Application of Test-Driven Development Framework for Environmental Software:
A Case Study in Long-Term Photosynthetic Process Simulation

Yasuyuki Egashira[a], **Miyuki Shibata**[a], **Korekazu Ueyama**[a] **and Koich Yamada**[b]

[a] *Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka, 560-8531 Japan*
[b] *Department of Materials and Life Science, Seikei University, Musashino, Tokyo 180-8633, Japan*

Abstract: In test-driven development, programs for tests are written before coding the main program, and using software frameworks for support testing, implementation of the program becomes easy by the frequent and low cost feedback of these tests. In this work, application of test-drive development framework for environmental software was attempted. As a case study, a simulator for photosynthesis and plant growth was developed using JUnit framework as a base of testing tool for test-driven development. In development phase, test case class which checking all equations in the model is constructed first, using JUnit with some extensions. And afterward, simulator for the model is developed, with the aid of this test case class. The test case class and the simulator were developed independently by different programmers and checking misunderstandings of equations, in addition to simple coding errors. This test code is also utilized during application phase of the model simulator. All equations in the model are tested for every simulation calculation, therefore, some rare combination of variables which results exceptional case of some equation will be checked and then users to be alerted. 2 years of simulation with 1 hour time-step was performed using series of measured data of photo-irradiation, temperature, humidity, and wind speed at an arid land in Leonra, Western Australia. The simulation, running with the test case class, was alerted to the unsatisfied leaf energy balance equation in some conditions. Afterwards, it became clear that the empirical equation for gas exchange conductance was the cause of such errors. The case study showed that utilization of the test case class successfully found an error, easy to fix but difficult to find.
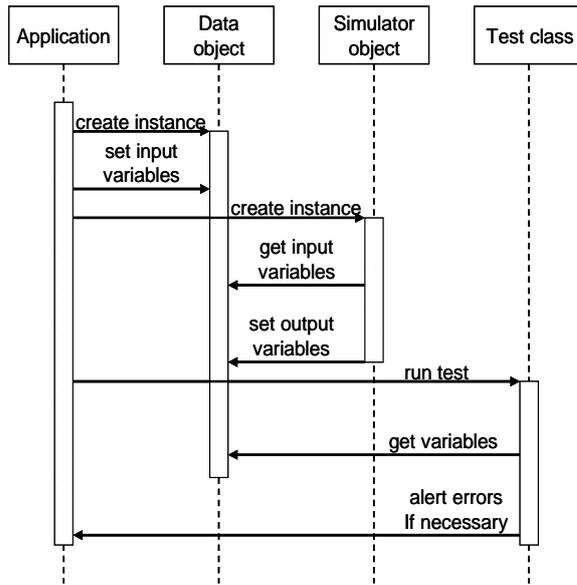
Keywords: Test-Driven Development; Framework; Modeling

1. INTRODUCTION

Test-driven development is now becoming popular programming technique in many fields. According to this technique, programmer writes tests before coding the program, and with the aid of a software framework, they can perform the automated tests. Implementation of the program becomes easy by the frequent and low cost feedback of these tests, and quality of the program will be improved. In this way, the program product is guaranteed to be the same as what was intended [Beck 2003]. However, such tests are not aimed to guarantee the soundness of the program, nor the suitability in application.

In the case of environmental software, such as simulator of ecological phenomena, the suitability of the program in application is mainly examined. Agreement between program and intended model,

and soundness of the model itself are often implicitly assumed. Some researchers focused on this point, and proposed software frameworks, such as EcoBAS [Benz, 2001], ECLPSS [Wenderholm, 2005][Woodbury, 2002], in order to overcome the troubles on the agreement between program and model. Within such software frameworks, models are described by documents in specific format defined by each framework, and program source codes are automatically generated from such documents. Consequently, the program generated is guaranteed to be in agreement with the model described in its original document. However, such framework will require some restriction for the program development.
Soundness of the environmental model is more difficult to guarantee, because, there remains possibility that some rare combination of variables cause exceptional case for some equation in the model, even after intentional tests.

Fig. 1. Method call relations between test class and simulation class.

In this work, application of test-driven development framework for environmental software was attempted. To ensure the agreement between model and software, test case class checking all equations in the environmental model was constructed first, using software framework for test-driven development. And afterward, simulator for the model was developed, with the aid of this test case class.

