Facultad de Ingeniería Mecánica y Eléctrica
Universidad Autónoma de Nuevo León

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

The Software Process

Chapter 2
Roger S. Pressman

What It is?

- When you work to build a product or system, It’s important to go through a series of predictable steps

Who does it?

- **Software engineers** and their managers adapt the process to their needs and then follow it
- The people who have requested the software have a role to play in the process of defining, building, and testing

Why is it important?

The process provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic

Modern process must be agile. We have to choose only those activities, controls, and work products that are appropriate for the project team and the product that is to be produced

What are the steps?

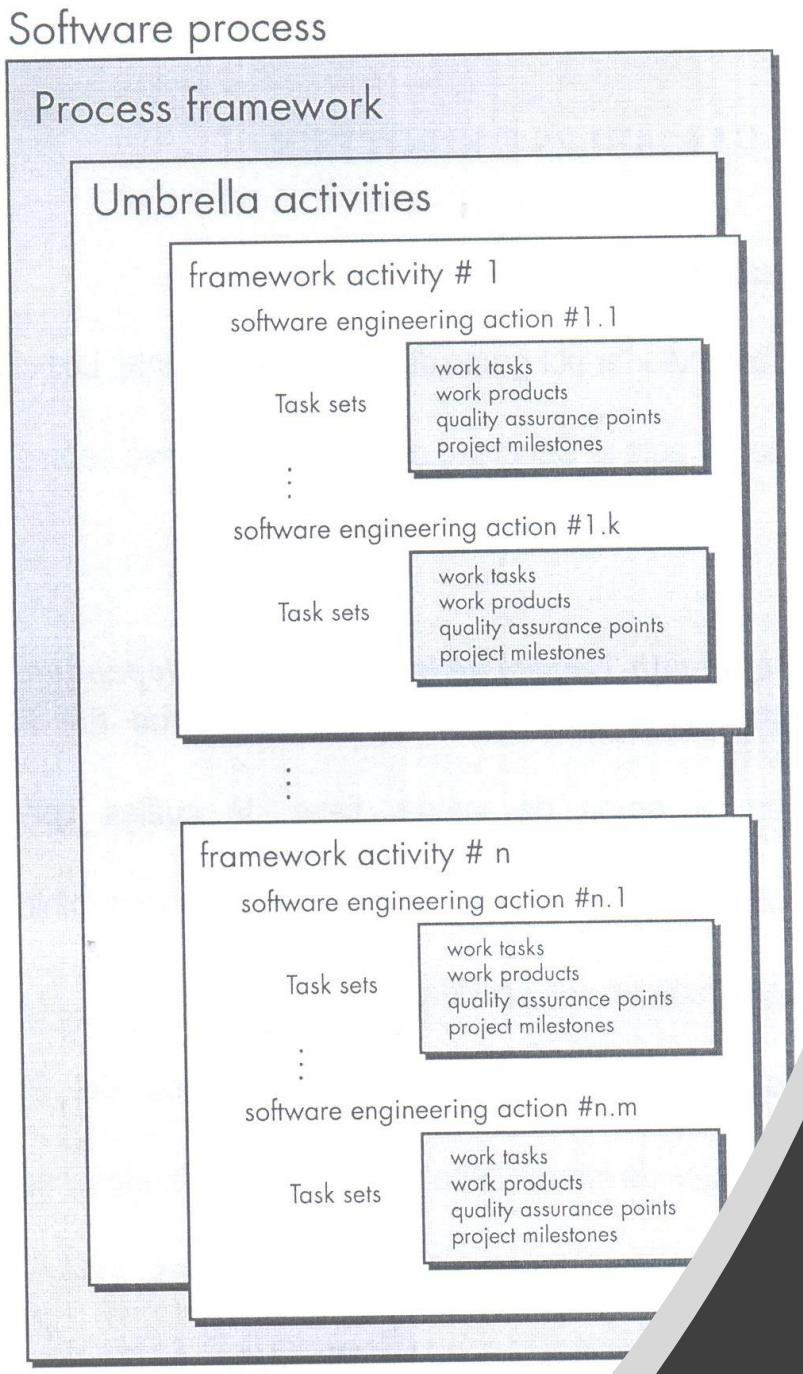
At a detailed level, the process that you adopt depends on the software that you're building

What is the work product?

- Point of view of a software engineer: the set of programs, documents, and data that are produced as a consequence of the activities and tasks defined by the process

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

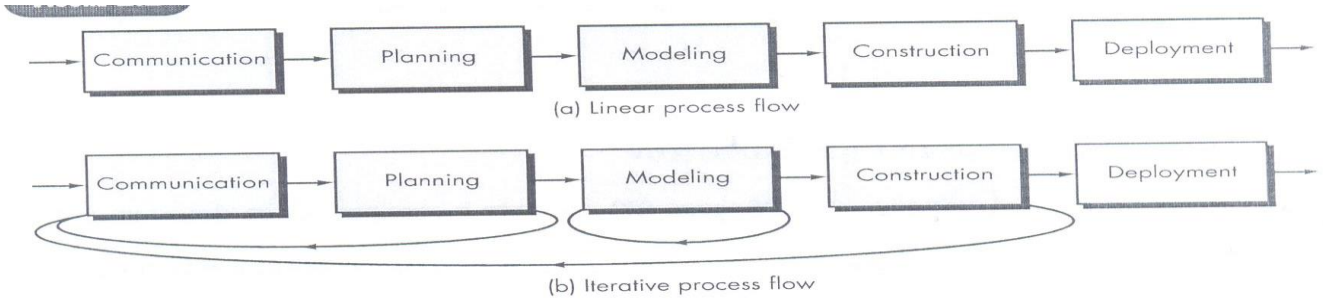
- There are a number of software process assessment mechanics that enable organizations to determine the ”maturity”of their software process
- Quality, timeliness, and long-term viability of the product you build are the best indicators of the efficacy of the software that you use

