

## **Resolution on the Retirement of David Krumme**

Adopted by the Faculty of Arts, Sciences and Engineering

The Computer Science department enthusiastically recommends that our first retiree, Professor David Krumme, be granted Emeritus status in the Department of Computer Science.

### **Introduction**

Professor David Krumme came to Tufts in 1977 as a member of the Mathematics Department, at a time when computer science was not even a subject of study at many schools. While his Ph.D. was in Applied Mathematics, as a Master's student he contributed software to the Berkeley UNIX project, and came to Tufts with remarkable software development and systems skills. Professor Krumme remains a “hacker” in the most positive sense of the word: a master programmer who makes complex systems function properly by a combination of perseverance, high standards for himself and his students, dedication to scientific truth, exacting scientific method, mathematical precision and proof, and a deep knowledge of the practical semantics and inner workings of each subsystem. His presence at Tufts was a remarkable and enduring gift to his students, many of whom have enjoyed the lucky chance to work by his side as an apprentice to a true master of the art of computer programming.

### **Research**

Professor Krumme's research has always straddled the line between theory and practice.

Professor Krumme's current theoretical work centers upon issues in parallel and distributed computing, including related graph theory and finite combinatorics. Early work considers the embedding problem of fitting large and irregular computational problems – such as calculating the orbits of planets and asteroids – into the regular structure of supercomputers. One early paper demonstrated that the problem of embedding a graph into a hypercube to minimize communication latencies is NP-complete. Later work considers the “gossip problem” of exchanging data between computing nodes in a large supercomputer. Professor Krumme has written several papers on optimal gossiping algorithms for various kinds of networks. Of late, he has turned his attention to the problem of creating optimal spanning and multicast trees within an ever-expanding and increasingly dynamic Internet.

But this is only one-half of the story. It is a rare Computer Science Professor who is able to contribute both to abstract theory and to real practice. Professor Krumme has written literally hundreds of thousands of lines of code in C, C++, and countless other languages. Projects of Professor Krumme and his students include two real-time operating systems, the “Sargasso C” compiler for DEC10 and DEC20 computers, the “Starfish” communications server and terminal switch, and operating system instrumentation for debugging of parallel programs. As one of the founding architects of the Computer Science Network for Teaching and Research, Professor Krumme wrote numerous tools to enhance day-to-day operations, including the “Upgrade” system

for automated grading of computer science projects, a user account management system, a tape backup and recovery system, and so on.

## Teaching

Professor Krumme believes that students should learn to write programs to conform to a rigid set of external specifications. To verify that a student's program works correctly, he tests it on several randomly generated problems that the student does not know about in advance. If the program crashes in solving a problem, it receives no credit. Appearance of code does not matter. Effort does not matter. In the end, the acid test of a student's program is whether it works or not in a realistic environment.

A student once asked Professor Krumme if he could perhaps be lenient in grading an assignment and give some points for a solution that crashes badly when tested on some inputs. Professor Krumme's response was that he hoped the student would not take up a career in writing *air-traffic control software*, because he did not want to fly on a plane that this student's software controls!

When Professor Krumme came to Tufts, the idea that students' programs must perform according to a rigid set of specifications was a revolutionary concept in Computer Science education. Professors at other universities were satisfied to grade programs based upon appearance or upon each program's behavior on simple test cases that the student is given in advance. Professor Krumme instead tested programs on realistic, randomly generated problems of varying sizes. Because students were up against the "luck of the draw," their programs really had to work or they would receive no credit at all.

His commitment to real-time performance grading led to one of Professor Krumme's most significant and least known contributions. The "Upgrade" system for automated grading of computer science assignments was the first of its kind: a system for managing the complete lifecycle of electronic assignment submissions. Upgrade included provisions for running automated grading programs, manually grading program appearance, detecting cheating, and reporting on student performance both to students and to the professor. Upgrade transformed the way we teach Computer Science at Tufts, and the teaching software that we utilize now – though tuned and perfected over the years by several other faculty – is still based upon concepts and techniques that Professor Krumme originally developed.

Professor Krumme's courses have always been structured around interesting, exciting, and confounding customized assignments of his own design. Always interactive, sometimes nearly impossible, often competitive between other students and even competing head-to-head with the professor, students are pushed by Professor Krumme's assignments to rise above their limits and strive for excellence. Professor Krumme is a firm believer in learning by doing. In his class on compilers, one writes a compiler. More surprising, in his class on operating systems, one writes part of an operating system.

Professor Krumme defined and refined the still successful format of Comp15, "Data Structures," the first course in the Computer Science major. Professor Krumme developed the original framework, goals, and assignments for the course. Comp15 students learn about data structures by

writing programs that interact with simulators of real-world environments. Though each environment is randomly generated, it has a serial number that can be used to reproduce it precisely for the purpose of debugging a program. To receive credit, each student's program must be able to solve thousands of potential real-world problems.

Professor Krumme wrote the original versions of the environment simulators that have confounded and excited generations of students, both in his classes and those of Professor Couch that followed. The best of these are fond memories of alumni: the “elevator problem,” in which students had to optimize delivery of goods through an automated elevator system; and “mankala,” in which students wrote programs that play the game of mankala against each other, with a tournament to decide final grades.

