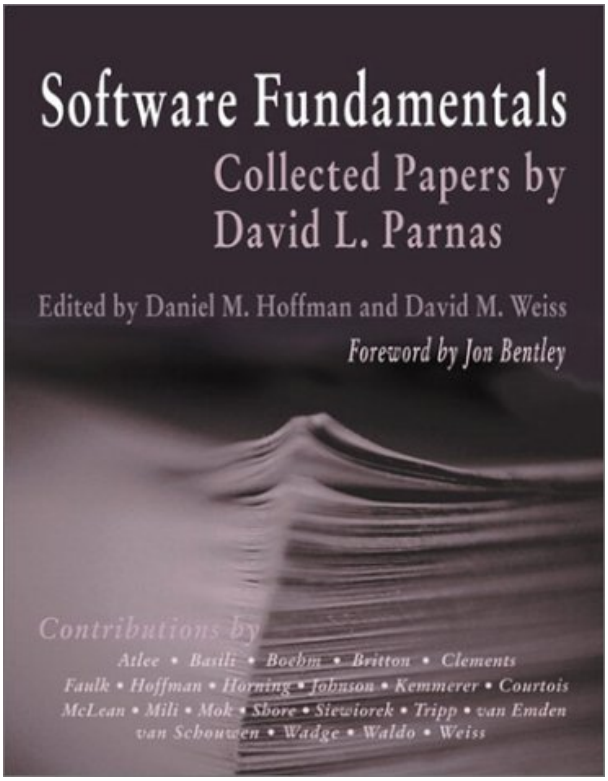
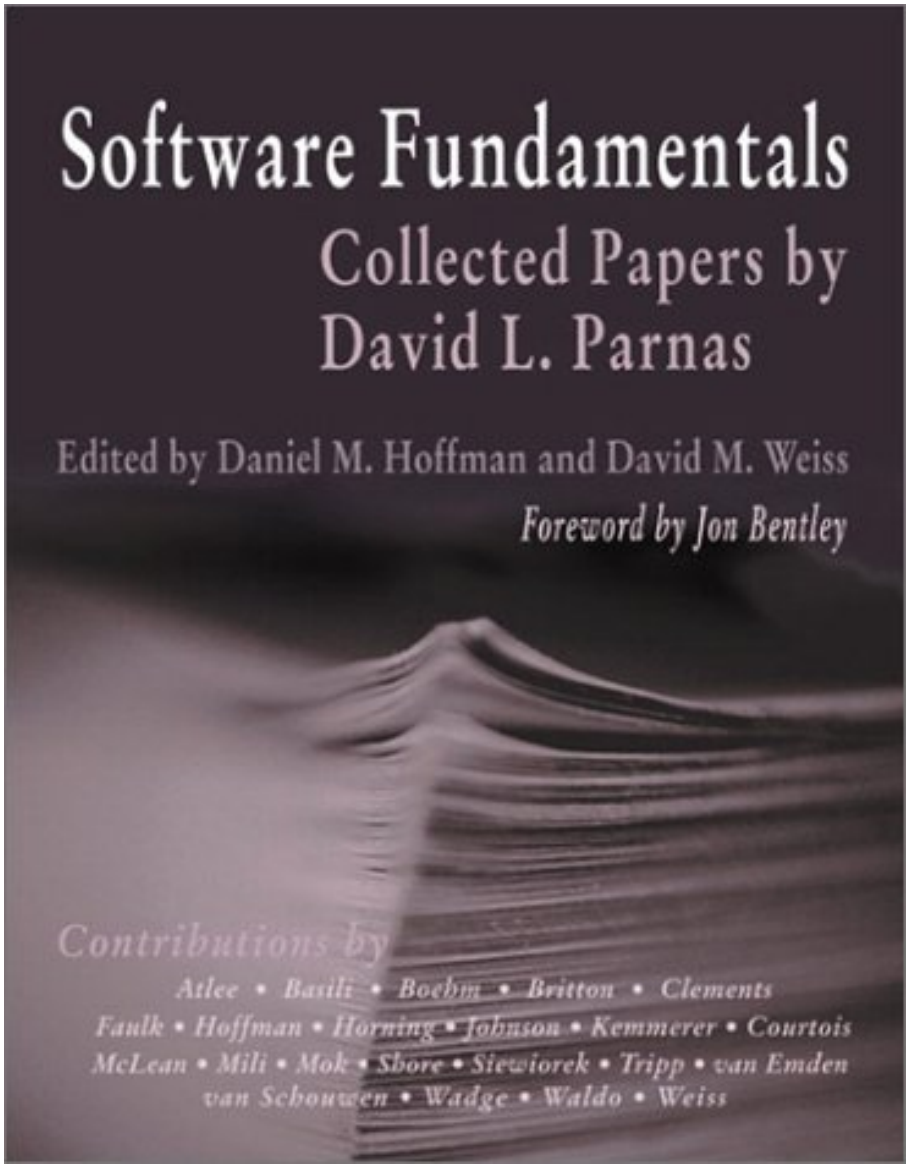
SOFTWARE FUNDAMENTALS: COLLECTED PAPERS





