [14,15,18] for iterations. Recently, some researchers concentrate on the inconsistent requirements' value from multi-stakeholder viewpoints [17,19] and dependencies among requirements [20,21]. They suggest that the inconsistent value and dependencies should be considered when the optimizing methods are used. However, these methods are not suitable for XP release planning. The premise of these methods is that there are detailed requirements and architecture design before implementation, which leads to a predictive plan naturally. In XP practice, this premise can rarely be satisfied [5].

The vague descriptions of techniques in XP method always make it difficult to develop, negotiate, and decide a release plan. To improve it, [12] uses "Dot Voting" or architectural significance, risk and value score to make a release plan. But some issues are still unclear such as negotiation. When XP practicers use the method for release planning, they still feel difficult in deciding what information they should collect and how to make feedbacks to customers to reach the agreement about a release plan. Some problems caused by unsuitable releasing plan in XP practice are reported such as that developers are unable to finish project for absence of integrated system viewpoints [9,10,22,23], inadequate negotiation during release planning [11], high development organization risks caused by excessive dependence on personal experiences [8], and so on. Therefore, an exact method is needed for developers to follow during XP release planning.

3. RISK-DRIVEN METHOD FOR RELEASE PLANNING

The risk-driven method for XP release planning is shown in Figure 1. It can be divided into three steps as follows:

Firstly, developers construct a set of feasible release plans from the project profiles which include those original ideas about the system's scope, cost, schedule, product quality and so on. In this step, developers not only write XP's stories based on the system's features as in general XP practice, but also combine stories into multiple feasible release plans according to their values, dependences, costs and available efforts per iteration. Some techniques, such as quantifying a story's business value using Analytic Hierarchy Process (AHP) method and constructing multiple feasible release plans automatically, are integrated in our method to help developers in this step.

Secondly, risks in each feasible release plan are analyzed. Risk analysis is used as the crucial tool when developers and customers plan releases. Risks are losses caused by uncertain things [27]; they maybe come from requirements, estimation or technologies and affect system scope, schedule or products quality.

At last, stakeholders decide a certain release plan for the next iteration according with the result of risk analysis. If there are not release plans whose risks are acceptable, developers should trace back and negotiate with customers about project profiles. Furthermore, after the iteration is finished, its results will be used to adjust project profiles.

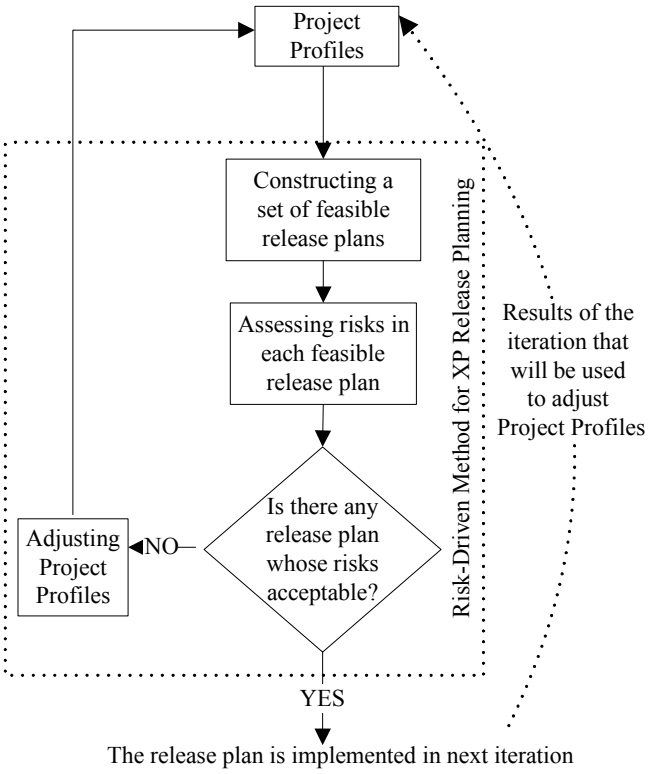

Figure 1. Risk-driven method for XP release planning

The method can be not only used for each XP iteration, but also applied for long-time plans such as milestone plans and even software life cycle. For example, developers use the method to get a release plan first for long-time milestone. Then, they use the method to get the first iteration's release plan. And when the first iteration is finished, they can adjust the project profiles by the results of the iteration. The adjustments lead to a more suitable plan for next iteration and further milestones.