Additionally, this test case class was also utilized during application phase of the model simulator, in order to check the soundness of the model. All equations in the model were tested for every simulation calculation, so, if some rare combination of variables caused exception to some equation, users were to be alerted. In this way, soundness of the model is ensured at least for the conditions calculated.

As a case study, development of a growth simulator for arid land plant, including photosynthesis process model, was attempted. A long-term simulation by developed program was also performed and examined.

2. SOFTWARE TOOLS AND METHOD

2. 1 JUnit

As a software framework supporting test-driven development, JUnit was selected in this case study. JUnit is an open source Java testing framework used to write and run repeatable tests. It was originally written by Erich Gamma and Kent Beck. (Website: http://www.junit.org)

At the start of the development process, all equations in the model were listed, and all variables appearing in equations were also listed. Before coding of tests and simulator, a class named `"Data"` class was prepared, for the purpose of holding all variables in the list. This data class has properties that correspond to variables, in other words, when a variable named `"X"` is in the list, object of this data class has `"setX"` method to write the value to it, and also has `"getX"` method to read the value from it. A simulation class, which contains implementation of simulation algorithm, was designed to read input variables from given data class object and, after calculation, write results to it. A test case class was also prepared in order to check all equations listed, using the values held in the data class object. In development phase, "TestRunner" classes prepared in JUnit framework were employed to perform this test case class to support simulation class development.

A main program of the simulator, developed for running simulation class during application phase, was incorporated with `"TestRunner"` function. This program set up input variables to a data class object, gives it to the simulation class to perform calculation, and afterwards, passes it to the test case class to check the results (Fig.1). The main program displays the calculated results and/or stores them into a file, and when test case class finds out an exception, this main program displays alert message. As shown in Fig.1, within main program, test case class is activated after simulation class, but during the process of software development, test case class was developed first, and then the implementation of simulation class followed.

2.2 Assertion method for test equations

JUnit framework provides methods for checking equivalence of two variables. For example

```
assertEquals(
  java.lang.String message,
  double expected,
  double actual,
  double delta)
```

method in junit.framework.Assert class compares two double variables `"expected"` and `"actual"`. And if their difference exceeds `"delta"` value, message held in the first argument of this method will be displayed.

```
List 1(a) Test code example.

  public void testEqTleaf() throws Exception {
    double[] terms = new double[5];

    double Ia = data.getD_I_a();
    double La = data.getD_L_a();
    double Le = data.getD_L_e();
    double SH = data.getD_S_H();
    double Lambda = data.getD_Lambda();
    double Eleaf = data.getD_E_leaf();

    terms[0] = Ia;
    terms[1] = La;
    terms[2] = -Le;
    terms[3] = -SH;
    terms[4] = -Lambda*Eleaf;

    assertEquation("Energy balance equation not satisfied", terms, 1.0E-06);
  }
```

```
List 1(b) Example of error message

junit.framework.AssertionFailedError: Energy balance equation not satisfied
 +1028.8… +899.6… -988.5… -859.5… -80.2…[= 0.099… ] != 0 with allowance=1.0E-6,
absAllowance=1.0E-10
  at junit.framework.Assert.fail(Assert.java:47)
  at jp.….TestSimulationBase.assertEquation(TestSimulationBase.java:1271)
  at jp.….TestSimulationBase.assertEquation(TestSimulationBase.java:1277)
     ...
  at junit.framework.TestSuite.run(TestSuite.java:203)
  at junit.framework.TestSuite.runTest(TestSuite.java:208)
     ...
```

When examining equivalence of left and right terms of equations, tolerance for comparison is often given as a relative value. However, in this "assertEquals" method, value of "delta" is defined as absolute value. Moreover, some equations are given in the form of balance between more than two terms. So, when an extension of "junit.framework.TestCase" class was developed, two methods for checking equations were implemented.

One is a method for checking equivalence of left and right terms in the equation. And its signature is

```
assertEquation(
  java.lang.String message,
  double left_term,
  double right_term,
  double relative_tolerance,
  double absolute_tolerance).
```

In this method, relative tolerance can specify directly. Input of absolute tolerance, or both two tolerances, can be omitted and in such case, previously defined values are used instead.

