Estimate of the appropriate iteration length in agile development by conducting simulation.

Ryushi Shiohama*, Hironori Washizaki*, Shin Kuboaki†, Kazunori Sakamoto*, Yoshiaki Fukazawa*
*Information Technology Dept.
Waseda University
Tokyo. Japan
r.shiohama@fuji.waseda.jp
†Afrel co., ltd.
Tokyo. Japan

***Abstract*—Agile development refers to the group of software development methodologies based on an iterative and incremental process model. It divides the development period into short time frames called iterations and uses a body of knowledge obtained from past experience called practice to ensure agile software development Although the iteration length is an important factor in agile development however it has so far been decided by the qualitatively and it has been reported that projects with an inappropriate iteration length tends to be failed. We thus propose a new methodology for estimating an appropriate iteration length through the conduct on of a simulation based on project constraints.**

In this paper we first, propose a method of calculating an appropriate iteration length for a particular project to promote the easy use of agile development. Second, the relationship between the iteration length and project constraints was investigated by varying the parameters to create diverse situations.

***Keywords*-Agile, Iteration, XP, Scrum, Iterative, Incremental, Simulation**

I. Introduction

Agile development refers to the group of software development methodologies based on an iterative and incremental process model[5]. It divides the development period into short time frames called iterations and uses a body of knowledge obtained from past experience called practice to ensure agile software development.

Figure 1 shows the model used to extract common processes from eXtreme Programming[8](XP) and Scrum[9] which are the most frequently used agile development processes[5]. In this common model, developers implement various requirements and show the functions to the customer in each iteration feedback from the customer is applied in the next iteration.

Each iteration has the three phases, ”iteration planning”, ”implementation” and ”release and review”[3]. In the iteration planning phase, picking the requirements from unimplemented functionalities are selected to decide the development scope of the iteration. The requirements to be implemented original from the development scope in the implementation phase and specifications are changed on the basis of customer feedback in release and review phase.

In software development no project has the same constraints and no process model is suitable for all projects. To increase productivity, some development parameters should be changed to match the project constraints. In particular, in agile development, there are numerous parameters that should be changed such as the iteration length, the type of practices and so forth. However most previous studies only considered verifying the applicability of recent software developments or performed comparisons with waterfall development[2][4].

We thus propose a new methodology for estimating an appropriate iteration length by conducting a simulation based on project constraints. As the project constrains we use five parameters to identify project features. Using our proposed method of calculating an appropriate iteration length for a particular project, the relationship between project constraints and iteration length is investigated by varying the parameters to create diverse situations. This method is expected to promote the easy use of agile development processes.

The reminder of the paper is organized as follow. Section II describes the problem of deciding an appropriate iteration length in agile development. Section Ⅲ explains the proposed methodology. Section Ⅳ gives the results of simulation and their implication. Section V briefly discusses related work. Section Ⅵ gives a summary of this paper along with the future works.

II. Problem of deciding the iteration length

In agile development, the iteration length affects the success of a software development project since it influences the scope of iteration[7], how changes are managed and implemented and how the development cycle is generated.

However there have been few studies on the iteration length and only qualitative values have been recommended such as, a week for XP and a month for Scrum[5]. Therefore managers must decide the iteration length exclusively from

