

Facultad de Ingeniería Mecánica y Eléctrica

Universidad Autónoma de Nuevo Leon

Material Didáctico Elaborado por
Dr. Edgar Danilo Domínguez Vera

Carrera: Ingeniería en Tecnología de Software

Unidad de Aprendizaje: Temas Selectos de la Ingeniería de Software

Elaborado: 01-Junio-2016

Tiempo de elaboración: 4 horas

Bibliografía

Software Engineering: A practitioner's Approach

Roger S. Pressman

Seven Edition

Mc Graw Hill

Chapter 1

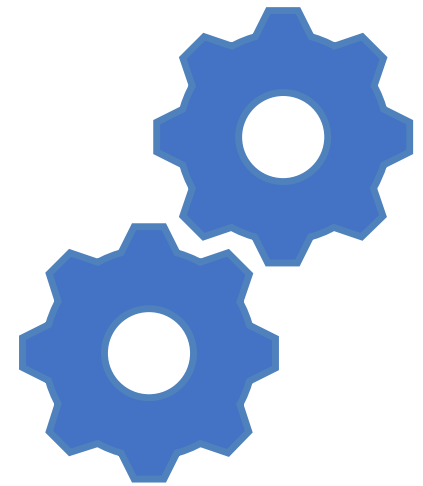
Software and Software Engineering



- Computer **software** is the product that software professionals build and then support over the long term.
- **Software Engineering** encompasses a **process**, a collection of **methods** (practice) and an array of **tools** that allow professionals to build **high-quality** computer software

Who does it?

- **Software engineers** build and support software, and virtually everyone in the industrialized world uses it either directly or indirectly



Why is it important?

Software is important because affects nearly every aspect of our lives and has become pervasive in our commerce, our culture, and our everyday activities

Software engineering is important because it enables to build complex systems in a timely manner with high quality

What are the steps?



You build computer software like you build any successful product, by applying an **agile, adaptable process** that leads to a high-quality result that **meets the needs** of the people who will use the product.



You apply software engineering approach

What is the work product?

- Point of view of a software engineer: the set of programs, content (data), other work product that are computer software
- Point of view of users: the resultant information that somehow makes the user's world better





How do I
ensure
that I've
done it
right?

- Read the remainder of this book, select those ideas that are applicable to the software that you build, and apply them to your work

1.1 The Nature of Software

- Software takes on a dual role. It is a product, and at the same time, the vehicle for delivering a product.
- Software is an information transformer
- Software delivers the most important product of our time : Information

Old Questions



Why does it takes so long to get software finished?



Why are development cost 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?

Defining Software

- Instructions (computer programs) that when executed provide desired features, function, and performance
- Data structures that enable the programs to adequately manipulate information
- Descriptive information in both hard copy and virtual forms that describes the operation and use of the programs
- Those definitions won't measurably improve your understanding, it is better to examine the characteristics of software



Definición de Software

- Es el texto que se escribe con base a reglas de sintaxis de un lenguaje de programación, utilizando palabras reservadas que se refieren a acciones computacionales para el tratamiento de información, las cuales se ejecutan en forma secuencial utilizando un dispositivo informático, y permitiendo la realización de una función útil para el usuario o que satisface una necesidad tecnológica
- It is a set of computational actions that are written under the rules of a computational language and are performance to manage an electronic and digital dispositive
- Input Data
- Output Data
- Process Data
- Option Selection
- Loop

Characteristics of Software

- Software is a logical rather than a physical system element, and has characteristics that are considerably different than those of hardware (fig.1.1)

