EDUCATION-ORIENTED JAVA COMPILER WITH ACTIVE E-TA FOR PROGRAMMING EDUCATION

Shun Hattori
School of Computer Science, Tokyo University of Technology
1404-1 Katakura-cho, Hachioji, Tokyo 192-0982, JAPAN
hattori@cs.teu.ac.jp

Hiroyuki Kameda
School of Computer Science, Tokyo University of Technology
1404-1 Katakura-cho, Hachioji, Tokyo 192-0982, JAPAN
kameda@cs.teu.ac.jp

ABSTRACT

In recent years, those who do not acquire enough programming ability and also hesitate in programming have been increasing, even though they are Computer Science students/graduates. One of the major reasons is that they could not avoid facing very difficult and unexpectedly massive compile errors with unknown technical words when they were beginners of programming. This paper proposes an education-oriented compiler with reactive/proactive e-TA (electronic Teaching Agent) for programming education, and shows an implementation of our education-oriented Java compiler to present not only static compile errors as-is but also easy-to-understand advice appropriately and on a timely basis (dynamically) depending on programming knowledge of learners who browse the compile errors as if at least a high-level human teacher or TA were always man-on-man standing at the learners' side.

KEYWORDS

Education-oriented Compiler, e-TA (electronic Teaching Agent), User-Adaptive Compiler, Proactive Compiler, Programming Education, Software Engineering Education.

1. INTRODUCTION

Today, software is one of the most important social infrastructures. Our societies require capable human resources who can develop software at higher level, and various educational institutions such as universities provide software engineering (programming) education aggressively. In Japan, however, those who do not acquire enough programming ability and also hesitate in programming have been increasing in recent years, even though they are CS (Computer Science) or IT (Information Technology) students/graduates.

One of the major reasons is that they could not avoid facing very difficult and unexpectedly massive compile errors with unacquainted English words or technical words and thus have given up learning about programming knowledge and skills when they were beginners of programming. Conventional compilers such as javac are for not programming education but software development, and are for not beginners of programming but intermediate and advanced developers. Another reason is that such human-powered supports by a teacher and a few TAs (Teaching Assistants) as deciphering compiler's error messages for learners and giving appropriate advice timely have several limitations in introductory programming exercises.

In this paper, we propose an education-oriented compiler with user-adaptive and active e-TA (electronic Teaching Agent) for introductory programming exercises, and describe an implementation of our education-oriented Java compiler to present not only static and often difficult-to-understand error messages by the most primary Java compiler, javac, but also easy-to-understand advice appropriately and on a timely basis (dynamically) depending on programming knowledge of learners who browse the compiler's error messages as if a high-level teacher or TA were always man-on-man standing at the learners' side.

2. DESIGN OF EDUCATION-ORIENTED JAVA COMPILER WITH E-TA

This section describes a design of our proposed education-oriented Java compiler, edujavac, in programming education for novices, as a first step to the universal compiler whose error messages are easy-to-understand for anybody with any level of knowledge and skills of programming. Not to overstress new learners of programming by difficult-to-understand compiler's error messages and/or human-to-human communication such as question-and-answering, not a human teacher or TA but a computer e-TA (electronic Teaching Agent) of our education-oriented Java compiler provides the learners with such supports as deciphering compiler's error messages for learners and giving appropriate advice about what wrong in their source code and/or how to modify their source code.

Our education-oriented Java compiler with e-TA for not software development but software engineering education, especially introductory programming exercises, should have the following requirements.

1. User-Adaptiveness: The primary Java compiler, javac, has "disability of User-Adaptiveness".
   - **Learner-Adaptiveness**: In programming exercises, a human teacher or TA seems to offer a student with appropriate and timely advice based on the student's level of knowledge and skills of programming inferred by observing the student's action history (e.g., transition of source code) and error tendency from behind, and/or asking the student additional questions arbitrarily (if necessary, referring the example source code of the student's tackling assignment). Therefore, for a learner, our proposed e-TA should generate appropriate and timely advice dependent on the learner's level of knowledge (e.g., reserved keywords and technical terms) of programming.
   - **Teacher-Adaptiveness**: Before programming exercises, a human teacher had better give her/his human TAs enough guidance about how to advise her/his students, e.g., timing control and kindness control. Therefore, for a teacher, our proposed e-TA should generate appropriate and timely advice according to the teacher's educational policies about timing, kindness, and programming knowledge and skills to teach.
2. **Activeness**: The primary Java compiler, javac, has "disability of Activeness". And some learners who cannot ask a human teacher or TA because of such a personality as shyness of strangers even if stuck in compiler's error messages and would leave them unsolved. Therefore, our e-TA should not reactively but proactively advise shy learners if necessary without their explicit search of the other's advice.