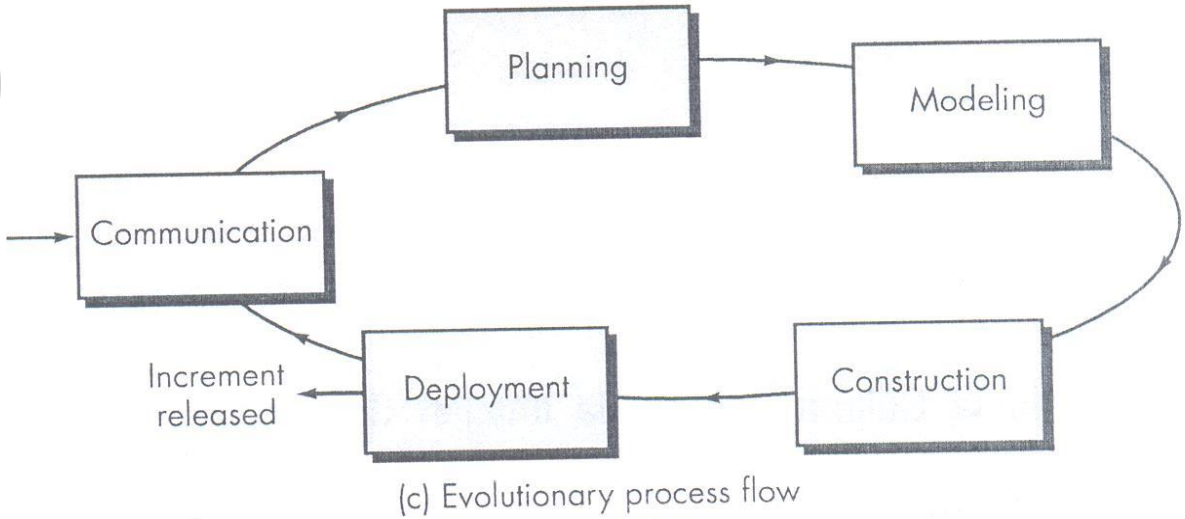


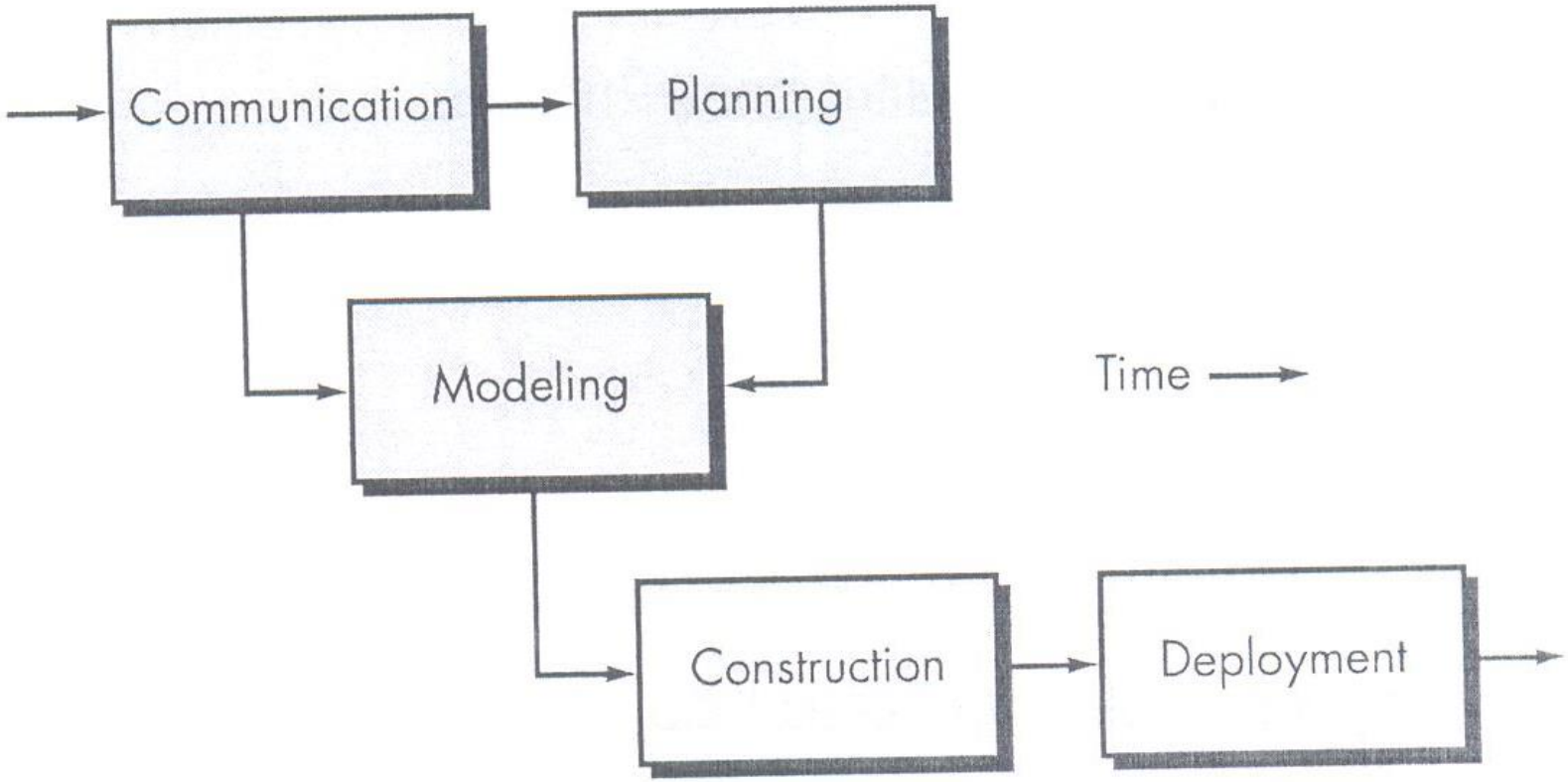
A Generic Process Model Fig 2.1 pag.32

A software process framework

Process Flow.
Fig 2.2 pag.33







(d) Parallel process flow

Process Flow. Fig 2.2 pag.33

Small project vs. Complex project

Small project

1. Make contact with stakeholder via telephone
2. Discuss requirements and take notes
3. Organize notes into a brief written statement of requirements
4. E-mail to stakeholder for review and approval

Complex project

- Communication Activity.
  Many stakeholders, each with different set of (sometime conflicting) requirements
- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation

2.1.2 Identifying a Task Set

- Each software engineering action can be represented by a number of different task sets – each a collection of SE work task, related work products, quality assurance, and project milestone.