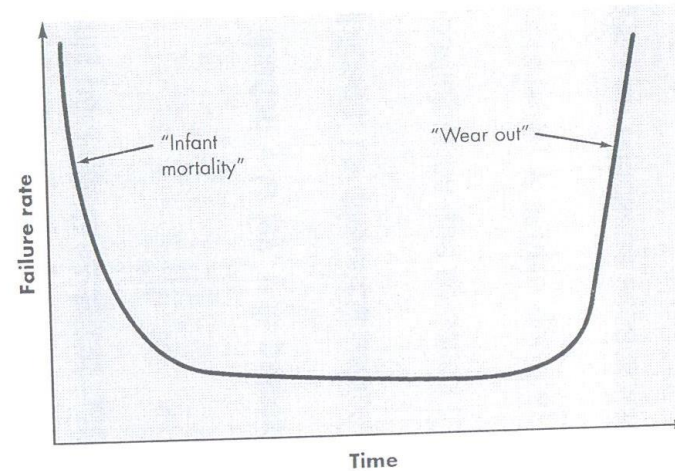
FIGURE 1.1Failure curve
for hardware

Fig. 1.1
pag.5

Software is
developed
or
engineered;

It is not a manufactured in
the classical sense

Software costs are
concentrated in engineering

Software projects cannot be
managed as if they were
manufacturing projects

Software doesn't “wear out”

- Hardware begins to wear out as an effect of dust, vibration, abuse, temperature extreme, and many other environmental maladies.
- Software is not susceptible to the environmental maladies. Undiscovered defects will cause high failure rates early in the life of a program
- Software does deteriorate because during its life will undergo change. It is likely that errors will be introduced. (fig.1.2)
- Every software error indicates an error in design or in the process through which design was translate into machine executable code

