IV. SYSTEMS & SOFTWARE ENGINEERING PROCESSES

IN THE COMPUTER WORLD, HARDWARE IS ANYTHING YOU CAN HIT WITH A HAMMER, SOFTWARE IS WHAT YOU CAN ONLY CURSE AT.

SOURCE UNKNOWN

Introduction

Systems and Software Engineering Processes are presented in the following topic areas:

- **Life Cycles and Process Models**
- **Systems Architecture**
- **Requirements Engineering**
- **Requirements Management**
- **Software Analysis, Design, and Development**
- **Maintenance Management**

Life Cycles and Process Models

The software process is a set of tools, methods, and practices used to produce a software product.

In general, a software engineering process can be described using three generic phases: definition, development, and maintenance. These phases are encountered in all software work, regardless of application area, project size, or complexity.

Life Cycles and Process Models (Cont'd)

Definition Phase

The definition phase focuses on "what?" During the definition phase, the software developer and customer attempt to identify the following "what" questions:

- **What information is to be processed?**
- **What functions and performances are desired?**
- **What interfaces are to be established?**
- **What design constraints exist?**
- **What validation criteria are required?**

By answering these what questions, the key requirements of the software and the system are defined.

Life Cycles and Process Models (Cont'd)

Definition Phase (Continued)

Although the methods applied during the definition phase will vary, three specific steps will occur in some form:

1. **Customer contact - This is a research and/or consultation activity. Through customer contact (in research and development projects) or through industry or regulatory specifications, each element in a computer based system is defined, and focus is established on the role that software will play in project and product implementation.**

2. **Project planning - Once the scope of the software has been established, risks are analyzed, resources are allocated, costs are estimated, and work assignments and schedules are defined.**

3. **Requirements analysis - The scope defined for the software provides direction, but a more detailed definition of the information, behavior, and function of the software is necessary before work can begin. This technical detail comes from a formal analysis of the requirements.**

Life Cycles and Process Models (Cont'd)

Development Phase

The development phase focuses on the "how": That is, during development, the software developer attempts to define "how":

- **Data structures are to be designed**
- **Software architectures are to be designed**
- **Procedural details are to be implemented**
- **The design will be programmed**
- **The testing will be performed**

Life Cycles and Process Models (Cont'd)

Development Phase (Continued)

The methods applied during the development phase will vary, but three specific steps will occur in some form:

Design - Design translates the software requirements into a set of representations (graphical, tabular, or language-based) that describe the data structure, architecture, and algorithmic procedures, as well as the nature of the human-computer interface.

Coding - Design representations must be translated into a programming language. The language may be a conventional programming language or a nonprocedural language (e. g., a language generated by a case tool) that results in instructions that can be executed by a computer.

Testing - The machine executable form must then be tested to uncover errors in function, logic, or implementation.

Life Cycles and Process Models (Cont'd)

Maintenance Phase

The maintenance phase focuses on change that is associated with:

- **Error correction**
- **Required adaptations**
- **Enhancements**

Four types of changes are encountered during the maintenance phase:

1. **Corrective - Using corrective maintenance techniques, changes are made to the software to correct defects.**

2. **Adaptive - Adaptive maintenance results in modifications to the software to accommodate changes to the external operation environment.**

3. **Perfective - Perfective maintenance extends the software beyond its original functional requirements.**

4. **Reengineering - Old software is reworked to improve performance.**

Related Quality Activities

A number of quality management activities take place in parallel within the software engineering process, as described below:

Quality assurance - Reviews are conducted to ensure that quality is maintained as each phase is completed. Documentation is also developed and controlled.

Configuration management - All information created as part of the definition, development, and maintenance phases should be uniquely identified and controlled.

Project monitoring - The software engineering process defines a set of milestones that provides an indication of progress. These must be monitored to ensure that scheduling and costs are under control.

Measurement - The software engineering processes and the products they generate should be measured. Both direct and indirect measures can be utilized.

