Chapter 2

Process Model

Process Model

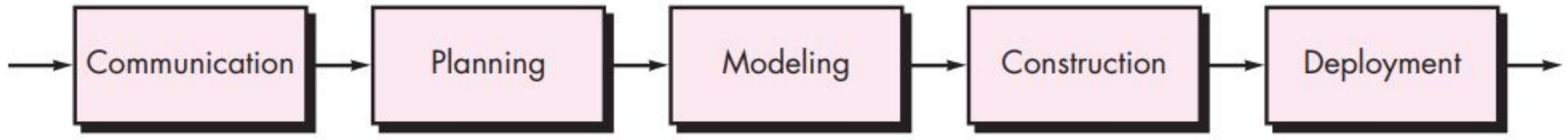
- When you work to build a product or system, it’s important to go through a series of predictable steps—a road map that helps you create a timely, high-quality result. The road map that you follow is called a “software process.”
- Software engineers, their managers and the people who have requested the software have a role to play in the process of defining, building, and testing it.
- One process might be appropriate for creating software for an aircraft
  avionics system, while an entirely different process would be indicated for the creation of a website.

Process Model

- A generic process framework for software engineering defines five framework activities—**communication, planning, modeling, construction,** and **deployment.**
- *process flow*—describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time.

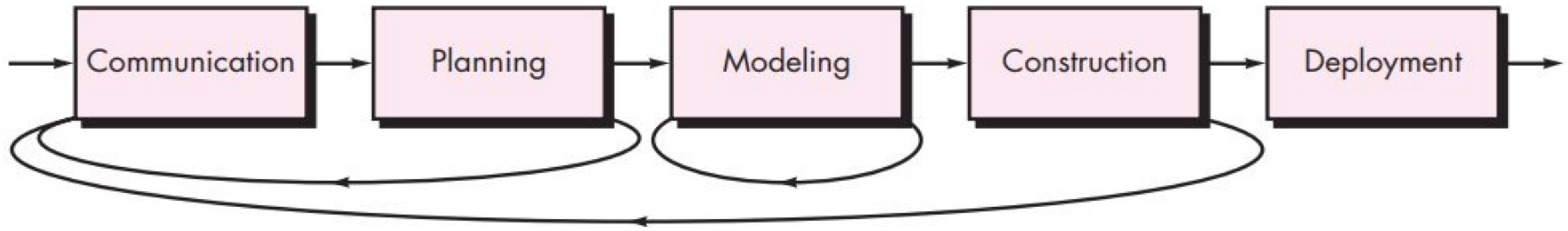
Process Flow

- A *linear process flow* executes each of the five framework activities in sequence, beginning with communication and culminating with deployment.



(a) Linear process flow

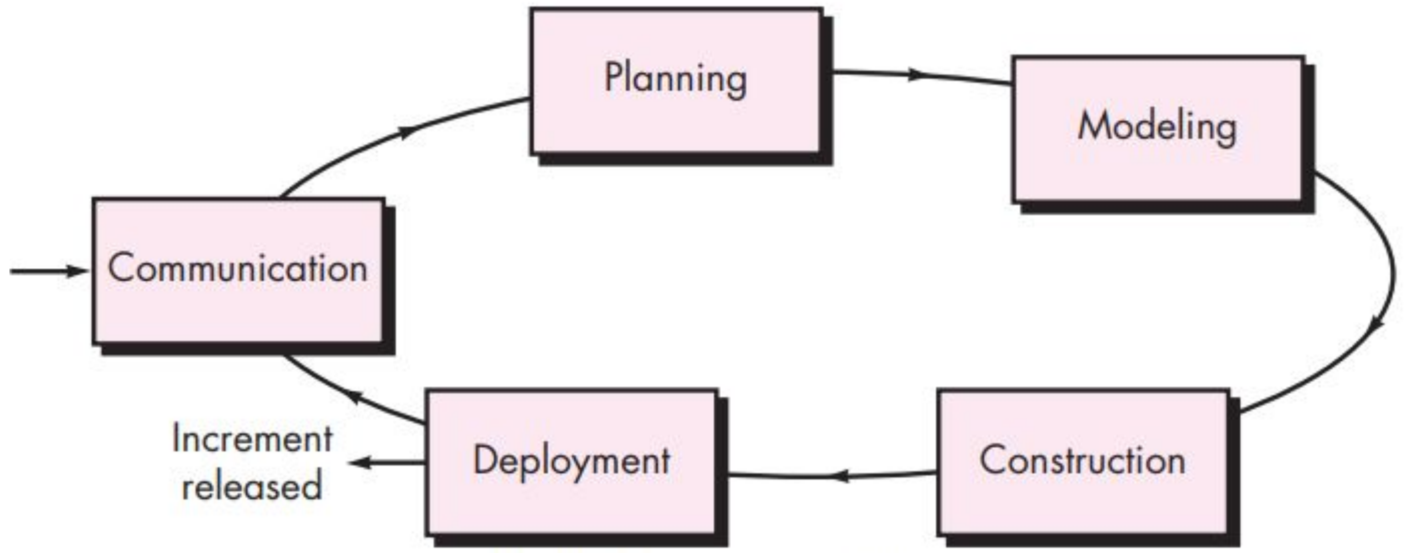
- An *iterative process flow* repeats one or more of the activities before proceeding to the next.