The other is a method for checking equations in the form of balance of terms. Its signature is

```
assertEquation(
  java.lang.String message,
  double[] terms,
  double relative_tolerance,
  double absolute_tolerance).
```

This method check the sum of values in "double" array defined as second argument. If the sum is recognized as non-zero value, with the tolerances defined as third and fourth arguments, this method will cause alert. Additionally, like first method described here, tolerances can be omitted.

List 1 demonstrates the usage of the second type of assertEquation method. Source code for the test method for checking a following equation is shown in list 1(a),

$$Ia + La - Le - SH - Lambda*Eleaf = 0. \qquad (1)$$

Firstly, an array variable with five double values is declared, because Eq.1 is representing balance between five terms. Secondly, all values of variables appearing in this equation are read out from data object by "getX" methods. Then, the values of each five terms in Eq.1 are calculated
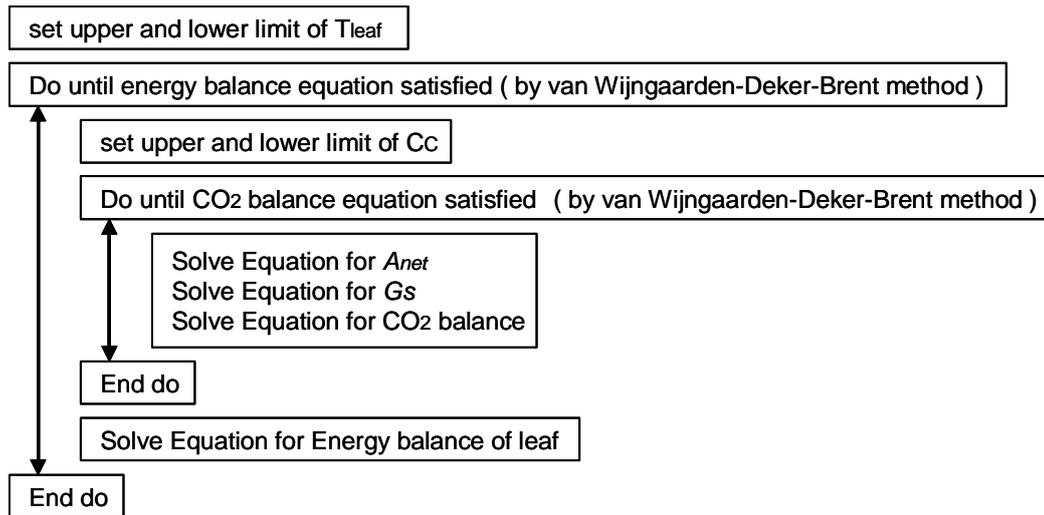
```
┌─────────────────────────────────────┐
│ set upper and lower limit of Tleaf   │
└─────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────┐
│ Do until energy balance equation satisfied ( by van Wijngaarden-Deker-Brent method ) │
└──────────────────────────────────────────────────────────────────────┘
      ┌─────────────────────────────────────┐
      │  set upper and lower limit of Cc     │
      └─────────────────────────────────────┘
      ┌──────────────────────────────────────────────────────────────────────┐
      │ Do until CO2 balance equation satisfied   ( by van Wijngaarden-Deker-Brent method ) │
      └──────────────────────────────────────────────────────────────────────┘
            ┌─────────────────────────────────────┐
            │  Solve Equation for Anet             │
            │  Solve Equation for Gs               │
            │  Solve Equation for CO2 balance      │
            └─────────────────────────────────────┘
      ┌──────────┐
      │ End do   │
      └──────────┘
      ┌─────────────────────────────────────────┐
      │ Solve Equation for Energy balance of leaf │
      └─────────────────────────────────────────┘
┌──────────┐
│ End do   │
└──────────┘
```

Fig. 2. Outline of procedure for solving the photosynthesis model.

and stored in the array. Finally, the array is handed to the assertEquation method so as to check the summation of the array. In other words, the assertEquation method check that the summation of the elements is not recognized as non-zero value

In this example, the relative tolerance, the third argument in assertEquation method, is defined as 1.0E-6. This means, if the summation exceeds 1.0E-6 of the maximum among the absolute value of values stored in the array, then the equation will be recognized as an exceptional case. If so, an error message stored in the String object defined as first argument of this method will be displayed, with some additional information about the cause of exception, as shown in list 1(b). Additionally, the fourth argument in assertEquation method is omitted and pre-defined value is used in this case.

