David L. Parnas Symposium

Dan Hoffman
Department of Computer Science
University of Victoria
dhoffman@csr.uvic.ca

Dave Weiss
Department of Software Technology Research
Avaya Laboratories
weiss@avaya.com

Abstract

David L. Parnas is one of the grandmasters of software engineering. His academic research and industrial collaborations have exerted far-reaching influence on software design and development. His groundbreaking writings capture the essence of the innovations, controversies, challenges, and solutions of the software industry. Together, they constitute the foundation for modern software theory and practice. This symposium is being held in recognition of Parnas’s work and in honour of his 60th birthday. It is an opportunity for everyone in the software engineering community to celebrate his contributions, and to think hard about where we are today and where we are going.

1. The Joy of Parnas

Speaker: Jon Bentley

Affiliation: Bell Laboratories

Some beautiful objects are not practical; the lovely Gossamer Condor won the prize for the first human-powered flight, then served no further purpose. Some practical objects are not beautiful; military transport aircraft often fall into this category. It is wrong to conclude, though, that practical objects cannot be beautiful: the images of certain fighter aircraft bring delight to their friends and fear to their foes.

Dave Parnas is famous for industrial-strength solutions to real-world software problems, and justly so. His ideas are also beautiful. This talk will celebrate the elegance of Dave’s insights, the power of his principles, the lucidity of his writing, and his passion for precision.

2. Software Product Lines — Parnas’s Program Families Come of Age

Speaker: Paul Clements

Affiliation: Software Engineering Institute

Abstract: A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Similar to product lines in manufacturing industries, software product lines embody a higher level of planned, strategic reuse than ever before. Companies adopting the product line approach are reporting phenomenal gains in productivity, time to market, and product quality. Typical are companies who say that for systems that used to take over a year to produce, the time to market is now about a week.

This talk will focus on the fundamentals of software product lines: the production of the core assets, the creation of products from those core assets, and the management necessary to orchestrate it all. We will explore some of the practice areas that organizations must master in order to achieve product line success, and relate some experiences of some successful software product line companies.

3. Parnas Tables: A Practical Formalism

Speaker: Jo Atlee

Affiliation: University of Waterloo

Abstract: An important aspect of “engineering” a software system is being able to precisely document intended and actual software behaviour. In traditional engineering disciplines, “precise documentation” means mathematical descriptions and models. In contrast, mathematical methods are not widely used to document
software—primarily because software behaviour can depend on a large number of conditions, making it difficult to express the behaviour in a compact mathematical description.

Parnas Tables are a collection of tabular templates that address this problem by organizing functional and relational descriptions into sub-relations. More specifically, a Parnas Table separates a case-based relation into its component cases, each of which is a simpler expression to write and to comprehend. This talk will discuss some of the challenges and open problems in writing and reviewing tabular documentation. It will also review industrial applications that have benefited from using Parnas Tables.

4. Abstract Interfaces, Distributed Computing, and Program Semantics

Speaker: Jim Waldo

Affiliation: Sun Microsystems

Abstract: Abstract interfaces are nothing more than the simple idea that what is being done can be distinguished from how that thing is done. Like many simple ideas, this one has had a profound impact, acting as the foundation for many of the major gains in software productivity and quality in the past 30 years, from structured programming to objects. Nor is the impact of this idea merely historical; current strides in distributed computing can be seen as exploiting this same idea in yet another area of computer science.

In this talk, we will look at some of the places that abstract interfaces have been used to manage complexity, help with maintainability, and allow extended lifetimes of computing systems. We will also look at some of the more recent uses of the notion in distributed computing. In taking this tour, we will also see some places where the notion has been simplified from its original characterization, and see how those simplifications are beginning to limit the use of the systems built around such interfaces. In particular, the notion of the semantic constraints on the implementation that was used behind an abstract interface, while present in the original discussion of the topic, has been lost. We will look at some problems faced by current systems that argue that it is time to bring back some kind of meaning, and discuss some of the approaches to that problem that might be used.

5. SE education Panel

Panelists: Dave Parnas, Tim Lethbridge, Mike Lutz, Jim Waldo, Dan Hoffman (moderator)


Speaker: Dave Weiss

Affiliation: Avaya Laboratories

Abstract: At ICSE 2001 we are honoring the work of one of the grandmasters of our field, highlighting the fundamental ideas that David L. Parnas invented and expounded, including such ideas as information hiding, abstract interfaces, the uses relation, program families, explicit layered exception handling, and deterministic scheduling for hard real-time systems.

Do you need to understand how to organize your software into modules so that it can be easily maintained and your modules are reusable, whether they are expressed as classes, packages, or other forms? Dave Parnas identified the information hiding principle and showed how to use it to construct workable, reusable modular structures that are stable over time.

Are you struggling to create APIs to make your software useful to application programmers? Dave Parnas devised the idea (and coined the term) for abstract interfaces, and showed how to design interfaces that provided services without revealing their implementations. Languages like C++ and Java directly support this idea with abstract classes.

Are you wondering how to create your software as a set of layers that define a hierarchical structure that meets your requirements, lets you build your system a few layers at a time, and lets others add to the structure that you have created? Dave Parnas clearly explained what a hierarchical structure is, what some of the important hierarchical structures that we use are, why people often confuse them, and how to create a layered structure that meets your needs.

Do you know that your software is going to exist in many different versions, but are having difficulty designing your software not just to accommodate the different versions, but to take advantage of your situation to make your development process more efficient? Dave Parnas
defined program families to help with just this situation and showed how to create them in a cost-effective way.

However, Dave has been busy in more than just technical areas. His work includes commentary on the social responsibility of software engineers, both by exposition and by example, and on how we should educate our students so that software development becomes an engineering profession. His stance on our inability to create trustworthy software for the Strategic Defense Initiative, as well as his thoughts on how to teach software engineering have influenced how we think, act, and teach, as well as how the public perceives us.

David Parnas is both a clear and creative thinker and an extraordinary expositor of seminal ideas. The issues that he addresses are at the heart of software engineering today; his explanations are still relevant and his solutions, trialed on real systems, transfer well to today's software development organizations and environments.


Speaker: Fred Brooks

Affiliation: University of North Carolina

Abstract: What kind of person do we honor today?

Fearlessly honest; honestly fearless.

Dave is brilliant; many people are brilliant. Dave is impressively productive; many are productive. Dave is articulate and lucid; many are articulate, some are lucid.

As much as we admire that profound, tough, clean mind, it is the attributes of character that we admire more. I submit that it is Dave's character attributes, as much as his mind, that have produced this incredible body of work.

First his honesty. Dave is ready to question his own assumptions, ready to accept such critique by others, and ready to let the chips fall where they may as the logical consequences of assumptions emerge. “Our first duty in research is to the truth.”

Second, his principled approach to every task. He has technical principles, which he articulates, which he follows in his research, and which he tests and demonstrates in practice. His principles of professional ethics he has clearly enunciated and consistently followed. His social conscience, his professional conscience, and his personal conscience are all keen.

Third, his boldness. He likes to question assumptions, to challenge widely held beliefs and attitudes. He is not shy about criticizing, even satirizing, work he considers shoddy. He freely states opinions, as such, that cannot yet be supported by data, but which he bases on experience and judgment. “I am not a modest man.”

Fourth, his commitment. Mili has put it well: Dave has not looked for quick or superficial contributions, but has made a “long-term, focused, painstaking effort.” Each of those four words is crucial for his contributions.

An honest sketch of an honest man must remark that any one virtue—even honesty—fully embraced, makes some other virtues difficult.