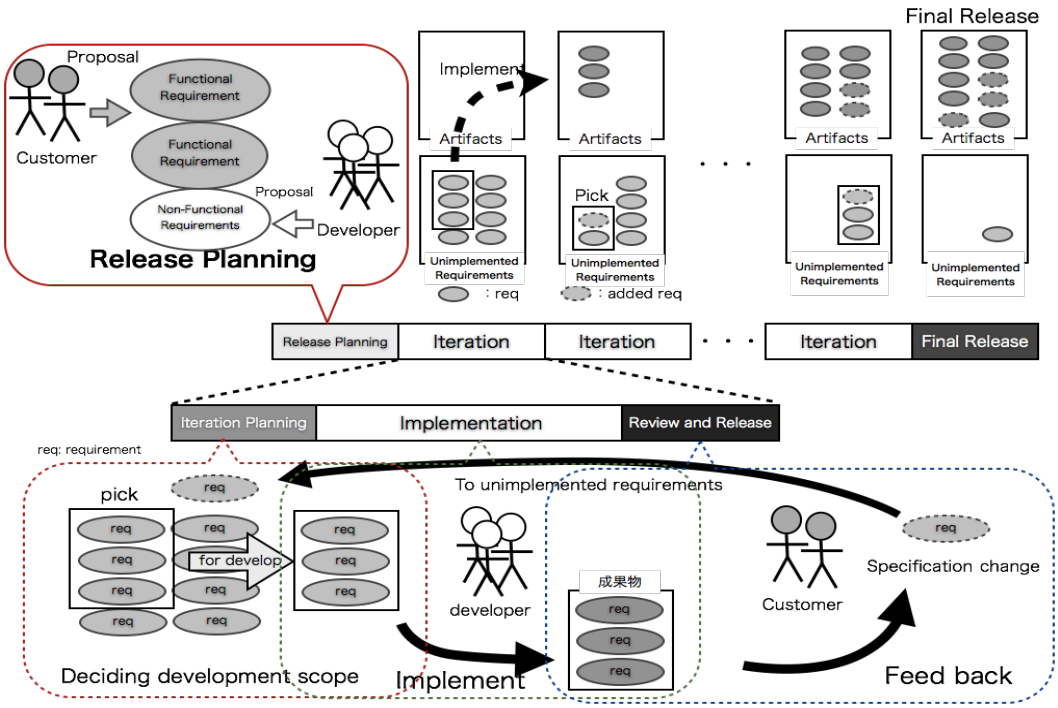

Figure 1. common model of agile development

their own experience and an incorrect decision may lead to project failure. Here, we explain how an inappropriate iteration length may lead to project failure[6].

An excessively long iteration length reduces the opportunities for obtaining feedback from customer and makes it difficult to deal with specification changes. Moreover, it increases the scope of iteration and makes the project more complex. In contrast, an excessively short iteration length increases the numbers of iteration planning phases, thus increasing overheads and leading to cost escalation. A developer may be forced to make the requirements much smaller to adjust the iteration length, thus increasing integration costs.

III. Estimation of Iteration Length by Conducting Simulation

We provide a methodology for estimating an appropriate iteration length for a project by conducting a simulation. Our methodology uses a common model extracted from XP and Scrum. This section describes the following points.

- The simulation methodology.
- The extraction of the common model.
- The flow of simulation in detail.

A. Simulation methodology

There are two main ways of investigating development processes by observing real projects and finding general laws or by creating a common model and using it to conduct simulations to estimate the real phenomenas.

The former way requires the observation for enormous amount of data because data may include artificial errors and because no two projects have the same constraints. In the latter way, one can compensate for a lack of data by using a simulation model that is constructed from well-known events. On the basis of these features, we considered the latter way to be more suitable in the case of agile development. However the results should only be applied to situations within the scope of the model and must be validated using some test cases. In this paper, we use two case studies for validation in Section Ⅳ.

B. Method of extracting the common model

As we mentioned above, analysis of whole project requires enormous data, however the part of the process can be investigated much easier. We thus referred to existing literatures of agile development and results of development process researches and combined them to construct the simulator model. Following shows the construction of simulator model. Figure 2 shows a class diagram of the simulator. Simulator creates the number of trials projects based on given Constraints. Each Project involves a number of some Iterations and has the Result of the project. Iterations are created by Iteration planning, Implementation and Review and it simulates the development processes using Developer, Requirements, Tasks and some parameters from Constrains.

Table I
PARAMETERS USED IN SIMULATOR

Parameter name		Values	unit
Project Constraints	Development Term	Integer greater than or equal to 1	days
	Variety	0, 5, 10, 15, 20, 25, 30	%
	Complexity	25, 50, 75	%
Developer Ability		0.25(Beginner), 1(Intermediate), 2.5(Advanced)	
Requirement	Required labor	Integer greater than or equal 1	man-days
	Importance	0-1	

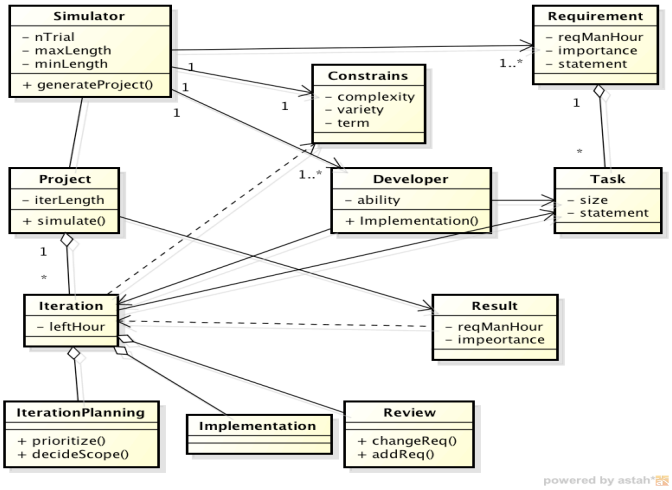

Figure 2. Class diagram of simulator

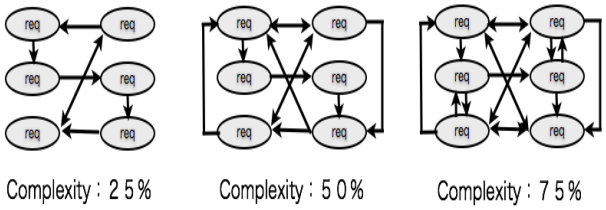

Figure 3. Relationship between complexity and volume of dependencies

Table I shows the parameters used in the simulator.

