Modelling the Process and Life Cycle

Session 16 delivered by:

PVR Murthy

ज्ञानं विज्ञानं च भक्तिसहितं

Session Objectives

At the end of this session, student will be able to

- Describe software engineering process
- Identify software development products, processes, and resources
- Discuss models of the software development process
- Describe tools and techniques for process modelling

Session Contents

- The Meaning of Process
- Software Process Models
- Tools and Techniques for Process Modelling
- Practical Process Modelling
- Information System Example
- Framework activities
- Unified process

The Meaning of Process

- A process
  - A series of steps involving activities, constrains, and resources that produce an intended output of some kind
- A process involves a set of tools and techniques

Process Characteristics

- Prescribes all major process activities
- Uses resources, subject to set of constraints
  - Such as schedule
- Produces intermediate and final products
- May be composed of subprocesses with hierarchy or links
- Each process activity has entry and exit criteria
- Activities are organized in sequence, so timing is clear
- Each process has guiding principles, including goals of each activity
- Constraints may apply to an activity, resource or product

The Importance of Processes

- Impose consistency and structure on a set of activities
- Guide us to understand, control, examine, and improve the activities
- Enable us to capture our experiences and pass them along

Reasons for Modelling a Process

- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goals
- To tailor a general process for a particular situation in which it will be used

Software Life Cycle

- When a process involves building a software, the process may be referred to as software life cycle
  - Requirements analysis and definition
  - System (architecture) design
  - Program (detailed/procedural) design
  - Writing programs (coding/implementation)
  - Testing: unit, integration, system
  - System delivery (deployment)
  - Maintenance