Software Engineering Paradigms

Like every other engineering discipline, software engineering encompasses a set of proven technical methods that are applied within the context of a process framework. This framework often refers to a software engineering paradigm, which is chosen based on:

- **The nature of the project and application**
- **The methods and tools to be used**
- **The controls and deliverables that are required**

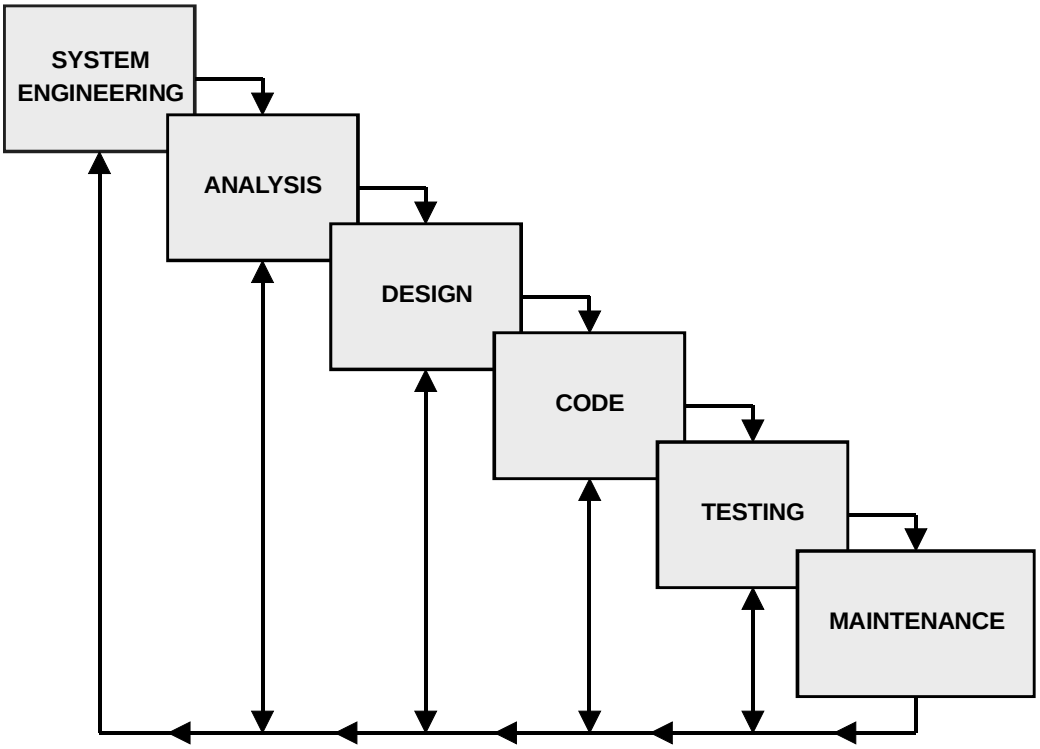
Because software engineering is still maturing, many different paradigms have been proposed, each of which is designed to support a different perspective of software development. A paradigm provides a basis for the creation of a work breakdown structure (WBS) for software engineering.

Waterfall Model

The oldest paradigm for software engineering is the classic waterfall model. This paradigm takes a linear, sequential view of the software engineering process. The name comes from its appearance when viewed in the following Figure.

The process begins at the system level and continues through the maintenance level until product retirement. Each box encompasses a set of tasks that result in the production of one or more work products. Time is tracked from left to right and top to bottom. Each new phase begins when the work products from the previous phase are completed, frozen, and signed off. The actual names and number of phases will often vary from project to project.

Waterfall Model (Continued)



The major features of the waterfall model follow:

System Engineering

System engineering and analysis encompasses requirements gathering at a system level with a small amount of top-level design and analysis.

Waterfall Model (Continued)

Analysis

In this phase, the requirements gathering process is intensified and focused specifically on software. The software engineer (or analyst) must understand the information domain for the software, as well as the required functions, behaviors, performances, and interfaces.

Design

Software design is actually a multi-step process that focuses on three distinct attributes of the program:

- **Data structures**
- **Software architecture**
- **Procedural details**

The design process translates requirements into a representation of the software.

Waterfall Model (Continued)

Coding

The design must be translated into code readable form. The coding step performs this task.

Testing

Once code has been generated, program testing begins. Testing focuses on the logical internals and the functional externals of the software.

Maintenance

Software will undoubtedly undergo change after it has been delivered to the customer. Change will occur because errors have been encountered and new functionality is required.

