Mastering Object-Oriented Technology Using a Self-Learning and Self-Assessment System

AHMED SEFFAH, MICHEL DESMARAI S, RAMZAN KHUWAJA

Computer Research Institute of Montreal, 550 Sherbrooke West, Suite 100, Montreal, Quebec H3A 1B9, Canada

Received 13 October 1998; accepted 10 June 1999

ABSTRACT: In this article, we describe a Web-based approach that defines training needs for object-oriented developers by identifying the strong and the weak areas of their knowledge. Our system is based on the use of two tools, Guide d’Auto-Apprentissage, Self-Learning Guide (GAA) and User Knowledge Assessment Technology (UKAT) developed at the Computer Research Institute of Montreal. UKAT uses a state-of-the-art user knowledge assessment method to create a user profile of the proficiency in a subject domain. GAA is a Web-based training system that uses the UKAT to personalize a training course to facilitate self-learning. As an exemplar, we are using a Java development environment. © 1999 John Wiley & Sons, Inc. Comput Appl Eng Educ 7: 162–170, 1999

Keywords: Internet technology-based training; intelligent training systems; object-oriented technology; knowledge assessment systems; self-learning; just-in-time training; intelligent learning resources

DIFFICULTIES IN LEARNING OBJECT-ORIENTED TECHNOLOGY

Training is a major challenge facing software development organizations in their effort to migrate toward object-oriented (OO) technology [1]. Theoretical classroom presentations introduce concepts and present simple case studies in an attempt to facilitate understanding of these concepts. Textbook case studies are also limited and restricted in their scope to reflect true-life experiences. According to Lato and Dreschler [2], what must be learned are not only concepts and skills, but also the mindset that will facilitate an effective and efficient transition toward OO technology and its practice. To achieve this, the following difficulties must be overcome:

- Difficulties in learning an OO programming language. Evidence gathered from our day-to-day experiences demonstrates that the OO programming language structures must be arranged in a specific order to program effectively. Several arrangements are possible; their effectiveness depends directly on the user’s level of skill regarding the language and the acquired experience and knowledge of OO programming principles.
• Difficulties in learning OO fundamental principles. It has been observed that students often do not distinguish and are sometimes confused between object-oriented principles and their use in a development environment and/or in the language. Hohmann [3] observed that students find it easy to differentiate conceptually among inheritance, aggregation, and association relationships. However, students find it extremely difficult to create solutions that will correctly integrate these relationships.

• Difficulties in using OO development tools. In most cases, an OO development environment offers a wide array of powerful but complex tools and services. An effective teaching strategy would demonstrate how to use the environment tools and explicitly highlight their relationship with OO fundamental concepts and OO design techniques. For example, browsers that show inheritance relationships will help students understand the difference between inheritance, aggregation, and association.

• Difficulties in learning OO design and techniques. Rosson and Carroll [4] observed that the problem of learning OO design and programming techniques is compounded by the fact that expert developers are highly confident in their ability to learn new programming languages and techniques. They also find the learning curve involved in learning OO design steep, and the programming techniques frustrating and demoralizing.

The following observations make these difficulties even more critical. According to Gehringer and Manns [5], taking an OO analysis and design course and an OO programming course does not appear to be recommendable. Teaching programming before design will lead to bad design practices, which would require effort to unlearn. Teaching design before programming could result in a sterile course where students are unable to fully apply what they learn. Whitelaw and Weckert [6] observed that at least some of the problems associated with learning OO programming occur because we do not usually think of everything as an object, and objects are not all of the same type.

This situation justifies not only the incipient existence of continuing education, but also the need to design and develop a innovative on-the-job training approach. Our opinion is that an innovative approach can be obtained by combining the two following computer-based training approaches into a system available upon demand:

• Intelligent training systems (ITS). ITS can indicate the best way to solve a problem and/or organize course content [7] and/or motivate and encourage the learner during a learning session. An ITS is also able to establish a review of the learner’s interaction and/or criticize the learner’s solutions [8] and/or cooperate and compete with the learner during problem solving [9].

