ISSN 2320 - 2602

International Journal of Advances in Computer Science and Technology (IJACST), Vol.2 , No.5, Pages : 06-10 (2013)
Special Issue of ICACSIT 2013 - Held during 09-10 May, 2013, Malaysia

myAgile: A XP-based Method for Modern Programming Education

Jason Jen-Yen Chen, Mike Mu-Zhe Wu
Dept. of Computer Science and Information Engineering
National Central University
Jhong-Li, Taiwan
jasonjychen@gmail.com

Abstract: This work presents the "myAgile" method that integrates the industry-hailed extreme programming (XP) method with someuniversity programming courses. The goal is to build a XP-based modern programming education. A Java grade system project has been designed for students to practice "myAgile". And, an experiment of the project is being conducted. In the long run, it is expected that some programming courses maybe redesigned based on this work to meet the industrial need.

Keywords: Agile Method, Extreme Programming, Programming, Programming Education.

INTRODUCTION

The extreme programming (XP) [1] is one of the best-known agile methods. It is widely-used by software industries around the world, especially after the announcement of the Agile Manifesto [2] in 2001. The 12 XP practices [3] bring us extremelyuseful programming techniques. With this innovative industrial trend, the programming education in the universities need to redesign some currentcourses, such as Java Programming, Data Structure, Algorithm, Software Engineering, Software Engineering Environment and so on, and to come up with a new XP-based method for programming educationthat integrates XP with the above-mentioned courses. An endeavor in this regard called the "myAgile"method is reported in this paper.

The goal is to use this new method to teach the undergraduate students of Computer Science departments in Taiwan so that the studentscan gain a modern and quality-conscious programming education. To do that, a grade system projectin Java that applies this method has been designed for the students to gain hand-on experience with it.

This paper is organized as follows. The "myAgile" method section describes the steps of "myAgile". Next, the new features of "myAgile" are depicted. An experiment of the grade system project is then described. And,the last section gives concluding remarks.

"MY AGILE" METHOD

Like XP, "myAgile" focuses on face-to-face communication. All of its steps are done in"pair programming". That is, two developers sit side by side, one being the driver, while the other the reviewer. And they can switch rolesat any time.

This work is sponsored by the National Science Council (NSC) in Taiwan under Grant No. NSC101-2221E-008-131.

The 11 steps (0 to 10) of the "myAgile" method are depicted below. Note that steps 5, 6, 9, and 10 are XP steps. Steps 0, 1, 2, and 3 are adopted from Software Engineering, especially Object-Oriented method. And steps 4, 7, and 8 are specially designedby the author, Professor Chen.

0) Exploring requirements

This step gathers the features (user stories) of the software system by a pair of two customers. Since this step is done in the customer company, rather than in the software company, it is omitted in the experiment.

1) Scenarios

For each user story, the "on-site customer" and a developer jointly develop various scenarios from the simplest case gradually to the more complex cases in a step-wise refinement manner. Fig.1 (a) shows an example of a simple scenario, while Fig. 1 (b) shows a complex one, which is derived from the simple one in Fig. 1 (a). Notice that the boldface in Fig 1 (b) comes from Fig, 1 (a).

The screen promptID
The user wants to quit
The screen showFinishMsg

Fig 1 (a):A simple scenario.

The screen promptID
The user inputs ID
The screen showWelcomeMsg

The screen promptCommand:
1)show grade 2)show rank 3)update weights 4)exit
The user wants to exit

The screen promptID
The user wants to quit
The screen showFinishMsg

Fig 1 (b):A complex scenario derived from 1 (a)

2) User manual and Acceptance test cases

For each scenario, add the exact data (system output data and user input data) to it. By doing so, a scenario turns into an acceptance test case.After all the cases are done, we summarize them to get a simple user manual, which gives users a quick overview of the software system. Fig. 2 shows the acceptance test case derived from the scenario shown in Fig. 1 (a). In Fig. 2, the system outputs "Enter ID or Q (Quit) ? " and the user inputs "Q" to quit. After that, the system outputs "Finished" meaning that the system is finished.

International Journal of Advances in Computer Science and Technology (IJACST), Vol.2 , No.5, Pages : 06-10 (2013)
Special Issue of ICACSIT 2013 - Held during 09-10 May, 2013, Malaysia

The screen shows：Enter ID or Q (Quit) ？ Q
The screen shows：Finished

Fig 2: An acceptance test case

3) *CRC session*[4]

For each acceptance test case, the on-site customer and 4 developers (2 pairs) sit around a table to trace the case to extract objects and object interactions. The objects relate to the classes, while the interactions refer to the public methods in the classes. A class name with its public method names makes up a class interface. And the system architecture is composed of all these class interfaces. Don't forget to add a class header on top of a class interface to document it.