- You should choose a task set that best accommodates the needs of the project and the characteristics of your team
- A SE action can be adapted to the specific needs of the software project and the characteristics of the project team

2.1.3 Process Patterns

- Chapter 12
- Pattern name
- Forces
- Type
  - Stage Pattern: e.g.Establising Communication
  - Task Pattern: e.g. Requirements Gathering
  - Phase Pattern: e.g. Spiral Modeling or Prototyping

- Initial context
- Problem
- Solution
- Resulting Context
- Related Patterns
- Know Uses and Examples

A Stage Pattern. (e.g. The Planning Pattern)

Customers and software engineers have established a collaborative communication

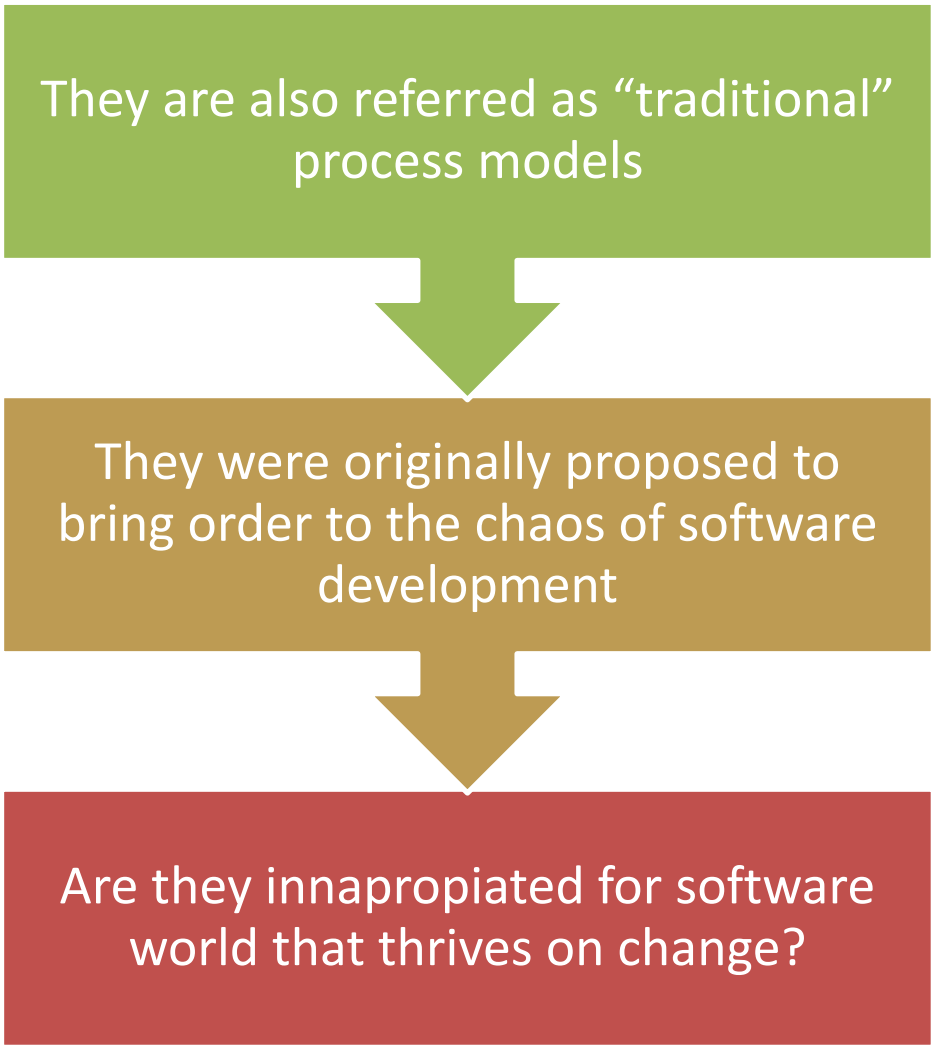
Successful completion of a number of taks patterns [specified] for the **Communication** has occurred