3.   CASE STUDY

To ensure the advantage of application of test-driven development framework, a case study on developing a simulator of a plant growth model in arid environment was attempted. A long-term simulation by developed programs was also performed and examined.

The plant growth model is based on a photosynthesis process model combined with a carbon balance model of plant bodies in order to simulate growth of plants, and, the adaptation effects of plants on dry conditions are included in the photosynthesis process model. Model development and estimation of parameters in the model were performed using the measurement results of a sample of *Eucalyptus camaldulensis* tree grown at a site located in Leonora, Western Australia where mean annual rainfall is around

200 mm/y. Continuous measurements of soil water content at 0.5, 1.0, 2.0 m from the soil surface were conducted at this site. Data of solar irradiation, air temperature, humidity and wind speed are also available.

3.1   Model and simulator

Following the model described in Amthor [1994], the process model of leaf photosynthesis consists of three parts. First is a part which represents photo-reactions and enzyme reactions of photosynthesis, second is a part which estimates the gas exchange of water and carbon dioxide through the stoma, and last is a part which calculates energy balance equation including absorption of light, radiation, and latent heat loss by transpiration of water.

In order to express the effect that trees are adapted for dry conditions, the equation for stomatal conductance, Gs (mol/m^{2}s) in the gas exchange part, was modified so that conductance of stoma will change with the averaged soil water potentials. The equation is similar to the Ball's equation [Ball, 1987] as show below.

$$G_S = m \frac{A_{net}}{C_s \times VPDL} + G_{S\min} \quad (2)$$

Where A_{net} is net photosynthesis rate (μmol/m^{2}s), C_s is CO2 mol fraction at leaf surface (μmol-CO2/mol), $VPDL$ is vapor pressure deficit at leaf (Pa). Values of m and $G_{S\min}$ are given by the function of soil water potentials.

Gs is related to A_{net}, so, leaf temperature and CO2 concentration in the leaf cell, C_C (μmol- CO2/mol), affect to the value of *Gs* through photosynthesis rate. Meanwhile, *Gs* also affect to temperature and C_C by changing transpiration and CO2 exchange rate through stoma. Therefore, consistent values for both leaf temperature and C_C must find out during simulation calculation. As shown in Fig.2, duplicated root-finding algorithm was employed in the simulator. Inner root-finding algorithm solves carbon dioxide balance equation to find out C_C, and, outer root-finding algorithm solves energy balance equation to find out leaf temperature. As a root-finding algorithm, the "van Wijngaarden-Deker-Brent method", a modification of "bisection method", was employed.

3.2 Implementation phase

As mentioned before, test case class was implemented first, and then simulation class was developed. Writing a test class code is simple and straightforward task, in contrast to implementation of simulation algorithm. And repeated and frequent test using the test class was helpful for development of somewhat complicated simulation programs.

Additionally, test case class and simulation class were separately developed by different person. This practice sometimes triggered useful discussion about understanding of the detail of the model equation.

3.3 Application phase

Using the simulator developed, two years of simulation with 1 hour time-step was performed using series of measured data sets of photo-irradiation, temperature, humidity, and wind speed at an arid land in Leonra, Western Australia.

In almost all time steps, calculation results pass the test successfully. But in rare cases, combination of input variables cause error message. Cases of nighttime without photo-irradiation, at the same time 100 % of humidity were responsible for such errors. The error messages indicated the unsatisfied energy balance equation(Eq.1), and the leaf temperatures (*T*leaf), derived from this equation by root-finding algorithm, seems miscalculated.

