Name: KATUMBIRE BOB BELLS

Reg.no: S20B23/219

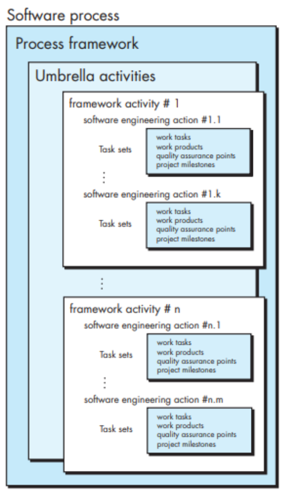
BSCS 3:2

# CHAPTER 3: SOFTWARE PROCESS 3STRUCTURE

Software process defines the approach that is taken for designing, implementing, and testing a software system. But software engineering also encompasses technologies that populate the process technical methods and automated tools.

## A GENERIC PROCESS MODEL

For a process, each of these activities, actions, and tasks reside within a framework or model that defines their relationship with the process and with one another. From the software process figure below, each framework activity is populated by a set of software engineering actions.



Each software engineering action is defined by a task set that I identify the work tasks that are to be completed , the work products, the quality assurance required, and the milestones that indicate progress.

A generic process framework defines five  framework activities: communication, planning, modeling, construction, and deployment.

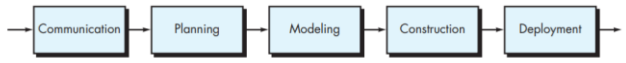
## DEFINING A FRAMEWORK ACTIVITY:

What actions are appropriate for a framework activity?

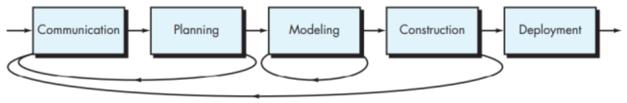
The nature of the problem to be solved, the characteristics of the people doing the work, and the stakeholders who are sponsoring the project helps to solve the problem.

Process flow: it describes how the frame work activities, actions and tasks are organized with respect to sequence and time.

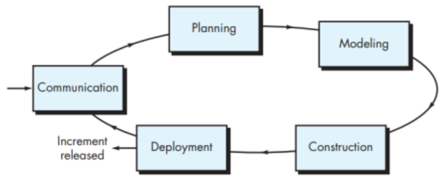
- Linear process flow:



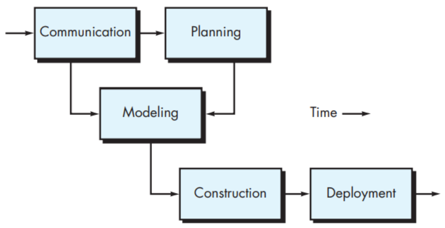
- Iterative process flow:



- Evolutionary process flow:



- Parallel process flow:



For a small software project requested by one person;

I. Contact stakeholder via phone.
II. Discuss requirements and develop notes.
III. Organize notes into a brief written statement of requirements.
IV. Email to stakeholder for review and approval.

For a complex project with many stakeholders, conflicting requirements, the communication activity might have  distinct actions; inception, elicitation, elaboration, negotiation, specification, and validation.

PROCESS PATTERNS:

It describes a process related problem that is encountered during software engineering work, identifies the environment in which the problem has been encountered, and suggests one or more proven solutions to the problem.

It also provides you with a template a, a consistent method for describing solutions within the context of the software process.

Types.

1. Stage pattern—defines a problem associated with a framework activity for the process.
2. Task pattern—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice (e.g., Requirements Gathering is a task pattern).
3. Phase pattern—define the sequence of framework activities that occurs within the process, even when the overall flow of activities is iterative in nature. An example of a phase pattern might be Spiral Model or Prototyping.

# Chapter 4: PROCESS  MODELS.

Process models were originally proposed to bring order to the chaos of software development.

a). PRESCRIPTIVE PROCESS MODELS: it strives for structure and order in software development. Activities and tasks occur sequentially with defined guidelines for progress.