Either planning for next iteration or for long-time milestone, the method can establish the links between the overview of the whole project and essential activities of stakeholders. This helps stakeholders construct a successful software product under the circumstance's constraints.

3.1 Constructing Feasible Release Plans

A feasible release plan means a group of requirements (stories) that can be completed in the next iteration under the iteration efforts and stories' dependencies constraints. Usually, the feasible release plan is not unique. Since developers and customers always expect to complete as many stories as possible per iteration, we have to define a criterion to evaluate the business value of stories in a release plan under the efforts and dependencies constraints. Accordingly, we can select those feasible release plans with higher business value as candidates. To construct multiple

feasible release plans and select those with high business value, developers should quantify every story's business value, size, available efforts per iteration, and locate the dependencies among stories.

The business value of a story is defined by customers. During release planning, the relative value should be assessed. Analytic Hierarchy Process (AHP) method is a useful tool to evaluate the stories' relative business value [24]. The stories' relative values are evaluated by performing pair wise comparisons and checking result consistency. After customers evaluate the stories' relative value by AHP, a number between 0 and 1 will be assigned to each story as a score that means the relative value of a story. And the sum of all scores is 1. In the paper, the symbol "V_i" denotes story i's business value.

Man-hour is used as the measurement of size estimation in XP. We use c_i to denote story i's size, C^t denotes available efforts of No. t iteration.

Although some researchers think the dependencies among stories are unimportant in XP release plan [5], we consider the dependencies as an important factor because a useful partial system should be delivered to customers after each iteration. The small release of each iteration should fulfill some customers' business needs. Consequently, the dependencies among stories caused by business process should be considered during planning release. The dependencies of stories, which mean the business process described by the stories, are denoted by d. If story i depends on story k, we endue d_{ik}=1. It means that story i cannot be implemented later than story k. If there is no dependency between i and k, we endue d_{ik}=0. For $\forall i = k$, d_{ik}=1.

The assumption is that there are n uncompleted stories in the storyboard. x_i=1 denotes story i will be completed in the next iteration; otherwise, x_i=0. Then, a function like formula (1) is designed to construct feasible release plans according to the criterion "maximizing business value per iteration".

$$Max \quad g=\sum_{i=1}^{n} V_i * x_i \tag{1}$$

Dependencies and available effort constraints can be expressed by formulas (2) and (3):

$$\sum_{i=1}^{n} c_i * x_i \le C^t \tag{2}$$

$$\forall x_i, x_k, x_i * d_{ik} \le x_k, i,k \in \{1,\cdots,n\} \tag{3}$$

Each solution of the function is an assignment combination of $\{x_1,...,x_n\}$ which means a feasible release plan. An algorithm is designed to find solutions. Firstly, the solution space is organized as Figure 2. From a node, e.g. node A, the left branch denotes a solution that includes x_i = 1; right branch denotes a solution that includes x_i = 0. A path from the root to a leaf denotes an entire solution. X_w denotes a path of the entire solution. In these paths, some are feasible solutions and others are not. $X = \{x_1,...,x_l\}$ denotes an assignment combination of $x_1 \sim x_l$, called partial solution. Feasible partial solution (*FPS*) means an X which satisfies formulas (2) and (3).

The algorithm is shown in Figure 3. Deep First Searching (DFS) and backtrack are used in the algorithm. A feasible solution queue S is used to cache solutions when the algorithm is running.

Supposing we want to construct top 5 solutions by g, there will be 5 X_w in S that are arranged in descending order according to the results of g. The solution in S whose result is the smallest is denoted by X_{wmin} and g_{min} denotes the result.