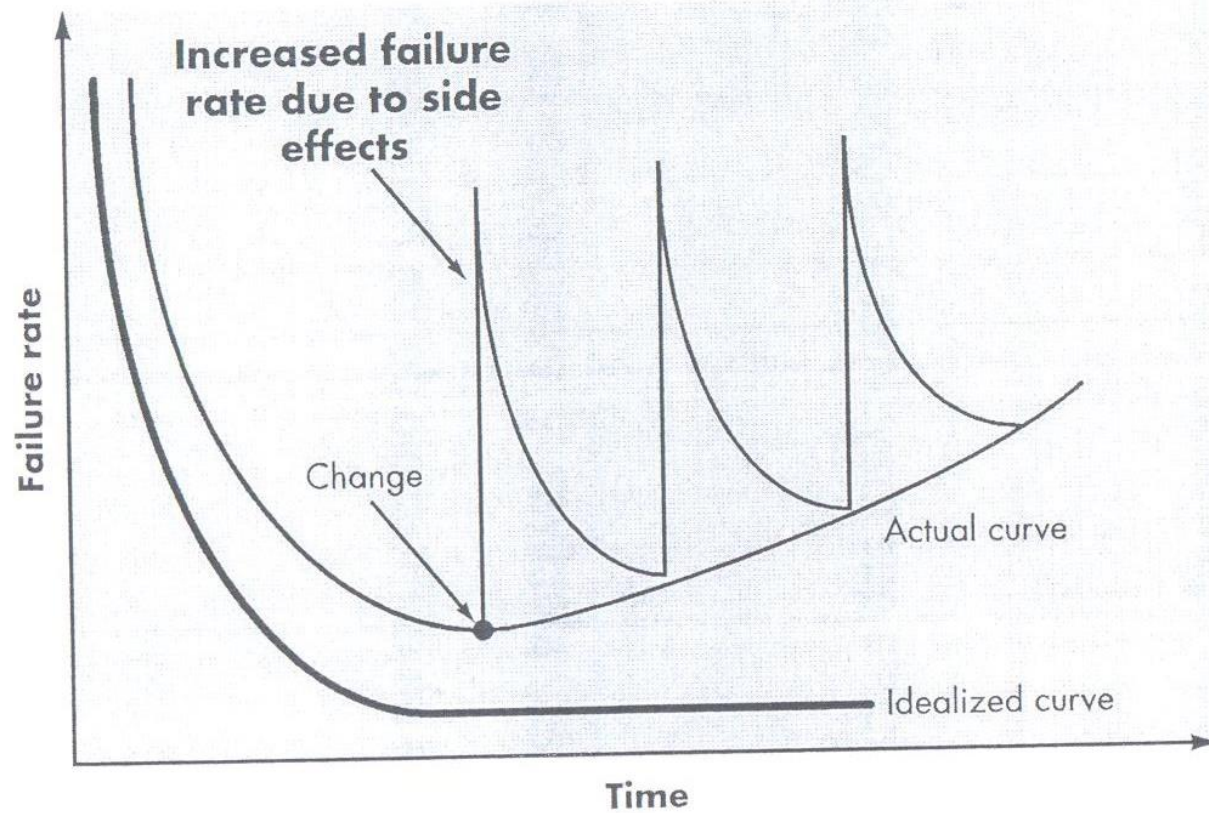


Fig. 1.2 page 6

Failure curves for
software

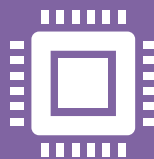
Component-based Industry



Although the industry is moving toward component-based construction, most software continues to be custom built



In the hardware world, component reuse is a natural part of the engineering process.



A software component should be designed and implemented so that it can be reused in many different programs