(b) Iterative process flow

Process Flow

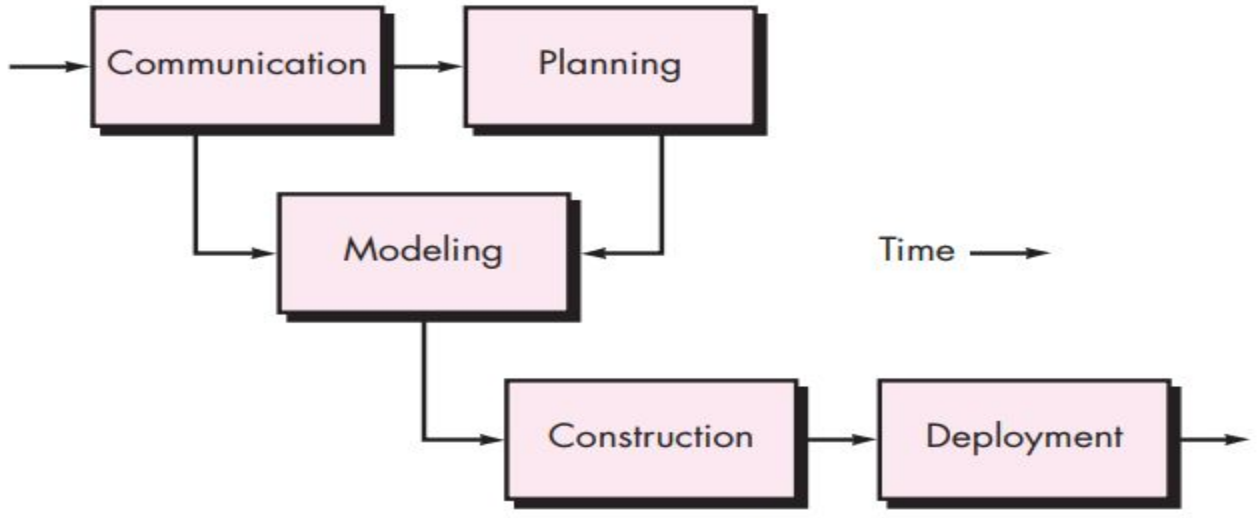
- An *evolutionary process flow* executes the activities in a “circular” manner. Each circuit through the five activities leads to a more complete version of the software.



(c) Evolutionary process flow

Process Flow

- A *parallel process flow* executes one or more activities in parallel with other activities (e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).



