Finding Reasons and Conclusion in a Basic Computer Science Course

Kanaan A. Faisal Ahmed H. Bagais M.R.K. Krishna Rao
Information and Computer Science Department
College of Computer Sciences and Engineering
King Fahd University for Petroleum and Minerals
Dhahran 31261, Kingdom of Saudi Arabia
{faisal,bagais, krishna}@ccse.kfupm.edu.sa

Abstract

*This paper discusses infusing the critical thinking skill of **finding reasons and conclusions** in a basic computer science course, ICS 201: Introduction to CS, which is the second course in the series of three introductory courses in our curriculum [8]. Software engineers usually use certain design in a program or algorithm without fully considering the effects of that decision. The critical thinking skill of finding reasons and conclusions enable them to reason more about their choice of design. Also, it helps them to convince the proponent to follow the decision they made with conviction.*

1. Introduction

Learning is an important skill that any student should acquire. The way knowledge and information are transferred to students in a traditional fashion is considered inadequate. This is because learning is a skill that is not used only in academic life; it is needed even afterwards when students face new developments in the technology and business, especially in information technology area. This can be achieved by improving the thinking skills that will enable the students to analyze and understand any new concept and enhance their communication with the outside world. In this paper we focus on one of the major thinking skills that can be applied in various contexts. This skill is called *finding reasons and conclusion* and it is used to analyze people's arguments and see whether they are acceptable or convincing.

Computer scientists use various design techniques in developing new computerized systems. In software life cycle, they should not do or accept a certain design or system environment unless there are accurate and sufficient reasons to do so. The skill of finding reasons and conclusions is important for software engineers in this process. This thinking skill helps them to either accept or reject a given argument after considering its supporting justifications and investigating for any evidence against this argument. For example, developing a new program that just meets the user requirements is not enough. Other factors like performance or maintenance should also be considered before delivering the final output in order to avoid undesirable results. Delivering a course without the emphasis in the thinking skill to adopt a certain structure or design does not improve students' creativity and thinking. Teaching this skill of finding reasons and conclusions explicitly in computer science courses is important to enhance the students' thinking skill (how to evaluate an argument?) and enable them to make more reasonable judgment when they develop a program. This is because learning a new concept involves two steps: (1) transmitting the subject or the knowledge itself and (2) the correct way to analyze this subject and find why it behaves in a certain manner. The second step is called critical thinking and when combined with a subject (or domain) knowledge results in better insight about the subject. It will

also be useful in the future when software developers need to convince the proponent about the feasibility of the product they develop.

This paper discusses finding reasons and conclusion skill and how to infuse it into an introductory computer science course.

2. Why is Finding reasons and Conclusion important?

Usually people try to convince us to do or accept something and they provide some reasons for accepting their argument. This thinking skill prevents premature judgments people make without looking for sufficient reasons in support of the argument and evaluating the accuracy of these reasons. There are cases where we accept an argument without considering the reasons first. The appeal of visual images and fiery speeches makes us emotional and we do not ask for the adequate reasons in support of the argument. Also, we only consider the stated reasons but not other ones. Maybe there are unstated reasons that count against the argument and searching for them will lead to rejecting the argument instead of accepting it. The second issue is that we do not take time to evaluate the given reasons if they are adequate to accept the conclusion. For example, a salesman is trying to convince you to buy a car and he is listing its features and showing pictures of how comfortable the car is. Buying the car from the salesman without checking this information with a reliable source (e.g. Consumer Reports) may cost you more than its value. Consumer reports may have information about the repair record of that car and indicate that it needs frequent repairs in its life time. This search may lead to a conclusion that this car is not good to buy and therefore you reject the salesman's offer. These difficulties in finding reasons may be summarized as follows [12]:

1. We may mistake emotions or visual images for reasons.
2. We may not search for unstated reasons.
3. We may only consider the reasons in favor of the conclusion but not search for reasons against.