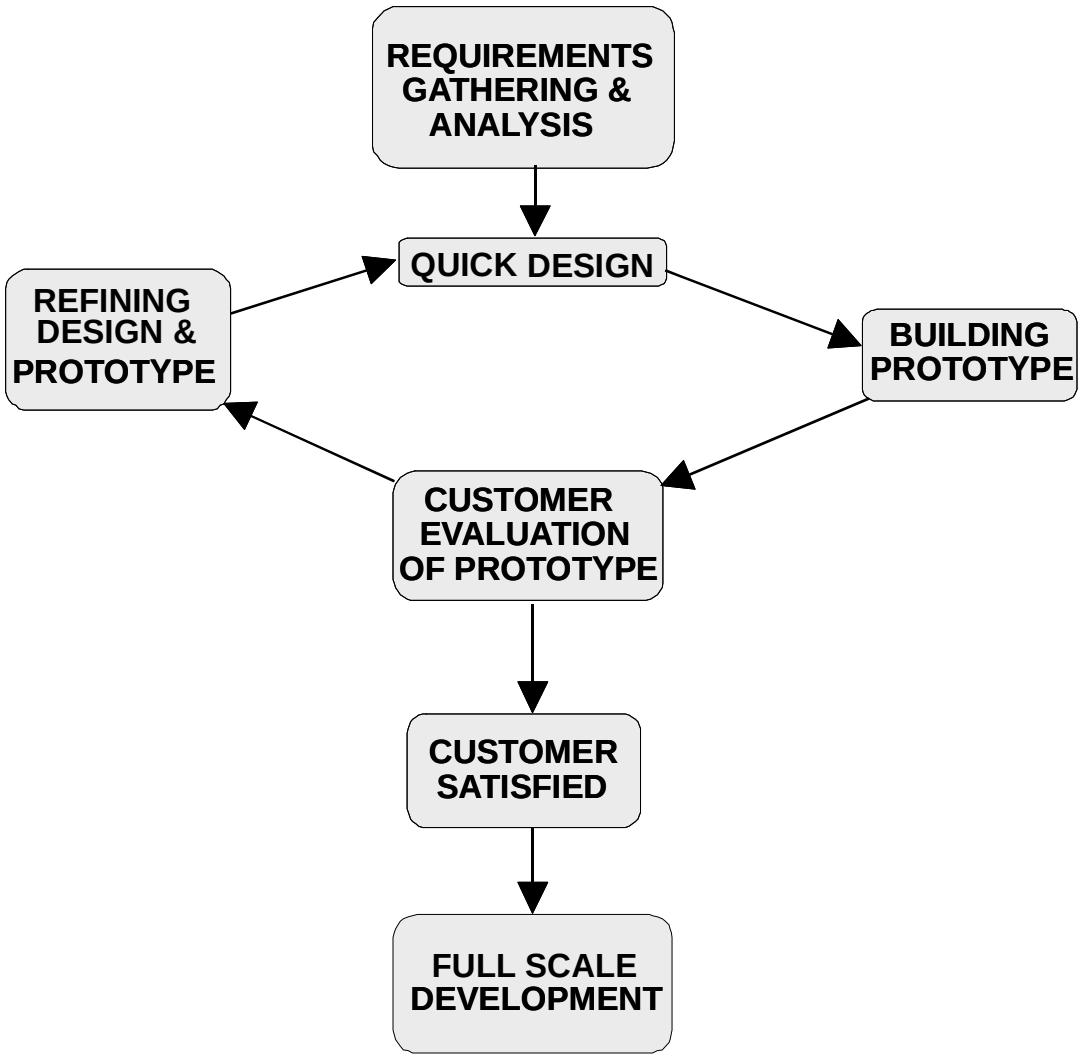
Waterfall Advantages/ Disadvantages

The classic waterfall model has an important place in software engineering. Although it is rarely practiced in pure form, it provides a template into which methods, for analysis, design, coding, testing, and maintenance can be placed. Disadvantages of the sequential waterfall approach are:

- **Real projects rarely follow this sequential flow. In real life, iterations occur which create problems in application of the paradigm.**
- **It is often difficult for a customer to explicitly state the requirements.**
- **A working version of the software is not available until late in the project. A major defect, if undetected until system test, will cause losses and delays to the project.**

Prototyping

Prototyping is a process that enables the developer to create a model of the software which is built in an evolutionary manner. A typical sequence of events for prototyping software is illustrated in the Figure below.



Prototyping (Continued)

The prototype may either be built on the target environment or on a convenient platform. Either way, the prototype is evaluated by the customer/user and is used to refine the requirements.

If the prototype is built on the target platform, the prototyping naturally progresses until the prototype becomes complete enough that the user can begin using it. Additional work is usually required to stabilize and complete the product.

If the prototype is built on a convenient platform, other than the system platform, the accepted prototype becomes the specification of the system. Full scale development is launched using the prototype as a detailed specification.