(d) Parallel process flow

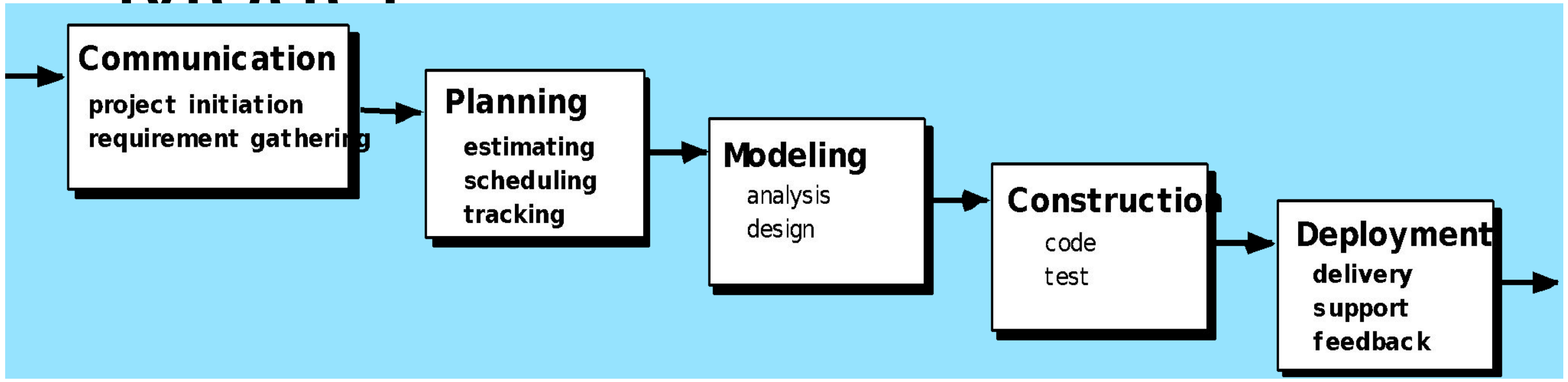
Prescriptive Process Models

- For a small software project requested by one person with simple, straightforward requirements, the communication activity might encompass little more than a phone call with the appropriate stakeholder.
- Therefore, the only necessary action is *phone conversation*, and
- the work tasks (the *task set*) that this action encompasses are:
  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.
- If the project was complex with many stakeholders, each with a different set of (sometime conflicting) requirements, the communication activity might have six distinct actions: *inception*, *elicitation*, *elaboration*, *negotiation*, *specification*, and *validation*.

Prescriptive Process Models

- Prescriptive process models advocate an orderly approach to software engineering
- These models also called Traditional Process Models.
- prescriptive process models strive for structure and order.