3. Finding Reasons and Conclusion Skillfully

Arguments should be dealt in a systematic way to avoid the difficulties mentioned before. Consider that a salesman is trying to sell you his car by stating its feature. He says that you should buy the car because it costs less than any comparable car and shows you some commercials that suggest how large and comfortable the car is. Also, he says that it gets good gas mileage and has a better repair record than other cars. Accepting or rejecting his offer (argument) should involve two main steps:

1. **Breaking out the argument**: from the information people present, the conclusion and its reasons need to be extracted. For stated reasons, keywords like because or therefore help in identifying the conclusion and its supporting reasons. Sometimes people rely on common knowledge as reasons to accept their argument. For the salesman example, he tells you that the car is large and comfortable because he thinks that this feature is important for big families and this can be a reason to buy his car. These questions (thinking map) can be used to break out an argument:
    a. What is the salesman trying to convince you to accept or do?
    b. What reasons does he provide to support accepting or doing that?
        i. Are there any words that indicate support (e.g. "therefore", "so", "because")?
        ii. Does he provide any other indication as to why he concludes what he does?
    c. Is there anything that you think he believes is common knowledge that he does not state but uses to support the conclusion?

    The answers to these questions can be organized in a graph similar to one in figure 1.

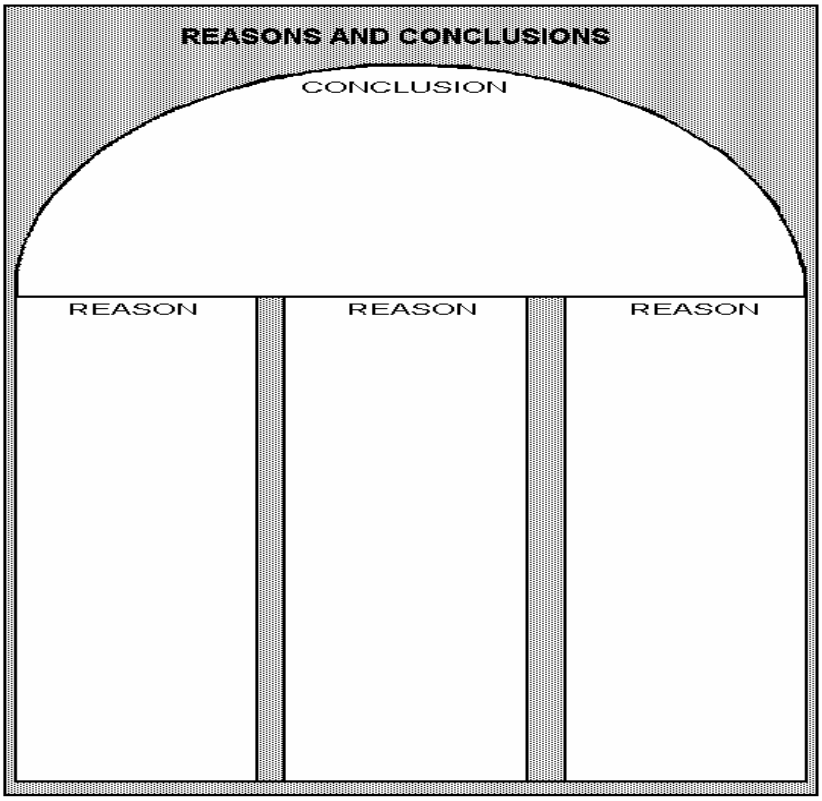

Figure 1: Graphic organizer for the salesman argument

This graphic organizer depicts the argument and illustrates the idea that conclusions should be supported by reasons.

2. **Assessing or evaluating the given reasons**: after identifying the conclusion and reasons in the graphic organizer, evaluating them should be done to find out if they are accurate and sufficient to accept the argument. This involves answering two questions:
    a. Is there anything you need to find out in order to determine whether the reasons are accurate?
        i. If so, what information do you need to find out?
    b. Given that the reasons are accurate, is additional information needed before you can accept the conclusion?
        i. If so, what information do you need?

