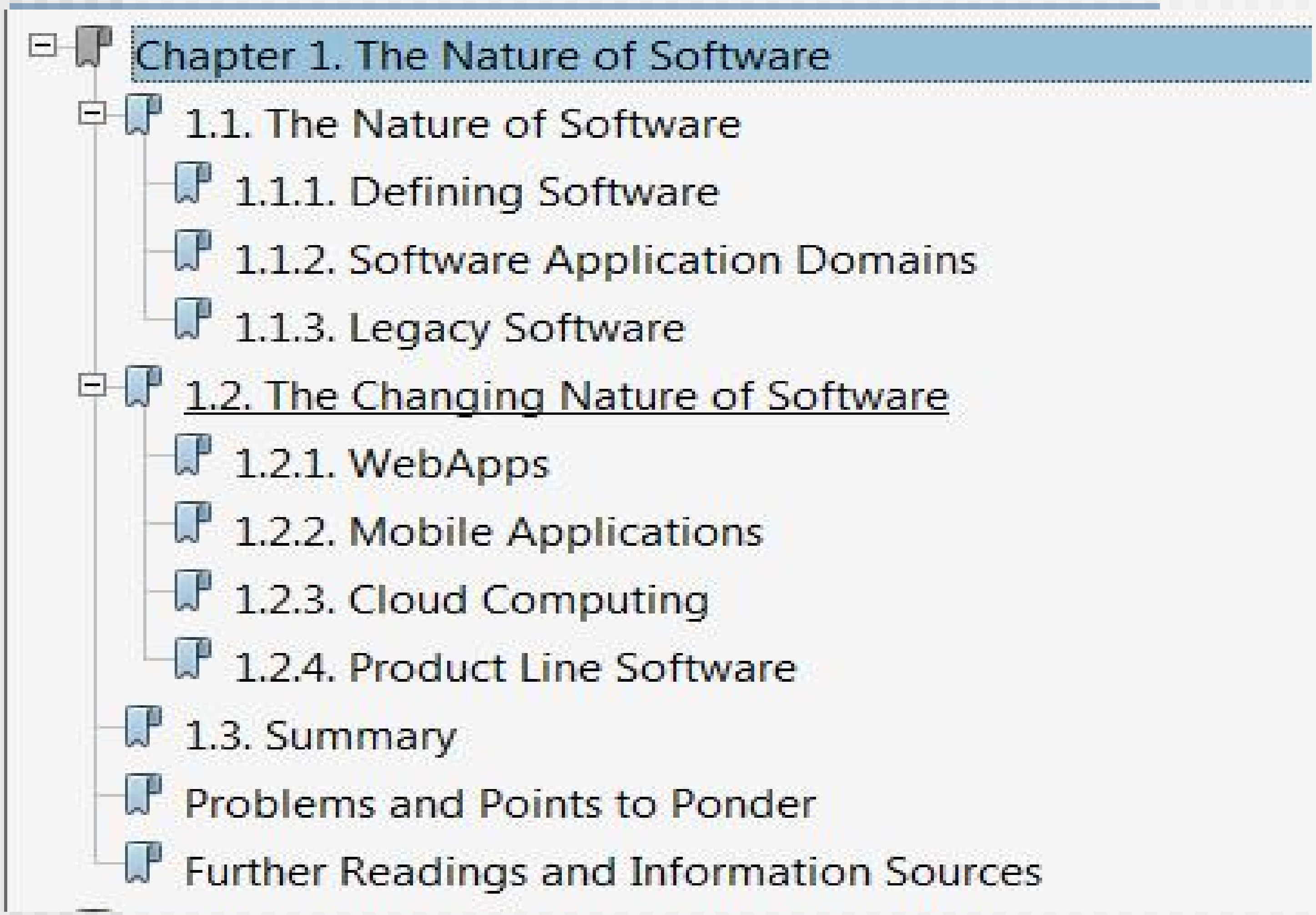
Chapter 1 THE NATURE OF SOFTWARE



Chapter 1 THE NATURE OF SOFTWARE

Computer software continues to be the single most important technology on the world stage.