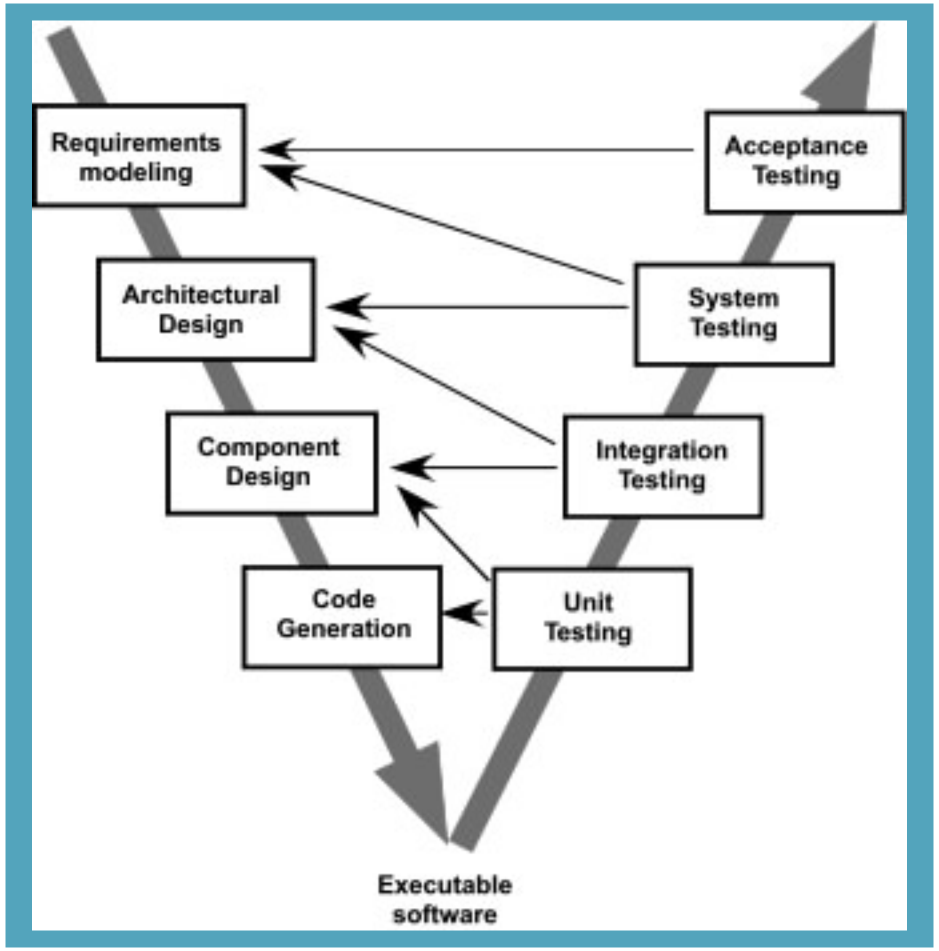
The Waterfall Model



Also called the *classic life cycle*, suggests a systematic, sequential approach

The V-Model

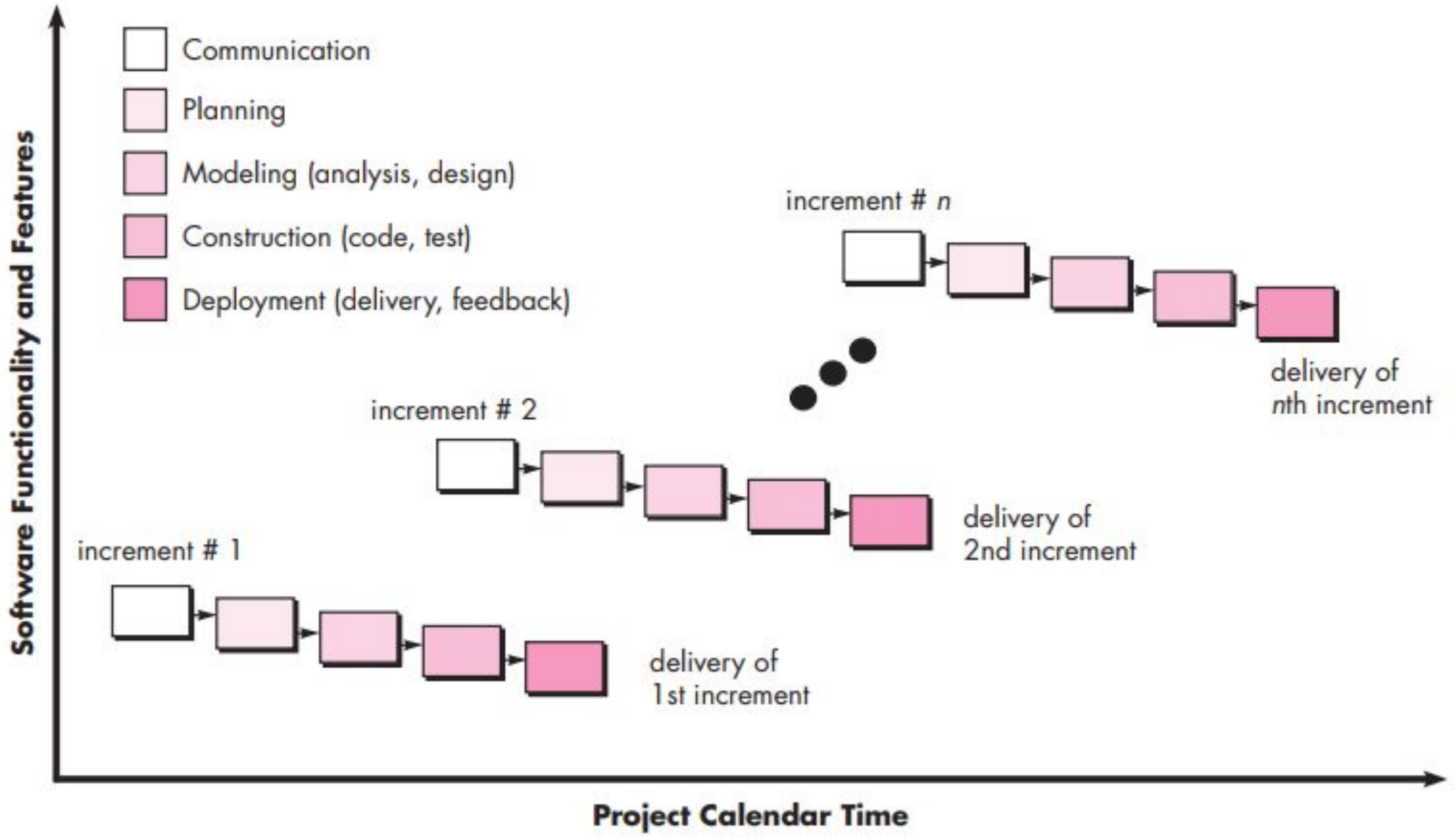
The V-model provides a way of visualizing how verification and validation actions are applied



Problems of Waterfall Model

1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.
2. It is often difficult for the customer to state all 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.