The project scope, basic business requirements, and project constraints are know

2.2 Process Assessment and Improvement

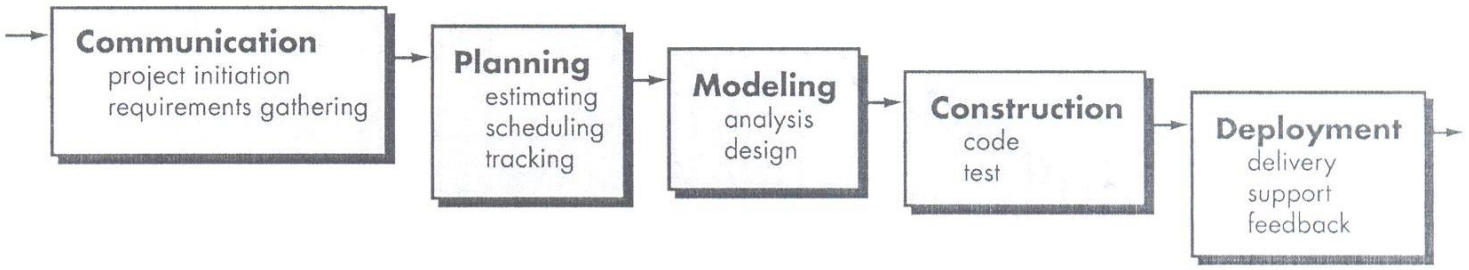
- Standard CMMI Assessment Method for Process Improvement (SCAMPI)
- CMM – Based Appraisal for Internal Process Improvement (CBA IPI)
- SPICE (ISO/IEC15504)
- ISO 9001:2000 for Software

2.3 Prescriptive Process Models



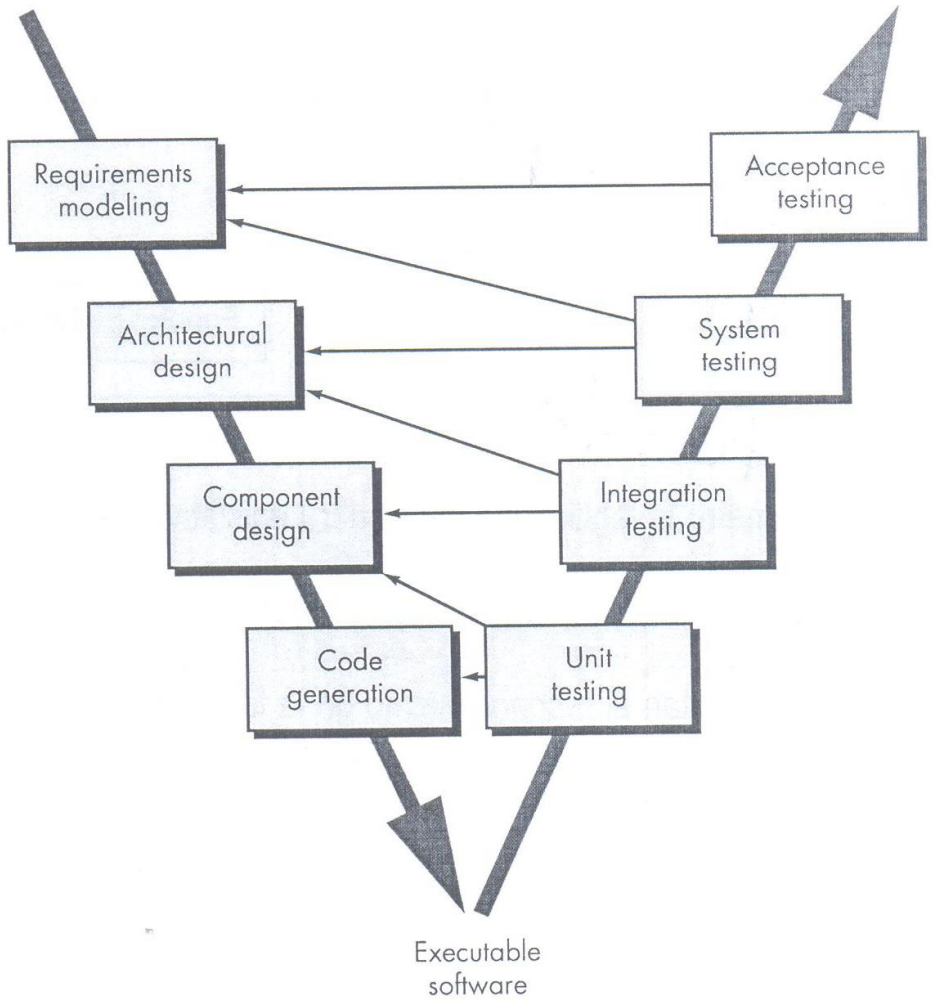
2.3.1 The Waterfall Model

- There are times when the requirements for a problem are well understood –when work flows from **communication** through **deployment** in a reasonably linear fashion
- The requirements are well defined and reasonably stable
- It is called classic life cycle (SDLC)



The Waterfall Model. Fig. 2.3 pag. 39

- It suggest a systematic sequential approach to software development that begins with customer specification of requirements and progress through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software



The V-model. Fig.2.4 p40

It is a variation in the representation of the water fall model.