Development Term

This is the number of days from the beginning to the deadline of the project excluding holidays. We assume that the project starts at the beginning of the first iteration because our method assumes that the initial requirements have already been decided.

Variety

The variety describes the probability that specifications change in each iteration. We can add, change and remove requirements as the specification change. However, it is impossible to know the exact amount of variety, so it should be estimated by considering the novelty, area and concreteness of the project on the basis of past experiences.

In our method, specification changes occur for two reasons: customer feedback based on the artifacts of each iteration and the volatility of the market or advances in technology. For simplification, we assume that variety results in both types of change, The former is incorporated using the model discussed in [16] and latter is incorporated by considering the elapsed time from the beginning of the project.

Complexity

Complexity indicates the probability that dependences between requirements are generated. As shown in Figure 3, we assume that a high complexity corresponds to a large number of dependences between requirements. Integrity costs are generated when requirements that have been already released depend on the requirement that is currently implementing or changing. Integrity costs are added as tasks as explained below. Similarly to variety, the complexity cannot be determined before the project starts, meaning that it should be estimated by considering the area and size of the project on the basis of past experiences.

Developer ability

Developer ability indicates the ability of a developer in the specified project. To decide a basic value for developer ability, we consider a previous report on how developer ability affects productivity, [14] and [15] In the former study, it was concluded that there is an approximately 28-fold difference between the productivity of a beginner and an advanced on developers.

However, in the latter study, an investigation of the evolution of integrated development environment (IDE), object oriented programing (OOP), web frameworks and testing frameworks, a 10-fold difference between beginner and advanced developers and a 2.5 times difference between intermediate and advanced developers were concluded. We considered the latter to be more suitable for fitting our model and used values of 0.25, 1, 2.5 for the developer ability.

Required labor of a requirement

This is the amount of labor required to develop the requirement. using a developer intermediate ability as the unit.

Importance of a requirement

This indicates the relative importance of the requirement as a value from 0 to 1. For example, considering the contents availability management system which has four requirements, ”searching for contents”, ”registering contents”, ”sorting results” and ”supporting input”, ”searching for contents” and ”registering contents” are the essential parts of the system and should have an importance of 0.8 - 1.0. Requirements that are not essential but important, such as ” sorting results” should be marked 0.5 - 0.8. Moreover additional requirements such as ”supporting input” should be marked 0.1 - 0.5.

The simulator uses the above parameters and calculates the result as describing below.

Progress

Progress is given by the sum of the relative importances of the implemented requirements.

Cost

Cost is the total number of man-hours of work including the customer man-hours spent in iteration planning. We do not focus on the material cost and equipment cost because they are basically constant even if the iteration length is changed.

As our method uses a comparable way to obtain an appropriate iteration length, we can use relative values in the results.

C. Flow of the simulation

On the basis of the flow of the simulation shown in Figure 4, we next explain the simulator model in detail.

1) There are three phases in iteration planning: requirement prioritization, deciding the scope and requirement splitting.
   a) Requirements are prioritized in accordance with their importance and the dependences between them. Concretely requirements are sorted by importance and then raise the priority of requirement thats that are dependent on other requirements are raised. This strategy originates from practice of agile development: value-driven development and develop the available functionality of each iteration.
   b) To decide the development scope, a scope-box method and a time-box method should be considered. The former is a means of deciding the length of the iteration from the development scope and the latter is a means of deciding the development scope from the iteration length. In our method, it is assumed that the iteration length is constant throughout the whole project, thus we use the time-box method for deciding the development scope.
   c) In our method, requirements are defined in terms of their functionality, regardless of whether they are functional or non-functional. Thus, we assumed that the requirements are divided into smaller tasks in the requirement splitting phase and that a developer is assigned to implement each task for. To avoid a procedure error due to requirement splitting, we fix the size of tasks to 0.5-2 man-days, which was reported as an appropriate value in [5].