The Incremental Model


Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # 1
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # n
delivery of nth increment
Software Functionality and Features
Project Calendar Time

Incremental Model

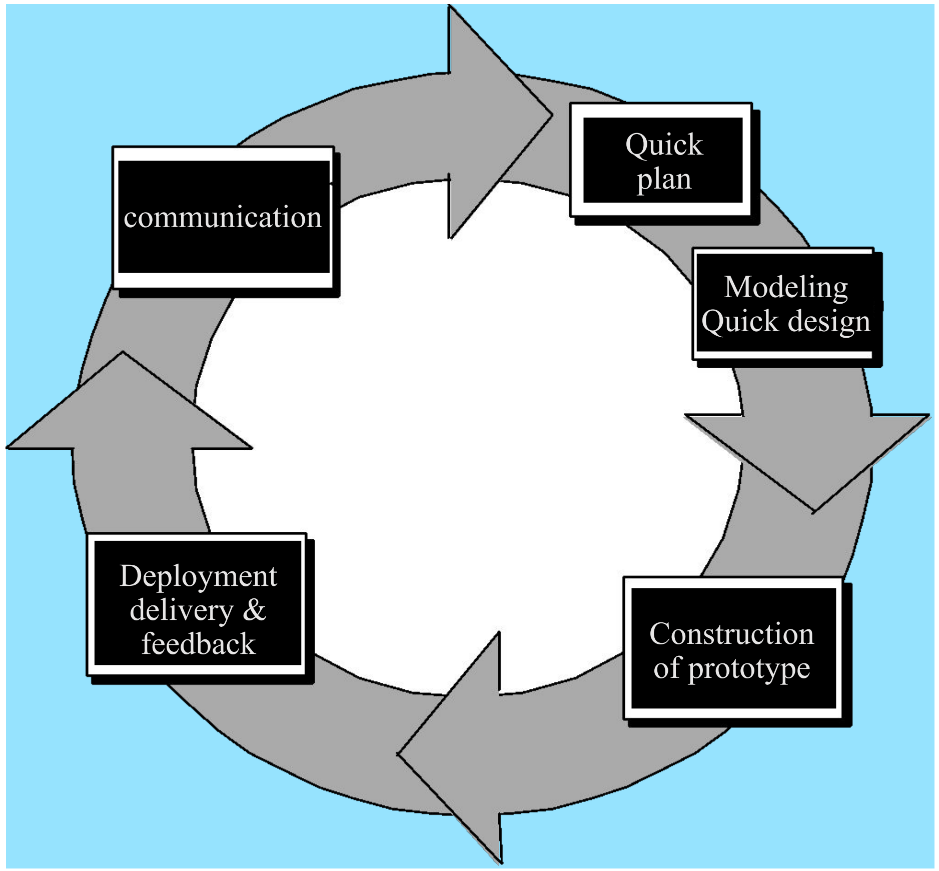
- The *incremental* model combines elements of linear and parallel process flows
- Each linear sequence produces deliverable “increments” of the software in a manner that is similar to the increments produced by an evolutionary process flow.
- For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment can incorporate the prototyping paradigm

Incremental Model

- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.
- Early increments can be implemented with fewer people. If the core product is well received, then additional staff (if required) can be added to implement the next increment.

Evolutionary Models: Prototyping

- Evolutionary models are iterative
- prototyping paradigm assists you and other stakeholders to better understand what is to be built when requirements are fuzzy.



Problems of Prototyping

- prototyping can be problematic for the following reasons:
  - Stakeholders see what appears to be a working version of the software, unaware that the prototype is held together haphazardly, unaware that in the rush to get it working you haven't considered overall software quality or long-term maintainability.
  - stakeholders demand that "a few fixes" be applied to make the prototype a working product.
  - As a software engineer, you often make implementation compromises in order to get a prototype working quickly.