It depicts the relation of quality assurance to the actions associated with communication, modeling, and early construction activities

Criticism of the Waterfall Model

- It is the oldest paradigm

1. Real problems rarely follow the sequential flow that the model proposes
2. It is often difficult for the customer to state al requirements explicitly
3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span

2.3.2 Incremental Process Models

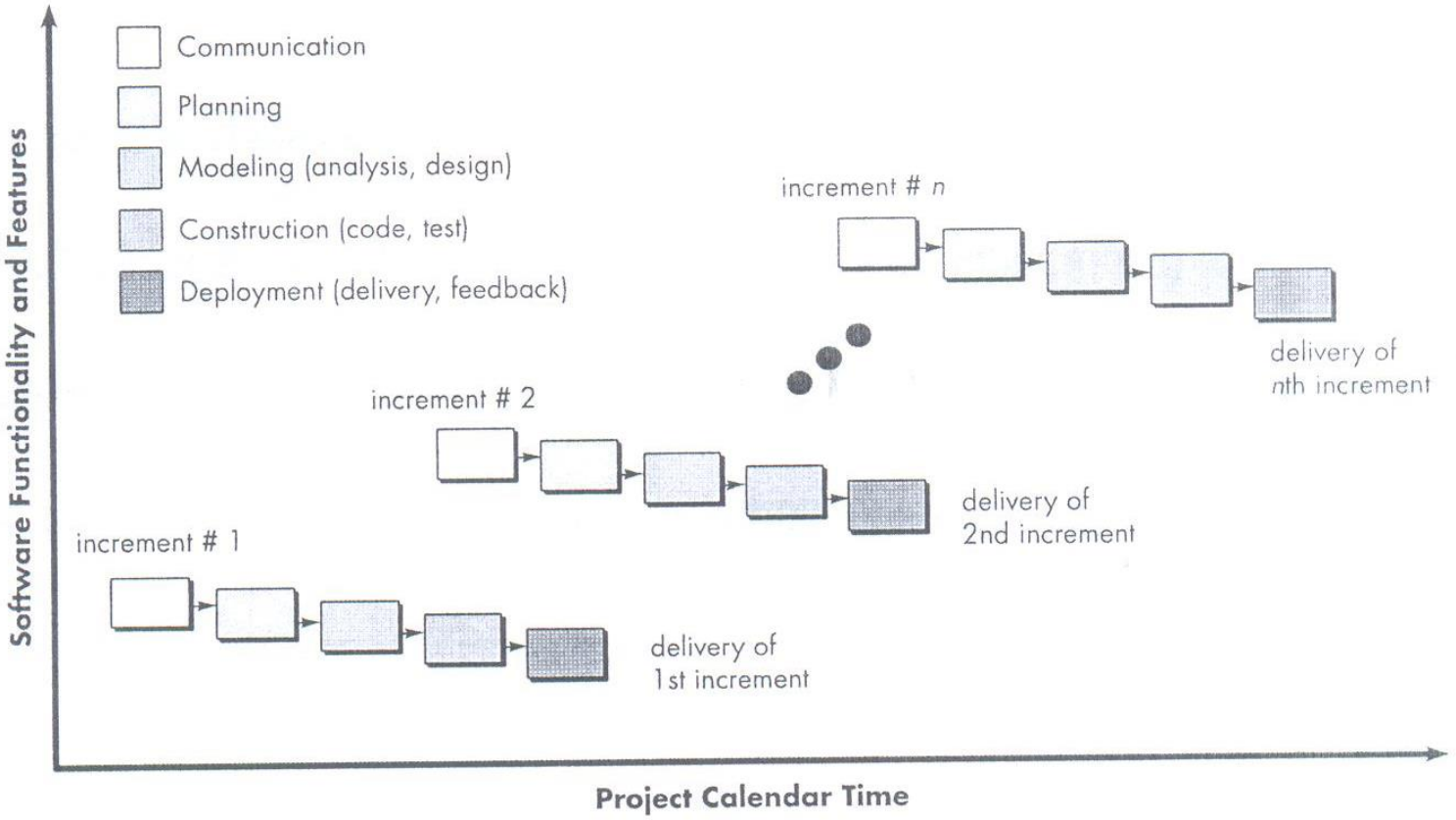
There are many situations in which initial software requirements are reasonably well defined, but the overall scope of the development effort precludes a purely linear

It combines elements of linear and parallel process flows

The first incremental is often a *core product*

It focused on the delivery of an operational product

It is useful when staffing is unavailable for complete implementation by business deadline that has been established for the project



The incremental model. Fig. 2.5 p 42

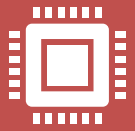
2.3.3 Evolutionary Process Model

Software evolves over a period of time. Business and products requirements change as development proceeds, making a straight line path to the end product unrealistic