When the algorithm terminates, there are five solutions in S. Then, the next step is to analyze risks in each solution.

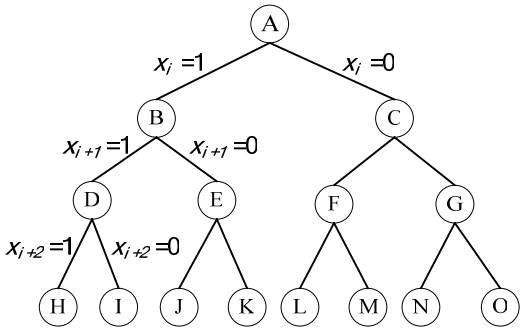

Figure 2. A solution space

1. Do DFS from i=1;
2. If $i \le n$, x_i = 1 is added into X, and X is still a *FPS*
   then add x_i = 1 into X, i=i +1, goto 2;
   else goto 3;
3. If $i \le n$, x_i = 0 is added into X, and X is still a *FPS*
   then add x_i = 0 into X, i=i +1, goto 2 ;
   else goto 5;
4. Compare g_i and g_{min},
   if $g_i > g_{min}$
   then delete X_{wmin} from S, add X_i into S ,
   and re-order S;
5. If for each x in X: x=0
   then exit;

Figure 3. Algorithm for finding solutions

3.2 Analyzing Risks of Feasible Release Plans

The first step of risk analysis is to identify risks. Then the probability and loss of each risk are estimated. At last, all kinds of risks are combined to show the whole risks of a release plan.

The risk taxonomy is used in identifying risks. Some studies have contributed effective risk taxonomies [25, 26]. Although those taxonomy tables are designed for traditional development process, they can be revised for XP. Table 1 shows a risk taxonomy defined by us. XP practicers can add risk types and risk items according to their experiences.

Probability and loss of a risk can be estimated quantitatively or qualitatively. Quantitative estimation requires a lot of time and cost, and sometime it is difficult for developers to collect enough data for quantitative analysis [26,27]. Thus, qualitative risk

analysis is more suitable for XP. Furthermore, risks maybe cause losses in multiple aspects of a software project. In our method, we use scope, schedule and product quality as criterions of losses. Scope loss means that development activities are not toward the software system's goals. Schedule loss means that the release is postponed. Product quality loss means that the product cannot fulfill the needs of functions and performance.

Table 1. XP risk taxonomy

Risk Type	Risk Item	Description
Requirements Risks	Unstable story	Story is volatile because of the volatile environment
	Vague story	Story is unclear in business goals or for system design
Estimation Risks	Size	Wrong estimation of story size
	Team productivity	Wrong estimation of team productivity
Technology Risks	Architecture conflict	How to combine new stories into existent architecture
	Difficult implementation	How to implement stories
Personnel Risks	Customers	Customers are not domain experts in business

We use the qualitative risk estimating method provided in [27]. The degrees of "Low", "Medium" and "High" are used as the description for the probability and loss of risks, and then the Risk Exposure (RE) is defined according to Table 2.

Table 2. A qualitative method for risk assessment

Risk Exposure		Probability		
		Low	**Medium**	**High**
Loss	**Low**	Minor	Significant	Critical
	Medium	Significant	Critical	Unacceptable
	High	Critical	Unacceptable	Unacceptable

A release plan may have multiple risks. To compare differences among feasible release plans, RE should be accumulated according to the type of loss. Based on existent methods [27], we present a method to combine multiple risks by scores. Developers may use the score table (refer to Table 3) to accumulate RE, according to scope, schedule and product quality losses, respectively. The scores give a straight way for comparing multiple release plans. The risks, business value and needed efforts of each feasible release plan are then submitted to stakeholders to help them make decisions.