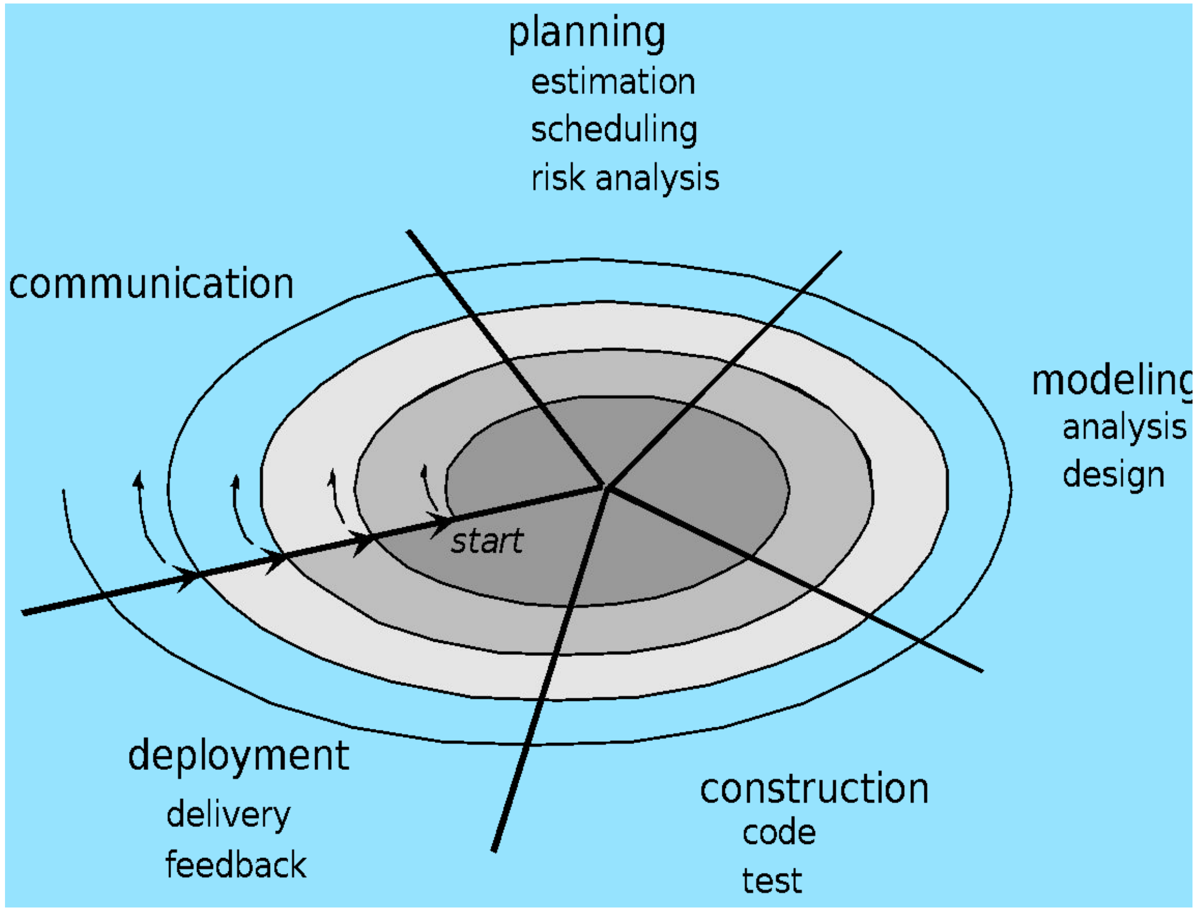
Evolutionary Models: The Spiral

couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

Using the spiral model, software is developed in a series of evolutionary releases

The spiral development model is a *risk*-driven *process model*

The spiral model is a realistic approach to the development of large-scale systems and software.



Problems of Spiral

- the spiral model is not a panacea.
- It may be difficult to convince customers (particularly in contract situations) that the evolutionary approach is controllable.
- It demands considerable risk assessment expertise and relies on this expertise for success. If a major risk is not uncovered and managed, problems will undoubtedly occur.