No one could foresee that software would become embedded in systems of all kinds: transportation, medical, telecommunications, military, industrial, entertainment, office machines, . . . the list is almost endless. And if you believe the law of unintended consequences, there are many effects that we cannot yet predict.

No one could predict that millions of computer programs would have to be corrected, adapted, and enhanced as time passed. The burden of performing these “maintenance” activities would absorb more people and more resources than all work applied to the creation of new software.

As software’s importance has grown, the software community has

Chapter 1 THE NATURE OF SOFTWARE

continually attempted to develop technologies that will make it easier, faster, and less expensive to build and maintain high-quality computer programs. However, we have yet to develop a software technology that does it all, and the likelihood of one arising in the future is small. And yet, people bet their jobs, their comforts, their safety, their entertainment, their decisions, and their very lives on computer software. It better be right.

This book presents a framework that can be used by those who build computer software—people who must get it right. The framework encompasses a process, a set of methods, and an array of tools that we call ***software engineering***.

1.1 THE NATURE OF SOFTWARE

Today, software takes on a dual role. It is a product, and at the same time, the vehicle for delivering a product.

Chapter 1 THE NATURE OF SOFTWARE

Software delivers the most important product of our time—information. It transforms personal data (e.g., an individual’s financial transactions) so that the data can be more useful in a local context; it manages business information to enhance competitiveness; it provides a gateway to worldwide information networks(e.g., the Internet), and provides the means for acquiring information in all of its forms. It also provides a vehicle that can threaten personal privacy and a gateway that enables those with malicious intent to commit criminal acts.

Today, a huge software industry has become a dominant factor in the economies of the industrialized world. Teams of software specialists, each focusing on one part of the technology required to deliver a complex application, have replaced the lone programmer of an earlier era. And yet, the questions that were

Chapter 1 THE NATURE OF SOFTWARE

asked of the lone programmer are the same questions that are asked when modern computer-based systems are built:

- **Why does it take so long to get software finished?**
- **Why are development costs so high?**
- **Why can't we find all errors before we give the software to our customers?**
- **Why do we spend so much time and effort maintaining existing programs?**
- **Why do we continue to have difficulty in measuring progress as software is being developed and maintained?**

These, and many other questions, are a manifestation of the concern about software and the manner in which it is developed—a concern that has *led to the adoption of software engineering practice.*

Chapter 1 THE NATURE OF SOFTWARE

1.1.1 Defining Software

Today, most professionals and many members of the public at large feel that they understand software. But do they?A textbook description of software might take the following form:

Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Software is a logical rather than a physical system element. Therefore, software has one fundamental characteristic that makes it considerably different from hardware: Software doesn't "wear out."

