Pet Formalisms versus Industry-Proven Survivors: Issues on Formal Methods Education

S.D. Cha
Department of Computer Science and Advanced Information Technology Research Centre (AITrc)
Korea Advanced Institute of Science and Technology (KAIST)
373–1 Kusong-dong, Yusong-gu
Taejon, Korea
Phone: 82-42-869-3535
Fax: 82-42-869-3510
Email: cha@cs.kaist.ac.kr

Formal methods are gaining steady industrial acceptance as a promising approach to developing high-quality software. Although educators find it increasingly important to include formal methods in their software engineering curriculum, they face a dilemma: there are too large numbers and diverse flavors of formal methods to cover in a limited time, and little empirical data exists on relative strengths and weaknesses among them. How can we best develop a curriculum on formal methods given several practical constraints? In this paper, we describe our approaches, share experiences, and evaluate results. Our goal is to stimulate further discussion among software engineering education communities on how formal methods can be most effectively introduced to current and next generation software engineers.

Keywords: software engineering education, formal methods

1. INTRODUCTION

As the use of safety-critical systems in industrialised nations continues to grow, so does our society’s dependency on high assurance software systems. There are some safety-critical applications where the most serious software mishaps must be avoided at all practical cost (Leveson, 1995). Examples include the fly-by-wire commercial jets, nuclear power plant shutdown systems, patient monitoring systems, and radiation therapy machines.

Formal methods are generally defined as techniques based on mathematical principles for the specification, development and verification of software and hardware systems. Although once regarded as a subject of research interest only to academics and considered impractical for industrial applications, formal methods are surely gaining slow but steady acceptance as a promising approach to overcome the quality ceiling problem commonly associated with traditional software development methods (NASA, 1995).

Formal methods are not, and should not be regarded as, “silver bullets” (Dean and Hinchey, 1996), and several issues, including their cost-effectiveness, are still being debated. However, widely accepted consensus among experts is that proper application of formal methods can increase