    Considering opposing viewpoints can help in this process. Open-mindedness is one of the important dispositions of critical thinking. Information that is against the argument may reveal reasons to reject the conclusion and this may save you from undesired situations. Therefore, if the answers to these questions are negative, then the argument is convincing; otherwise it should not be accepted.

4. Infusing the Skill

The skill of finding reasons and conclusions can be used in many contexts in computer science discipline. The context in which we infuse this skill is whether teaching Java programming language is better than other languages in the introductory courses. Convincing the students about this argument has two benefits. First, it

gives them a good example of how this skill can be applied in computer science. Second, it will reveal the advantages of Java so that they can make full use of it whenever possible.
The argument is started by stating the reasons of teaching Java in basic courses. Java is an Object-Oriented language, a concept that can be used to represent real world problems and deployed in Databases and Software Engineering design process. It is safe in the sense that it catches beginners' mistakes and report them (through exception handler) and secure in the sense that applets (little programs that run inside a web browser) cannot touch the local disk, otherwise there would be a potential virus threats. Java also supports network programming through servlets (server-side program) and applets (client-side program). Another important advantage of Java over other languages is that it is a portable language i.e. the same Java program can run on different machines without any change. The only requirement for this is that the computer should have a Java virtual machine that handles the execution of java code.

Following the thinking map of the skill to collect the reasons of teaching Java, we answer the following questions:

a. What is the proponent trying to convince you to accept or do?
   Teaching Java in basic courses.
b. What reasons does (s)he provide to support accepting or doing that?
   1. It is an Object-Oriented programming language.
   2. It is safe and secure.
   3. It support network programming.
   4. Its compatibility because of the JVM.
c. Is there anything that you think he believes is common knowledge that he does not state but uses to support the conclusion?
   OO concept is useful in computer science discipline.

Organizing these ideas in the following graph gives more clarity to the argument.

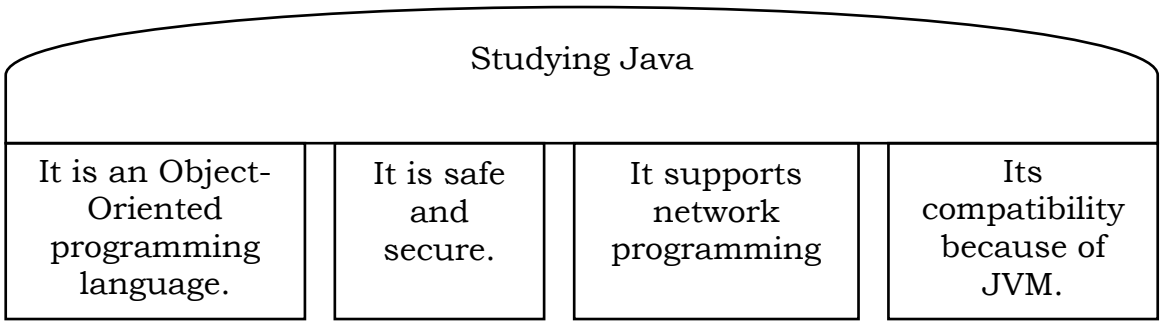

Figure 2: Collecting reasons in the graphic organizer

The second step before accepting the argument is to evaluate these reasons using the check list of questions:

a. Is there anything you need to find out in order to determine whether the reasons are accurate?
   No, because these are widely accepted features that Java is designed on.
b. Given that the reasons are accurate, is additional information needed before you can accept the conclusion?
   No. They are sufficient because these are the main advantages in Java and the only consideration is efficiency. This can be an issue in some cases, but you can compensate by using a certain type of compilers. Also, other languages' features unavailable in Java can be used through Java Native Interface (JNI).

Since the answers to the two questions are negative, the argument for teaching Java is convincing.

This skill can also be used to state why Polymorphism is implemented in Java. Polymorphism enables the computer to execute the actual method of an object at its method call in run time. This concept has major advantages. Polymorphism allows us to write generic code in which it has different types of objects and get different result based on the object and using the same method call. Generic code is also more readable and easy to understand which makes the code easy to debug. Polymorphism also allows the functionality of the code to be extended for any added data type which is a useful feature enabling modification of the code when it needs to support larger application.