   The time spent on iteration planning depends on the project, the customer and team decisions. However, for modeling purpose we assumed that it depends on the length of the iteration and size of the development scope.
2) In the implementation phase, developers implement the tasks that are selected in iteration planning. Figure 5 shows a state transition diagram of the implementation, developer has two states: empty and assigned. Each task has three states: todo, doing and done. A statement of both are updated every an hour, and the rules of the state changes are as follows.
   - A developer in the empty state:
     – If there is at last a task which state is todo, the developer is assigned to the task. It changes

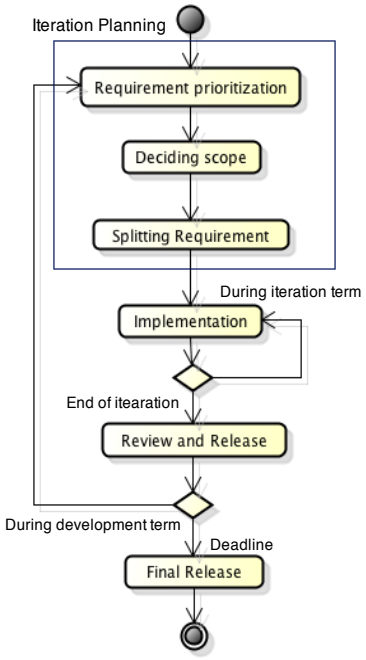

Figure 4. Flow of the simulation

the developer state to assigned and task state to doing.
  - if there is no task which state is todo, nothing is happened.
- A developer in the assigned state:
  - if the task still have the work to implement, the developer implements the assigning task.
  - if the task has been implemented, it enters the done state and the developer returns to empty state.

In our method, a developer works a fixed 8 hour per day and never expand their working time. It comes from the practice called 40 hours working per week from XP. Moreover to simplify the calculation, simulation should only consider variations of the iteration length. The total implementation time is given by
Iteration length (days) * 8 (hours) - iteration planning time (hour)

For example, in the case of 7 days per iteration and 6 hours of iteration planning, the total implementation time is 50 hours. A task that is in the done state is considered as an implemented task, and the requirement that all tasks are in the done state becomes an artifact of the iteration. If a requirement includes a task in the todo or doing state task, it must be implemented in the next iteration[5].

3) In the review and release phase, specifications may be changed or previously implemented requirements may be released. The probability of this occurring depends on the project complexity. Changed requirements are added tasks for specification changes or are integrated and set to the todo state. After that, new requirements are added that are related to the released functionalities; thus these requirements have higher importance and stronger dependences than the average values for the initial requirements[5]. The number of added requirements is limited by the initial number of requirements and project variety [16].
4) The above iterations are repeated until the last day of the project. At last there is final release phase. In this phase sum of the implemented requirements and the total number of man-hours of work are calculated and outputted as the result.

D. Use of our method

Our method can be used to help apply the agile development processes to actual project. The expected users are project managers or product managers who decide the iteration length. The results of the simulation can be used to decide the iteration length before starting the project or used to review the project after its completion by comparing the project results and those can be obtained from simulator. Moreover, it can be used in the middle part of a project, particularly when an unexpected accident or delay to the process occurs.

E. Threats to validity

In our model, we use only five parameters to identify a project feature. Hence it is the simulator model for estimating appropriate iteration length, we eliminate some parameters to avoid unnecessarily complexity. To choose the parameters, we refer to paper of Information-technology Promotion Agency, Japan which mention about non-waterfall development[24]. However in the real project, no parameters can be ignored for considering result. Thus it should be recognized as threats to validity and need to be examined for assuring validity of the model.

IV. Results of simulation and discussion

We select two projects which are published the detailed project tracking and review of them as case study. We then simulate these projects using our method and compare the simulation results for the projects to with their review to validate the simulator. After that, the parameters are varied to create different situations, and analyze the relationship between the project constraints and appropriate iteration length. Each simulation is performed 1000 times using a computer with the following specifications, and the results given are average values.

OS: Mac OSX 10.6.7
Processor: 2.3GHz Intel Core-i5
Memory: 8GB 1333 MHz DDR3

A. Case study 1

We uses the XP Practice Report from Eiwa System Management, inc. for case study 1[18]. This report describes a project that was launched to evaluate agile development, which gives detailed project constrains and reviews about the iteration length. In this project, a Web application for patient management system is developed in XP using five developers: a beginner programmer who joined the company the previous year, two intermediate-level programmers who have worked at the company for a few years and two advanced level-programmers.