SOFTWARE FUNDAMENTALS: COLLECTED PAPERS PDF

So, when you need quickly that book **Software Fundamentals: Collected Papers**, it does not have to await some days to get the book Software Fundamentals: Collected Papers You could directly obtain guide to conserve in your device. Even you like reading this Software Fundamentals: Collected Papers everywhere you have time, you can enjoy it to check out Software Fundamentals: Collected Papers It is undoubtedly handy for you that intend to get the a lot more priceless time for reading. Why do not you invest five mins and also spend little money to obtain guide Software Fundamentals: Collected Papers here? Never allow the brand-new thing quits you.

From the Inside Flap
Daniel M. Hoffman and David M. Weiss

Why Create a Book Around Dave Parnas's Work? It is sometimes said that progress in a scientific discipline can be measured by how quickly its founders are forgotten. Software development, sometimes called software engineering, is not a scientific discipline and is still young: Many of those who formulated fundamental principles in the field are still active in it. Unfortunately, we have the worst of both worlds: Our founders seem dimly remembered, and we are making little progress towards becoming a discipline. Fundamental ideas, such as information hiding and abstraction, are only vaguely understood by those who need them most and are constantly reinvented. Those who practice software development and those who teach software engineering seem uneducated in, and unaware of, the history of their profession. This book is our attempt to provide a view of the work of one of the grandmasters of our field, highlighting the fundamental ideas that he and his colleagues invented and expounded. We hope to provide a reference for those who teach and those who do, giving them both an historical record, a clear explanation of fundamental ideas that will help them in their work, and a set of examples to use and emulate. David L. 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 to today's software development organizations and environments. Do you need to understand how to organize your software into modules so it can be easily maintained and so that 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 to use it to construct workable, reusable modular structures that are stable over time. (See Chapters 2 and 16.) 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 provide services without revealing their implementations. (See Chapter 15.) 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. (See Chapter 8.) 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. (See Chapters 10 and 14.) 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. His stance on our inability to create trustworthy software for the Strategic Defense Initiative is represented (Chapters 26 and 27), as well as his thoughts on how to teach software engineering (Chapter 31 and 32), and how to make software engineering a profession (Chapters 28 and 33).

Why Did We Pick These Papers? The preceding are just a few examples of the ideas described in the papers that constitute this book. Out of the more than two hundred papers that Dave has published, we selected thirty-two, plus one special one that he did not write, but strongly influenced. We picked technical papers that expressed fundamental ideas that were groundbreaking when they were published, that have an enduring message, and that are models of exposition, and nontechnical papers that had an influence on the opinions of the time. Some were controversial when published and remain so. An outstanding aspect of Dave's career is his insistence that his ideas be tested on real problems, where one cannot define away the complexity of the world in the interest of devising an elegant solution. Perhaps the best known examples are the operational flight program (OFP) for the U.S. Navy's A-7E aircraft and the shut-down software for the Darlington nuclear power plant. The A-7E project, also known as the Software Cost Reduction (SCR) project, was conducted by Dave and colleagues at the U.S. Naval Research Laboratory (NRL). It was a demonstration of how to apply ideas such as information hiding, abstraction, cooperating sequential processes, deterministic scheduling, program families, formal specification, hierarchical structuring, and undesired event handling to the design of a hard-real-time system. Many of the same approaches now appear in modern designs and modern languages under different names; a few diverse examples are exception handling (Chapter 12) and the observer pattern (Chapter 22). Several years of Dave's time and effort were directed at making the SCR software and its documentation an engineering model of how to develop and document software. The papers derived from the project that appeared in the research literature; such as Chapters 6, 12, 15, 16, 17, 18, and 22, only tell part of the story. The complete set of requirements and design documentation (including what we now term architecture), was published as technical reports by NRL and serve as detailed guides and templates for those wishing to use the ideas.

How Is the Book Organized? This book contains thirty-three papers divided into four sections. Dave has written a short introduction to each section and we have invited a guest author to write an introduction to each paper. Specification and Description contains six papers, focusing on the most important kinds of software engineering documentation and the roles that they play. Relational and tabular documentation are presented in depth, including both the underlying mathematical basis and practical notations suitable for use by working programmers. Design contains thirteen papers, covering the principles and techniques that have been central to Dave's work for the past three decades. Information hiding is emphasized, including the role of information hiding in abstract interfaces, its application in complex systems, and its implications in the design of program families. Concurrency and Scheduling contains two early papers on the use of semaphores and two more recent papers on new approaches to synchronization and scheduling. The latter focus on achieving both good performance and a module structure that supports maintainability and comprehensibility. Finally, Commentary contains ten papers on a wide variety of topics including education, social issues, the role of the engineer, and the status of software engineering as an engineering profession. In the interests of preserving the historical record and of leaving Dave's writing style unperturbed, we have tampered as little as possible with the papers that appear here, only correcting a few typographical errors in most papers.