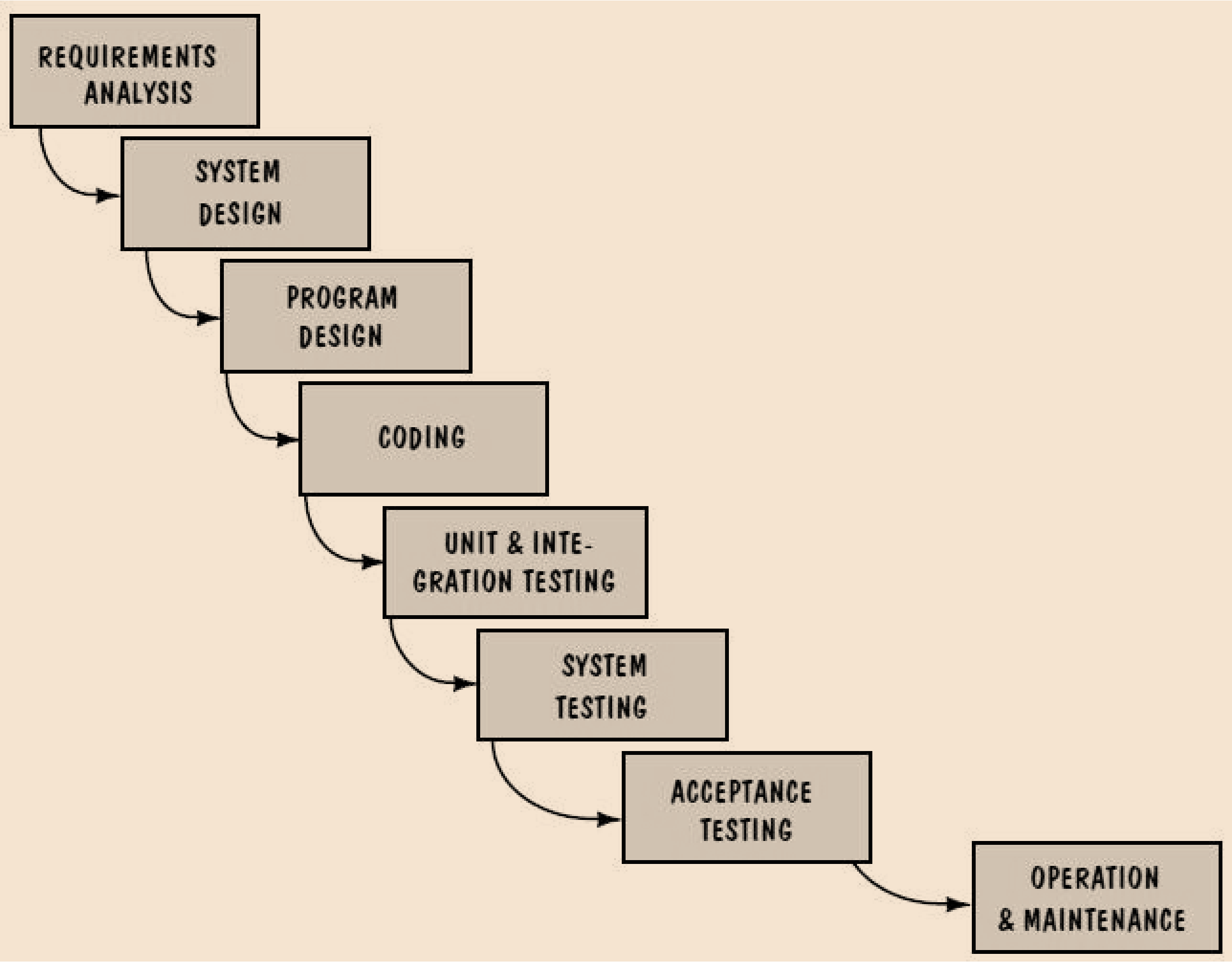
Software Development Process Models

- Waterfall model
- V model
- Prototyping model
- Operational specification
- Transformational model
- Phased development: increments and iterations
  - Spiral model
  - Agile methods

Waterfall Model

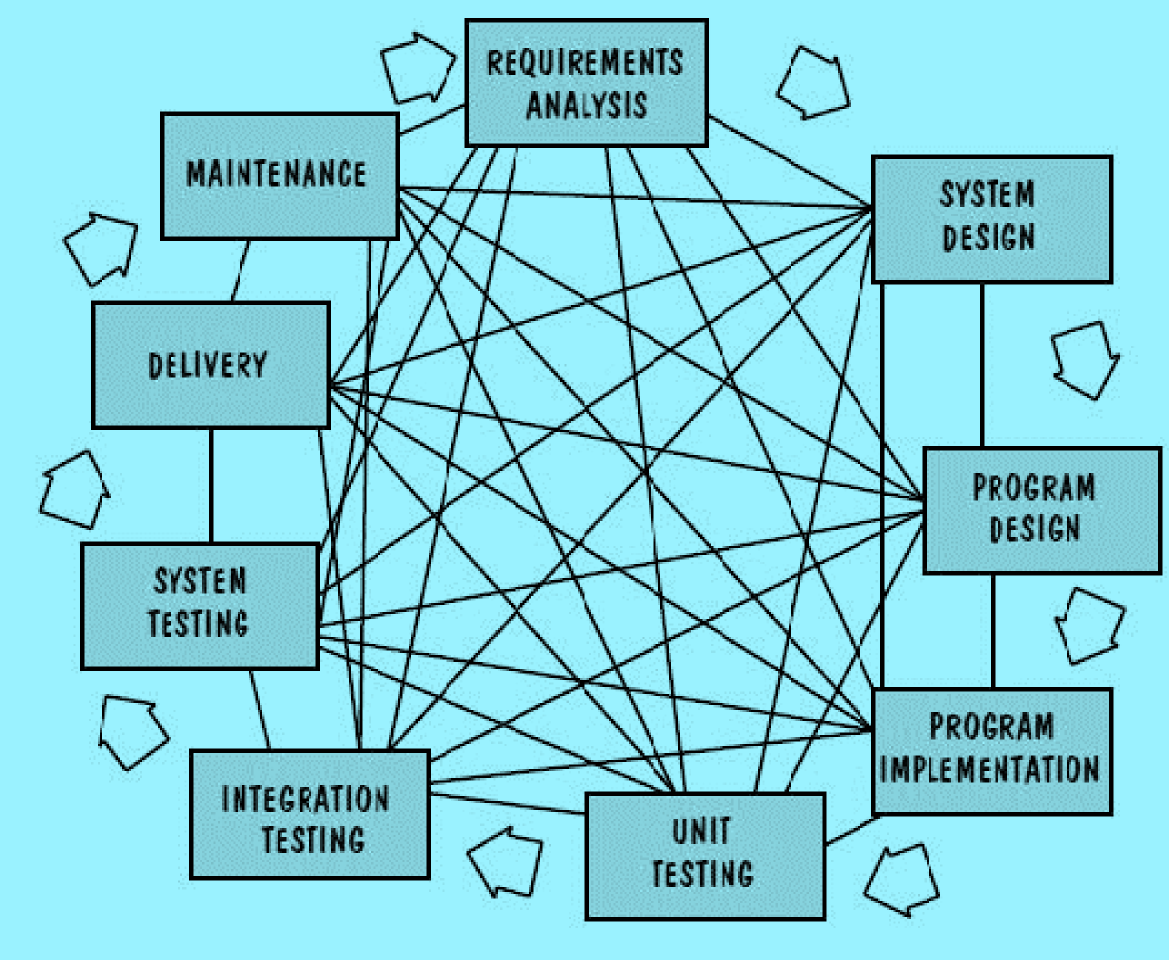
- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase is marked by milestones and deliverables (artifacts)