• Internet technology-based training. Internet technology can provide learners and educators with new tools and innovative means to communicate and disseminate information, as well as develop resourceful, constructive and outcome-driven training strategies [10]. Internet-based training has the potential to be used either internally by companies developing their own corporate Intranet [11] or externally with an Extranet targeting their client base.

In this article, we show how this combination addresses some issues related to the difficulties listed above. To illustrate our approach, we consider a course on learning Java as an object-oriented programming language for Internet-based software development. We will focus on the knowledge structure that is responsible for organizing the course content. This organization is one of the major factors when personalizing a course for different types of learners. It also directly influences the reusability and sharing of subject content between overlapping courses that are tailored for use in diverse learning scenarios.

GAA: A WEB-BASED INTELLIGENT TUTORING AND ASSESSMENT SYSTEM

The Power of Internet-Based Training

From the manager perspective, the successful experiences with Internet-based training for software engineering in general [12] and for OO technology in particular [13] show that Internet tools can provide a less costly and more efficient alternative to traditional training and continuous education approaches. Web-based performance systems can also actively support today’s demanding workforce by integrating information systems, job aids, as well as anchored instruction into unified systems that are available upon demand.

From a learner perspective, the Internet platform has the potential to support highly up-to-date, just-in-time, media-rich content and modify it whenever
needed. It also offers a flexible structure allowing self-directed, self-paced instruction on any topic that is capable of being supported by adaptive remedial and assessment strategies. Internet technology is also presented as an ideal vehicle for helping software developers perform their jobs, share expertise on how to develop well, and at the same time learn through several kinds of resources, such as real-life, “best-use” case studies, demonstrations, courseware, etc.

A Brief Overview of the GAA Architecture

The long-term goal of GAA has been to develop a generic self-learning guide that could provide user knowledge assessment and pedagogical feedback for a number of domains. We especially focused on domains that require the user to master a number of concepts and/or skills to achieve a satisfactory level of competence in the domain [14].

The GAA architecture is based on the typical Internet client/server infrastructure and the intelligent training/tutoring system (ITS) architecture. It is composed of the following four components (Fig. 1):

- The intelligent training system
- Web server (HTTP)
- The Web manager generates an HTML page based on the pedagogical engine decisions and the information contained in the subject database. The Web manager is a set of common gateway interface scripts
- A Web browser–based user (learner) interface

The ITS used in GAA includes four main components [14]:

- The knowledge network contains knowledge units (KUs) or domain entities such as OO principles and Java programming techniques. KUs are connected via multiple domain relationships (e.g., prerequisite). A course designer using a curriculum development language (see “The Knowledge Network”) creates the network. The knowledge network is the basis for the domain intelligence in our system.
- The subject database includes the domain entities that are communicated to the user. For example, the various question items are part of the database.
- The user knowledge assessment tool is the central component of the GAA. It is responsible for identifying strong and weak points. The next section describes in detail the theoretical background and how it works.
- The pedagogical engine is responsible for interpreting the knowledge state of the student, and depending on the currently selected goals and the state of interaction with the user, it determines what should come next in a session.

The Assessment Process

The user knowledge assessment module is responsible for providing a user profile of the knowledge network’s state of mastery. Based on the information it receives from the pedagogical engine about which knowledge unit is mastered by the user, the knowledge assessment module will infer the likelihood that every other KU is mastered.

The knowledge assessment tool adopts the overlay approach to defining the whole domain knowledge. It
assigns a state, which is a collection of numerical attribute values attached to the each knowledge network. Each value indicates the likelihood (i.e., probability) of a user knowing a specific KU. In the knowledge structure, implication (precedence) relationships connect KU. An implication relationship is a gradation constraint that expresses whether a certain concept must be understood before another difficult one or whether a certain skill must be acquired prior to an advanced one. These implication relationships are what enable the inferences about the mastery of KU.

The KU state is built and updated as soon as a few observations are made (e.g., questions and exercises are answered). Each observation can be viewed as a piece of evidence. This new bit of information may be propagated to other KU in compliance with the gradation constraints (inference structure).

In contrast to other work that adopted similar approaches to knowledge assessment, the knowledge structure is induced entirely from empirical data comprising samples of knowledge states. Further details about the knowledge assessment method are given in Desmarais et al. [15].