Why Have Guest Introductions? The papers span the period from the 1970s through the 1990s. Some use old examples and notations that may not seem relevant to today's Internet world. We asked leading members of our field to write short introductions to the papers to explain the papers' historical and modern relevance.

Right from the start, we knew that the introductions must be fun to read and worth reading. They must tell the reader something worth knowing that is not in the paper or is not obvious from reading the paper. We were most fortunate in gathering an impressive collection of authors. Some have been involved with Dave since his work at NRL and earlier. Others participated in the SCR Workshops that continued the NRL work. Some have never directly collaborated with Dave. All are excellent writers with special insights about the significance of the papers both at the time of writing and today. All wrote with enthusiasm and skill. The thirty-three paper introductions are an important contribution in their own right. The fact that these people were all willing, indeed eager, to contribute speaks highly of Dave's work. Dave collaborated with us on the selection of the papers in this book. On several occasions he commented that we were likely to get people angry once again. That is the nature of the man and his ideas: insightful, creative, stimulating, provocative. We hope you find that the papers in this book have the same qualities. It is our present to Dave on his sixtieth birthday.

Acknowledgments We would like to say that we had the idea for this book on our own, but it actually originated with Brad Appleton. Thanks, Brad, for giving us the chance to carry out the idea. Organizational and production details for a book of this sort can get quickly out of hand without an experienced professional editor to guide you. Debbie Lafferty at Addison-Wesley has been a cheerful, steadfast guide for us, appreciating the idea for the book from the first, and working with us to make it happen. During the course of production, all of the papers contained herein were retyped. Dorene Brummel happily took on the job of proofreading them, for which we are very grateful. Joanne Glazer Weiss showed outstanding forebearance and support when her husband plunged into this project immediately after finishing his first book. He thanks and loves her. Duck Bay, British Columbia September, 2000

0201703696P04182001

From the Back Cover

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 book contains thirty-three of his most influential papers in various areas of software engineering. Leading thinkers in software engineering have contributed short introductions to each paper to provide the historical context surrounding each paper's conception and writing.

Software Fundamentals: Collected Papers by David L. Parnas is a practical guide to key software engineering concepts that belongs in the library of every software professional. It introduces and explains such seminal topics as:

- Relational and tabular documentation
- Information hiding as the basis for modular program construction
- Abstract interfaces that provide services without revealing implementation
- Program families for the efficient development of multiple software versions
- The status of software engineering as a profession
- Why complex software, such as for the Strategic Defense Initiative, is unlikely to work the first time that it is used in the field

As a celebration of one of the fathers of modern software engineering, and as a practical guide to the key concepts underlying software development, Software Fundamentals is valuable for professionals, especially those who are interested in teaching the fundamentals of software.

David Parnas is highly regarded for his many valuable contributions to software engineering. He developed and applied cutting-edge software technology to the U.S. Navy's A-7E aircraft, and he advised the Atomic Energy Control Board of Canada on the use of safety-critical, real-time software. During his career, he has contributed more than 200 papers to ACM, IEEE, and ICSE publications. He won an ACM "Best Paper" award, two "Most Influential Paper" awards from ICSE, and the 1998 "Outstanding Researcher" award from ACM SIGSOFT. In May 2001, Dr. Parnas was recognized at the International Conference on Software Engineering for his lifetime of outstanding achievements.

About the editors:

Daniel Hoffman is an Associate Professor of Computer Science at the University of Victoria in British Columbia. David Weiss is the Director of the Software Technology Research Department at Avaya Laboratories.

0201703696B04062001

About the Author

Daniel Hoffman is an Associate Professor of Computer Science at the University of Victoria in British Columbia.

David M. Weiss is the Director of the Software Production Research Department at Avaya Laboratories. His technical work has evolved into the invention of processes that incorporate ideas from families, design for change, measurement, precise specification, and technology transfer. The result has been a software production process based on family-oriented abstraction, specification, and translation, known as FAST.

0201703696AB04062001

SOFTWARE FUNDAMENTALS: COLLECTED PAPERS PDF

Software Fundamentals: Collected Papers. The developed innovation, nowadays sustain every little thing the human demands. It consists of the everyday activities, tasks, workplace, home entertainment, as well as more. Among them is the terrific net connection as well as computer system. This condition will alleviate you to assist among your pastimes, reading routine. So, do you have going to read this publication Software Fundamentals: Collected Papers now?