Fig.3 shows an example of class interface with class name C1 and two method interfaces, namely, a constructor C1 and another method m1 with two parameters p1 and p1 and the return type C2. Of course, there is a class header on top of it.

```
/*---------------------------------
        CLASS HEADER
---------------------------------*/
public class C1 {
        public void   C1 ();
     public C2     m1 (p1, p2);
} //end class C1
```

Fig 3: An example class interface C1

4) *Reverse engineering tool*

The Java class interfaces can be automatically processed by a reverse engineering tool to get the graphical documents, such as UML class diagram [5]. Currently, the tool eUML2 is used in the experiment. The developers do not need to spend time to draw diagrams and verify them anymore. It is guaranteed that the diagrams are consistent with the source code. What a relief!

In the experiment, the project is a new development. However, quite a few real-world projects are maintenance jobs in which this step is particularlycrucial. In this case, the maintenance engineers should first get a hard copy of the class diagram by using the tool. Then, use pencil and eraser to carefully modify the diagram and trace the modified diagram against acceptance test cases by pair programming or group (2 pairs) review.After the diagram is settled, modify the class interfaces and class headers in .java fileaccordingly.

5) *Dispatching and scheduling*

After step 3 CRC session, the class interfaces are available and posted on the white board in the software company. Each class interface is "picked-up" by a pair of 2 developers based on the pair's experience and preference. To be exact, this is not dispatching work. Instead, it is picking-up work. About scheduling, a pair themselves decides the number of days they need to finish a class, again based their experience plus a time allowance. Like XP, we use whiteboard to record and control the dispatching and scheduling activities.

6) *Unit test code*

Every public method (calledunit in XP) in a class interface needs to develop a number of test cases. And, each case is a combination of input parameter values of the method. For example, if a method "m1"has two input parameters, each of which has 2 values. Then, there are 4 cases to be tested for the unit. Next, develop JUnit test code for each test case. Briefly, what test code does is to assert equality of 1) hand-calculated expected result and 2) the actual result by executing the method. Fig. 4 illustrates the test cases and the simplified JUnit test code for method "m1".

```
testM1
test case 1: p1 with value1     p1 with value1
test case 2: p1 with value1     p1 with value2
     test case 3: p1 with value2     p1 with value1
     test case 4: p1 with value2     p1 with value2
     public testM1Case1 () {
          1. Construct an object of C1 called o1
          2. hand-calculate expectedResult of o1.m1(value1, value1)
          3. actualResult = o1.m1(value1, value1)
          4. assertEqual(expectedResult, actualResult)
     public testM1Case2 () { ……}
     public testM1Case3 () {…….}
     public testM1Case14() {…….}
```

Fig 4: An illustrationof unit test code

7) *Data structure design*

In object-oriented design, presumablythe public methods of a class work on the common data structure of the class. Further, the data and the methods are inter-related. If an abstract data structure is used, the algorithms of the methods will be simplified. On the contrary, if low-level data structures, such as arrays, are used, the algorithms will get complicated. In this case, the source code will get longer, more complex, and more difficult to maintain.

We use a graphical document called "design sketch" to record the data structure along with a simple set of input data. This design sketch helps the developer to figure out how to get the output data from the input data, thus help build the problem solving thinking. Fig. 5 shows a design sketch with an array data structure and the input data "3, 1, 4, 2". And the task is to sort "3, 1, 4, 2" into "1, 2, 3, 4".Note that the task is to sort n data. But, the design sketch shows only 4 data to ease the problem solving thinking and the algorithm tracing later on.

The sketch shows that we select the minimal number or "min" (1),from a series "3, 1, 4, 2" and swap "min" with the leftmost number (3). Next, we do the same thing to the smaller series without the leftmost number, that is "3, 4, 2". Repeat this until the series has only two numbers in it, that is "4, 3".Amazingly, we now have a sorted series "1, 2, 3, 4". Note the array index 0, n-2, n-1 in the sketch that will help the developer a lot in the next step, algorithm design.

ISSN 2320 - 2602

International Journal of Advances in Computer Science and Technology (IJACST), Vol.2 , No.5, Pages : 06-10 (2013)
Special Issue of ICACSIT 2013 - Held during 09-10 May, 2013, Malaysia

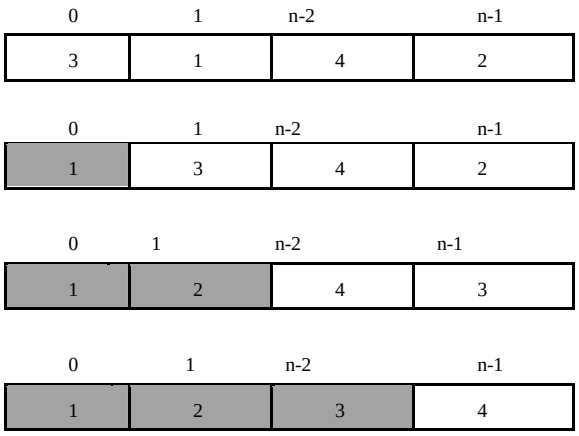

Fig 5:A design sketch of SORT

When it comes to Java Programming, we use the Java data structure classes of the Java collection framework (JCF) [6] such as ArrayList, LinkedList,TreeMap, TreeSet, HashMap, HashSet, and PriorityQueue. Notably, the methods of these classes are all equipped with Big O time estimate. This enables a development team to get time estimates for all the methods they developed, which is so desperately needed by the customer!