Evolutionary models are iterative

Prototyping and The Spiral Model

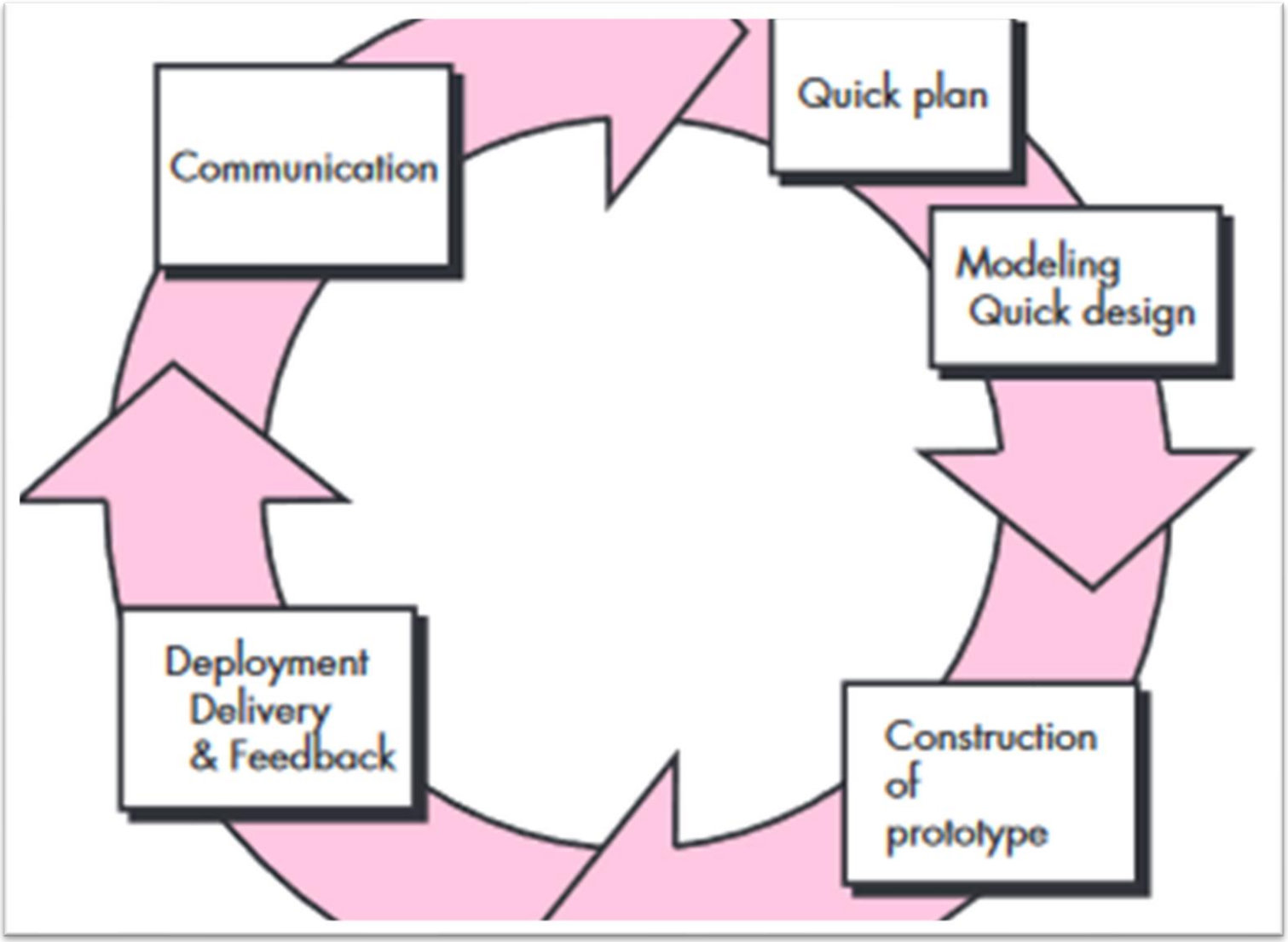
Prototyping



Often, a customer defines a set of general objectives for software, but does not detailed requirements for functions and features

It can be used as a stand-alone process model

It is more commonly used as a technique that can be implemented whitin the context of any one of the process models noted in this chapter