Chapter 1 THE NATURE OF SOFTWARE

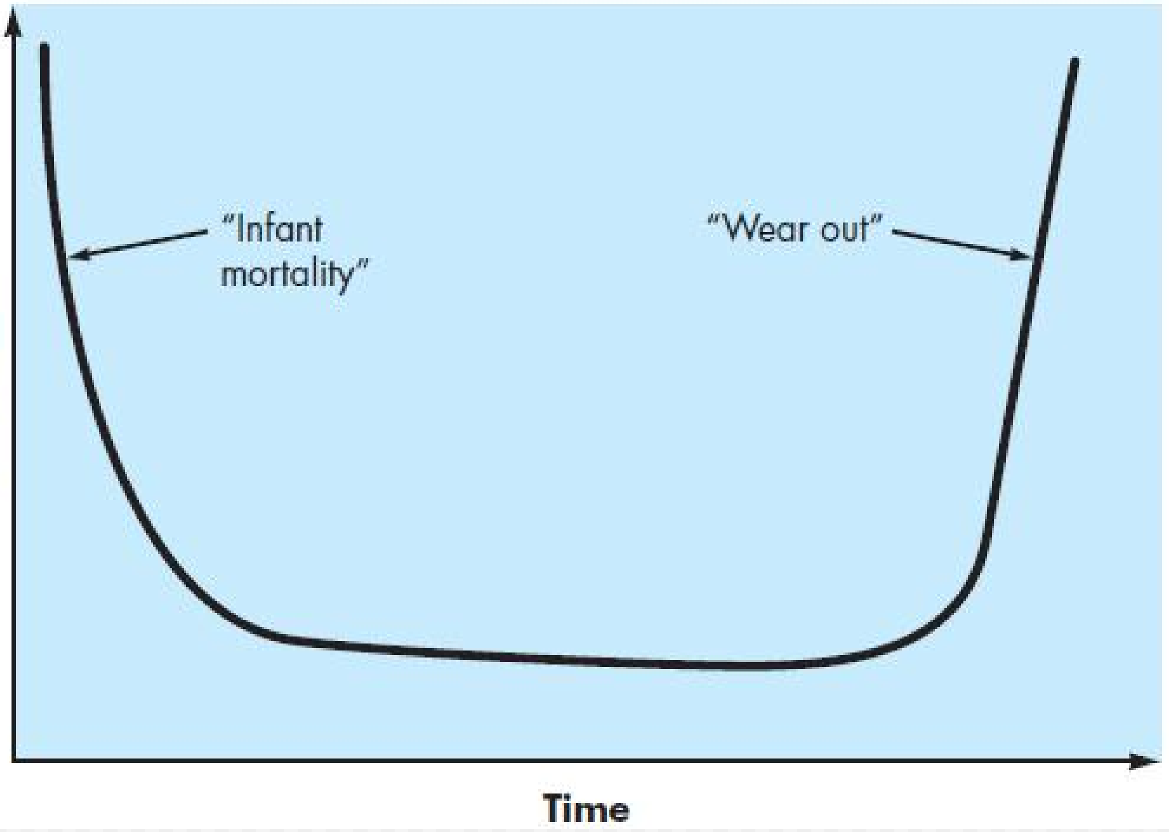
Figure 1.1 depicts failure rate as a function of time for hardware,and often called the "bathtub curve".

FIGURE 1.1

Failure curve for hardware

ADVICE

If you want to reduce software deterioration, you'll have to do better software design (Chapters 12 to 18).