Problems of Evolutionary Model

In many cases, time-to-market is the most important management requirement. If a market window is missed, the software project itself may be meaningless

- However,
  - 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
  - evolutionary software processes do not establish the maximum speed of the evolution
  - software processes should be focused on flexibility and extensibility rather than on high quality.we should prioritize the speed of the development over zero defects.

Other Specialized Process Model

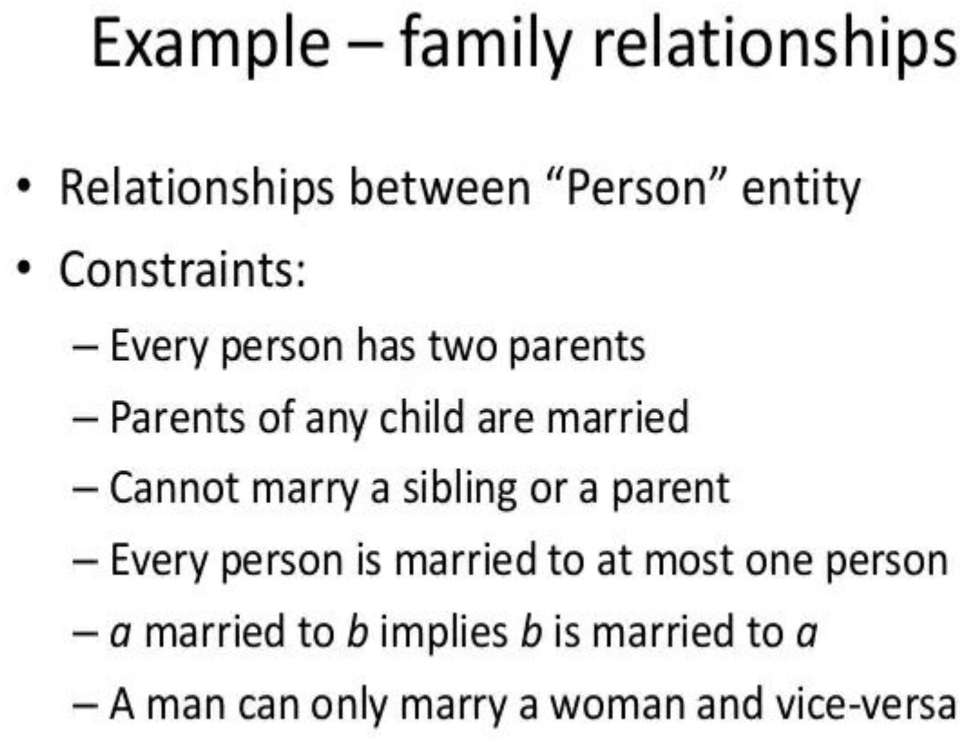
- Component Based Development
- Formal Methods Model
- Aspect oriented Development

Component-Based Development

- Commercial off-the-shelf (COTS) software components, developed by vendors who offer them as products, provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software that is to be built.
- The component-based development model incorporates many of the characteristics of the spiral model
- the component-based development model incorporates the following steps (implemented using an evolutionary approach):
  - Available component-based products are researched and evaluated for the application domain in question.
  - Component integration issues are considered.
  - A software architecture is designed to accommodate the components.
  - Components are integrated into the architecture.
  - Comprehensive testing is conducted to ensure proper functionality.

Formal Methods

- The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software.
- Formal methods enable you to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.



Example – family relationships

- Relationships between "Person" entity
- Constraints:
  - Every person has two parents
  - Parents of any child are married
  - Cannot marry a sibling or a parent
  - Every person is married to at most one person
  - *a* married to *b* implies *b* is married to *a*
  - A man can only marry a woman and vice-versa

Formal Methods

```
abstract sig Person{
    children: set Person
}
sig Man, Woman extends Person{
}
```