Our education-oriented Java compiler with e-TA generates appropriate and timely advice to error messages by compiling a learner's source code for a teacher's assignment based on Learner and Teacher models [Hattori, 2010]. Each learner has a knowledge directory, know-dir, on her/his local computer as her/his Learner model. Technical keywords with high frequency to some extent extracted from electronic documents in the learner's knowledge directory are identified as her/his well-known keywords. Unknown technical keywords in its error messages are masked depending on the user's knowledge (i.e., electronic documents in the knowledge directory), or are supplemented with some additional explanation and by referring textbooks and Web pages. Whether to mask or supplement them for each learner and what to refer, and the interval of updating content of e-TA's advice are specified in advance and can be always switched by each teacher as her/his Teacher model.

First, our e-TA finds the most effective (but maybe the least educational) method to modify a targeted source code with errors by referring its original error messages of the most conventional Java compiler, javac. In the case of generating the most effective modification method to the javac's 1st error message for "HJW.java (typo)" in Figure 1, e-TA collects the following 6 kinds of modification candidates by inferring "insert" or "replace" as its operation from a pattern "expected" and extracting its keyword and location, and selects one based on the number of re-compile errors after applying each modification candidate:

(insert, 1, 1, "class")	3 → 1 error
(insert, 1, 1, "interface")	3 → 1 error
(insert, 1, 1, "enum")	3 → 2 errors
(replace, 1, 1, "class")	**3 → 0 error**
(replace, 1, 1, "interface")	3 → 2 errors
(replace, 1, 1, "enum")	3 → 4 errors

Next, our e-TA generates a sequence of advices by rule-based interpolating between the javac's original error message(s) and the most effective modification method, e.g., (replace, 1, 1, "class").

3. DEMOS OF EDUCATION-ORIENTED JAVA COMPILER WITH E-TA

As an implementation of our proposed education-oriented Java compiler, edujavac, in programming education, this section describes a prototype of our user-adaptive (knowledge-dependent) Java compiler with reactive or proactive e-TA to present not only static error messages by the existing Java compiler, javac, but also easy-to-understand advice appropriately and on a timely basis (dynamically) depending on programming knowledge of learners who browse the compiler's error messages. The current prototype can deal with a typo, missing of "}" or ")" etc., and not-found symbol as a part of various kinds of compile errors.

3.1 Advice Examples of Reactive e-TA

The primary Java compiler, javac, offers the quite same error messages indefinitely unless the inputted source code is modified as shown in Figure 1. Meanwhile, our proposed education-oriented Java compiler, edujavac, with reactive e-TA offers the following sequence of advices about the source code with a typo, "HJW.java (typo)", to the learner with known technical keyword(s), know-dir ={"class"}. Our edujavac with e-TA masks the other technical keywords than a learner's know-dir = {"class"} on the left as below, while it supplements them with their explanations as her/his teacher's knowledge to teach on the right.

[mask except know -dir={"class"}]

```
> cat HJW.java
clas HJW {
  public static void main(String[] args) {
    System.out.println("Hello, Java World.");
  }
}
> javac HJW.java
HJW.java:1: class, interface, or enum expected
clas HJW {
^
HJW.java:2: class, interface, or enum expected
  public static void main(String[] args) {
                ^
HJW.java:4: class, interface, or enum expected
  }
  ^
3 errors
> edujavac HJW.java
HJW.java:1: class expected
clas HJW {
^
HJW.java:2: class expected
  public static void main(String[]  args) {
                ^
HJW.java:4: class expected
  }
  ^
3 errors
> edujavac  HJW.java
HJW.java:1: class expected
clas HJW {
^
> edujavac HJW.java
HJW.java:1 : there is a typo anywhere
> edujavac HJW.java
HJW.java:1: there is a typo in the line no.1
> edujavac HJW.java
HJW.java:1: replace clas with something in the
line no.1
> edujavac HJW.java
HJW.java:1: replace clas with class in the
line no.1
```