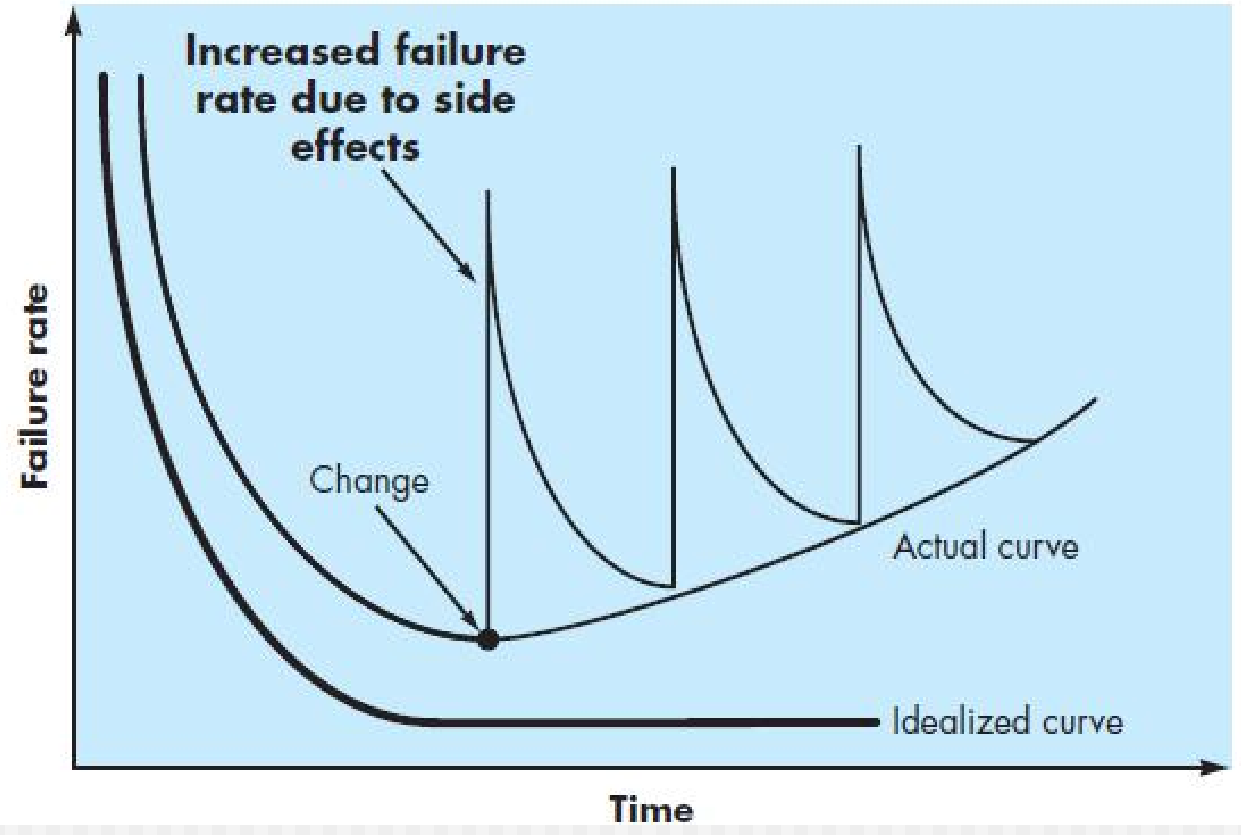


Chapter 1 THE NATURE OF SOFTWARE

Figure 1.2 depicts the "idealized curve" and the actual curve for software failure rate.

FIGURE 1.2

Failure curves for software



Chapter 1 THE NATURE OF SOFTWARE

Another aspect of wear illustrates the difference between hardware and software. When a hardware component wears out, it is replaced by a spare part. There are no software spare parts. Every software failure indicates an error in design or in the process through which design was translated into machine executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance.

1.1.2 Software Application Domains

Today, seven broad categories of computer software present continuing challenges for software engineers:

- **System software**
- **Application software —stand-alone programs that solve a specific business need.**

Chapter 1 THE NATURE OF SOFTWARE

- **Engineering/scientific software —a broad array of "number-crunching programs that range from astronomy to volcanology, from automotive stress analysis to orbital dynamics, and from computer-aided design to molecular biology, from genetic analysis to meteorology.**
- **Embedded software— resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.**
- **Product-line software —designed to provide a specific capability for use by many different customers.**
- **Web/Mobile applications —this network-centric software category spans a wide array of applications and encompasses both browser-based apps and software that resides on mobile devices.**

Chapter 1 THE NATURE OF SOFTWARE

- **Artificial intelligence software— makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artifi cial neural networks.**

1.1.3 Legacy Software

Hundreds of thousands of computer programs fall into one of the seven broad application domains discussed in the preceding subsection. Some of these are state-of-the-art software—just released to individuals, industry, and government. But other programs are older, in some cases much older.

These older programs—often referred to as *legacy software* have been the focus of continuous attention and concern since the

Chapter 1 THE NATURE OF SOFTWARE

1960s. Dayani-Fard and his colleagues describe legacy software in the following way:

Legacy software systems . . . were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve.

Many legacy systems remain supportive to core business functions and are ‘indispensable’ to the business.” Hence, legacy software is characterized by longevity and business criticality.