This project have a month for development and XP normally adopts an iteration length of a week, so this project involved four iterations. Init requirements are constructed from seven main stories and some requirements are gradually fixed through the feedback from the customer. From the above conditions we extract the given parameters for the simulator, which are shown in Table II. Because there are no exact details about variety and complexity, we calculate the variety from the actual specification changes. Moreover, we estimated the number of dependences between

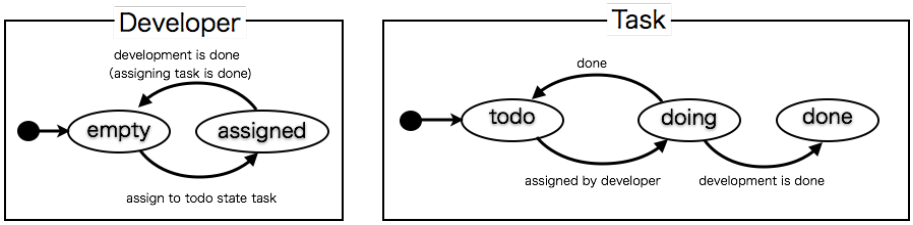

Figure 5. State transition diagram of implementation

Table II
GIVEN PARAMETERS FOR CASE STUDY 1

Name	Value
Development term	20
Variety	10%
Complexity	50%
Developers	0.25, 1, 1, 2.5, 2.5 (five people)
Required man-day for requirements	10, 10, 15, 10, 10, 10, 5 man-days
Importance of requirements	0.3, 0.6, 1, 0.3, 0.6, 0.6, 1

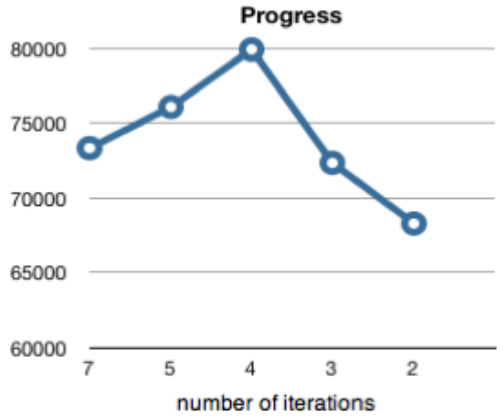

Figure 6. Simulation results for case study 1. x: iteration length, y: progress

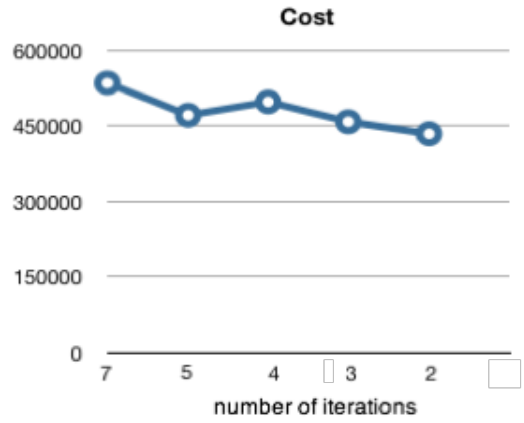

Figure 7. Simulation results for case study 1. x: iteration length, y: cost

the requirements from the story names and determined the complexity as its reciprocal.

Table Ⅲ shows result values of the simulations and Figure 6 shows the progress of the simulations of case study 1, x-axis shows the number of iterations and y-axis shows the progress value. We can see progress value is the highest the case in four iterations, meaning that five days for each iterations. Normally progress should be higher when the number of iterations are increased. However in this case, the result involved five and seven iterations are lower than four iterations. It indicates that small scope reduces productivity in high complexity.

Figure 7 shows the cost of the simulations of case study 1, x-axis shows the number of iterations and y-axis shows the progress value. Cost tend to be reduced by decreasing the number of iterations though a case of five iterations is lower than an law of others. It implicates total labor times for development are almost same except the case of five iterations.

Figure 8 shows that unit cost and it tends to be higher in the case of five iterations and the review of the project commented that iteration length was slightly shorter than the appropriate length though this project almost succeeded". This demonstrates that the simulator works well for this project.

B. Case Study 2

As case study 2, we consider a project about a correspondence education system carried out by Probizmo, Co. LTD[17]. This project was reported in an IT development plan of Shimane prefecture and the project title was "Research project for validating the business model of Ruby".

In this project, the development term was 100 days and developer was an intermediate level programmer who used the scrum model. There were 16 initial requirements, and as because a Scrum model was used, the iteration length was a month, meaning that this project was completed in 4 iterations. Regarding the project features, it was reported that the complexity was very low with few specification changes. We thus extracted the parameters present them in Table Ⅳ.