Table 3. Score for risk exposure

Risk Exposure	Score
Unacceptable	4
Critical	3
Significant	2
Minor	1

3.3 Making Decision and Adjusting Project Profiles

During making decision, if every release plan always includes a type of risks whose score is very high, it means there are crucial defaults in the project profiles. Then developers must go back to check the project plan and negotiate with customers about the project profiles. The scores' criterion of high risks comes from historical projects' data and former iterations' results of the current project. A release plan is selected based on the project progress and risks scores. For example, in early iteration, developers should choose the release plan whose scope risk's score is higher for understanding the software system's goals more quickly. When near the milestone, developers should choose the release plan with low risks scores to ensure a useful system at the milestone point.

Although it shows in our method that constructing feasible release plans, analyzing risks and making decision are necessary for per iteration, it does not mean that stakeholders should execute every activity from the beginning during the development process. In fact, if stakeholders affirm the early results of analysis such as stories' business value and risks, these data can still be used in latter iterations until the events happen which cause the changes of the early analysis results.

4. RISK-DRIVEN XP PROCESS

When the risk-driven method is applied in XP practice, the XP process becomes a risk-driven process. To help developers apply the risk-driven method for XP release planning easily, this section introduces a specific XP process with the risk-driven release planning method applied, as shown in Figure 4.

The activities and their inputs, outputs, and steps are defined as follows:

Activity 1: Defining stories

(1) Input: Project Profiles, New Stories
(2) Steps:
  a. Constructing stories based on the project profiles
  b. Building test scenarios
(3) Output: Stories, Test Scenarios

Activity 2: Constructing feasible release plans

(1) Input: Stories
(2) Steps:
  a. Estimating stories' value and size as well as available effort and analyzing stories' dependencies
  b. Making a set of feasible release plans
(3) Output: Release Plans

Activity 3: Analyzing risks and making decision

(1) Input: Release Plans
(2) Steps:
  a. Analyzing risks in each release plan
  b. Comparing analysis results and making decision
(3) Output: Adopted Release Plan (in the next iteration)

Activity 4: Adjusting project profiles

(1) Input: Results of Risks Analysis, Results of Acceptance Test
(2) Steps:

a. Negotiating with customers and adjusting project profiles to mitigate risks

(3) Output: Guides of Project Profiles Adjustment

Activity 5: Iteration

(1) Input: Adopted Release Plan
(2) Steps:
   a. Coding and unit test
(3) Output: Latest Version

Activity 6: Acceptance test

(1) Input: Latest Version, Test Scenarios
(2) Steps:
   a. Testing and collecting customers' feedback
   b. Collecting new stories and information of iterations' accumulative efforts (project velocity)
(3) Output: Small Release

Following the process, developers organize their development orderly in accordance with the method described in section 3. When they plan a release (from Activity 2 to Activity 4), all techniques for constructing feasible release plans, analyzing risks and adjusting project profiles will contribute to a sensible release plan. Moreover, the developers have the chance to adjust the project profiles and improve their project process according to the feedbacks.