Software Application Domains Categories of computer software

System Software

Application software

Engineering/scientific software

Embedded software

Product line software

Web applications

Artificial intelligence software

System Software



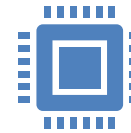
A collection of programs written to service other programs.



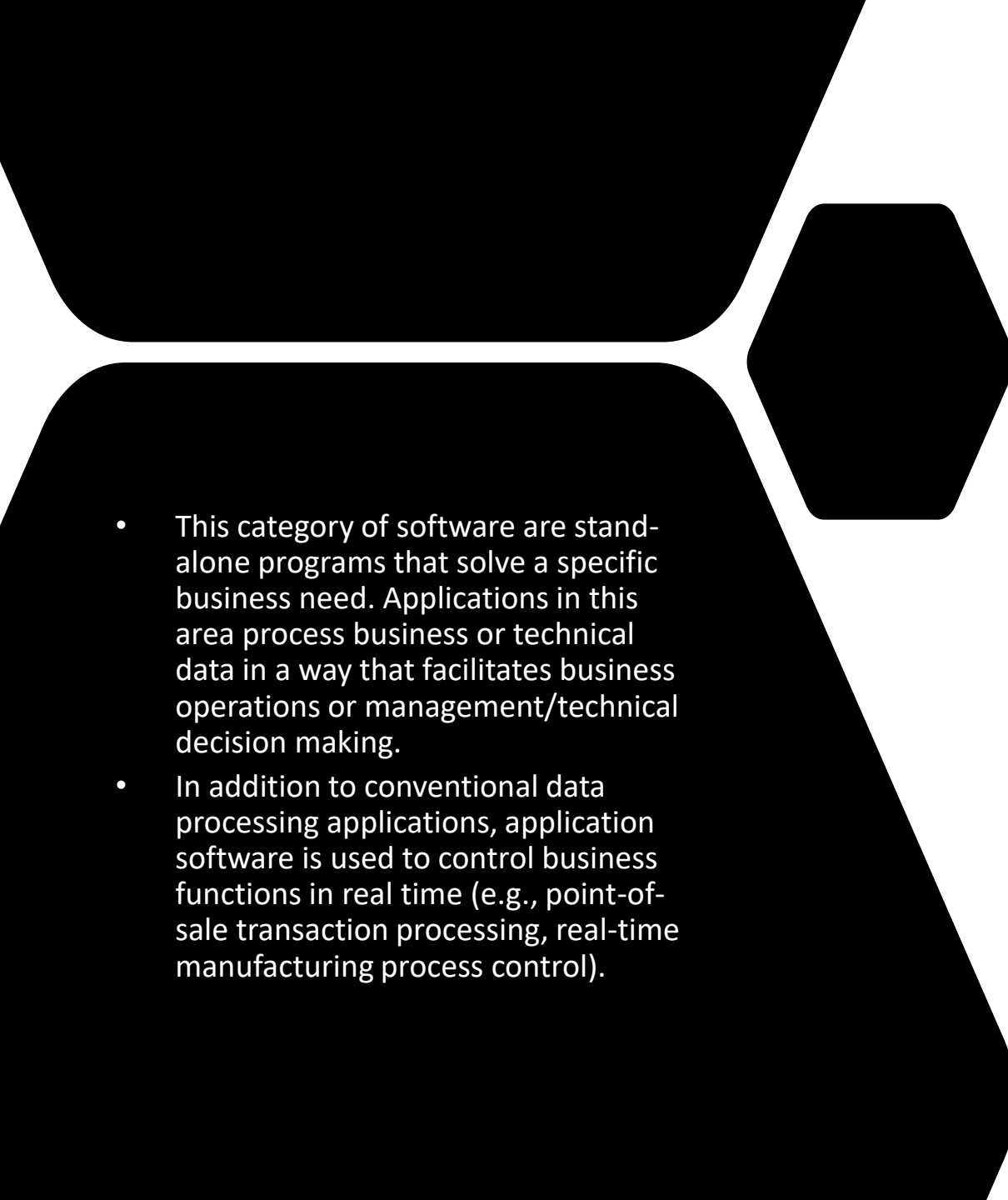
Compilers, editors, and file management utilities processes complex, but determinate, information structures.