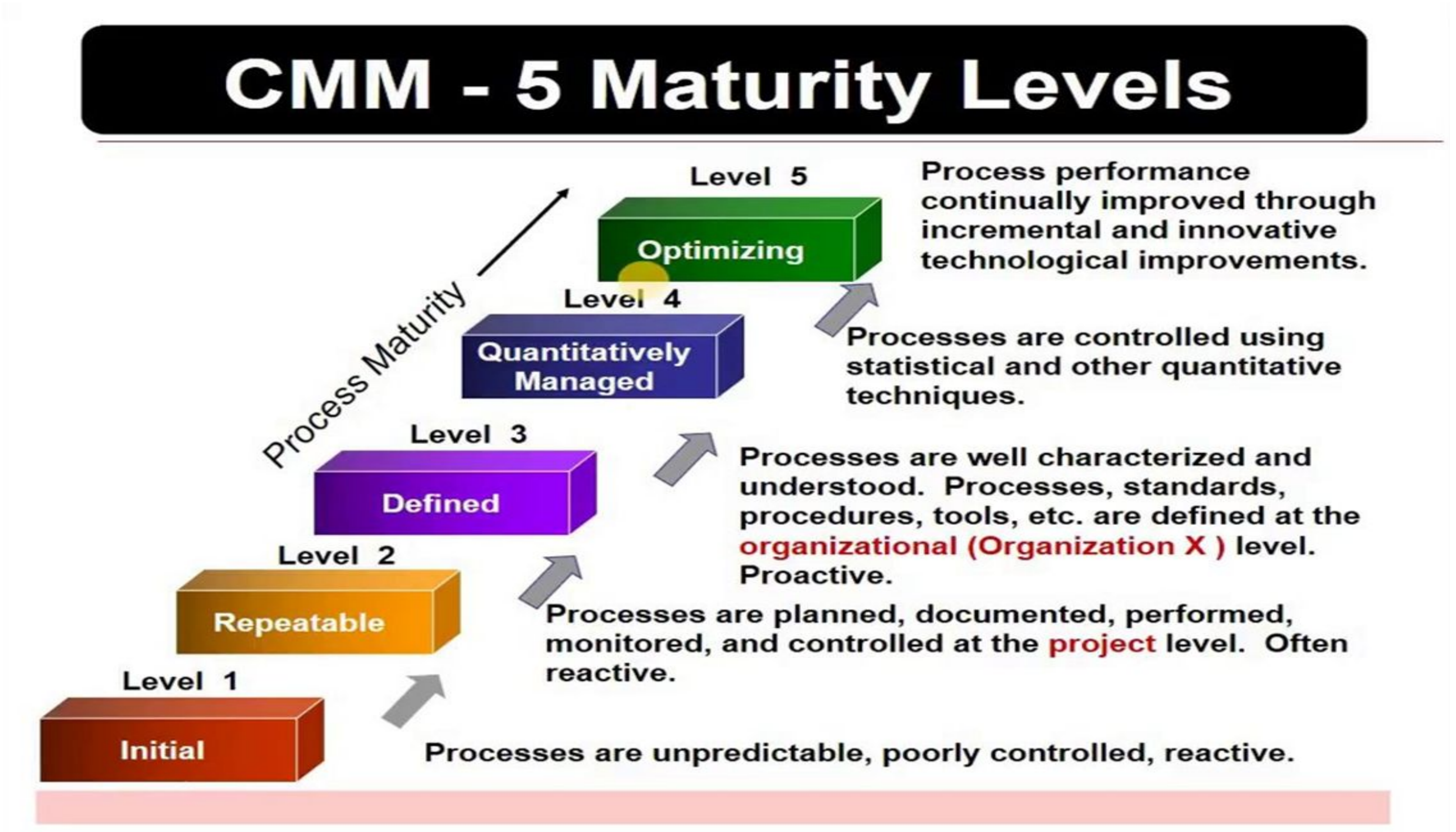
Formal Methods

- The development of formal models is currently quite time consuming and expensive.
- Because few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the models as a communication mechanism for technically unsophisticated customers

Team Software Process (TSP)

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of three to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
- The Capability Maturity Model (CMM), a measure of the effectiveness of a software process.
- Provide improvement guidance to high-maturity organizations.

Capability Maturity Model



CMM - 5 Maturity Levels

Process performance continually improved through incremental and innovative technological improvements.

Processes are controlled using statistical and other quantitative techniques.

Processes are well characterized and understood. Processes, standards, procedures, tools, etc. are defined at the organizational (Organization X) level. Proactive.

Processes are planned, documented, performed, monitored, and controlled at the project level. Often reactive.

Processes are unpredictable, poorly controlled, reactive.