THE KNOWLEDGE NETWORK: REPRESENTING A COURSE CURRICULUM

Course Example: Building a Software Application Using Visual J++

In this section, we describe the organization of the knowledge network by using the Building a Software Application Using Visual J++ (J++_BApp) course as an example.

To detect the strong and weak areas of a developer’s knowledge about a course, the fundamental concepts that make up the body of the course and a list of questions that can assist in testing the knowledge about these concepts must be identified. Table 1 shows two questions about applet capabilities and restrictions.

A question is always linked to one or more lessons. Lessons are at the lowest level of the knowledge network hierarchy. A lesson can have multiple questions linked to it. Some of the lessons in the J++_BApp course are:

- Lesson 5: Overview of Applets
- Lesson 6: Creating an Applet User Interface
- Lesson 7: Communicating with Other Programs
- Lesson 8: Understanding Applet Capabilities and Restrictions

<table>
<thead>
<tr>
<th>Table 1 Questions from the J++_BApp Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1: Current browsers impose some restrictions on applets that are loaded over the network. Which of the following is not a restriction (An applet can:).</td>
</tr>
<tr>
<td>A. Load libraries or define native methods</td>
</tr>
<tr>
<td>B. Read or write files on the host that is executing it</td>
</tr>
<tr>
<td>C. Make network connections to hosts other than the one it originated from</td>
</tr>
<tr>
<td>D. Start any program on the host that is executing it</td>
</tr>
<tr>
<td>E. Read certain system properties</td>
</tr>
<tr>
<td>F. Invoke public methods of other applets on the same page</td>
</tr>
<tr>
<td>G. I do not know</td>
</tr>
<tr>
<td>Question 2: Applets can find other applets and send messages to them. However, all applets must be running:</td>
</tr>
<tr>
<td>A. On the same page</td>
</tr>
<tr>
<td>B. In the same browser window</td>
</tr>
<tr>
<td>C. On the same page and in the same browser window</td>
</tr>
<tr>
<td>D. On the same page or in the same browser window</td>
</tr>
<tr>
<td>E. No conditions are required</td>
</tr>
<tr>
<td>F. I do not know</td>
</tr>
</tbody>
</table>

Lesson 9: Completing an Applet
Lesson 10: Common Applet Problems and Their Solutions

Lessons in the knowledge network can be connected through prerequisite relationships. For instance, the Overview of Applets lesson is a prerequisite to Lessons 6–10. Besides lessons, the knowledge network can have additional levels to further organize the course hierarchy. In the J++_BApp course, we have three additional levels that are connected through the part-of relationship. These levels are described as follows.

- Chapter level: Lessons are grouped into chapters. For example, Lessons 5–10 are parts of the Writing Applets chapter. A lesson may be linked to one or more chapters. An example list of the chapters in the J++_BApp course is provided below:

Chapter B1: Getting Started
Chapter B2: Writing Java Programs
Chapter B3: Writing Applets
Chapter B4: Using the Core Java Classes
Chapter B5: Creating a User Interface
Chapter B6: Custom Networking and Security
Chapter B7: Integrating Native Methods into Java Programs
Module level: At this level, chapters are grouped into modules. We have divided the J++ _BApp course into the following four modules.

**Module 1:** Object-Oriented Programming Fundamentals Concepts. This module describes the important Object Technology concepts, such as class, instance, generalization-specialization, inheritance, and polymorphism.

**Module 2:** Java Programming Language. This module addresses the language aspects of Java.

**Module 3:** Development Environments and Tools (in our case, Visual J++): This includes the Java API, resource tools, etc.

**Module 4:** Object-Oriented Programming and Design Techniques. Once the students have a solid understanding of object technology, how to express them in the Java language and how to use the development environments and tools, this module provides case studies. A case study is a set of easy-to-follow guidelines, illustrations, and sample programs that can give the students an opportunity to engage in object-oriented programming and design.

This breakdown is based on the four difficulties identified in the first section. Many practitioners suggest the same breakdown, especially for developers who are making the transition to object-technology [3].