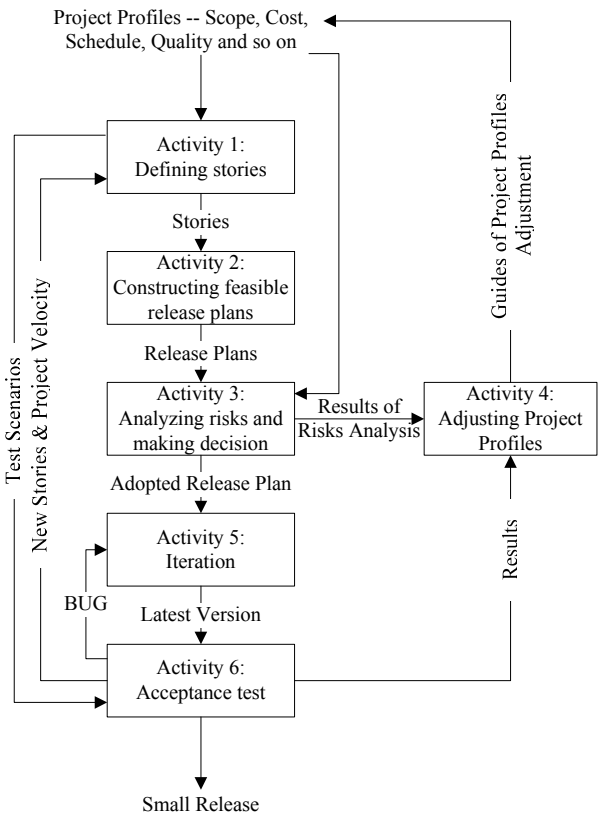

Figure 4. Risk-driven XP process

5. CASE STUDY

The proposed method was applied to a Web-based application project in our organization for verifying its feasibility and effectiveness. A team in our organization had developed the supposed system. This team (B) practised XP without any specific release planning method. To minimize the influences of personnel experiences, a group of people who had never been involved in the project was organized in a team (A) to develop the project again. The data were established as comparable as possible by assigning similar background of personnel according to team B (refer to Table 4). The project data were collected and compared, and the result demonstrates that the proposed method in the paper is feasible and can help developers improve their XP practices.

5.1 Data Collection and Analysis

The background of the case is summarized in Table 4.

Table 4. Background of the case

Project Characteristics	Description	
Type of end product	Web-based application	
Development tools	JBuilder 9, JDK1.4, Tomcat4, Mysql Bugrat, Winrunner QMP2.5, Firefly2.5	
Developer team size	A	6 developers
	B	
Developers' experience in XP	A	1 experienced
	B	5 novice
Developers' experience in end product	A	6 experienced
	B	
Developers' experience in coding	A	6 experienced
	B	
Iterations	A	4
	B	5

A release plan's information of risks, business value and effort needed is presented to stakeholders. Table 5 is an example of such information used by team A for an iteration. Based on the information, stakeholders negotiate with each other and make the release plan decision for the next iteration.

Both objective and subjective data were collected from team B and team A. The objective data are from the organization's CASE tools, such as defect tracking, project tracking, source code analysis and test suite tools. All these data are shown in Table 6 and Figure 5. Furthermore, we collected developers' subject feedback in team A by a questionnaire. Some questions are designed for developers' opinions about the proposed method in the paper. There are four selections for every question: "Very good", "Good", "Fair", and "Poor". There also are some open questions in the questionnaire for developers' to express their viewpoints about the method freely. The team A developers' subject feedbacks are summarized in Table 7.

One of the characteristics of the development process of team A is that the number of story changes declines rapidly. It means developers and customers catch the scope and constraints of the system and reach a consensus more quickly. We compare the rate of story changes (the number of stories that change in each iteration / total number of stories), as shown in Figure 6. The results indicate a lower rate of story changes exists in later iteration in the development process of team A using the risk-

driven method for XP release planning. The result also can answer the question "Why in the similar team productivity, more efforts are needed in the previous XP practice of team B?" Table 6 shows there are much more story changes in late phrase of the team B. The story changes cause extra needless reworks. Developers also agree that the proposed method in the paper is helpful in negotiating the project's scope and available resources.

Table 5. An example of risks, business value and effort information provided for customers' decision making

Feasible Release Plans	Risks Score				Business Value (%)	Effort Needed (Hour)
	Scope	Schedule	Product Quality	Total		
NO. 1	17	21	35	73	37	289
NO. 2	24	19	27	70	33	312
NO. 3	29	16	33	78	21	210
NO. 4	11	23	29	63	20	247
NO. 5	18	19	28	65	19	188