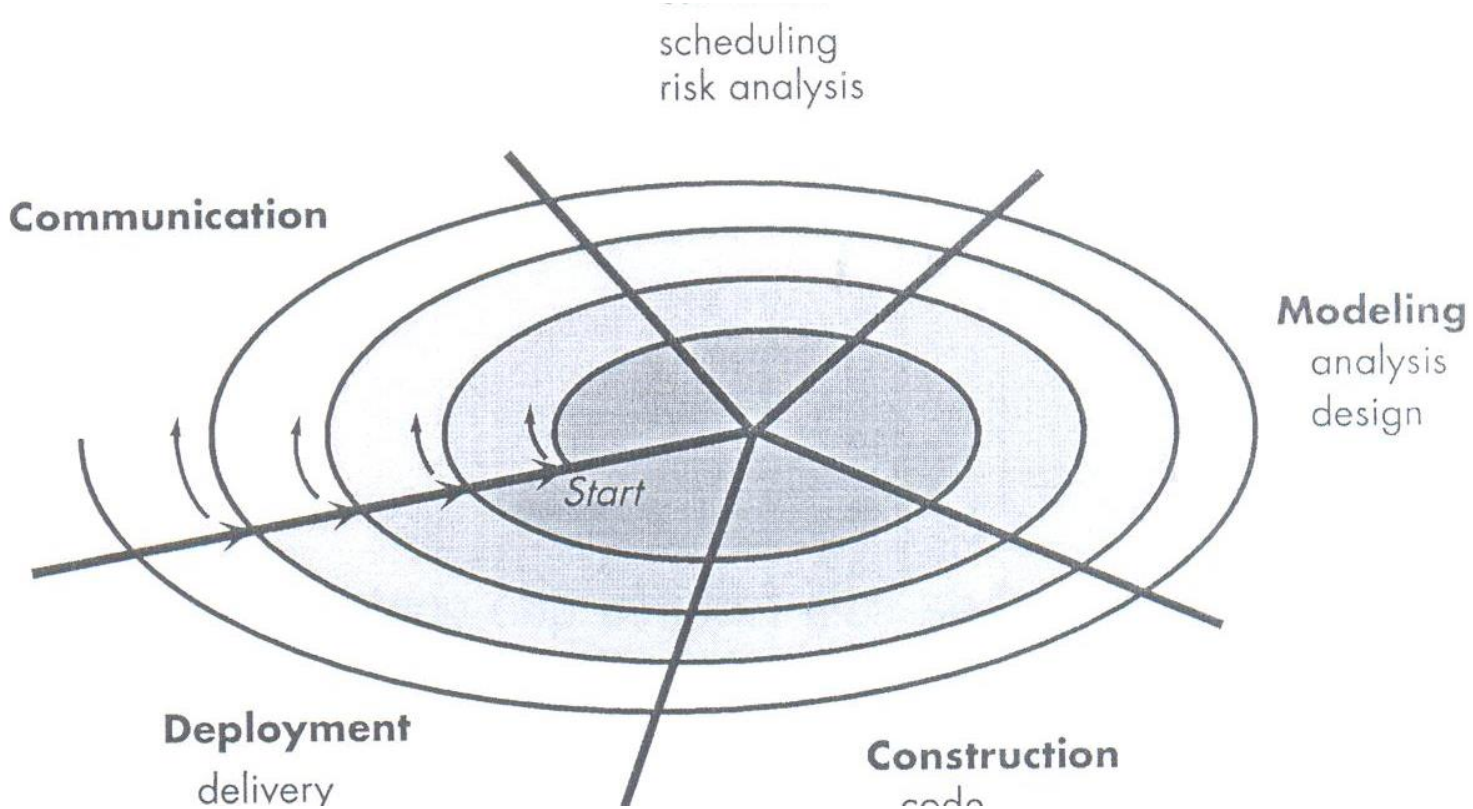
The prototyping paradigm. Fig. 2.6 p 43

Prototyping

- A quick design focused on a representation of those aspects of the software that will be visible to the end users (e.g., human interface or output display)
- Stakeholder provides feedback that is used to further requirements
- It serves as a mechanism for identifying requirements
- It can be as “the first system”

The Spiral Model

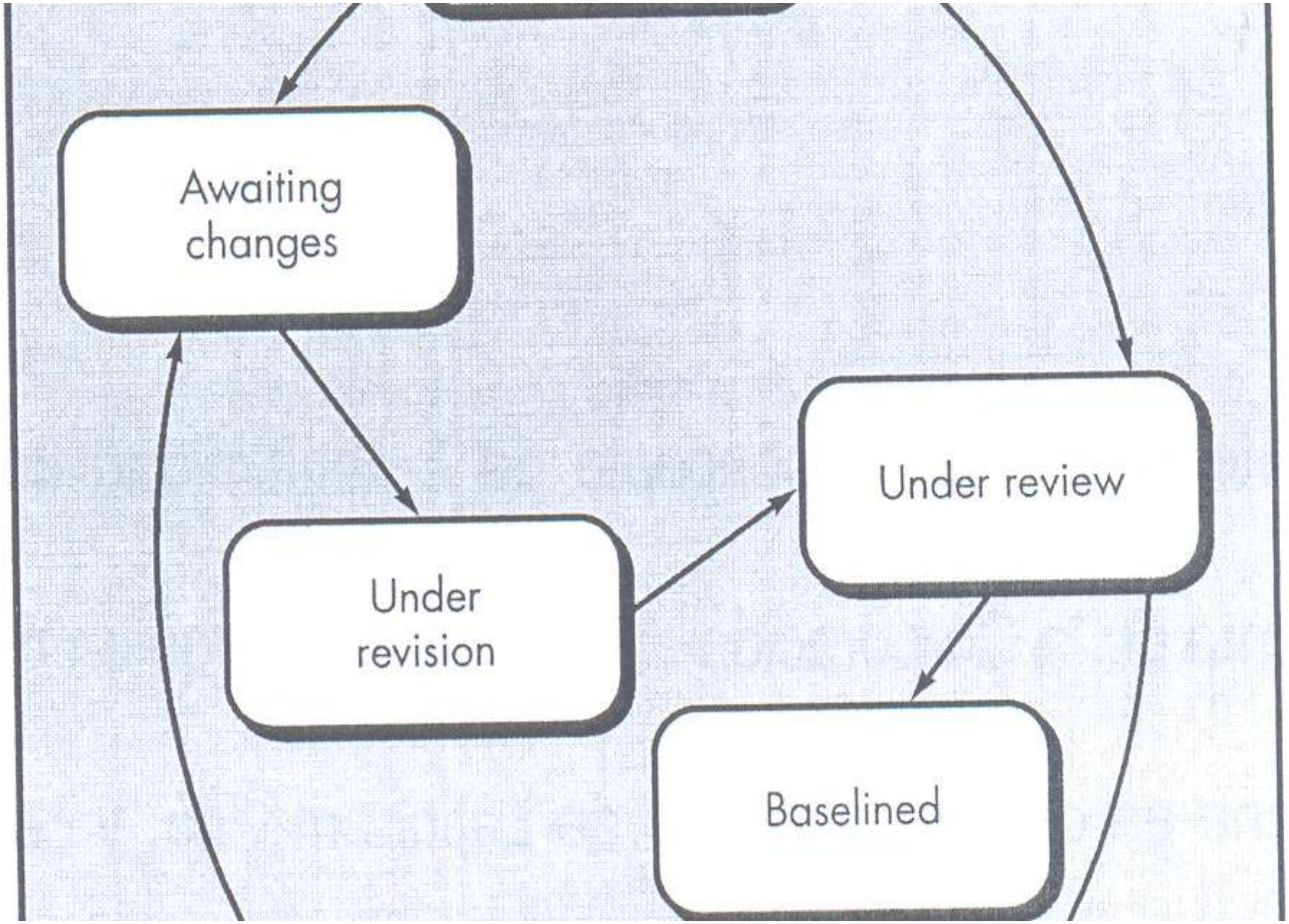
- It is an evolutionary process model that couples the iterative nature or prototyping with the controlled and systematic aspects of the water fall model



A typical spiral model. Fig. 2.7 page 46

2.3.4 Concurrent Models

- It allows a software team to represent iterative and concurrent elements of any if the process models described en this chapter