The reason of why you can obtain and get this *Software Fundamentals: Collected Papers* faster is that this is the book in soft file type. You could check out guides Software Fundamentals: Collected Papers any place you desire even you remain in the bus, office, home, as well as other places. Yet, you might not have to relocate or bring guide Software Fundamentals: Collected Papers print wherever you go. So, you will not have bigger bag to bring. This is why your selection to make far better concept of reading Software Fundamentals: Collected Papers is actually handy from this instance.

Recognizing the means ways to get this book Software Fundamentals: Collected Papers is likewise useful. You have been in appropriate site to begin getting this details. Get the Software Fundamentals: Collected Papers web link that we give right here and visit the web link. You can get the book Software Fundamentals: Collected Papers or get it as quickly as possible. You could promptly download this Software Fundamentals: Collected Papers after getting deal. So, when you require the book quickly, you could straight receive it. It's so very easy therefore fats, right? You must like to by doing this.

SOFTWARE FUNDAMENTALS: COLLECTED PAPERS PDF

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 book contains 33 of his most influential papers in various areas of software engineering. Leading thinkers in software engineering have contributed short introductions to each paper to provide the historical context surrounding each papers conception and writing. Software Fundamentals: Collected Papers by David L. Parnas is a practical guide to key software engineering concepts that belongs in the library of every software professional. It introduces and explains such seminal topics as: Relational and tabular documentation Information hiding as the basis for modular program construction Abstract interfaces that provide services without revealing implementation Program families for the efficient development of multiple software versions The status of software engineering as a

- Sales Rank: #1176137 in Books
- Published on: 2001-04-19
- Original language: English
- Number of items: 1
- Dimensions: 8.90" h x 1.50" w x 7.10" l, 2.60 pounds
- Binding: Paperback
- 688 pages

From the Inside Flap
Daniel M. Hoffman and David M. Weiss

Why Create a Book Around Dave Parnas's Work? It is sometimes said that progress in a scientific discipline can be measured by how quickly its founders are forgotten. Software development, sometimes called software engineering, is not a scientific discipline and is still young: Many of those who formulated fundamental principles in the field are still active in it. Unfortunately, we have the worst of both worlds: Our founders seem dimly remembered, and we are making little progress towards becoming a discipline. Fundamental ideas, such as information hiding and abstraction, are only vaguely understood by those who need them most and are constantly reinvented. Those who practice software development and those who teach software engineering seem uneducated in, and unaware of, the history of their profession. This book is our attempt to provide a view of the work of one of the grandmasters of our field, highlighting the fundamental ideas that he and his colleagues invented and expounded. We hope to provide a reference for those who teach and those who do, giving them both an historical record, a clear explanation of fundamental ideas that will help them in their work, and a set of examples to use and emulate. David L. 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 to today's software development organizations and environments. Do you need to understand how to organize your software into modules so it can be easily maintained and so that 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 to use it to construct workable, reusable

modular structures that are stable over time. (See Chapters 2 and 16.) 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 provide services without revealing their implementations. (See Chapter 15.) 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. (See Chapter 8.) 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. (See Chapters 10 and 14.) 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. His stance on our inability to create trustworthy software for the Strategic Defense Initiative is represented (Chapters 26 and 27), as well as his thoughts on how to teach software engineering (Chapter 31 and 32), and how to make software engineering a profession (Chapters 28 and 33).

Why Did We Pick These Papers? The preceding are just a few examples of the ideas described in the papers that constitute this book. Out of the more than two hundred papers that Dave has published, we selected thirty-two, plus one special one that he did not write, but strongly influenced. We picked technical papers that expressed fundamental ideas that were groundbreaking when they were published, that have an enduring message, and that are models of exposition, and nontechnical papers that had an influence on the opinions of the time. Some were controversial when published and remain so. An outstanding aspect of Dave's career is his insistence that his ideas be tested on real problems, where one cannot define away the complexity of the world in the interest of devising an elegant solution. Perhaps the best known examples are the operational flight program (OFP) for the U.S. Navy's A-7E aircraft and the shut-down software for the Darlington nuclear power plant. The A-7E project, also known as the Software Cost Reduction (SCR) project, was conducted by Dave and colleagues at the U.S. Naval Research Laboratory (NRL). It was a demonstration of how to apply ideas such as information hiding, abstraction, cooperating sequential processes, deterministic scheduling, program families, formal specification, hierarchical structuring, and undesired event handling to the design of a hard-real-time system. Many of the same approaches now appear in modern designs and modern languages under different names; a few diverse examples are exception handling (Chapter 12) and the observer pattern (Chapter 22). Several years of Dave's time and effort were directed at making the SCR software and its documentation an engineering model of how to develop and document software. The papers derived from the project that appeared in the research literature; such as Chapters 6, 12, 15, 16, 17, 18, and 22, only tell part of the story. The complete set of requirements and design documentation (including what we now term architecture), was published as technical reports by NRL and serve as detailed guides and templates for those wishing to use the ideas.