Course level: This is the highest level in the knowledge network. Each course node at this level can have multiple models linked to it. Examples at this level are Building Applications using Visual J++ (J++ _BApp), Developing a Graphical User Interface with Visual J++ (J++ _GUI), and Connecting a Database to Applets Using JDBC (J++ _BD). Note that these courses are not totally independent of each other. Each requires some aspects of others to be comprehensive in its coverage.

Besides these major levels, there are several other items linked to the objects in these levels. One of the major reasons for all these items is to enhance the system’s pedagogical capabilities. Two of these items are solutions and Note-Box.

Solution objects are attached to questions in the knowledge network. Each solution can have a description and a justification for the question it represents. Table 2 shows a solution to Question 1.

The GAA displays a solution for a question after a test or quiz. Our experience with GAA and other training systems shows us that the solution for a question is sometimes incomplete, ambiguous, and/or incomprehensible to the learner.

For this reason, we have attached Note-Box objects in the knowledge network to supplement the learner’s understanding of the subject matter. A Note-Box object can be attached to any object in the knowledge network. A Note-Box helps to achieve a greater understanding of the domain knowledge by dispensing further information about a lesson. These objects can point to a book, journal, or URL, and/or provide a mail address of an expert that can be contacted for help. Table 3 shows a Note-Box attached to the Understanding applets capabilities and restrictions lesson.

**Developing a Course Structure: Difficulties, Issues, and a Solution**

Our past experience in designing training systems for different educational settings led us to address three

---

**Table 2** Example of a Solution

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The correct answer to this question is F. An applet can find another applet either by looking it up by name (using the AppletContext and GetApplet( ) methods) or by finding all the applets on the page (using the AppletContext and the GetApplets( ) methods). Both methods, if successful, give the caller one or more Applet objects. Once the caller finds an Applet object, the caller can invoke methods on the object.</td>
</tr>
<tr>
<td>Comments: Be careful with applets restrictions and capabilities</td>
</tr>
<tr>
<td>The java.applet package provides an API that gives applets some capabilities that applications do not have. For example, applets can play sounds, which other programs cannot yet do. However, every browser implements security policies to keep applets from doing damage. The implementation of the security policies differs from browser to browser. Also, security policies are subject to change. For example, if a browser is developed for use only in trusted environments, then its security policies will likely be much more lax than those described here.</td>
</tr>
</tbody>
</table>
main software engineering issues while designing ITS Knowledge structure. For the GAA, they are as follows.

First, the course structure for J11_BApp (see above) has clearly demonstrated the potential reuse of certain parts of a course in others courses. For example, a course on designing graphical user interfaces using Visual J11 needs to have a module on object-oriented principles. This means that a course designer should be able to reuse items of a knowledge network without any modification or by specializing them for specific needs when constructing new courses. This can increase designer productivity and reduce the overall cost of developing a new course.

Second, a knowledge network is likely to be reusable and flexible when its construction does not rely on the details of a specific context in which it is going to be used. When the architecture has laid out this separation correctly, the interface can be customized to each user’s characteristics and goals. This means that the semantics and use of each knowledge network item should be adjustable according to the course context at hand.

Third, each learner has a different background and knowledge level about the course. Their needs and requirements for feedback/guidance vary. To meet these diverse requirements, the knowledge network should contain a set of generic items that the course designer could use to provide multiple learning resources.

These reasons led us to design an organization of the knowledge network that is flexible and generic enough to be useful in designing courses for different contexts with relative ease. We have developed an object-oriented language that takes full advantage of this flexibility in the architecture of the knowledge network and makes the course development process much simpler and more powerful. This language supports the three characteristics listed below:

- Reusability of network parts when developing other networks
- Modifying the number of network levels and their meaning to meet demands for different kinds of training contexts and developer needs
- Enabling the use of a generic item in the knowledge network to contain varying learning resources for different types of users. The Note-Box object is an example of this type of generic item that enables the course designer to plug in multiple learning resources.

A first version of this language was developed using the Lex, Yacc, and M4 tools on UNIX. The J++_BApp course is implemented using this language. Annex A and B show some parts of the J++_BApp implementation using this language.