- The Waterfall Model: suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software.
  problems that are sometimes encountered when the waterfall model is applied are:
  I. Real projects rarely follow the sequential flow that the model proposes.
  II. It is often difficult for the customer to state all requirements explicitly.
  III. A working version will not be available until in the project time span.

  A variation of the waterfall model can be; The V-model
- Incremental Process Models
  The incremental model combines the elements' linear and parallel process flows. The incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces deliverable "increments" of the software. The process flow for any increment can incorporate the prototyping paradigm.
  The process is repeated following the delivery of each increment until the complete product is produced.
- Evolutionary Process Models
  Evolutionary models are iterative. They are characterized in a manner that enables you to develop increasingly more complete versions of the software.
  1. Through Prototyping , often, a customer defines a set of general objectives for software but does not identify detailed requirements for functions and features. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take.
  2. We can also use the spiral model which is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model. It provides the potential for rapid development of increasingly more complete versions of the software.
- Concurrent Models
  The concurrent development model, sometimes called concurrent engineering, allows a software team to represent iterative and concurrent elements of any of the process models.
- A Final Word on Evolutionary Processes
  Evolutionary process models were conceived to address these issues, and yet, as a general class of process models, they too have weaknesses. Despite the unquestionable benefits of evolutionary software processes, we have some concerns.
  I. The first concern is that prototyping [and other more sophisticated evolutionary processes] poses a problem to project planning because of the uncertain number of cycles required to construct the product
  II. Second, evolutionary software processes do not establish the maximum speed of the evolution. If the evolutions occur too fast, without a period of relaxation,

it is certain that the process will fall into chaos. On the other hand, if the speed is too slow then productivity could be affected.

III. software processes should be focused on flexibility and extensibility rather than on high quality.

b). SPECIALIZED PROCESS MODELS

Specialized process models take on many of the characteristics of one or more of the traditional models presented in the preceding sections. This model is applied when a specialized or narrowly defined software engineering approach is chosen.

- Component-Based Development. Here we can use Commercial off-the-shelf (COTS) software components, developed by vendors.
  <u>Steps</u>
  I. Available component-based products are researched and evaluated for the application domain in question.
  II. Component integration issues are considered.
  III. A software architecture is designed to accommodate the components.
  IV. Components are integrated into the architecture.
  V. Comprehensive testing is conducted to ensure proper functionality.
- The Formal Methods Model. Formal methods enable you to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation. Variation on this method , called classroom software engineering is, is also used. It also eliminates many problems difficult to overcome using other software engineering paradigms.
- Aspect-Oriented Software Development

c). THE U NIFIED PROCESS

Unified Process is an attempt to draw on the best features and characteristics of traditional software process models, but characterize them in a way that implements many of the best principles of agile software development.

d). PERSONAL AND TEAM PROCESS MODELS

The team itself can create its own process, and at the same time meet the narrow needs of individuals and the boarder need of the organization. It is possible to create;

- Personal software process
- Team software process, but both hard work, training, and coordination,  and yet these are achievable

e). PROCESS TECHNOLOGY

Process technology tools allow a software organization to build an automated model of the process framework, task sets, and umbrella activities. The model, normally represented as a network, can then be analyzed to determine typical workflow, and examine alternative process structures that might lead to reduced development time or cost. Once an acceptable process has been created, other process technology tools can be used to allocate, monitor, and even control all software engineering activities, actions, and tasks defined as part of the process model.

# CHAPTER 5: AGILE DEVELOPMENT

Agile methods were developed in an effort to overcome perceived and actual weaknesses in convectional software engineering. It can provide benefits but not applicable to all products ,people and situations. Most compelling characteristics of agile is the ability to reduce the costs of change through the software process. But if process models are to work, they must provide a realistic mechanism for encouraging the discipline that is necessary.

## WHAT I S AGILITY?

Software engineers must be quick on their feet if they are to accommodate the rapid changes.