Table 6. Project data

No	Data Collected	Case	Release No.					Total
			1	2	3	4	5	
1	Calendar time (weeks)	A	3	3	2	1	/	9
		B	3	2	2	2	2	11
2	Effort (man-hour)	A	613	703	476	199.5	/	1991.5
		B	609	638	423.5	399	467	2536.5
3	KLOC implemented in the last release	A	21.89	26.5	19.52	8.02	/	76.9
		B	13.22	11.99	15.46	20.71	21.6	82.98
4	Team productivity (loc/hour)	A	35.7	37.7	41.0	40.2	/	38.6
		B	36.1	38.0	39.7	40.9	40.9	39.1
5	Numbers of story changes after iteration	A	7	4	1	1*	/	13
		B	5	4	6	5	3*	23
6	Numbers of stories implemented	A	11	9	10	6	/	36
		B	10	10	8	2	4	34
7	BUG numbers after the acceptance test	A	0	1	0	0	/	1
		B	0	0	0	0	0	0
* *Preserved for future implementation*								

Table 7. Developers' subject feedbacks

The risk-driven method is helpful in	Feedback of developers			
	Very good	Good	Fair	Poor
Speeding up release planning	3	3	0	0
Balancing project scope and available resources	4	2	0	0
Controlling project progress	2	3	1	0

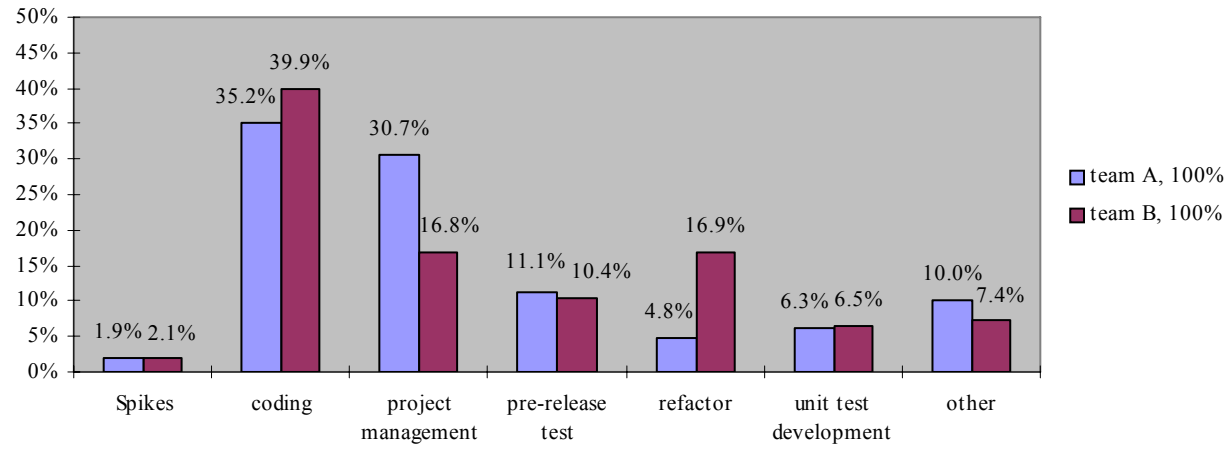

Figure 5. Comparison of task efforts distribution between team A and team B

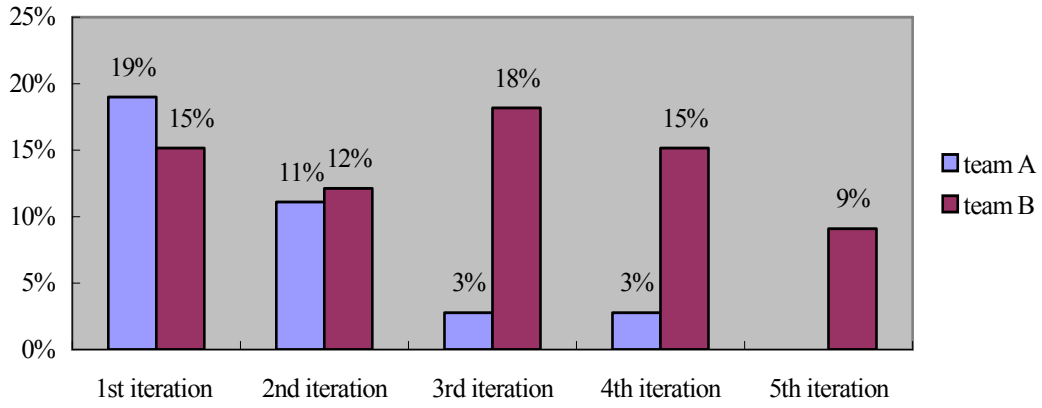

Figure 6. Comparison of story change rate between team A and team B

5.2 Lessons Learned

We got at least three benefits through the risk-driven method for XP release planning in this paper:

- The risk-driven method for XP release planning is well-defined for XP practice. It can help XP practicers make better release plan decision.
- Identifying and assessing risks during release planning can improve stakeholders' understanding of the system's goals and developers' capacity. Based on scope, estimation and technology risks, stakeholders may make a balance between the system's goals and available resources, and ensure the software development process towards correct goals within the schedule and resource constraints.
- The method improves negotiation between customers and developers. Developers can construct multiple feasible release plans and expose their risk levels. The groundwork for negotiation is established by these release plans and information of the risks when developers and customers choose suitable release plans for XP iterations.

Also, we noticed some weaknesses in the method. For example, since one story may be included in multiple feasible release plans, risk data can be reused during the risks analysis phase. However, we do not have related reuse scheme yet. Actually, developers in the case study already advised the method would be more effective if there were some data reuse mechanism.

6. CONCLUSIONS

There are no detailed requirements and architecture design before the system implementation in XP practice. Release planning is the key for successful development where the scope and constraints, design, and implementation are considered synthetically in a software system. However, the vague techniques in traditional XP release planning often muddle XP practicers. And developers often hesitate in making a release plan and negotiating with customers about the implementation order of stories in iterations.

In this paper, we put forward a risk-driven method for XP release planning. XP practicers can follow it to decide a suitable release plan and control the development process. The features of the method can improve XP practices. Firstly, multiple choices of feasible release plans are provided for stakeholders to enlarge the scopes of the developers and customers' negotiations when they decide a release plan for the next iteration. Therefore, stakeholders can avoid making a poor decision based on a single candidate release plan. Secondly, risks analysis can improve the ability of the stakeholders to understand the circumstance and to collaborate with each others [28]. In our method, the scores of scope, schedule and product quality risks provide clear criteria for stakeholders' consensus when they are making release plans choice. The comparable risks' scores, which differ from unstructured risks analysis method in traditional XP practice, reduce problems when release plans are negotiated. Thirdly, the method will make out of a risk-driven XP process for developers to guide their XP practice. Furthermore, the method not only can be used for XP iterations, but also for long-time plans such as milestone plans and/or even software life cycle development with adaptive planning method. This general framework of the method suggests that it could also be used in other IID methods to improve their release planning.

We have applied the method in a case study and the results show the method is feasible and effective. In the future work, we are going to study an approach of reusing risk analysis data and develop an integrated tool to improve the method's usability.

7. ACKNOWLEDGMENTS

This research is supported by the National Natural Science Foundation of China under grant Nos. 60273026, 60573082 and the National "863" High-Tech Program of China under grant Nos. 2004AA112080, 2005AA113140.

Professors Qing Wang and Yongji Wang gave many constructive suggestions. Their works are highly appreciated.

8. REFERENCES

[1] Macias, F., Holcombe, M., Gheorghe, M., A Formal Experiment Comparing Extreme Programming with Traditional Software Construction, *Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03)*, 2003.

[2] Abrahamsson, P., Koskela, J., Extreme Programming: A Survey of Empirical Data from a Controlled Case Study, *Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE'04)*, 2004.

[3] Layman, L., Williams, L., Cunningham, L., Motivations and Measurements in an Agile Case Study, *Proceedings of the Workshop on Quantitative Techniques for Agile Processes (QUTE-SWAP '04)*, 2004.

[4] Sommerville, I., *Software Engineering*, 6th edition, Addison-Wesley, 2000.