Table V shows result values of the simulations and Figure

Table III
RESULT OF THE SIMULATION

Number of iterations	Progress	Cost	Progress/Cost
7	73337	535373	0.136983
5	76089	471283	0.161450
4	79916	497689	0.160574
3	72345	458321	0.157847
2	68270	434585	0.157092

Table IV
GIVEN PARAMETERS FOR CASE STUDY 2

Name	Value
Development term	100
Variety	25%
Complexity	25%
Developpers	2.5 (a person)
Required man-day of Requirements	5, 5, 10, 10, 15, 10, 10, 10, 5, 10, 10, 15, 10, 10, 10, 10man-day
Importance of Requirements	0.3, 0.3, 0.6, 0.6, 1, 0.6, 0.6, 0.6, 0.3, 0.6, 0.6, 1, 0.6, 0.6, 0.6, 0.6

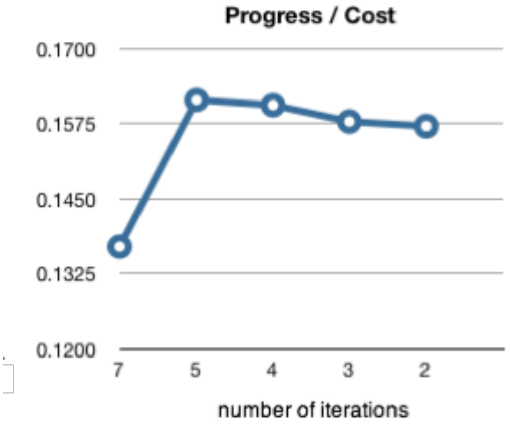

Figure 8. Simulation results for case study 1. x: iteration length, y: progress/cost

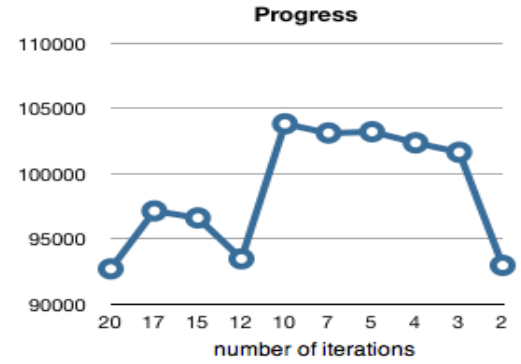

Figure 9. Simulation result of case study 2 x:iteration length, y: progress

9 shows the progress of the simulations of case study 2. Progress increases extremely in the cases of less than 10 iterations except the case of two iterations. It indicates that in the cases of more than 12 iterations are too short for this project complexity and it also exceeds the amount of specification change limit.

Figure 10 shows the cost of the simulations of case study 2. Cost tends to be reduced by decreasing number of iterations. Similar to progress, it can be separated boundary of the 10 iterations. Progress per unit cost for the case study 2 are shown in Figure 11. It tends to be higher in the case of three or four iterations. Moreover, the review of the project commented that the project was successfully finished in four iterations and the iteration length fitted the project". Demonstrating that the simulator works reasonably well for this project.

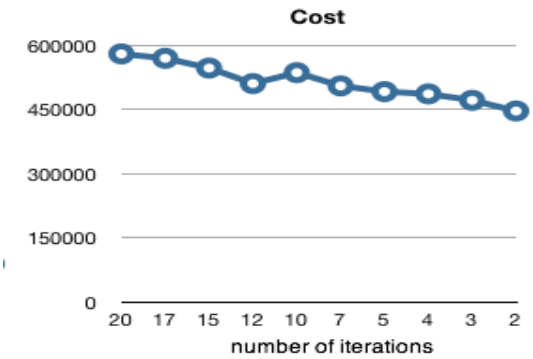

Figure 10. Simulation result of case study 2 x:iteration length, y: cost

C. Relationship between project constraints and iteration length

We next varied the parameters, particularly the variety and the complexity, and used the simulation results to investigate the relationship between project constraints and appropriate

Table V
RESULT OF THE SIMULATION

Number of iterations	Progress	Cost	Progress/Cost
20	92688	569624	0.159915
17	97124	569341	0.170590
15	96591	546541	0.176731
13	93455	510257	0.183152
10	103807	535765	0.19375
7	103104	504808	0.204243
5	103201	491463	0.209998
4	102349	486195	0.210510
3	101626	471163	0.215691
2	92952	446110	0.2083611

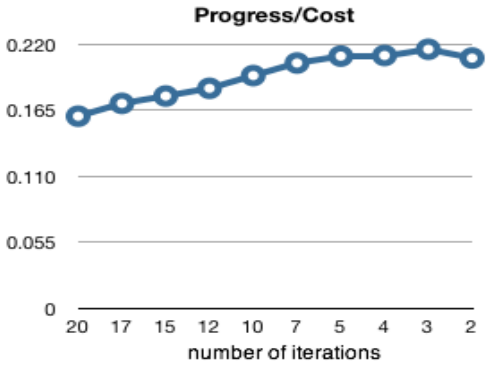

Figure 11. Simulation result of case study 2 x:iteration length, y: progress/cost