How Is the Book Organized? This book contains thirty-three papers divided into four sections. Dave has written a short introduction to each section and we have invited a guest author to write an introduction to each paper. Specification and Description contains six papers, focusing on the most important kinds of software engineering documentation and the roles that they play. Relational and tabular documentation are presented in depth, including both the underlying mathematical basis and practical notations suitable for use by working programmers. Design contains thirteen papers, covering the principles and techniques that have been central to Dave's work for the past three decades. Information hiding is emphasized, including the role of information hiding in abstract interfaces, its application in complex systems, and its implications in the design of program families. Concurrency and Scheduling contains two early papers on the use of semaphores and two more recent papers on new approaches to synchronization and scheduling. The latter focus on

achieving both good performance and a module structure that supports maintainability and comprehensibility. Finally, Commentary contains ten papers on a wide variety of topics including education, social issues, the role of the engineer, and the status of software engineering as an engineering profession. In the interests of preserving the historical record and of leaving Dave's writing style unperturbed, we have tampered as little as possible with the papers that appear here, only correcting a few typographical errors in most papers.

Why Have Guest Introductions? The papers span the period from the 1970s through the 1990s. Some use old examples and notations that may not seem relevant to today's Internet world. We asked leading members of our field to write short introductions to the papers to explain the papers' historical and modern relevance. Right from the start, we knew that the introductions must be fun to read and worth reading. They must tell the reader something worth knowing that is not in the paper or is not obvious from reading the paper. We were most fortunate in gathering an impressive collection of authors. Some have been involved with Dave since his work at NRL and earlier. Others participated in the SCR Workshops that continued the NRL work. Some have never directly collaborated with Dave. All are excellent writers with special insights about the significance of the papers both at the time of writing and today. All wrote with enthusiasm and skill. The thirty-three paper introductions are an important contribution in their own right. The fact that these people were all willing, indeed eager, to contribute speaks highly of Dave's work. Dave collaborated with us on the selection of the papers in this book. On several occasions he commented that we were likely to get people angry once again. That is the nature of the man and his ideas: insightful, creative, stimulating, provocative. We hope you find that the papers in this book have the same qualities. It is our present to Dave on his sixtieth birthday.

Acknowledgments We would like to say that we had the idea for this book on our own, but it actually originated with Brad Appleton. Thanks, Brad, for giving us the chance to carry out the idea. Organizational and production details for a book of this sort can get quickly out of hand without an experienced professional editor to guide you. Debbie Lafferty at Addison-Wesley has been a cheerful, steadfast guide for us, appreciating the idea for the book from the first, and working with us to make it happen. During the course of production, all of the papers contained herein were retyped. Dorene Brummel happily took on the job of proofreading them, for which we are very grateful. Joanne Glazer Weiss showed outstanding forebearance and support when her husband plunged into this project immediately after finishing his first book. He thanks and loves her. Duck Bay, British Columbia September, 2000

0201703696P04182001

From the Back Cover

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 book contains thirty-three of his most influential papers in various areas of software engineering. Leading thinkers in software engineering have contributed short introductions to each paper to provide the historical context surrounding each paper's conception and writing.

Software Fundamentals: Collected Papers by David L. Parnas is a practical guide to key software engineering concepts that belongs in the library of every software professional. It introduces and explains such seminal topics as:

- Relational and tabular documentation

- Information hiding as the basis for modular program construction
- Abstract interfaces that provide services without revealing implementation
- Program families for the efficient development of multiple software versions
- The status of software engineering as a profession
- Why complex software, such as for the Strategic Defense Initiative, is unlikely to work the first time that it is used in the field

As a celebration of one of the fathers of modern software engineering, and as a practical guide to the key concepts underlying software development, Software Fundamentals is valuable for professionals, especially those who are interested in teaching the fundamentals of software.

David Parnas is highly regarded for his many valuable contributions to software engineering. He developed and applied cutting-edge software technology to the U.S. Navy's A-7E aircraft, and he advised the Atomic Energy Control Board of Canada on the use of safety-critical, real-time software. During his career, he has contributed more than 200 papers to ACM, IEEE, and ICSE publications. He won an ACM "Best Paper" award, two "Most Influential Paper" awards from ICSE, and the 1998 "Outstanding Researcher" award from ACM SIGSOFT. In May 2001, Dr. Parnas was recognized at the International Conference on Software Engineering for his lifetime of outstanding achievements.