Operating system components, drivers, networking software, telecommunications processors, process largely indeterminate data.

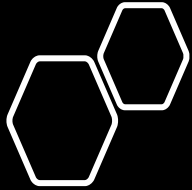


This software is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces



Application software

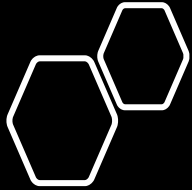
- This category of software are stand-alone programs that solve a specific business need. Applications in this area process business or technical data in a way that facilitates business operations or management/technical decision making.
- In addition to conventional data processing applications, application software is used to control business functions in real time (e.g., point-of-sale transaction processing, real-time manufacturing process control).



Engineering/scientific software

This software has been characterized by “number crunching” algorithms. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

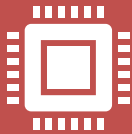
The engineering/scientific area are moving away from conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics



Embedded software

- This 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.
- Embedded software can perform limited and esoteric functions (e.g., key pad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems)

Product line software



This software is designed to provide a specific capability for use by many different customers.



Product-line software can focus on a limited and esoteric marketplace (e.g., inventory control products) or address mass consumer markets (e.g., word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, and personal and business financial applications).

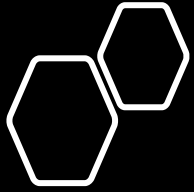
Web applications

- called “WebApps,” this network-centric software category spans a wide array of applications. In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics.
- As Web 2.0 emerges, WebApps are evolving into sophisticated computing environments that not only provide stand-alone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications



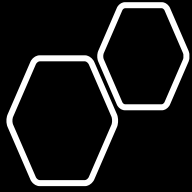
Artificial intelligence software

- This 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), artificial neural networks, theorem proving, and game playing.



Challenges in the horizon

- **Open-world computing:** the rapid growth of wireless networking may soon lead to true pervasive, distributed computing. The challenge for software engineers will be to develop systems and application software that will allow mobile devices, personal computers, and enterprise systems to communicate across vast networks.
- **Netsourcing:** the World Wide Web is rapidly becoming a computing engine as well as a content provider. The challenge for software engineers is to architect simple (e.g., personal financial planning) and sophisticated applications that provide a benefit to targeted end-user markets worldwide
- **Open source:** a growing trend that results in distribution of source code for systems applications (e.g., operating systems, database, and development environments) so that many people can contribute to its development. The challenge for software engineers is to build source code that is self-descriptive, but more importantly, to develop techniques that will enable both customers and developers to know what changes have been made and how those changes manifest themselves within the software



Legacy software

- 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
- An additional characteristic: poor quality
- The goal of modern software engineering is to “devise methodologies that are founded on the notion of evolution”; that is, the notion that software systems continually change...