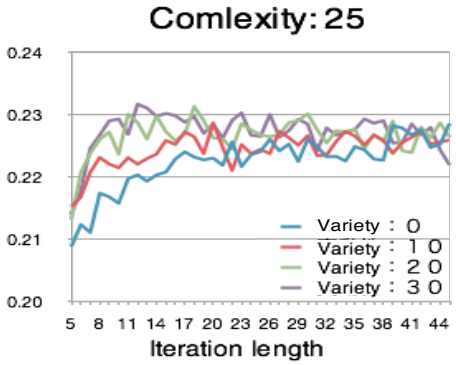

Figure 12. Relationship between project constrains and iteration length

iteration length. Parameters were fixed as follows

Development term: 60 days

Developers: 0.25, 1, 1, 1, 2.5 (five people)

Number of requirements: 30 (Suitable size for development in 60 days by five people)

Figure 12, 13, 14 shows the simulation results. Each graph represents a certain complexity value and each line in the graph represents a certain value of variety. Concretely all figures have 4 legends representing: variety 0, 10, 20, 30 and Figure 12 represents complexity: 25, Figure 13 represents complexity: 50 and Figure 14 represents complexity: 75. Moreover x-axis describes the length of iterations and y-axis shows the progress per unit cost values.

We first focused on the variety. Regardless of the complexity, a high variety tends to decrease the appropriate iteration length and a low variety tends to increase it. This may be because the specifications are changed frequently meanings that dealing with changes sooner makes reduces the integration cost and allows the preferred requirements to be implemented first. On the other hand, when the variety is low, wasteful planning may occur if the iteration length is too short.

Then, focusing on the complexity.It can be seen that a high complexity increases the appropriate iteration length. In the

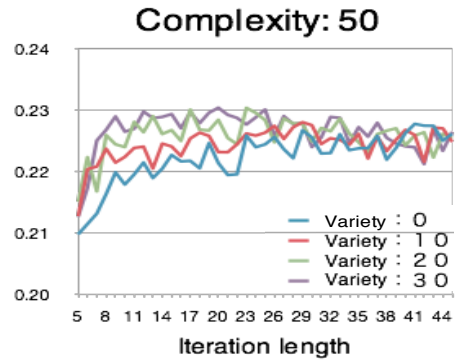

Figure 13. Relationship between project constrains and iteration length

case that there are many dependences between requirements, a short iteration length reduces the development scope regardless of requirement's dependencies and increases integration cost.

Additionally, results are more variable in the cost of high complexity. This is due to the method used to generate the dependences, which are set randomly using the complexity values; This may have reduced the accuracy of the results

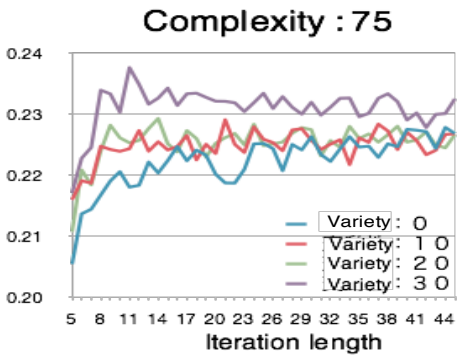

Figure 14. Relationship between project constrains and iteration length

V. Related works

To estimate the impact of processes before start of a project, some software process simulation methods have been proposed over the years. Barghouti and Rosenblum [19] proposed methods for simulating and analyzing software maintenance process. Otero et al. [22] use simulation to optimize resource allocation and the training time required for engineers and other personnel. Joana Rus, James Collofello and Peter Lakey helped apply the plan-based development using system dynamics simulation model[13]. Researches above are focused on plan-based development or part of the development process. As the research of agile development process, Dan Port and Alexy Olkov[2] created the original simulation model and used it to investigate requirements prioritization strategies. Moreover they proposed the new prioritization methodology that are combined agile development and waterfall development ways. Melis et al. [20][21] proposed an event-driven simulator for Extreme Programming practices such as test-driven programming and pair programming. David J. Anderson et al. [23], authors presented an event-driven simulator of the Kanban process and used it to study the dynamics of the process, and to optimize its parameters. And they were also using simulation to evaluate the Scrum and Kanban approaches on the basis of actual software maintenance processes.

VI. Conclusion and future works

In this paper, we simulated agile development while focusing on the iteration length to help apply agile development processes to actual projects. The main points of this study are as follows.

