

Statement of Purpose
Joseph D. Kulisics

I began my professional and academic relationship to information systems and computer science seven years ago. Five years after finishing my AB in mathematics at the University of Chicago and after completing a few introductory courses in computer science at the University of Wisconsin–Madison, I started working for Softmart, Inc., a Microsoft launch-partner for Windows 95, where I assisted callers with technical difficulties with the use of the Windows 95 operating environment. The brief training at Softmart covered general concepts in operating systems, troubleshooting, and information systems and computer science, and I regarded each topic in the broad range of material covered in training as intrinsically interesting as well as essential to my work in technical support. As a way to refine my technical skills, I continued to study alone topics in computer science directly relating to my work in technical support or holding academic interest.

As I matured as a technical professional, I changed jobs taking a position in the quality assurance group of Databeam, Inc., a programming company engaged in contract, quality assurance work. Quality assurance approaches computing from a different angle than technical support; while technical support is concerned with troubleshooting a fixed build of a program and working around problems and is dependent on knowledge of the structure of the specific program to be supported, quality assurance often views programs as opaque and benefits from an understanding of machine organization, abstract data types, implementations of abstract data types in data structures, and the intuition that these areas of knowledge provide during testing. I shifted my focus to the study of programming languages, algorithms, techniques of implementation, and the relationship of these topics to my work in testing and quality assurance, and in time, I became interested in programming.

While teaching myself system programming in C, I worked on my personal computer with several UNIX and UNIX-like environments. User interactions with UNIX environments emphasize ease of use rather than ease of learning, so to facilitate use, many of the most common and fundamental interfaces to a UNIX environment, such as the interactive shells, make available to the user, operator, and system administrator great expressive power in the form of scripting languages; moreover skillful administration of a UNIX environment seems to require scripting. Recognizing the importance of scripting to UNIX administration and the essential character of scripting as programming, I sought work as a UNIX system administrator. I took a position with JWT Specialized Communications, a recruitment advertising company, as an MIS administrator and programmer in charge of SCO OpenServer administration. As I became fluent in the details of operations, I was able to free much of my time to work on large projects requiring careful planning and design. For a system administrator I had considerable freedom to pursue scripting and programming projects, ranging from basic text processing in shell scripts to network programming with C/C++ and the System V Transport Layer Interface. I was able to propose and undertake major projects in database design and normalization and information architecture improvement by document analysis, DTD design, and SGML implementation, projects touching areas of computer science that I had never before studied. While my position at JWT Specialized Communications presented me with many technical opportunities and intellectual challenges, I understood that there was an absolute ceiling on the complexity of available work; the mission of the company was not technical and could not support an intensely technical program. As I exhausted interesting, technical opportunities within the company, I decided to find a position in an engineering, scientific, or academic computing environment, an environment where technology would be central to the mission of the organization.

I accepted a position as a Programmer/Analyst II at the Laboratory of Neuro Imaging at UCLA, where my duties included work on enterprise programming projects in an object-oriented development environment. I rose to the level of *de facto* project manager for the Brain Data Pipeline prototype project, a Java implementation of a graphical, data-flow tool to be used by researchers to connect processes filtering arbitrary, streamed data. Since the goal of the project was the construction of a software system intended for widespread use within the medical-imaging community and anticipated to be large and complex, methodical development was absolutely essential to the successful completion of the project; the project demanded attention to issues of knowledge capture, knowledge transfer, and the development process itself, so development

for the project differed from all development that I had done before. In response to these new concerns about organization, I began to study software engineering and techniques for managing and controlling complexity in the development of large systems, and I led the project in drafting a basic specification.

The laboratory invested heavily in computer hardware; to meet the demand of the research staff for CPU-time to execute compute-intensive algorithms for image registration, the laboratory bought a 32-processor configuration of an SGI Origin 2000, a shared-memory, NUMA, parallel-processing computer. I bore part of the responsibility for the management and effective use of the computing resources of the laboratory, and I eventually assumed responsibility for profiling the use of computer resources and recommending hardware to accommodate future demand. My responsibility for the management of parallel-processing resources combined with my background in programming to stimulate interests in the design of digital systems and computer hardware and the optimization of the use of parallel-processing resources, and I enrolled in summer sessions in my first classes in the Department of Computer Science at UCLA completing with high marks *Logic Design of Digital Systems* with Professor Ercegovic and *Computer Systems Architecture* with Professor Tamir.

Two years ago I took a position as a Programmer/Analyst III in the School of Engineering and Applied Science at UCLA. My duties included maintenance of programs written to help undergraduate students audit previous coursework to determine progress toward a degree and plan a future course of study in compliance with the requirements of the school and the university. As the programs that perform these functions had become difficult to maintain, I performed an analysis of the implementation of the auditing and planning programs with the goal of simplifying the auditing and planning processes and streamlining the implementation of the processes as programs; from previous study of programming languages, models of evaluation, and metalinguistic abstraction, in particular, I concluded that the description of the requirements of an academic program had simple, declarative formulations, and the auditing and planning programs, instead of capturing declarative knowledge about requirements in a declarative style, supported by SQL, pressed C, an imperative language, into service to interpret coerced, imperative formulations of requirements. By my work with programming languages, I was able to make a compelling case before my supervisor for replacement of the existing programs and offer an outline of a design directly capturing the readily available, declarative knowledge about academic program requirements.

By necessity as my professional responsibilities have changed or for personal interest, I have turned my attention to the study of various areas of computer science, and though I lack formal, undergraduate training in computer science, my academic and professional development in the field has covered much of the same ground as a typical, undergraduate curriculum. I have discovered academic interests in areas ranging from hardware design and quantitative analysis of the performance of novel architectures to metalinguistic abstraction, models of evaluation, and the effective, procedural interpretation of the means of combination and the means of abstraction of languages supporting experimental programming styles. Formal training in mathematics equipped me with some of the sensibilities and sensitivities necessary to independent, academic inquiry into computer science: an appreciation of rigor, a sense of organization, and an inclination to methodically approach problems. Self-study was adequate to the task of preparing to effectively treat the problems that I encountered in the past, but I have reached the point in my professional development and individual study where a formal exposure to computer science is required for my continued growth in the field. I hope to continue to mature academically in computer science so that I can approach new problems in a capable manner, and as part of my continued development, I look forward to the opportunity to study computer science at UCLA.

Thank you,

Joseph D. Kulisics

6 November 2002