8) Algorithm design

Following the previous step, we write down the problem solving thinking in natural language text called "pseudo-code". This usually will take several rounds from a very abstract level down to the algorithm level. Fig. 6(a) shows ahigh-level abstract pseudo-code, while Fig. 6(b) shows the low-level pseudo-code that is the algorithm. Note that the loop indexes specified in Fig. 6(b) such as 0 and N-2 can be traced all the way back to the design sketch!

1. select min from the initialseries, and swap it with the leftmost number.
2. from the smaller series (that without the leftmost number), select min and swap it with the leftmost number
3. repeat(2) until the smaller series contains only two numbers

Fig 6(a): A high-level pseudo-code of SORT

```
for i from 0 upto N-2
1.min points at leftmost number of array[i..N-1] (that is i)
2.fromarray[i..N-1]selectmin
for j from i+1 upto N-1
ifthe number at j is less than that at min let it be min end if
end for
3.swapthe number at min with the leftmost number (that is
      array[i])
end for
```

Fig 6(b): A low-level pseudo-code of SORT

One last thing about this step is that once the pseudo-code is done, the students need to manually trace it against a simple set of input data. With this abstract pseudo-code, the tracing would be not as difficult as tracing source code. And, if the tracing is done in a "slow but firm" manner, all the bugs will go away, even before the coding starts! It surely takes a lot of mental practice and quality-consciousness to do it right.

9) Coding

Coding becomes a relatively simple job in this"myAgile" method. All you need to do is to follow exactly the structure and the logic of pseudo-code to add source code beneath it. Fig. 7 shows the sort source code in Java.

```
for (int i = 0; i <= array.length-2; i++)
{     int min=i;
for (int j = i + 1; j <= array.length-1; j++)
if (array[j] < array[min]) min = j;
swap (i,min);
}
```

Fig 7:A source code of SORT

10) Unit testing and Acceptance testing

Unit testing refers to running the test code of a unit to test the source code of it. In XP, unit testing is equivalent to integration testing. The reason for this is that we continuously (one unit at a time) unit-test and integrate the units into the system in a bottom-up manner, thus making the low-level units naturally integrated with the current unit being tested. This is called "continuous integration" [7], which turned out to be a powerful feature of XP.

When all the units of a user story (feature) are integrated, the on-site customer tests the user story again the acceptance test cases. That is, the on-site customer reads the test cases one by one and manually run the system accordingly. This is the acceptance testing of XP.Next, we will reveal the new features of the "myAgile" method.

NEW FEATURES

We here point out the new features of "myAgile" method from three perspectives: 1) requirement engineering, 2) architecture design, and 3) detailed design.

Requirement engineering

In this regard, we offer 3 new features:

First, we add Step (0) "Exploring requirements", which is done in the customer company to gather user stories (features) to be developed by the software company.

Second, we combine two documents, namely, scenarios and acceptance test cases, into one. This is agile in a sense. XP does not require producing the two documents. But, we figure that they are needed, especially for programming education. In "myAgile" we first develop scenarios. Then we add exact data (inputs and outputs) to them, and convert them into acceptance test cases.

Third, we use step-wise refinement technique to develop multiple test cases from the simplest case to more complex cases. In Software Engineering courses, normally they do not do this kind of breaking-down a case. They just do complex cases. This, we believe, will facilitate the testing, especially when bugs occur.

International Journal of Advances in Computer Science and Technology (IJACST), Vol.2 , No.5, Pages : 06-10 (2013)
Special Issue of ICACSIT 2013 - Held during 09-10 May, 2013, Malaysia

Architecture design

We use the industry-used CRC sessionto develop the architecture design of the system. This work is usually done by two pairs (4 developers) sitting around a table, brain-storming the scenarios to extract objects and object interactions.

After the session, we require the team to develop the class interfaces as system architecture, which can be automatically transformed into UML design diagrams by using a reverse engineering tool. By doing so, we essentially eliminate the tiresome, error-prone work of drawing and verifying design diagrams!

We also stress the importance of documenting the class interface with a class header, which makes source code much easier to read and maintain.

Detailed design

Traditionally, all the Computer Science departments offer Data Structure and Algorithm courses, but not from the software development perspective. We want to think "out of the box", and make these two courses fit into this XP-based new method.