[supplement except know -dir={"class"}]

```
> edujavac HJW.java
HJW.java:1: class, interface, or  enum expected
clas HJW {
^
HJW.java:2: class, interface, or  enum expected
  public static void main(String[]  args) {
                ^
HJW.java:4: class, interface, or  enum expected
  }
  ^
3 errors
interface is an abstract type that ...
  TEXT: Objects First with Java, pp.328
  URL:http:// en.wikipedia.org /wiki/Interface_(Java)
enum is a data type that ...
  TEXT: Objects First with Java, pp.233
  URL: http://en.wikipedia.org/wiki/Enumerated_type
> edujavac  HJW.java
HJW.java:1: class expected
clas HJW {
^
interface is an abstract type that ...
  TEXT: Objects First with Java, pp.328
  URL:http://en.wikipedia.org/wiki/Interface_(Java)
enum is a data type that ...
  TEXT: Objects First with Java, pp.233
  URL: http://en.wikipedia.org/wiki/Enumerated_type
> edujavac  HJW.java
HJW.java:1: class expected
clas HJW {
^
> edujavac HJW.java
HJW.java:1: there is a typo anywhere
> edujavac HJW.java
HJW.java:1: there is a typo in the line no.1
> edujavac HJW.java
HJW.java:1: replace  clas with something in the
line no.1
> edujavac HJW.java
HJW.java:1: replace  clas with class in the
line no.1
```

Figure 1. edujavac HJW.java (with a typo) with Reactive e-TA

3.2 Advice Examples of Proactive e-TA

Figure 2 shows advices offered to a learner with known technical keywords, know-dir ={“class”, “interface”}, by our education-oriented Java compiler with proactive e-TA. For a Java filename (.java) at the 1st learner’s prompt (**you>>**), our e-TA reactively offers all of its javac’s error messages in which her/his unknown technical keywords such as “enum” are masked. Even if no inputs as the 2nd learner’s prompt, after such a period of time as 15 minutes specified by her/his teacher, e-TA proactively offers only the 1st error message in which her/his unknown technical keywords are masked. For a technical keyword at the 3rd learner’s prompt, e-TA reactively offers the explanation of the keyword specified by her/his teacher. Even if no inputs as her/his prompt, e-TA proactively offers one by one from a generated sequence of advices after the period.

```
[mask except know-dir={"class","interface"}]
> edujavac
eTA>> Hello.
you>> HJW.java
HJW.java:1: class or interface expected
clas HJW {
^
HJW.java:2: class or interface expected
  public static void main(String[]  args) {
                ^
HJW.java:4: class or interface enum expected
  }
  ^
3 errors
you>>
eTA>> HJW.java:1: class or interface expected
clas HJW {
^
you>> interface
(continues to the right)
```

```
interface is an abstract type that ...
  TEXT: Objects First with Java, pp.328
  URL: http:// en.wikipedia.org /wiki/Interface_(Java)
you>>
eTA>> HJW.java:1: class expected
clas HJW {
^
you>>
eTA>> HJW.java:1: there is a typo anywhere
you>>
eTA>> HJW.java:1: there is a typo  in the line
no.1
you>>
eTA>> HJW.java:1:  replace clas with something
in the line no.1
you>>
eTA>> HJW.java:1: replace  clas with class in
the line no.1
you>>
```

Figure 2. edujavac HJW.java (with a typo) with Proactive e-TA

4. CONCLUSION

In this paper, we have proposed an education-oriented compiler with reactive or proactive e-TA (electronic Teaching Agent) about compile errors for not intermediate and advanced program developers but new learners of programming in the context of software engineering education, especially introductory programming exercise. And we have described an implementation of our proposed education-oriented Java compiler to present not only static and often difficult-to-understand error messages by the most primary Java compiler, javac, but also easy-to-understand advice appropriately and on a timely basis (dynamically) depending on programming knowledge of learners who browse the compiler's error messages as if at least a high-level teacher or TA were always man-on-man standing at the learners’ side. The prototype can deal with a typo, missing of “}” or “)” etc., and not-found symbol as a part of various kinds of compile errors. We are working up the registered patterns and rules to deal with as many kinds of compile errors as possible.

In the future, we plan to evaluate our proposed education-oriented Java compiler, edujavac, with user-adaptive and active e-TA who gives advices proactively or reactively by masking or supplementing a user’s unknown technical keywords, by applying it to practical introductory Java programming exercises for Computer Science (CS) freshmen at our university. And we try to invent a mechanism that not single but multiple different e-TAs collaboratively give advices to the user as if a human TA asked the other TA or teacher for help in a practical exercise classroom.

REFERENCES

Hattori, S. and Kameda, H., 2010. Knowledge-based Compiler with e-TA for Software Engineering Education. *Proc. of the 9th Joint Conference on Knowledge-Based Software Engineering (JCKBSE’10)*. Kaunas, Lithuania, pp. 265-278.