About the editors:

Daniel Hoffman is an Associate Professor of Computer Science at the University of Victoria in British Columbia. David Weiss is the Director of the Software Technology Research Department at Avaya Laboratories.

0201703696B04062001

About the Author

Daniel Hoffman is an Associate Professor of Computer Science at the University of Victoria in British Columbia.

David M. Weiss is the Director of the Software Production Research Department at Avaya Laboratories. His technical work has evolved into the invention of processes that incorporate ideas from families, design for change, measurement, precise specification, and technology transfer. The result has been a software production process based on family-oriented abstraction, specification, and translation, known as FAST.

0201703696AB04062001

Most helpful customer reviews

19 of 20 people found the following review helpful.
At last, the classics begin to emerge
By Norman L. Kerth
If you are old enough to have studied David Parnas' papers, buy this book - all his treasures are collected in

one convenient place.
If you are too young to have studied Parnas' work, then know there is great value in studying the classics no matter what field you are in. Until now, we have not been able to study classics, as our field was too young for true classics to have been identified. As I read this book, I realize Parnas' papers have stood the test of time and are worthy of serious study by all in our field who wish to be thought of as professional software engineers.
During the early decades of our profession, David wrote some of the most insightful papers published. He uniquely wove a scholarly approach to understanding how we might develop our field, with a pragmatic view of what really happens as we set out to build software systems.
As I developed my career, his papers influenced how I thought about and approached my discipline. More than that, his papers influenced the foundation of all software engineering. For example, he first applied the word "module," to our field - the term since has been abused to the point where it means nothing, but he was talking about what we have come to call an object! With the term "information hiding," he was telling us how to design fine objects.
Beyond objects, he explored how to approach reuse, he laid the foundation for application frameworks, and showed us that methodical system specification was possible. These are just a few examples, he addressed so much more.
Adding value to Parnas' collection of papers, some of the most influential leaders in the field of software engineering introduce his papers, explaining how Parnas' ideas are being put to work in our modern day practice.
I remember a time, sitting beside my grandfather as he showed me pictures he had taken throughout his life. For him it was a chance to remember important times from his past. For me it was my opportunity to learn about a history that had great influence on me. His stories answered who I was and why my family was the way it was.
"Software Fundamentals," provides just such a valuable experience. If you are "one of the old guys," you will enjoy looking again at these wonderful ideas. If you are a "young'n" then learn about the foundations of software engineering, learn where we have come from, learn the lessons of an earlier generation and you will have my respect. After all, those who do not learn from history, are condemned to relive it.

12 of 12 people found the following review helpful.
There's nothing new under the sun...
By Brent Fulgham
The software world is full of "revolutionary" ideas that seem to be periodically rediscovered. Topics such as refactoring, data hiding, and "design for change" have all made recent rounds in the development world. However, most of these concepts have been part of the research literature for decades.
Much of the software development work done today is done by people lacking the requisite fundaments for the job. Very few are capable of assessing the true technical strengths of software products. Most are content to read the glossy sales brochures or shallow write-ups in trade magazines to maintain their knowledge of the state of the art. A careful reading of the collected papers in this volume go a long way towards protecting the reader from the modern snake oil salesmen of the software industry.
This book should be required reading for all software developers who strive to deserve the title "Engineer."

13 of 14 people found the following review helpful.
Classics and unknown gems
By James J. Horning
Despite a half-century of practice, a distressingly large portion of today's software is over budget, behind schedule, bloated, and buggy.
To those who wonder why, and whether anything can be done about it, I have long recommended the book {\it The Mythical Man-Month}, by Frederick P. Brooks, Jr. This book has stayed continuously in print since 1975, and remained remarkably relevant.

Now there is another book I would put beside it. {\it Software Fundamentals: Collected Papers by David L. Parnas} is more technical and less management-oriented, but equally thought-provoking.
Parnas has been writing seminal and provocative papers about software and software development for more than 30 years. This book collects more than 30 of these papers. It includes well-known classics such as "On the Criteria to Be Used in Decomposing Systems into Modules," "On a 'Buzzword': Hierarchical Structure," "On the Design and Development of Program Families," "Designing Software for Ease of Extension and Contraction," "A Rational Design Process: How and Why to Fake It," and "Software Engineering: An Unconsummated Marriage." It also has some lesser-known gems, such as "Who Taught Me About Software Engineering Research?", "Active Design Reviews: Principles and Practices," and "Software Aging."
Because the papers were written to stand alone, and because each has its own introduction, the reader can browse them in just about any order.
Browsing or reading this book, I think you'll be struck by how much of today's "conventional wisdom" about software was introduced (or championed very early) by Dave. Equally surprising is the number of his good ideas that have still not made their way into current practice. Anyone who cares about software should ask, Why?
Parnas isn't always right, but he's never dull. One of the most valuable things to do with this book is to pick something he says that you disagree with (preferably something you think is "obviously wrong"), and try to construct a convincing counter-argument. You'll probably find it harder than you expect, and you'll almost surely learn something valuable.