One element of the concurrent process model. Fig. 2.8 pag. 48

Modeling Activity

A final word on Evolutionary Processes

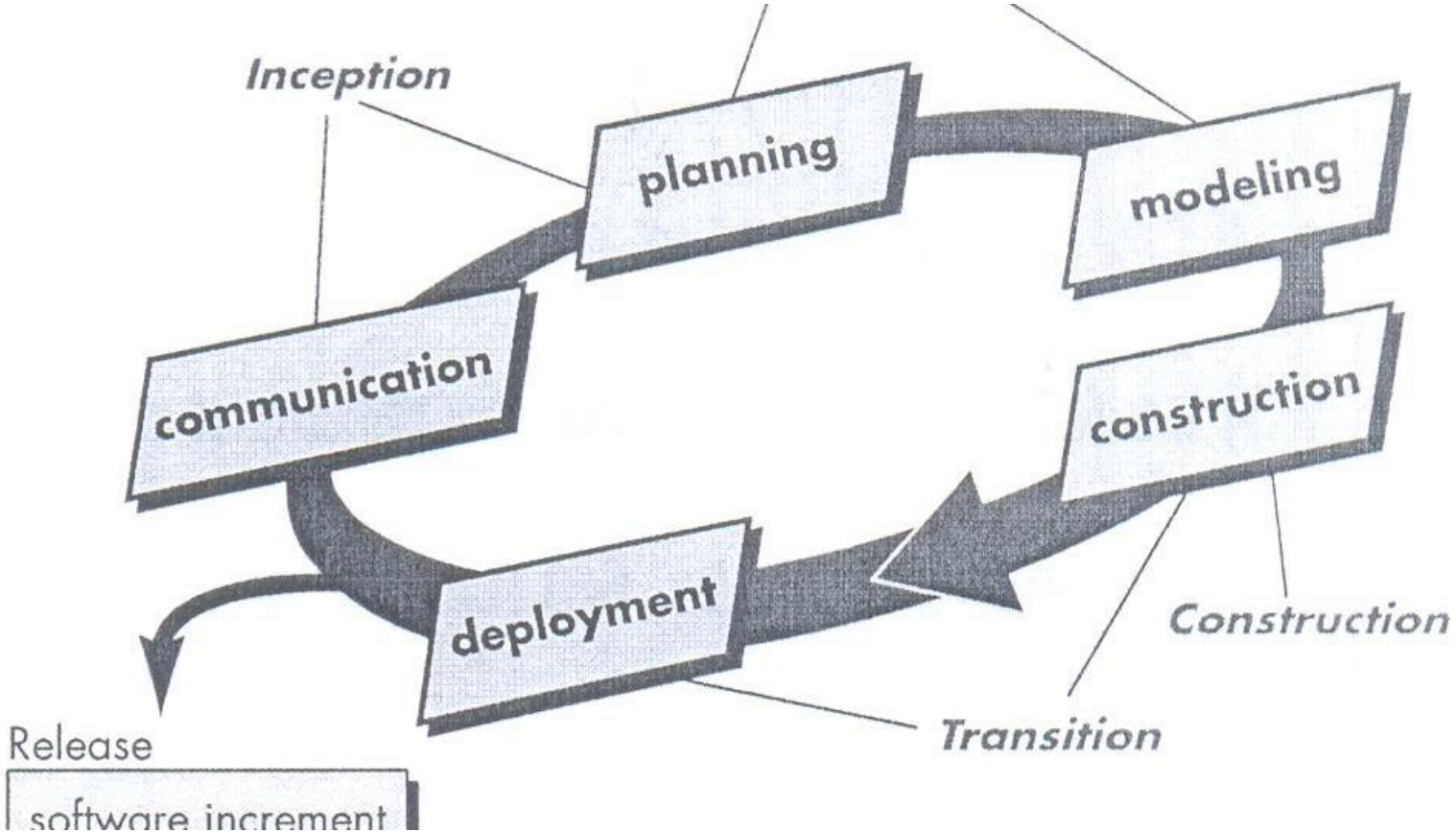
- Modern computer software is characterized by continual change, by very time lines, and by emphatic need for customer-user satisfaction
- Evolutionary process model were concieved to address these issues

2.4 Specialized Process Models

- They take on many characteristics of one or more of the traditional models presented en the preceding section
- Component-Based Development
- The Formal Methods Model
- Aspect-Oriented Software Development

2.5 The Unified Process

- It is an attempt to draw on the best features and characteristics of traditional software process models, but characterized them in a way that implements many of the best principles of agile software development (chapter 3)
- UML. The Unified Model Language containts a robust notation for modeling and development of object-oriented systems (part 2 of this book).
- UML does not provide a framework to guide project teams in their application of the technology



The Unified Process. Fig. 2.9. page 55.

2.6 Personal and Team Process Models

- 2.6.1 Personal Software Process (PSP)
  - Planning
  - High-level design
  - Hig-level design review
  - Development
  - Postmortem

2.6.2 Team Software (TSP)

- PSP is an introduction for TSP
- Objectives of TPS
  - Build self-directed teams that plan an track their work, establish goals, and their own process and plans
  - Show managers how to coach and motivate their teams an how to help them sustain peak performance
  - Accelerate software process improvement by making CMM level behavior normal and expected
  - Provide improvement guidance to high-maturity organizations
  - Facilitate university teaching of industrial-grade team skills

2.7 Process Technology

Process Technology tools have been developed to help software organizations analyze their current process, organize work task, control and monitor progress, and manage technical quality

The model can be analyzed to determine typical workflow and examine alternative process structures that might lead to reduce development time or cost

2.8 Product and Process

If the process is weak, the end product will undoubtedly suffer. But an obsessive overreliance on process is also dangerous

People derive as much (or more) satisfaction from the creative process as they do from the end product