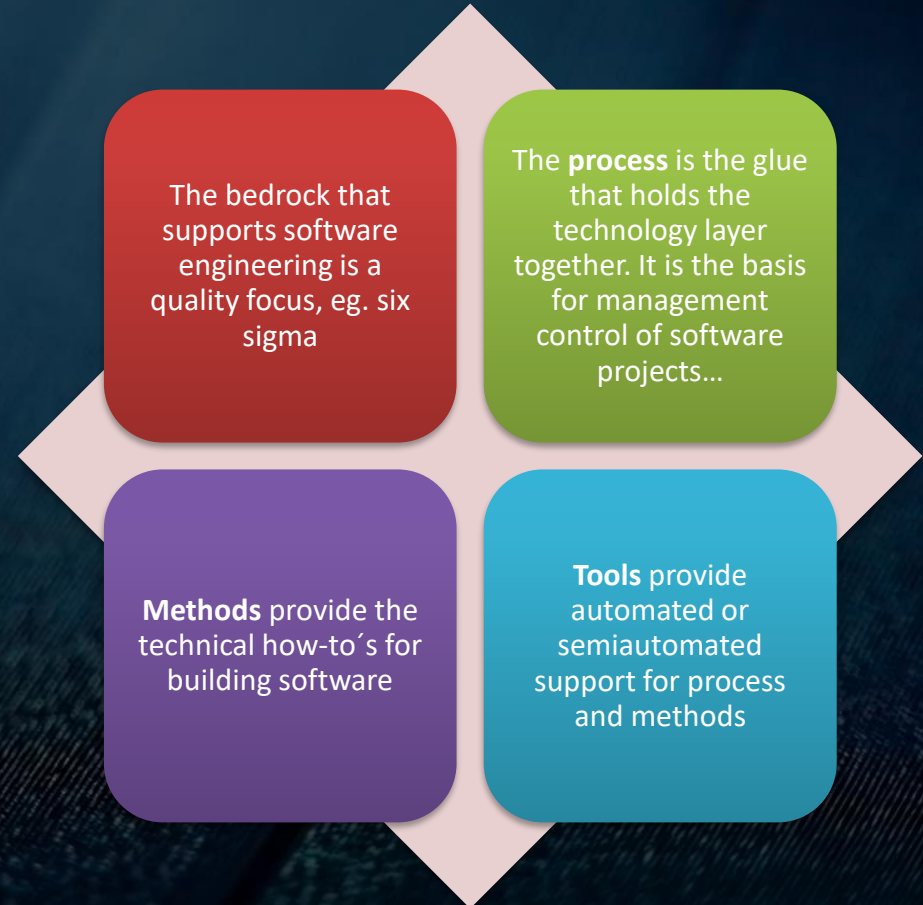
1.2 The Unique Nature of WebApps

- Network intensiveness
 - Concurrency
 - Unpredictable load
 - Performance
 - Availability
 - Data driven
-
- Content sensitive
 - Continuous evolution
 - Immediacy
 - Security
 - Aesthetics

1.3 Software Engineering

- Software has become deeply embedded in virtually every aspects of our lives
- The information technology requirements demands by individuals, business, and government grow increasing complex with each passing year.
- Individuals, business, and government increasingly rely on software for strategic and tactical decision as well as day-to-day operations and control
- As the perceived value of a specific applications grows, the likelihood is that its user base and longevity will also grow
- Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines

Software Engineering layers



Ingeniería del Software

- Proceso
 - Comunicación
 - Planeación
 - Modelado
 - Construcción
 - Despliegue

Métodos

- Diagrama Entidad-Relación
- Diagrama de Flujo
- Diagrama de Flujo de datos
- Diccionario de Datos
- Inglés Estructurado
- UML

Herramientas

- IDE's
- Compiladores

Software Engineering Definition

- IEEE definition for Software Engineering is: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Layered Technology

- Software engineering is a layered technology. Referring to Figure 1.3, any engineering approach (including software engineering) must rest on an organizational commitment to quality.