Figure 2 shows the residual of energy balance equation (Eq.1) as a function of the assumed leaf temperature (*T*leaf). With typical conditions, the residual changes with the *T*leaf continuous and monotonous manner. In contrast, with error-
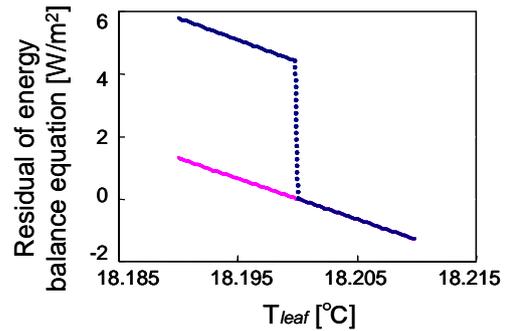
Fig. 3. Residual of energy balance equation (Eq.1) for a leaf as a function of T*leaf*. Blue line show the exceptional case, dark and 100 % humidity. And red line corresponds to the simulation with fixed equation.

reported conditions, the residual changes monotonous but discontinuous manner. The root-finding algorithm is trying to find out the *T*leaf at cross point of horizontal axis and residual energy curve, so, in such cases, discontinuous point of the curve is reported as the balance point. Therefore, the cause of the error is not a problem of calculation method, rather, is a problem of model equations.

Examination of the model equations gives following explanation as the cause of this discontinuous energy curve.

The cause of discontinuity is the Eq.(2), an empirical equation gives the relationship between net photosynthesis rate (A_{net}) and stomatal conductance (*Gs*). Under dark conditions, no photosynthesis reaction occurs but respiration rate remains, so A_{net} becomes negative value. In such case, Eq.2 gives negative value as *Gs*, and this value is replaced by a minimum limit of stomatal conductance during actual simulation. However, when both 100 % humidity condition and air temperature higher than *T*leaf are realized, the value of *VPDL* also takes negative value, and then, stomatal conductance takes large positive values. In changing *T*leaf, the value of *Gs* is jumping up from the minimum limit to some large positive value, around the air temperature.

This discontinuity is the result of both negative A_{net} and negative *VPDL*. The negative A_{net} is not uncommon, and was already taking care. But negative *VPDL* is the result of 100 % humidity condition, which is uncommon, especially in arid lands.

Checking negative *VPDL*, this problem was easily avoided. But, such exceptional case of model

equation seems very difficult to find out, because there will be no message on exception nor error during root-finding calculation, without aids of the test case class. Additionally, combination of conditions responsible for such exceptional case is not typical and not likely to be selected as a sample case for examination of simulator and evaluation of model. Utilization of the framework of test-driven development method was shown to be useful to find the error, difficult to find otherwise, in this case study. The author expects that the utilization of this technique is also useful in other cases.

4.   CONCLUSION

In this work, application of framework for the test-driven development to environmental software was attempted. The test case class was utilized not only for development phase but also for application phase of the simulator. The case study showed that utilization of the test case class successfully found an error, easy to fix but difficult to find.

5.   ACKNOWLEDGEMENTS

6.   REFERENCES

Amthor, J. S., Scaling CO_2 photosynthesis relationships from the leaf to the canopy, *Photosynthetic Research*, 39,321-350,1994.

Ball, J. T., E. W. Woodrow, and J. A. Berry, A Model Predicting Stomatal Conductance and its Contribution to the Control of Photosynthesis Under Different Environmetal Conditions, *Progress in Photosynthesis Research*, IV, 5.221-5.225, 1987.

Beck, K., *Test-Driven Development By Example*, Addison-Wesley, Boston, 2003.

Benz, J., R. Hoch, and T. Legovic, ECOBAS - modelling and documentation, *Ecological Modelling*, 138, 3–15, 2001.

Egashira, Y., M. Shibata, K. Ueyama, H. Utsugi, N. Takahashi, S. Kawarasaki, T. Kojima, and K. Yamada., Development of tree growth simulator based on a process model of photosynthesis for *Eucalyptus camaldulensis* in arid land, In proceeding of Desert technology VIII (DT8), Nasu, Japan, November 28 – December 2, 2005.

Wenderholm, E., Eclpss: a Java-based framework for parallel ecosystem simulation and modeling, *Environmental Modelling & Software*, 20(9), 1081-1100, 2005.

Woodbury, P. B., Beloin, R. M., Swaney, D. P., Gollands, B. E., and Weinstein, D. A., Using the ECLPSS software environment to build a spatially explicit component-based model of ozone effects on forest ecosystems, *Ecological Modelling*, 150(3), 211-238, 2002.