Agility encourages team structure and attitude that make communication more casual.

I. It emphasizes rapid delivery of optional software and deemphasizes the importance of intermediate work products.
II. It adopts the customer as a part of the development team and works o eliminate the "us and them" attitude.

Agility can be applied to any software process.

## WHAT I S AN AGILE PROCESS?

Any agile software process is characterized in a manner that addresses several key assumptions about many software projects:

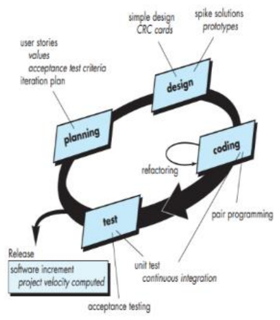
I. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds.
II. For many types of software, design and construction are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is diffi cult to predict how much design is necessary before construction is used to prove the design.
III. Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.

## EXTREME PROGRAMMING

Extreme programming (XP), is the most widely used approach to agile software development.

a). The XP Process

Extreme programming uses an object-oriented approach.



- Planning. The planning activity begins with listening, a requirement gathering activity that enables the technical members.
- Design. XP design rigorously follows the KIS principle. A simple design is always preferred over a more complex representation.
- Coding. After stories are developed and preliminary design work is done, the team does not move to code, but rather develops a series of unit tests that will exercise each of the stories that is to be included in the current release.
- Testing. The unit tests that are created should be implemented using a framework that enables them to be automated.

b). Industrial XP

IXP is an organic evolution of XP. It is imbued with XP's minimalist, customer-centric, test-driven spirit. IXP differs most from the original XP in its greater inclusion of management, its expanded role for customers, and its upgraded technical practices.

IXP incorporates six new practices designed to help that XP project works successfully for significant projects within a large organization:

I. Readiness assessment.
II. Project community.
III. Project chartering.
IV. Test-driven management.
V. Retrospectives.
VI. Continuous learning.

c). Scrum

d). Dynamic Systems Development Method

e). Agile Modeling

f). Agile Unifi ed Process

# CHAPTER 6: HUMAN ASPECTS OF SOFTWARE ENGINEERING

Software engineering has an abundance of techniques, tools, and methods designed to improve both the software development.

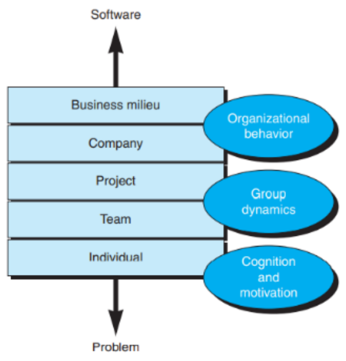
## a). CHARACTERISTICS OF A SOFTWARE E NGINEER

I. An effective software engineer has a sense of individual responsibility.

II. An effective software engineer has an acute awareness of the needs of other members of his team, of the stakeholders that have requested a software solution to an existing problem, and the managers who have overall control over the project that will achieve that solution.
III. An effective software engineer is brutally honest.
IV. An effective software engineer exhibits resilience under pressure.
V. An effective software engineer has a heightened sense of fairness.
VI. An effective software engineer exhibits attention to detail.
VII. An effective software engineer is pragmatic. She recognizes that software engineering is not a religion in which dogmatic rules must be followed, but rather a discipline that can be adapted based on the circumstances at hand.

b). THE PSYCHOLOGY OF SOFTWARE E NGINEERING

A layer's behavioral model for software engineering.



At an individual level, software engineering psychology focuses on recognition of the problem to be solved, the problem-solving skills required to solve it, and the motivation to complete the solution within the constraints established by outer layers in the model.

At the team level, teams often establish artificial boundaries that reduce communication and, as a consequence, reduce the team effectiveness.

c). The software teams.

d). Team structures

e). Agile teams

- The Generic Agile Team
- The XP Team

f). The impact of social media

g). Software e engineering using the cloud.

h). Collaboration tools.