### A SCENARIO INVOLVING GAA

The GAA interacts with the learner for multiple sessions. Each login with the GAA invokes a new session. At the first login, the system recommends that the learner take a pretest. An initial user profile is thereby created. If the learner ignores this recommendation, a default profile is used to initialize the system. Each test or quiz taken by the learner causes a change in the user profile that drives the training and assessment process. Learners can also start their interaction with the system at the point where they left off (the last session).

The GAA provides two principal modes of interaction: free learning and Test/Diagnosis. No restriction is placed on interacting between these modes of interaction. Students are left in charge of their training and assessment processes.

The free-learning mode makes available the course contents through a hierarchical browser that presents the course structure into an easy-to-internalize format. In addition to presenting the course contents, this mode also provides the user with a list of learning resources. This list includes for a course item (e.g., How to write an applet) links to the relevant URLs available through the Internet or Intranet, a list of selected examples and tutorials, etc. The kind and number of learning resources are limited only by the imagination of the course designer who, depending upon the objectives of course item, can make available a set of complementary learning resources that can foster internalizing the knowledge for the student.

The test/diagnosis mode provides learners with opportunities to obtain an evaluation of their knowledge of the course. This diagnosis is presented in a variety of ways so that an overall and a detailed view

---

**Table 3 Example of a Note-Box Object**

<table>
<thead>
<tr>
<th>Note-Box: To learn more about applets restriction and restrictions, consult:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- The Java tutorial: Object-oriented programming for the Internet, M. Campione and K. Walrath, Addison-Wesley, New York, 1996</td>
</tr>
<tr>
<td>E-mail to <a href="mailto:news_java@crim.ca">news_java@crim.ca</a></td>
</tr>
<tr>
<td>FAQ on Java restrictions</td>
</tr>
<tr>
<td><a href="http://www.coopeere.com/discussion">http://www.coopeere.com/discussion</a></td>
</tr>
</tbody>
</table>

---

**Mastering Object-Oriented Technology 167**
of the progress learning the course material are made clear to the learner. The test phase is what uses the strength of the assessment tool to provide knowledge assessment effectively and efficiently to the learner. The results of the assessment are available via a histogram (Fig. 2). When the student double-clicks on his or her particular column, a pop-up menu appears on the screen to display a list of a value added resources that will reinforce the students’ apprenticeship.

CONCLUSIONS

In this article, we presented a self-assessment and self-learning approach to deal with the difficulties of providing effective OO training and continuing education. Our system has the capability to assess the learner’s knowledge state and then suggest resources which can promote cognitive learning by OO developers. We note the following advantages and benefits of our approach. First, the system is able to identify the real difficulties of software developers and highlight their strong skills. This can be very helpful for a project manager when identifying the best developer for a certain task. Second, by using this system, developers can get their learning profile periodically. This self-monitoring places learners in charge of their learning rate. This empowerment can cause the learning curve to become less steep, frustrating, and demoralizing.

Compared to paper manuals and traditional training in the classroom, the system holds the opportunity to implement an open and a distributed approach for software engineering distance continuing education. The system offers the following advantages:

- Searching and retrieval functions of on-line learning resources and information
- Integration of multimedia material (image, video, sound)
- Using material available on the World Wide Web, such as online bibliographies, technical reports, and remote courseware
- Integration with the company-wide information system (locations, dates, staff, lecture information)

However, the system’s potential is highly dependent on the quality of the course content. We have suggested a solution to some fundamental issues that are essential to designing an effective and useful course. This solution includes the potential reuse of a course or part of a course and makes the content highly relevant in various contexts.