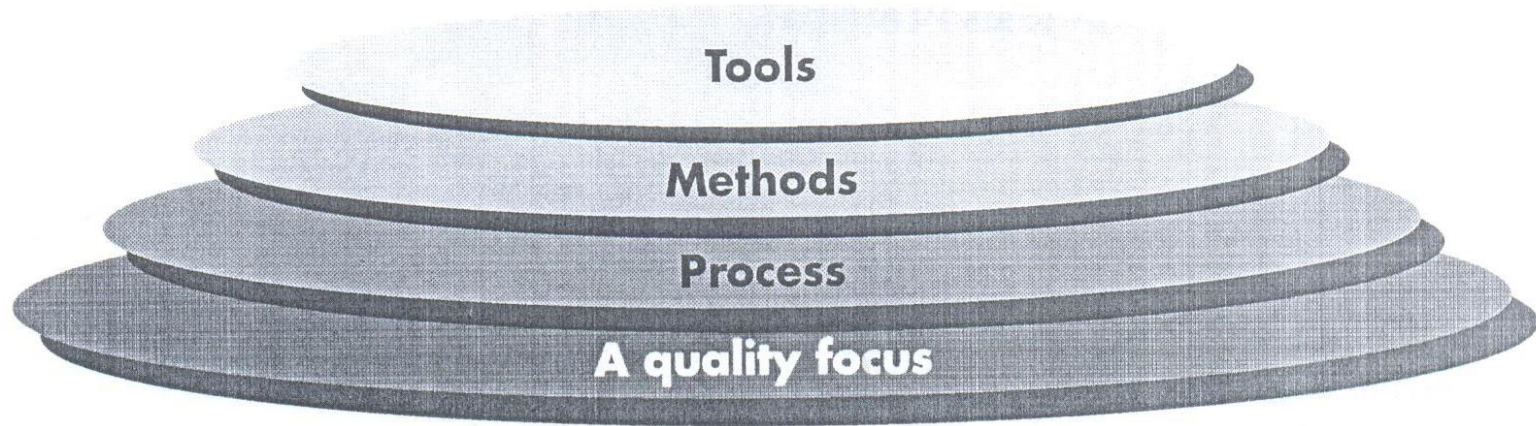


Fig. 1.3 page 14

Software engineering layers

1.4 The Software Process

- A process is a collection of **activities**, **actions**, and **tasks** that are performed when some work product is to be created
- An **activity** strives to achieve a broad objective (e.g., communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which software engineering is to be applied.
- An **action** (e.g., architectural design) encompasses a set of tasks that produce a major work product (e.g., an architectural design model).
- A **task** focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.

Una descripción gráfica

EL PROCESO DE SOFTWARE



The Software Process as a prescription

- The SP is not a rigid prescription for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work (the software team) to pick and choose the appropriate set of work actions and tasks. The intent is always to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Process Software as a glue

- The software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software
- The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed



The Process framework

Communication

Planning

Modeling

Construction

Deployment

Communication

- Before any technical work can commence, it is critically important to communicate and collaborate with the customer (and other stakeholders). The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions

Planning

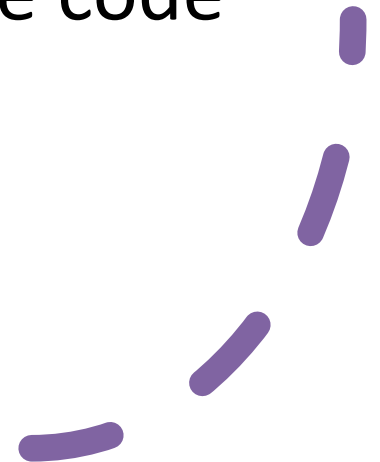
- Any complicated journey can be simplified if a map exists. A software project is a complicated journey, and the planning activity creates a “map” that helps guide the team as it makes the journey.
- The map—called a *software project plan*—*defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a workschedule.*

Modeling

- Whether you're a landscaper, a bridge builder, an aeronautical engineer, a carpenter, or an architect, you work with models every day. You create a “sketch” of the thing so that you'll understand the big picture—what it will look like architecturally, how the constituent parts fit together, and many other characteristics. If required, you refine the sketch into greater and greater detail in an effort to better understand the problem and how you're going to solve it.
- A software engineer does the same thing by creating models to better understand software requirements and the design that will achieve those requirements.