The advantages of the prototyping approach are that it provides a mechanism for identifying requirements and facilitating customer buy-in and agreement.

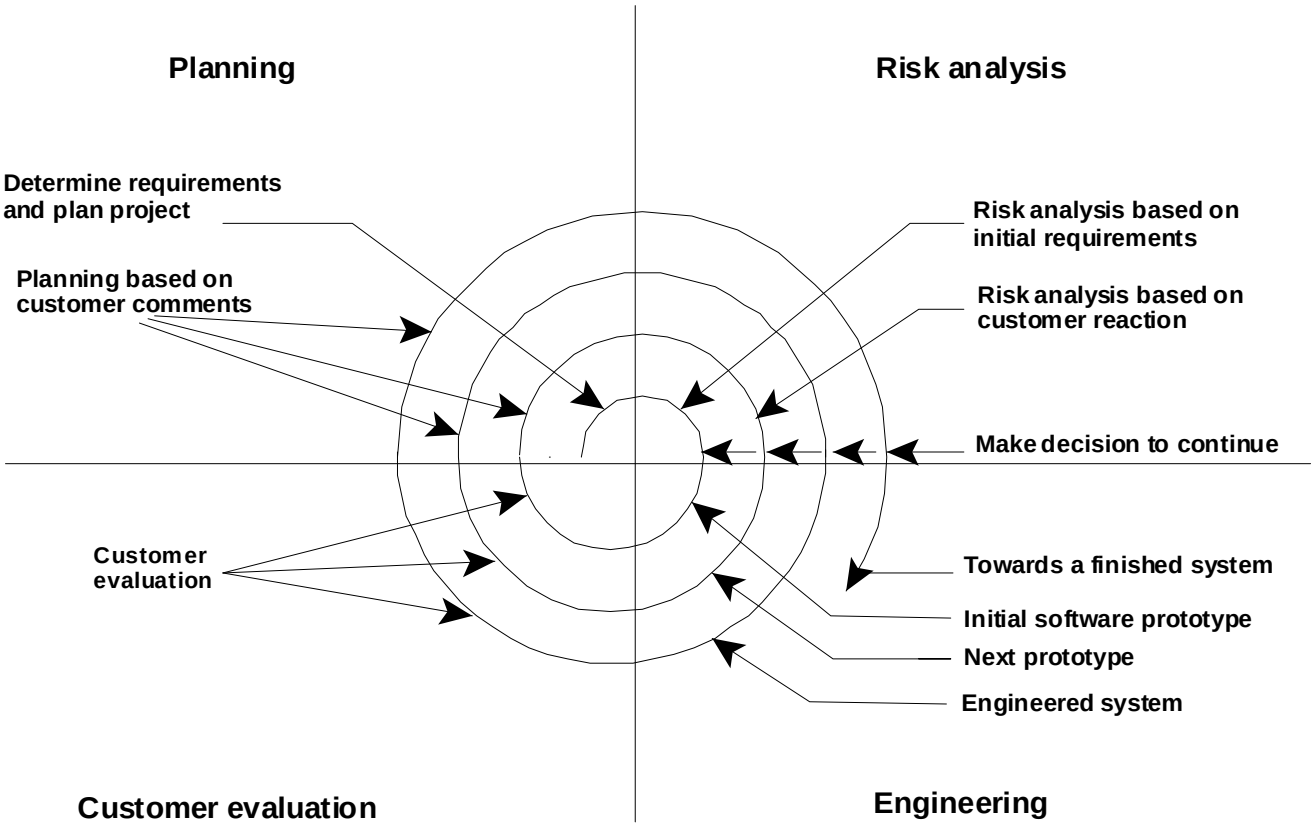
The disadvantages are that continued change tends to corrupt the software structure and that the prototype may provide false impressions to the customer of the availability of functionality.

Spiral Model

The spiral model is another type of evolutionary model. It has been developed to provide the best features of both the classic life cycle approach and prototyping, while at the same time adding the risk analysis element missing from each of these other paradigms. The model, represented by a spiral in the following Figure, defines four major activities represented by the four quadrants of the Figure:

1. **Planning - Determination of objectives, alternatives, and constraints**
2. **Risk analysis - Analysis of alternatives as well as an identification and/or resolution of risks**
3. **Engineering - Development of the next level of product**
4. **Customer evaluation - Assessment of the results of engineering**

Spiral Model (Continued)



With each iteration around the spiral, beginning at the center and working out, progressively more complete versions of the software are built.