Following the thinking map in the same way before, we answer the questions:

a. What is the author trying to convince you to accept or do?
   Polymorphism is useful in programming.
b. What reasons does he provide to support accepting or doing that?
   1. It makes the code generic.
   2. It allows extensibility to the code.
   3. It makes the code more readable.
c. Is there anything that you think he believes is common knowledge that he does not state but uses to support the conclusion?
   Readability of the code makes debugging and maintenance easier.

Organizing these points in this graph gives a summary of the argument:

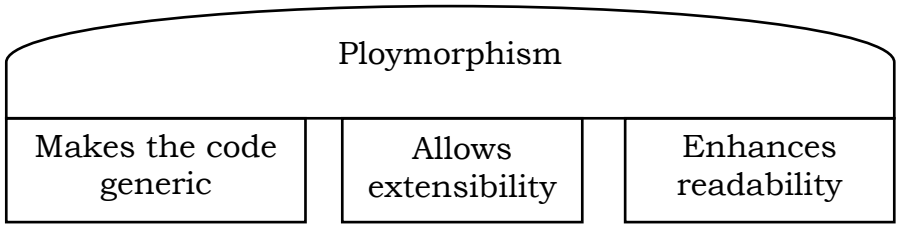

Figure 3: Polymorphism argument

Evaluating these reasons for using Polymorphism is done by answering the questions:

- Is there anything you need to find out in order to determine whether the reasons are accurate?
  - No, the reasons given are accurate as suggested by the popularity and extensive use of polymorphism in the industry.
- Given that the reasons are accurate, is additional information needed before you can accept the conclusion?
  - No. The only concern that may count against polymorphism is the need for down-casting when using an object reference. But it becomes natural after practice.

Because the answers to these questions are no, the use of Polymorphism in programming is viable.

In this course, the use of Java Collections Framework can also be convinced by this thinking skill. This framework defines a set of important utility classes and interfaces in the java.util package for working with collections. It consists of interfaces for the abstract data types that the framework supports, implementations of these data types and predefined actions or methods for their implementations. There are many legitimate reasons for developing JCF. It provides efficient implementations for important data structures like linked lists and trees and using JCF enables uniform manipulation for these structures. It also improves program speed and quality by providing high-performance, high-quality implementations for the most common data

structures. JCF allows interoperability among unrelated APIs. If a network API provides a Collection of node names, and a GUI toolkit expects a Collection of column headings, they will interoperate seamlessly even though they were written independently. Because of its abstraction, JCF fosters software reuse in a number of applications which is an important feature in software engineering.

Applying the skill of finding reasons and conclusion to see if this argument is valid, we follow the thinking map:

a. What is the author trying to convince you to accept or do?
    i. Developing JCF has major advantages to programming.
b. What reasons does he provide to support accepting or doing that?
    1. It provides efficient implementations for important data structures like linked lists and trees and enables uniform manipulation for these structures.
    2. It also improves program speed and quality.
    3. It allows interoperability among unrelated APIs.
    4. It fosters software reuse in a number of applications.
c. Is there anything that you think he believes is common knowledge that he does not state but uses to support the conclusion?
    Reusability is an important feature of a program.

This argument can be summarized in the following graph:

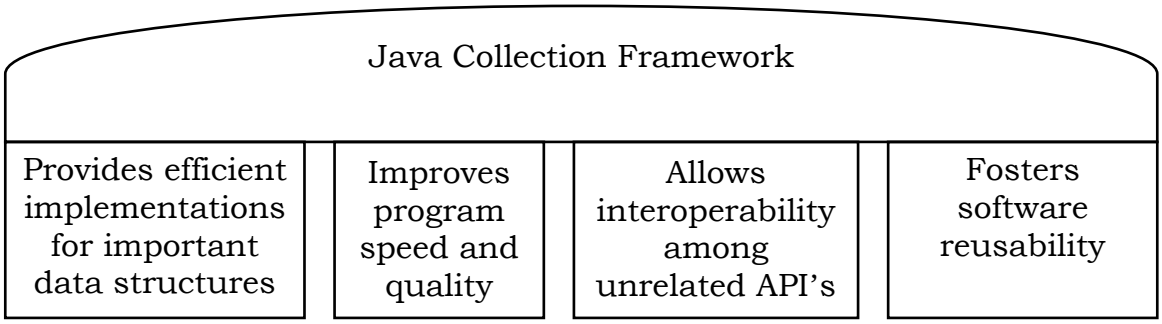

Figure 4: JCF argument

For argument evaluation, these questions are asked:

- Is there anything you need to find out in order to determine whether the reasons are accurate?
    - No, the reasons given are accurate suggested by the use of this framework.
- Given that the reasons are accurate, is additional information needed before you can accept the conclusion?
    - No. JCF has some concerns that can be worked around by careful programming. JCF work only with objects not primitive data types and it can contain incompatible types at the same time. Therefore, down casting must always be used with object references.

These answers indicate that developing JCF has major advantages in programming that are needed in software.

5. Conclusion

In this paper, the skill of finding reasons and conclusion of an argument is introduced. It is applied on the decision to teach Java in introductory programming courses to see that the supporting argument is valid. Also

applied on why Polymorphism and Java Collection Framework are useful in programming languages. The benefits from the infusion of critical thinking skills into course content include the following:

1. Improved thinking skills in the students,
2. Improved searching skill for information: this thinking skill gives the students opportunity to search for any related information regarding the argument before accepting it,
3. Improved communication skills: because of the writing they have to do for critical think assignments, students communication skills improve, and
4. Lively classroom atmosphere: in view of the active learning techniques used in the course, student participation is naturally higher.

Acknowledgements

The authors would like to thank the Deanship of Academic Development at King Fahd University of Petroleum and Minerals for supporting this research work.

References

[1] ACM//IEEE. Computing Curricula 2001. Electronic version available at http://www.acm.org/sigcse/cc2001/.

[2] Bransford, J., Brown, A.L., and Cocking, R.R. ***How People Learn: Brain, Mind, Experience***, *and School*. NAP, 2000.

[3] Broadbear, J.T. *Essential elements of lessons designed to promote critical thinking*, The Journal of Scholarship of Teaching and Learning, 3, 3 (2003), 1-8.

[4] Browne, M. N., & Freeman, K. *Distinguishing features of critical thinking classrooms*. Teaching in Higher Education, 5, 3 (2000), 301-309.

[5] De Bono, E. ***De Bono's Thinking Course***. Ariel Books 1985.

[6] De Bono, E. ***Six thinking hats for schools***. Hawker Brownlow, 1992.

[7] M.R.K. Krishna Rao (2005), *Infusing critical thinking skills into content of AI course*, Proc. of the 10th annual SIGCSE conference on Innovation and technology in computer science education, ITICSE'2005, pp. 173-177.

[8] M.R.K. Krishna Rao, S. Junaidu, T. Maghrabi, M. Shafique, M. Ahmad and K. Faisal (2005), *Principles of curriculum design and revision: a case study in implementing computing curricula CC2001*, Proc. of the 10th annual SIGCSE conference on Innovation and technology in computer science education, ITICSE'2005, pp. 256-260.

[9] Norris, S. P., and Ennis, R. H. *Evaluating critical thinking.* Critical Thinking Press and Software, 1989.

[10] Paul, R. & Elder, L. ***Critical Thinking: Tools for Taking Charge of Your Professional and Personal Life***. Prentice Hall, 2002.

[11] Popper, K.R. ***The Logic of Scientific Discovery***. 1934. Recent edition (15^{th}) , Routledge publishers, 2002.

[12] Swartz, R. *Infusing the Teaching of Critical and Creative Thinking into Content Instruction*, in ***Developing Minds***, Association of Supervision and Curriculum Development, Alexandria, Virginia, 2001.

[13] Van Tassel-Baska, J. ***Comprehensive curriculum for gifted learners***. Allyn & Bacon., 1994.