See all 14 customer reviews...

SOFTWARE FUNDAMENTALS: COLLECTED PAPERS PDF

Simply link your tool computer or gadget to the net attaching. Obtain the modern-day technology to make your downloading **Software Fundamentals: Collected Papers** completed. Also you do not want to review, you could directly close the book soft data and open Software Fundamentals: Collected Papers it later. You can likewise effortlessly get guide all over, considering that Software Fundamentals: Collected Papers it is in your device. Or when remaining in the office, this Software Fundamentals: Collected Papers is likewise advised to read in your computer gadget.

From the Inside Flap
Daniel M. Hoffman and David M. Weiss

Why Create a Book Around Dave Parnas's Work? It is sometimes said that progress in a scientific discipline can be measured by how quickly its founders are forgotten. Software development, sometimes called software engineering, is not a scientific discipline and is still young: Many of those who formulated fundamental principles in the field are still active in it. Unfortunately, we have the worst of both worlds: Our founders seem dimly remembered, and we are making little progress towards becoming a discipline. Fundamental ideas, such as information hiding and abstraction, are only vaguely understood by those who need them most and are constantly reinvented. Those who practice software development and those who teach software engineering seem uneducated in, and unaware of, the history of their profession. This book is our attempt to provide a view of the work of one of the grandmasters of our field, highlighting the fundamental ideas that he and his colleagues invented and expounded. We hope to provide a reference for those who teach and those who do, giving them both an historical record, a clear explanation of fundamental ideas that will help them in their work, and a set of examples to use and emulate. David L. 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 to today's software development organizations and environments. Do you need to understand how to organize your software into modules so it can be easily maintained and so that 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 to use it to construct workable, reusable modular structures that are stable over time. (See Chapters 2 and 16.) 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 provide services without revealing their implementations. (See Chapter 15.) 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. (See Chapter 8.) 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. (See Chapters 10 and 14.) 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. His stance on our inability to create trustworthy software for the Strategic Defense Initiative is represented (Chapters 26 and 27), as well as his thoughts on how to teach software engineering (Chapter 31 and 32), and

how to make software engineering a profession (Chapters 28 and 33).

Why Did We Pick These Papers? The preceding are just a few examples of the ideas described in the papers that constitute this book. Out of the more than two hundred papers that Dave has published, we selected thirty-two, plus one special one that he did not write, but strongly influenced. We picked technical papers that expressed fundamental ideas that were groundbreaking when they were published, that have an enduring message, and that are models of exposition, and nontechnical papers that had an influence on the opinions of the time. Some were controversial when published and remain so. An outstanding aspect of Dave's career is his insistence that his ideas be tested on real problems, where one cannot define away the complexity of the world in the interest of devising an elegant solution. Perhaps the best known examples are the operational flight program (OFP) for the U.S. Navy's A-7E aircraft and the shut-down software for the Darlington nuclear power plant. The A-7E project, also known as the Software Cost Reduction (SCR) project, was conducted by Dave and colleagues at the U.S. Naval Research Laboratory (NRL). It was a demonstration of how to apply ideas such as information hiding, abstraction, cooperating sequential processes, deterministic scheduling, program families, formal specification, hierarchical structuring, and undesired event handling to the design of a hard-real-time system. Many of the same approaches now appear in modern designs and modern languages under different names; a few diverse examples are exception handling (Chapter 12) and the observer pattern (Chapter 22). Several years of Dave's time and effort were directed at making the SCR software and its documentation an engineering model of how to develop and document software. The papers derived from the project that appeared in the research literature; such as Chapters 6, 12, 15, 16, 17, 18, and 22, only tell part of the story. The complete set of requirements and design documentation (including what we now term architecture), was published as technical reports by NRL and serve as detailed guides and templates for those wishing to use the ideas.

How Is the Book Organized? This book contains thirty-three papers divided into four sections. Dave has written a short introduction to each section and we have invited a guest author to write an introduction to each paper. Specification and Description contains six papers, focusing on the most important kinds of software engineering documentation and the roles that they play. Relational and tabular documentation are presented in depth, including both the underlying mathematical basis and practical notations suitable for use by working programmers. Design contains thirteen papers, covering the principles and techniques that have been central to Dave's work for the past three decades. Information hiding is emphasized, including the role of information hiding in abstract interfaces, its application in complex systems, and its implications in the design of program families. Concurrency and Scheduling contains two early papers on the use of semaphores and two more recent papers on new approaches to synchronization and scheduling. The latter focus on achieving both good performance and a module structure that supports maintainability and comprehensibility. Finally, Commentary contains ten papers on a wide variety of topics including education, social issues, the role of the engineer, and the status of software engineering as an engineering profession. In the interests of preserving the historical record and of leaving Dave's writing style unperturbed, we have tampered as little as possible with the papers that appear here, only correcting a few typographical errors in most papers.