APPENDIX A

The overall Java course includes four modules. The course organization is inspired from the Java tutorial [16]

Module A: Object-Oriented Programming Fundamentals Concepts

Chapter A1: Class, Methods and Attributes
Chapter A2: Instance and Object
Chapter A3: Inheritance
Chapter A4: Functions and Operators Overloading
Chapter A5: Generic Classes
Chapter A5: Exceptions and Assertions

Module B: Java Programming Language

Chapter B1: Getting Started
Lesson 1: The “Hello World” Application
Lesson 2: The “Hello World” Applet
Chapter B2: Writing Java Programs
Lesson 3: The Nuts and Bolts of the Java Language
Lesson 4: Objects, Classes, and Interfaces
Chapter B3: Writing Applets
Lesson 5: Overview of Applets
Lesson 6: Creating an Applet User Interface
Lesson 7: Communicating with Other Programs
Lesson 8: Understanding Applet Capabilities and Restrictions
Lesson 9: Completing an Applet
Lesson 10: Common Applet Problems and Their Solutions
Chapter B4: Using the Core Java Classes
Chapter B5: Creating a User Interface
Chapter B6: Custom Networking and Security
Chapter B7: Integrating Native Methods into Java Programs

Module C: Development Environment and Tools
Module D: Object-Oriented Programming and Design Techniques

APPENDIX B

The formal description of the course is as follows:
/* Course J+++_BApp declaration
   Module B
     Chapter B2
       Lesson {5, 6, 7, 8, 9}
     End_Chapter
     Chapter {B1, B2, B4, B5, B6, B7}
   End_Module
   Module {A, B, C}
End_Course

Lesson 8
   Resource R_8
   Question Q_110
   Solution SQ_110
End_Question
End_Lesson

/* Attributes definition of J+++_BApp objects
   Question Q_110
   Definition: question_110.html
   Name: Question 110
   Choice: 6
   Correct_choice: 1
End_Question

Solution SQ_110
   Definition: solution_110.html

REFERENCES

[16] M. Campione and K. Walrath, The Java tutorial: Ob-
ject-oriented programming for the Internet, Java Series, Addison-Wesley, New York, 1996.

ADDITIONAL BIBLIOGRAPHY


M. Glinz, The teacher: Concepts! The student: Tools!—On the number and importance of concepts, methods, and tools to be taught in software engineering education, Third International Workshop on Software Engineering Education, 18th International Conference on Software Engineering, Berlin, Germany.

H. Hermann and I. Metz, Teaching OO software engineering by example, ECOOP’96 Educator’s Symposium (ECOOP-ES’96), 1996.


S. Lilly, Pedagogical design patterns, Object (January 1996).

M. L. Manns, Pedagogical patterns: Successes in teaching object technology, ECOOP’96 Educator’s Symposium, 1996.


A. Seffah and R. Bouchard, A discussion of pedagogical strategies and resources for software engineering online training, ED-MEDIA 98—World Conference on Educational Multimedia and Hypermedia, Germany, 1998.

BIOGRAPHIES

Ahmed Seffah is a researcher at the Computer Research Institute of Montreal and adjunct professor of computer science at the University of Quebec at Montreal. He earned his PhD in software engineering from the École Centrale de Lyon in France. His areas of interest range from software engineering and human-computer interaction to human factors in software development, software process especially requirements and usability engineering processes and training methods and tools in software development organizations. He is member of the IEEE computer and education societies, ACM, the International Association for Continuing Engineering Education (IACEE), and the UNESCO International Centre for Engineering Education.

Michel Desmarais is currently the team leader of the Human–Computer Interaction Group at the Computer Research Institute of Montreal. He earned his doctorate degree in psychology from the University of Montreal. He has directed a number of R&D projects, from an expert system in meteorology to a computer coach for text editing. His previous work experience includes Xerox PARC, Stanford University, and HP Labs. His areas of interest range from cognitive science and artificial intelligence to human factors, user interfaces, and performance support systems.

Ramzan Khuwaja is an advisor of pervasive and nomadic computing in the Computing Technology Lab (CTL), Nortel Networks. He has been working in these advanced technology areas since 1996 and is involved with several nomadic and mobile computing projects. Dr. Khuwaja was one of the original architects of a CTL project, Distributed Network Services Platform (DNSP), which is currently a flagship product of a Nortel joint venture, Saraide.com. Before joining Nortel Networks, Dr. Khuwaja was a researcher at the Computer Research Institute of Montreal and worked in creating and making available a smart computer-based teaching system using Web technologies. Dr. Khuwaja has a PhD from Chicago in the field of knowledge engineering.