Unfortunately, there is sometimes one additional characteristic that is present in legacy software— poor quality . Legacy systems sometimes have inextensible designs, convoluted code,

Chapter 1 THE NATURE OF SOFTWARE

poor or nonexistent documentation, test cases and results that were never archived, a poorly managed change history—the list can be quite long. If the legacy software meets the needs of its users and runs reliably, it isn't broken and does not need to be fixed. As time passes, legacy systems often evolve for one or more of the following reasons:

- **The software must be adapted to meet the needs of new computing environments or technology.**
- **The software must be enhanced to implement new business requirements.**
- **The software must be extended to make it interoperable with other more modern systems or databases.**
- **The software must be re-architected to make it viable within a evolving computing environment.**

Chapter 1 THE NATURE OF SOFTWARE

1.2 THE CHANGING NATURE OF SOFTWARE

Four broad categories of software are evolving to dominate the industry. And yet, these categories were in their infancy little more than a decade ago.

1.2.1 WebApps

In the early days of the World Wide Web (circa 1990 to 1995), websites consisted of little more than a set of linked hypertext files that presented information using text and limited graphics. As time passed, the augmentation of HTML by development tools (e.g., XML, Java) enabled Web engineers to provide computing capability along with informational content. Web-based systems and applications (we refer to these collectively as WebApps) were born.

Chapter 1 THE NATURE OF SOFTWARE

Over the past decade, Semantic Web technologies (often referred to as Web 3.0) have evolved into sophisticated corporate and consumer applications. HTML5/CSS/JS

Exercise:by looking up information, to learn the history of web development.(web1.0~web4.0)

1.2.2 Mobile Applications

The term app has evolved to connote software that has been specifically designed to reside on a mobile platform (e.g., iOS, Android, or Windows Mobile). In most instances, mobile applications encompass a user interface,interoperability with Web-based resources,and local processing capabilities. In addition, a mobile app provides persistent storage capabilities within the platform.

Chapter 1 THE NATURE OF SOFTWARE

It is important to recognize that there is a subtle distinction between mobile web applications and mobile apps. A mobile web application (WebApp) allows a mobile device to gain access to web-based content via a browser that has been specifically designed. A mobile app can gain direct access to the hardware characteristics of the device (e.g., accelerometer or GPS location) and then provide the local processing and storage capabilities.

1.2.3 Cloud Computing

Cloud computing encompasses an infrastructure or "ecosystem" that enables any user, anywhere, to use a computing device to share computing resources on a broad scale. The overall logical architecture of cloud computing is represented in Figure 1.3.

Chapter 1 THE NATURE OF SOFTWARE

Referring to the figure 1.3, computing devices reside outside the cloud and have access to a variety of resources within the cloud. These resources encompass *applications, platforms,* and *infrastructure*. In its simplest form, an external computing device accesses the cloud via a Web browser. The cloud provides access to data. In addition, devices can access executable applications that can be used in lieu of apps that reside on the computing device.

The implementation of cloud computing requires the development of an architecture that encompasses front-end and back-end services. The front-end includes the client (user) device and the application software (e.g., a browser) that allows the back-end to be accessed. The back-end includes servers and related computing resources, data storage systems (e.g., databases),

