NSF CENTER FOR RESEARCH ON COMPLEX NETWORKS (CRCN)

NSC 128, 713-313-1290, TEXAS SOUTHERN UNIVERSITY ·3100 CLEBURNE AVENUE ·HOUSTON TX 77004

CENTER RESEARCH SEMINAR

When Good Algorithms Yield Bad Software

Ernst L. Leiss
Professor, Department of Computer Science
University of Houston
Houston TX 77204-3010

Thursday, September 19, 2013
3:00 p.m. – 4:30 p.m.
Room 150 at Science Building

Biography

Ernst L. Leiss earned graduate degrees in computer science (1974) from U. Waterloo, Canada, and mathematics (1975, 1976) from TU Vienna, Austria. He joined the University of Houston in 1979; from 1985 until 1994, he headed its Research Computation Laboratory. He wrote over 160 peer-reviewed papers, nine book chapters, and six books: *Principles of Data Security* (1982, Plenum), *Software Under Siege* (1990, Elsevier), *Parallel and Vector Computing* (1995, McGraw-Hill), *Language Equations* (1999, Springer), and *A Programmer's Companion to Algorithm Analysis* (2006, Chapman & Hall,), and with Jose Aguilar *Introducción a la Computación Paralela* (2004, U. Mérida, Venezuela). He has been the research supervisor for 16 Ph. D. dissertations and over 100 M. S. theses. He is an ACM Distinguished Speaker and has lectured in 32 different countries to date.

Abstract

This presentation focuses on the transition from an algorithm to a program. Algorithm analysis is a well-studied discipline, as is software development. However, at the interface between these two disciplines much can and does go wrong. In fact, many programmers have experienced situations where a good algorithm (that is, correct and efficient) resulted in either wrong or unacceptably slow software. Our aim is to describe what can go wrong, why it goes wrong, and how to avoid these problems.

We first outline the fundamental assumptions of algorithmic complexity. Then we describe scenarios where correct algorithms result in incorrect software. This is followed by a comprehensive examination of performance discrepancies, situations where the complexity analysis of the algorithm provides one answer and the observed behavior of a faithful implementation of the algorithm provides a dramatically different (worse!) answer. Both significantly slower behavior and unpredictable behavior are of concern. The causes of the differences between the behaviors of algorithms and software can be categorized into several areas, namely the implications of the non-uniform memory in real architectures (both caches and virtual memory management are implicated), system issues (memory mappings, passing of parameters, garbage collection, and optimization techniques are important here), implicit assumptions (including exception handling), and the finiteness of the number representation (which does not only have relevance for numerical applications, but is also important if one tests for equality or assumes mathematical identities hold). Our emphasis is on practically useful knowledge that helps software designers to produce correct and efficient code. We assume some familiarity with the analysis of algorithms as well as a working knowledge of software design.