[5] Beck, K., Fowler, M., *Planning Extreme Programming*, Addison-Wesley, 1st edition, 2000.

[6] Paetsch, F., Eberlein, A., Maurer, F., Requirements Engineering and Agile Software Development, *12th IEEE International Workshops on Enabling Technologies (WETICE 2003)*, 2003, 308-313.

[7] Orr, K., Agile Requirements: Opportunity or Oxymoron?. *IEEE Software, 21, 3* (2004), 71-73.

[8] Neill, C. J., The Extreme Programming Bandwagon: Revolution or Just Revolting?. *IEEE IT Professional*, *5, 5* (2003), 62-64.

[9] Nawrocki, J., et al., Extreme Programming Modified: Embrace Requirements Engineering Practices, *Tenth International IEEE Conference on Requirements Engineering*, 2002.

[10] Rand, C., Eckfeldt, B., Aligning Strategic Planning with Agile Development: Extending Agile Thinking to Business Improvement, *Proceedings of the Agile Development Conference (ADC'04)*, 2004.

[11] Cao, L., et al. How Extreme does Extreme Programming Have to be? Adapting XP Practices to Large-scale Projects. *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[12] Larman. C., *Agile Iterative Development A Management's Guide*, Pearson Education, 2004.

[13] Lehman, M. M., Ramil, J. F., *Rules and Tools for Software Evolution Planning and Management, Annals of Software Engineering, Vol 11*, 2001,15-44.

[14] Jung, HW. Optimizing Value and Cost in Requirements Analysis. *IEEE Software, 15, 4*(1998), 74- 78.

[15] Karlsson, J., Ryan, K., A Cost-Value Approach for Prioritizing Requirements. *IEEE Software, 14, 5*(1997), 67-74.

[16] Tran, T., Sherif, J.S., Quality Function Deployment (QFD): an effective technique for requirements acquisition and reuse, *2nd IEEE Software Engineering Standards Symposium*, 1995.

[17] Moisiadis, F., The fundamentals of prioritising requirements. *(Web) Proceedings of Systems Engineering/Test and Evaluation conference (SETE2002)*. 2002.

[18] Wiegers, K., *Software Requirements*, Microsoft Press, 2003.

[19] In, H., Olson, D., Rodgers, T., Multi-Criteria Preference Analysis for Systematic Requirements Negotiation, *IEEE International Computer Software and Applications Conference (COMPSAC 2002)*, 2002, 887-892.

[20] Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Dag, Johan N., An Industrial Survey of Requirements Interdependencies in Software Product Release Planning.

[21] Jackson, A., et al., Behind the Rules: XP Experiences. *Proceedings of the Agile Development Conference (ADC' 04),* 2004.

[22] Glass, L. R., Extreme Programming: The Good, the Bad, and the Bottom Line, *IEEE SOFTWARE, 18, 6* (2001), 112-113.

[23] Karlsson, J., Wohlin, C., Regnell, B., An Evaluation of Methods for Prioritizing Software Requirements, *Information and Software Technology, 39, 14-15* (1998), 938-947.

[24] Carr, M. J. Konda, S. L., Monarch, I., Ulrich, F. L and Walker, C. F., *Taxonomy-Based Risk Identifcation*, Software Engineering Institute, technical Report, CMU/SEI-93-TR-6, 1993.

[25] Simmons, E., Requirements Triage: What Can We Learn from a "Medical" Approach?, *IEEE Software, 21, 4* (2004), 86-88.

[26] Greer, D., Bustard, David W., Towards an Evolutionary Software Delivery Strategy based on Soft Systems and Risk Analysis. *IEEE symposium on Engineering of Computer Based Systems*, 1996, 126-133.

[27] Boehm, B., *Tutorial: Software Risk Management*, IEEE Computer Society Press, 1989.

[28] Boehm, B., Turner, R., Using Risk to Balance Agile and Plan-Driven Methods, *IEEE computer, 36, 6*(2003), 57-66.