We designed two documents, namely, "design sketch" and "pseudo-code" to do that. The design sketch draws the data structure and feeds into it a simple set ofinput data. The graphical sketch helps human brains of developers to come up with problem solving thinking. After that, the pseudo-code is used to document that thinking in a hierarchy from high abstraction level down to low-level (algorithm level). We feel that this process is rather human!Next, we will describe an on-going experiment of the "myAgile" method.

AN EXPRIMENT

We have designed a grade system project for the students to practice the "myAgile"method. A brief processdescription of the project follows:

The students use pair programming to apply the Extreme Programming (XP)-based "myAgile"method to developa grade system in Java. With regard to the software engineering environment, the students must use:

1) Eclipse [8]to develop easy-to-read Java source code, including header and pseudo-code;
2)JUnit[9] to develop Java unit test code;
3) eUML2 [10]to automatically generate UML class diagram.

Every method should not exceed10 source lines. If too long, break it down into low-level private method. Every public method needs Big-O time estimate [11]. At the end, turn in 3 documents:

1) User manual & Acceptance test cases (.doc file)
2) Source code (.java file),
3) Unit test code (.java file).

The grading will be based on readability of the documents. And, a questionnairewillbe distributed to collect students' experiences about using "myAgile" method.

Currently, we are particularly interested with how "Pair programming" and "Continuous integration" XP practices are done in the experiment. We are consideringto usethe "action research" technique to analyze the pair programming activities.

Next, a brief product description of the project follows:

This grade system allows a user (student) to get his/her total grade and rank. The total grade is based on the weights, which can be updated. The rank denotes the order of the total grade in the class.

Input file: The scores of all the students in a class. For example:

962001044 John Lyn 87 86 98 88 87
962001051 Wen Lee 81 98 84 90 93

Note the data field names above are:

ID name lab1 lab2 lab3 midTermfinalExam

We gave the students 6 scenarios along with 6 acceptance test cases. And we ask them to add 2 more cases. We did CRC session for the students so that all of them work on the same architecture design, which is composed of 4 classes, namely, "Main", "UI", "GradeSystems", and "Grades". The data structure used in class "GradeSystems" is Java "LinkedList". The pseudo-codes of all the methods are given. And, an example JUnit test code of a method is given too. Also, a simplified input file with just two lines (see the input file above) is provided for students to trace the algorithms. One-line input file will not do the job because executing a loop requires two lines at least. Of course, a real input file with about 60 lines is provided for testing.

The experiment is currently being conducted in 4 universities in Taiwan. There are totally about 230 undergraduate students participating in it. And we look forward to gathering valuable feedback from it.

CONCLUDING REMARKS

The agile method, especially the extreme programming (XP) method, has brought about a profound change to the programming profession, making programming more human, more enjoyable, and more quality-conscious. How should the programming education community respond to this trend?

Inspired by the industry-hailed XP method, a XP-based "myAgile" method is presented in this paper that integrates XP with some current university programming courses. We also designed agrade system project to experiment on the method, which is now being undertaken in several universities in Taiwan.

We expect to tune-up the "myAgile" method after the experiment. And, in the long run, we expect to propose a programming courses redesign initiative based on this workin order to meet the industrial need.

ACKNOWLEDGEMENT

The author wishes to thank Prof. C.Y. Huang, Prof. In-Hon Wang, and Prof. John Li for participating in the experiment of the Grade System Project. The author also wishes to thank Tung-Ying Tsai for his editing assistances.

REFERENCES

[1]K.Beck,*ExtremeProgramming Explained: Embrace Change,* U.S.:Addison-Wesley Professional,1999.

International Journal of Advances in Computer Science and Technology (IJACST), Vol.2 , No.5, Pages : 06-10 (2013)
Special Issue of ICACSIT 2013 - Held during 09-10 May, 2013, Malaysia

[2] R.C. Martin, *Agile Software Development, Principles, Patterns, and Practices*, 1st ed. Prentice Hall,2002.

[3]K.Beck,*ExtremeProgramming Explained: Embrace Change*, U.S.:Addison-Wesley Professional,1999.

[4] G. Booch,J.Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, 2nd ed. Addison-Wesley Professional,2005.

[5] G. Booch,J.Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, 2nd ed. Addison-Wesley Professional,2005.

[6]W.J.Collins, *Data Structures and the Java Collections Framework*, Wiley, 2011.

[7]P.M. Duvall, S. Matyas and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk*, 1st ed. Addison-Wesley Profession,2007.

[8]The Eclipse Foundation, "Eclipse."[Online]. Available:http://www.eclipse.org/

[9]GitHub, "JUnit."[Online]. Available:http://junit.org/

[10] Soyatec ,"eUML2."[Online].Available: http://www.soyatec.com/euml2/

[11]T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd ed.:The MIT Press, 2009.