- We proposed a way of calculating an appropriate iteration length for a particular project to promote the easy use of agile development.
- We investigated the relationship between iteration length and project constraints by varying the parameters to create diverse situations.

At least result of our research shows that appropriate iteration length is changed by condition of project constraints such as the increase variety reduce the term of appropriate iteration length and high complexity situation suit longer iteration length. However our model still has the threats to validity and need to be validated more carefully to estimating obvious appropriate iteration length for each project.

As the future work, use class diagram, activity diagram and so on for applying our method to various of projects easily and validating this model through much more cases. Moreover, to make simulator more elaborate, we consider taking in the common method for measuring and estimating the development processes such like the function point methodology.

References

[1] Independent administrative corporation, information processing promote organization, Software Engineering Center : ”Observation about non-waterfall development observation report”, 2009 information property 0507

[2] Dan Port, Alexy Olkov :”Using Simulation to Investigate Requirements Prioritization Strategies.”, 2008 Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering

[3] Masaru Amano : How to proceeding the agile development, http://www.slideshare.net/esmsec/ss-5656398

[4] Akira Hattori, Koichiro Ochimizu : Validating the organization pattern using probability petri-net, Computer software vol. 23 (2006) No.1

[5] Craig Larman : Agile and Iterative Development, Addison-Wesley Professional; 1 edition (August 21, 2003)

[6] Toshiya Ikegami : Why is agile failed, Nikkei System 21th December 2010 http://itpro.nikkeibp.co.jp/article/COLUMN/20101215/355245/

[7] Shusuke Shitara : Agile Transparency http://gihyo.jp/dev/serial/01/agile-transparency/0002, 2009/11/10

[8] Pekka Abrahamsson, Michele Marchesi and Giancarlo Succi : Extreme Programming and Agile Processes in Software Engineering: 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006, Proceedings , Springer July 26, 2006

[9] Ken Schwabe and Mike Beedle : Agile Software Development with Scrum, Prentice Hall October 21, 2001

[10] Boehm, B, and Papaccio, P. 1988 : ”Understanding and Controlling Software Costs.”, IEEE Transactions on Software Engineering, Oct. 1988.

[11] Jones, C. 2000. : ”Software Assessments, Benchmarks, and Best Practices.”,Addison-Wesley.

[12] Tetsuo Tamai : Software Engineering, Iwanami shoten（2004）

[13] Joana Rus, James Collofello, Peter Lakey : ”Software process simulation for reliability management”, Journal of Systems and Software Volume 46 Issues 2-3, 15 April 1999, pages 173-182

[14] H. Sackman, W. J. Erikson, E. E. Grant : “Exploratory experimental studies comparing online and offline programming performance”,Communications of the ACM Volume 11 , Issue 1 1968

[15] Tom DeMarco, Timothy Lister : ”Peopleware”, 日経 BP 社 (2001)

[16] Takako Nakatani, et al. : A case study of requirements elicitation process with changes, IEICE Transactions 93-D: 2182-2189, 2010

[17] IT development plan of Shimane 2010 : Validating the business model of Ruby, http://www.pref.shimane.lg.jp/sangyo/it/

[18] Eiwa System Management, inc. : XP practice report http://objectclub.jp/community/XP-jp/xprelate/xppracticereport

[19] Barghouti, N. S., Rosenblum, D. S.. : A Case Study in Modeling a Human-Intensive, Corporate Software Process. Proc. 3rd Int. Conf. On the Software Process(ICSP-3). 1994, IEEE CS Press.

[20] Melis M. Turnu I., Cau A. and Concas G. : Evaluating the Impact of Test-First Programming and Pair programming through Software Process Simulation. Software Process Improvement and Practice, vol. 11, 2006, pp. 345-360.

[21] Melis M., Turnu I., Cau A. and Concas G. : Modeling and simulation of open source development using an agile practice. Journal of Systems Architecture, vol. 52, 2006, pp. 610-618.

[22] Otero, L.D., Centeno, G., Ruiz-Torres, A.J., Otero, C.E.. : A systematic approach for resource allocation in software projects. Comput. Ind. Eng. 56(4)(2009) 1333-1339.

[23] Anderson, D.J.. Concas, G.. Lunesu, M.I., and Marchesi, M.,. : Studying Lean-Kanban Approach Using Software Process Simulation. Proc. Agile Processes in Software Engineering and Extreme Programming 12th international Conference, XP 2011, Madrid, Spain, May 10-13 2011.

[24] Information-technology Promotion Agency, Japan Software Engineering Center. : Investigation for non-waterfall development, Information-technology Promotion Agency, Japan, 30th March, 2010.