Why Have Guest Introductions? The papers span the period from the 1970s through the 1990s. Some use old examples and notations that may not seem relevant to today's Internet world. We asked leading members of our field to write short introductions to the papers to explain the papers' historical and modern relevance. Right from the start, we knew that the introductions must be fun to read and worth reading. They must tell the reader something worth knowing that is not in the paper or is not obvious from reading the paper. We were most fortunate in gathering an impressive collection of authors. Some have been involved with Dave since his work at NRL and earlier. Others participated in the SCR Workshops that continued the NRL work. Some have never directly collaborated with Dave. All are excellent writers with special insights about the significance of the papers both at the time of writing and today. All wrote with enthusiasm and skill. The

thirty-three paper introductions are an important contribution in their own right. The fact that these people were all willing, indeed eager, to contribute speaks highly of Dave's work. Dave collaborated with us on the selection of the papers in this book. On several occasions he commented that we were likely to get people angry once again. That is the nature of the man and his ideas: insightful, creative, stimulating, provocative. We hope you find that the papers in this book have the same qualities. It is our present to Dave on his sixtieth birthday.

Acknowledgments We would like to say that we had the idea for this book on our own, but it actually originated with Brad Appleton. Thanks, Brad, for giving us the chance to carry out the idea. Organizational and production details for a book of this sort can get quickly out of hand without an experienced professional editor to guide you. Debbie Lafferty at Addison-Wesley has been a cheerful, steadfast guide for us, appreciating the idea for the book from the first, and working with us to make it happen. During the course of production, all of the papers contained herein were retyped. Dorene Brummel happily took on the job of proofreading them, for which we are very grateful. Joanne Glazer Weiss showed outstanding forebearance and support when her husband plunged into this project immediately after finishing his first book. He thanks and loves her. Duck Bay, British Columbia September, 2000

0201703696P04182001

From the Back Cover

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 book contains thirty-three of his most influential papers in various areas of software engineering. Leading thinkers in software engineering have contributed short introductions to each paper to provide the historical context surrounding each paper's conception and writing.

Software Fundamentals: Collected Papers by David L. Parnas is a practical guide to key software engineering concepts that belongs in the library of every software professional. It introduces and explains such seminal topics as:

- Relational and tabular documentation
- Information hiding as the basis for modular program construction
- Abstract interfaces that provide services without revealing implementation
- Program families for the efficient development of multiple software versions
- The status of software engineering as a profession
- Why complex software, such as for the Strategic Defense Initiative, is unlikely to work the first time that it is used in the field

As a celebration of one of the fathers of modern software engineering, and as a practical guide to the key concepts underlying software development, Software Fundamentals is valuable for professionals, especially those who are interested in teaching the fundamentals of software.

David Parnas is highly regarded for his many valuable contributions to software engineering. He developed and applied cutting-edge software technology to the U.S. Navy's A-7E aircraft, and he advised the Atomic Energy Control Board of Canada on the use of safety-critical, real-time software. During his career, he has contributed more than 200 papers to ACM, IEEE, and ICSE publications. He won an ACM "Best Paper" award, two "Most Influential Paper" awards from ICSE, and the 1998 "Outstanding Researcher" award from

ACM SIGSOFT. In May 2001, Dr. Parnas was recognized at the International Conference on Software Engineering for his lifetime of outstanding achievements.

About the editors:

Daniel Hoffman is an Associate Professor of Computer Science at the University of Victoria in British Columbia. David Weiss is the Director of the Software Technology Research Department at Avaya Laboratories.

0201703696B04062001

About the Author

Daniel Hoffman is an Associate Professor of Computer Science at the University of Victoria in British Columbia.

David M. Weiss is the Director of the Software Production Research Department at Avaya Laboratories. His technical work has evolved into the invention of processes that incorporate ideas from families, design for change, measurement, precise specification, and technology transfer. The result has been a software production process based on family-oriented abstraction, specification, and translation, known as FAST.

0201703696AB04062001

So, when you need quickly that book **Software Fundamentals: Collected Papers**, it does not have to await some days to get the book Software Fundamentals: Collected Papers You could directly obtain guide to conserve in your device. Even you like reading this Software Fundamentals: Collected Papers everywhere you have time, you can enjoy it to check out Software Fundamentals: Collected Papers It is undoubtedly handy for you that intend to get the a lot more priceless time for reading. Why do not you invest five mins and also spend little money to obtain guide Software Fundamentals: Collected Papers here? Never allow the brand-new thing quits you.