Waterfall Model, Cont’d.



Waterfall Model, Cont’d.

- There is no iteration in waterfall model
- Most software developments apply a great deal of iterations



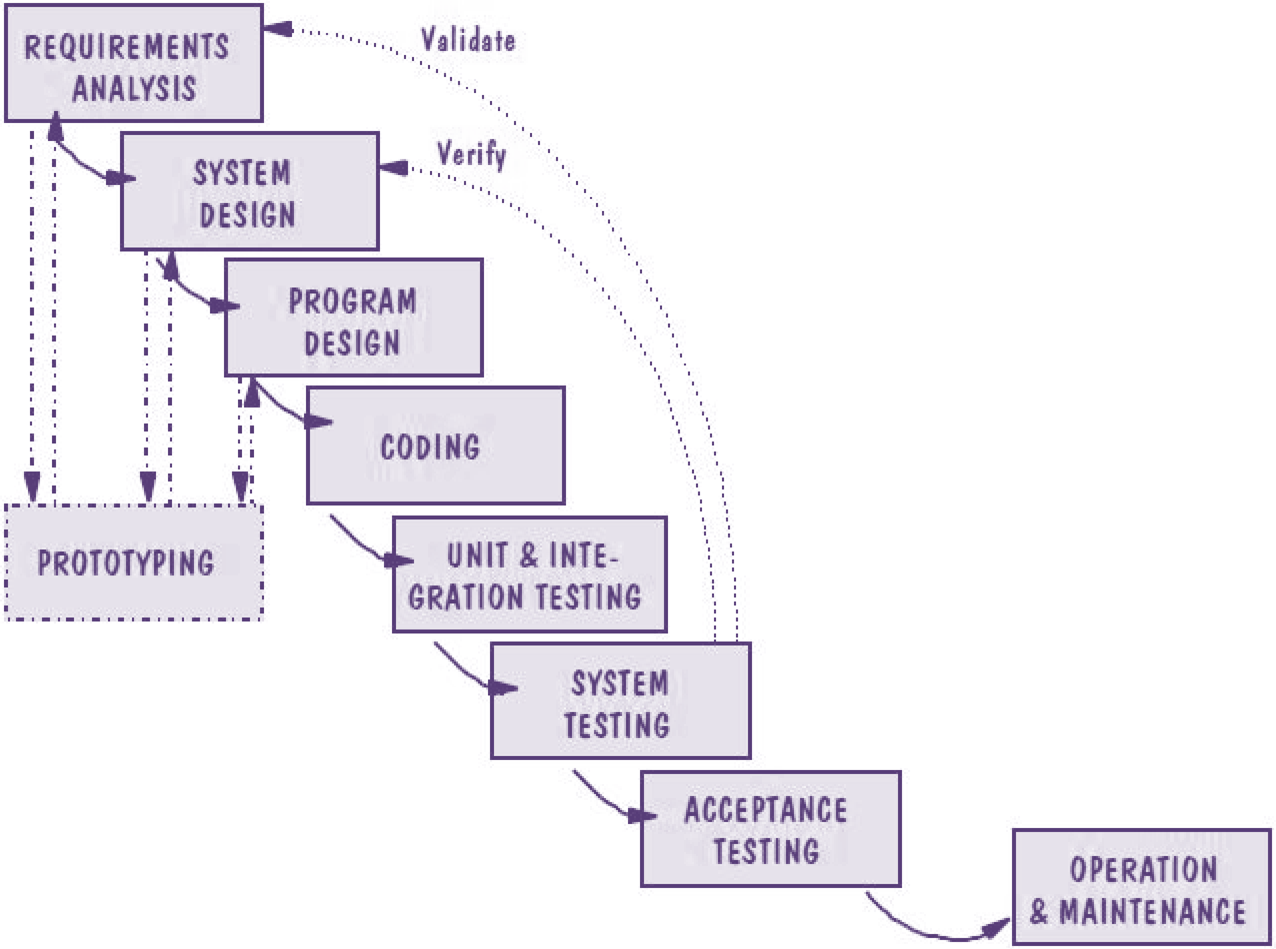
Drawbacks of the Waterfall Model

- Provides no guidance on how to handle changes to products and activities during development (assumes requirements can be frozen)
- Views software development as manufacturing process rather than as creative process
- There is no iterative activities that lead to creating a final product
- Long wait before a final product