Chapter 1 THE NATURE OF SOFTWARE

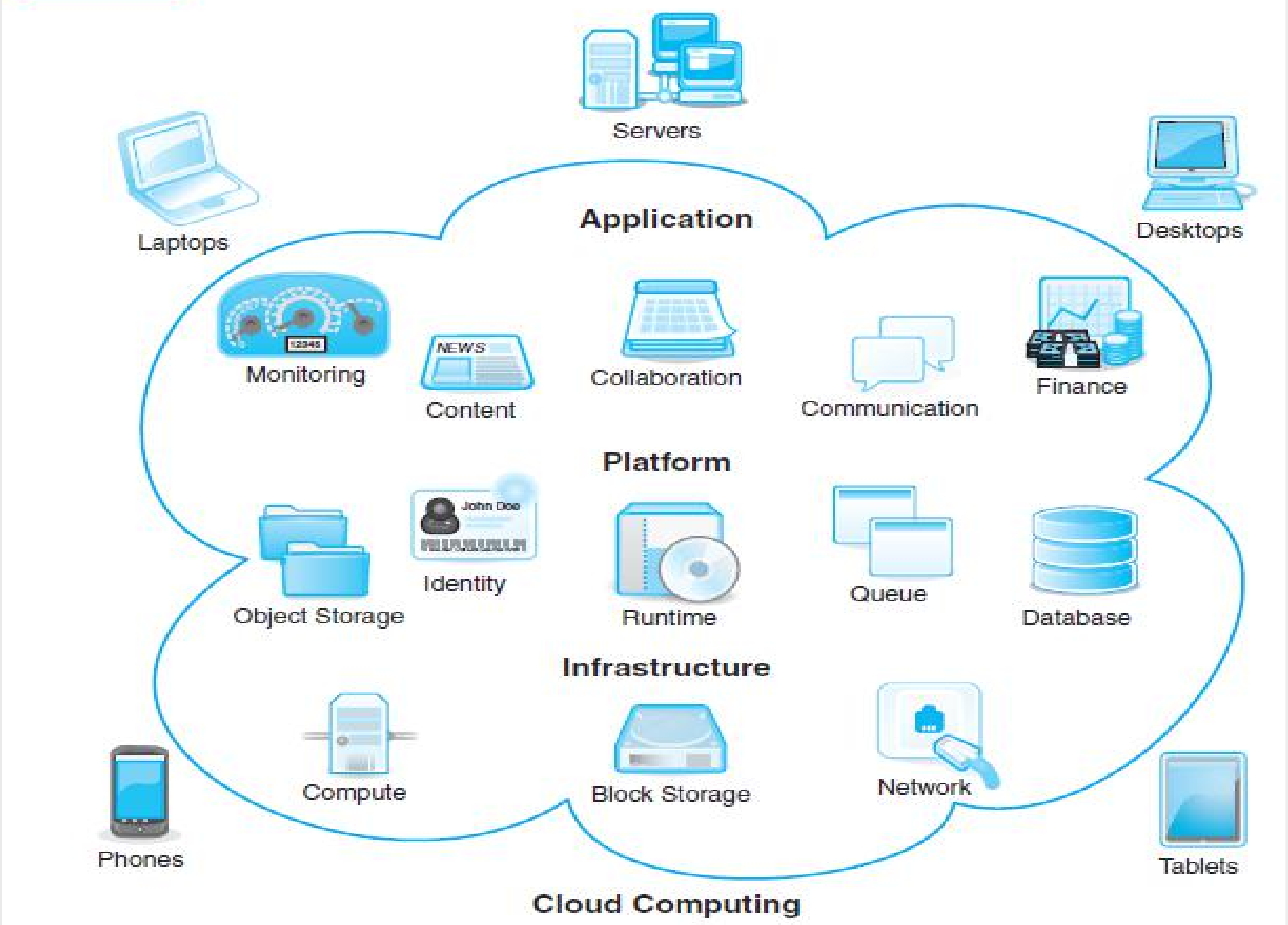
server-resident applications, and administrative servers that use middleware to coordinate and monitor traffic by establishing a set of protocols for access to the cloud and its resident resources.

The cloud architecture can be segmented to provide access at a variety of different levels from full public access to private cloud architectures accessible only to those with authorization.

See appendix 3.

Exercise: by looking up information, to learn cloud platform architecture.

FIGURE 1.3 Cloud computing logical architecture [Wik13]



Chapter 1 THE NATURE OF SOFTWARE

1.2.4 Product Line Software

The Software Engineering Institute defines a software product line as “a set of software-intensive systems that share a common, managed set of features satisfying 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.”

The concept of a line of software products that are related in some way is not new. But the idea that a line of software products, all developed using the same underlying application and data architectures, and all implemented using a set of reusable software components that can be reused across the product line provides significant engineering leverage.

Chapter 1 THE NATURE OF SOFTWARE

A software product line shares a set of assets that include requirements(Chapter 8), architecture (Chapter 13), design patterns (Chapter 16), reusable components (Chapter 14), test cases (Chapters 22 and 23), and other software engineering work products. In essence, a software product line results in the development of many products that are engineered by capitalizing on the commonality among all the products within the product line.

Refer to http://wiki.mbalib.com/wiki/软件产品线