Construction

- This activity combines code generation (either manual or automated) and the testing that is required to uncover errors in the code





Deployment

- The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation

Umbrella activities

They help a software team manage and control progress, quality change, and risk

Software project tracking and control

Risk management

Software quality assurance

Technical reviews

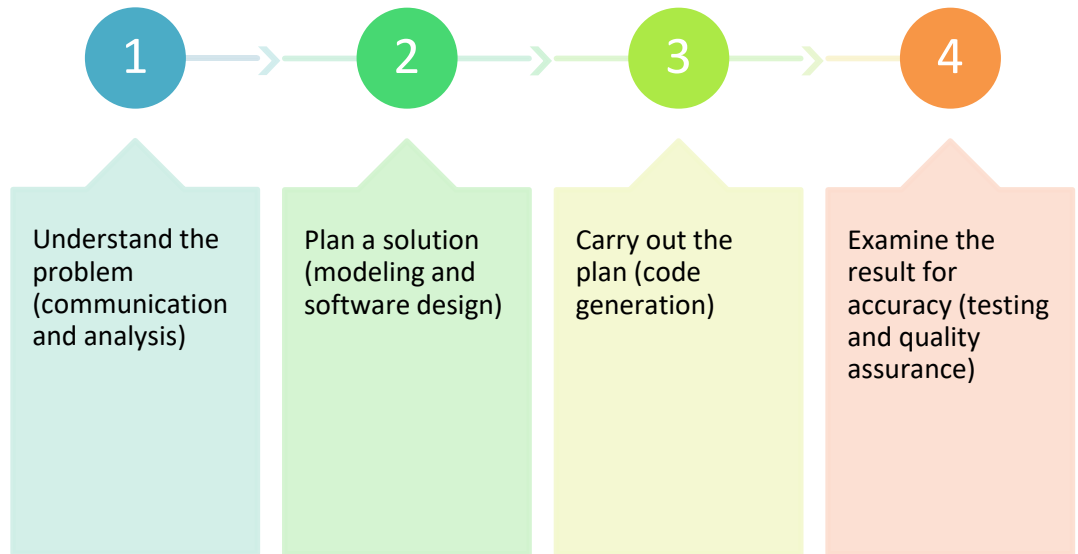
Measurement

Software configuration management

Reusability management

Work product preparation and production

1.5 The Software Engineering Practice



Understand the problem

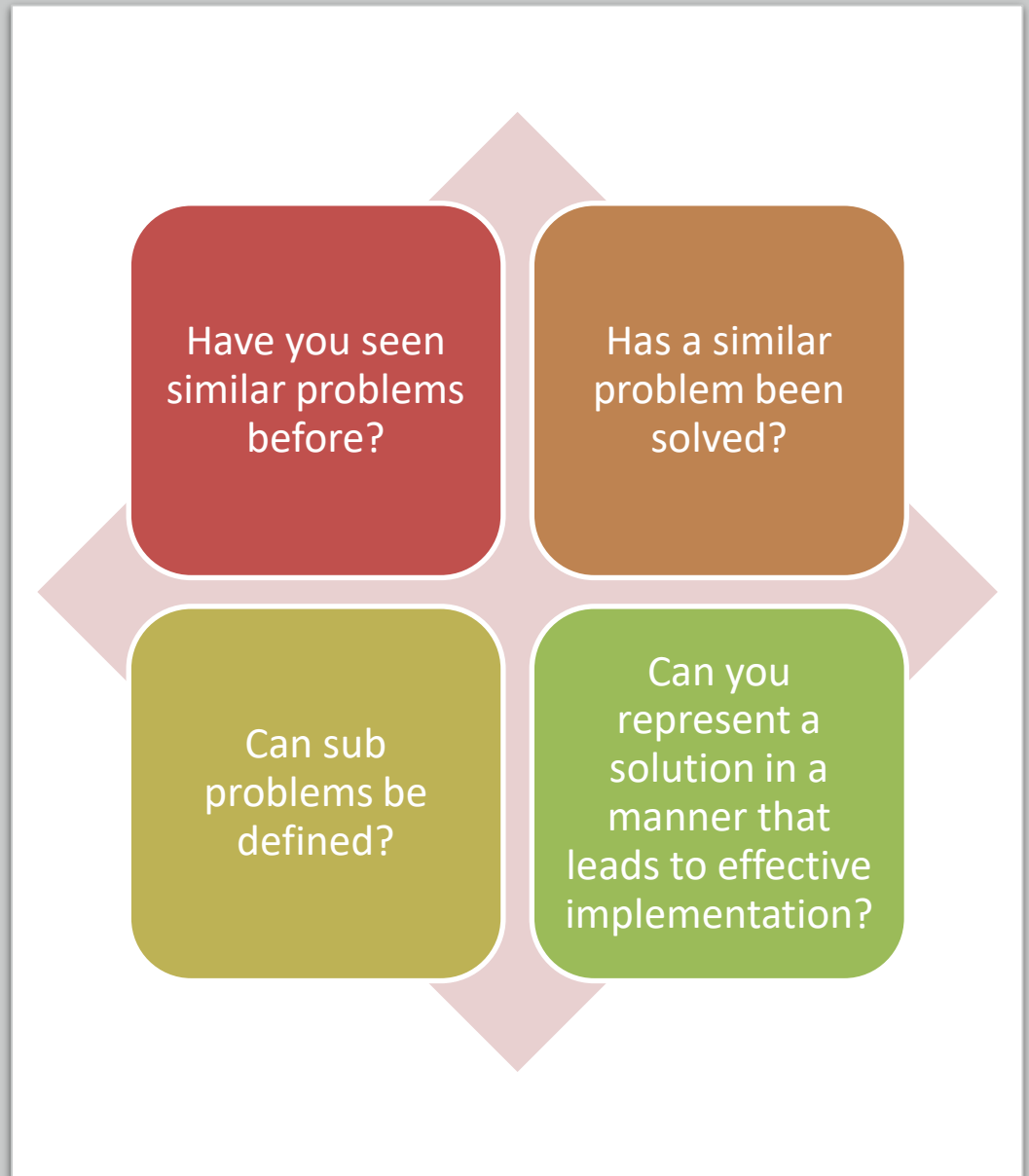
Who has a stake in the solution to the problem?

What are the unknowns?

Can the problem be compartmentalized?

Can the problem be represented graphically?

Plan the solution



Carry out the plan



DOES THE SOLUTION CONFORM TO
THE PLAN?



IS EACH COMPONENT PART OF THE
SOLUTION PROBABLY CORRECT?

Examine the result

Is it possible to test each component part of the solution?

Does the solution produce results that conform to the data, functions, and features that are required?

General Principles

The reason it all exists (provide value to its users)

KISS (Keep It Simple, Stupid!)

Maintain the Vision

What You Produce, Others Will Consume

Be Open to the Future

Plan Ahead for Reuse

Think!

Software Myths

Software myths are erroneous beliefs about software and the process that is used to build it. They have a number of attributes that make them insidious. For instance, they appear to be reasonable statements of fact (sometimes containing elements of truth), they have an intuitive feel, and they are often promulgated by experienced practitioners who “know the score.”

Management myths

Customer myths

Practitioner's myths

1.7 How It All Starts

- Every software project is precipitated by some business need
- The need to correct a defect in an existing application; the need to adapt a “legacy system” to a changing business environment; the need to extend the functions and features of an existing application; or the need to create a new product, service, or system