Waterfall Model with Prototype

- A prototype is a partially developed product
- Prototyping helps
  - developers assess alternative design strategies (design prototype)
  - users understand what the system will be like (user interface prototype)
- Prototyping is useful for verification and validation

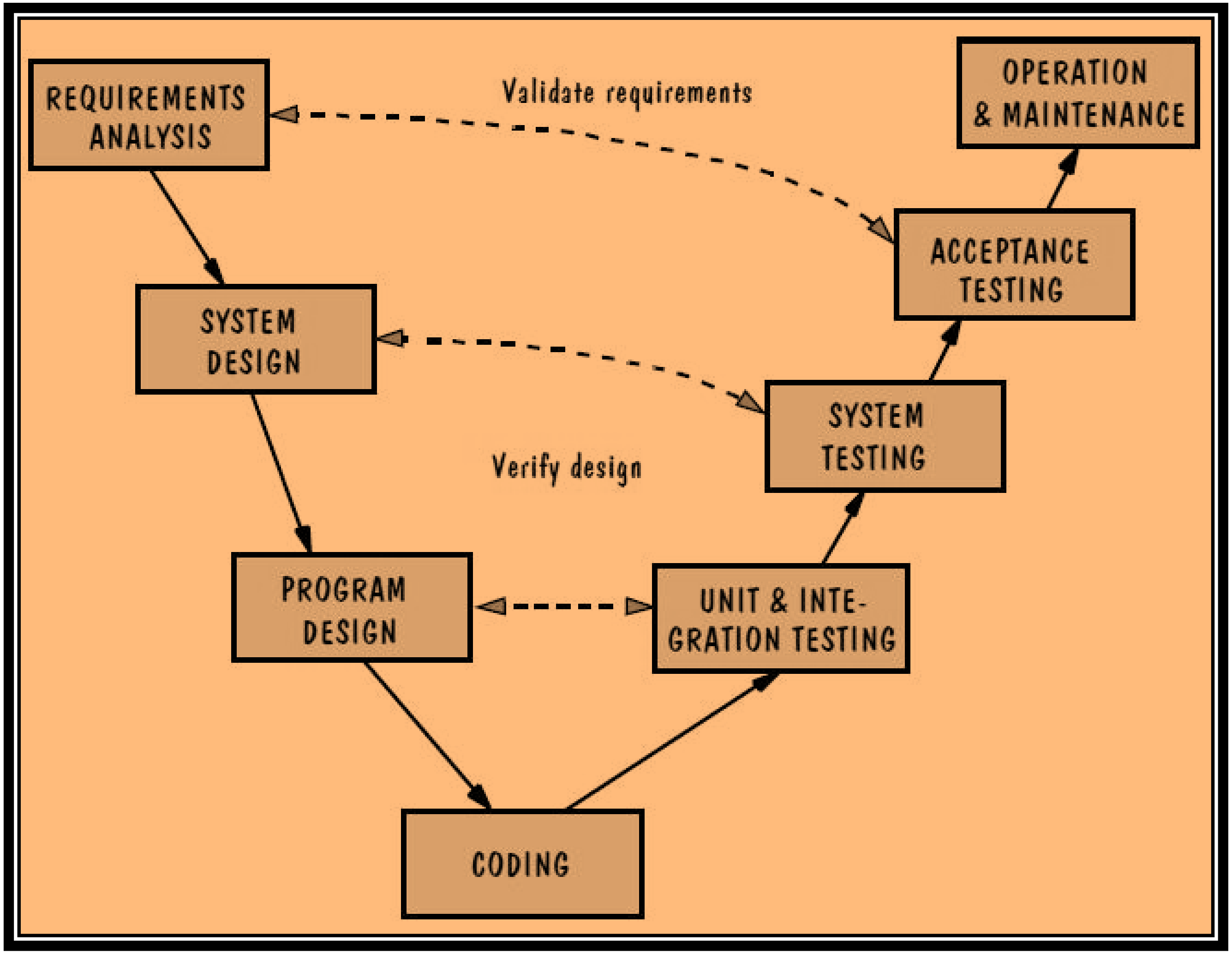
Waterfall Model with Prototype, Cont'd.