These simulators were the “hard way” to teach Data Structures. While instructors at other schools required students to write relatively simple stand-alone programs, Professor Krumme's assignments required the *professor* to both craft an incredibly complex simulator and then compete alongside students for the best possible solution. Lectures contain hints, insights, and other material relevant to the assignment at hand, providing an extremely cohesive and immersive hands-on experience in real-life programming.

In Comp40, “Computer Architecture,” Professor Krumme continued the tradition he had started in Comp15. As in Comp15, assignments revolve around complex simulators that Professor Krumme crafts himself. The most infamous of these include “core wars,” in which students craft assembly language programs that run concurrently and try to erase each other from memory. In Professor Krumme's “game of life,” students compete to optimize the assembly language version of a C program that implements Conway's game of life. In both cases, complex concepts such as runtime, memory maps, and frames are made real to students through practical and realistic programming experience.

Professor Krumme has long been the department specialist on efficiently scaling courses to large audiences and proactively thwarting attempts at academic dishonesty. A pioneer user of computer-readable forms for exams, he made several notable innovations in their use. Disappointed in multiple-choice exams in which each question has only four possible answers, he developed a multiple-choice exam format in which each question has 20 possible answers. To discourage cheating, he developed a computer program to generate and grade a unique examination for each student. The result is a very challenging multiple-choice exam on which it is virtually impossible to cheat.

In Comp6, “Computing on the Internet,” Professor Krumme brought these experiences to bear upon running a very large course on general computer literacy. At its peak, serving over 300 students a term, as it was estimated that 2/3 of all students at Tufts took the course as a Math distribution requirement. To disseminate the material from this groundbreaking course, Professor Krumme published a textbook by the same name.

Professor Krumme's current special topics courses are perhaps the capstone of an illustrious teaching career. In Comp150C++, “C++ Programming Practicum,” Professor Krumme gives students a “master's-eye-view” of a programming language from the inside, showing students not

just “how to program” but also “what it all means.” In Comp150LNX, “Linux Kernel Internals,” Professor Krumme shows students how to craft and customize the operating system to their needs. The tradition of confounding capstone experiences continues. In Comp150LNX, one capstone experience was to modify the file system so that files are automatically erased after a given time period. Students were amazed not only by the fact that they could do this, but by the fact that this was possible at all.

## **Service**

Last but not least, Professor Krumme has been the embodiment of the kind of dedicated service and self-sacrifice that many might call “the light on the hill.” Chair of the newly formed Computer Science Department between 1986 and 1994, he was a key influence upon the current Computer Science major in both Arts and Sciences and in Engineering. In forging a solid major of which he could be proud, he was faced with a difficult personal sacrifice.

It stands to reason, given the nature and challenge inherent in Professor Krumme's assignments, that using central computers that were shared by other departments would lead to difficulties. Indeed, his students were responsible for many system overloads and crashes of the central servers for Tufts as a whole. Eventually, it was clear that Computer Science needed its own machines upon which to teach and do research, unencumbered by the rules and constraints of shared resources. In creating the departmental computing environment, Professor Krumme tried his best to make it the exact opposite of what is normally done in a centralized computing facility. The most up-to-date software was always available when created. Emphasis was upon experimentation and creative work rather than reliability and robustness. The early departmental network was a “place of wonder,” where students got their first exposure to the cutting edge of Computer Science. It was a uniquely proactive environment, where students who requested a piece of software were often granted the privileges to install it for themselves. This resulted in the availability of literally thousands of programs, a virtual playground that – at the time – was available nowhere else. Only the most recent Linux releases have had similar capabilities.

It was an ideal environment for learning, including some lessons learned the hard way. Professor Krumme's golden rule for students was “you break it, you fix it.” Many a student system administrator learned the hard way that a minor typo in an administrative command can result in a frantic all-night work session recovering from the mistake, with literally thousands of users unable to work until the mistake is corrected.

One unwritten rule of the network was that if the network broke without assistance from the students, it was Professor Krumme who would come to the rescue. Professor Krumme's dedication is perhaps easiest to quantify in terms of the late night work and all-nighters that he endured in the service of Computer Science and its computing needs. When systems crashed, it was he and Professor Couch who burned the midnight oil and tried to pick up the pieces before the next dawn, often taking turns in shifts. There was no staff support, and no one else to do it. If no one else would do something important, Professor Krumme would find a way to do it.

## Conclusion

In countless ways, Professor Krumme helped make the Computer Science Department the “place of wonder” that he wanted it to become. His spirit lives on in the mission statement of Computer Science, to be a place of wonder, where one looks forward to coming to work in the morning, and anticipates the daily wonder of new discovery.

Professor Krumme has often said: “Some things are best taught one-on-one, by the side of a master.” The students who have sought him out know what this means, and with his retirement this year, we will lose something irreplaceable. His sense of wonder, his mastery of the field, his commitment to lifelong learning, his commitment to the truth and to science in its purest form, will all be missed.

*Be it resolved that this resolution be spread on the minutes of the faculty of Arts, Sciences and Engineering and that a copy be handed to Professor David Krumme.*

Sincerely and with deep gratitude and friendship,  
The faculty of Tufts Computer Science.

*May 19<sup>th</sup>, 2004*

*Read by Professor Alva Couch*