V Model

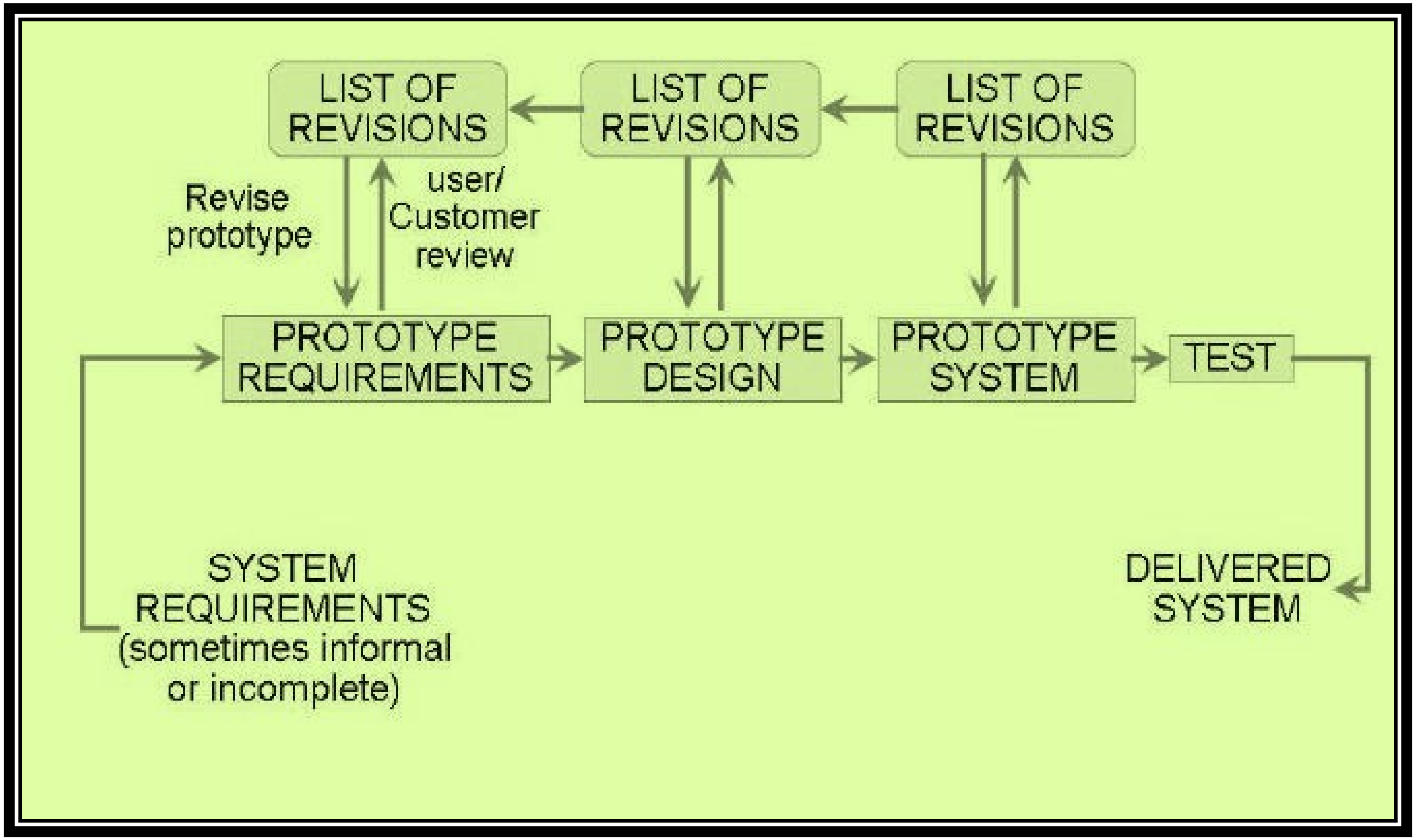
- A variation of the waterfall model
- Uses unit testing to verify procedural design
- Uses integration testing to verify architectural (system) design
- Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted

V Model, Cont'd.



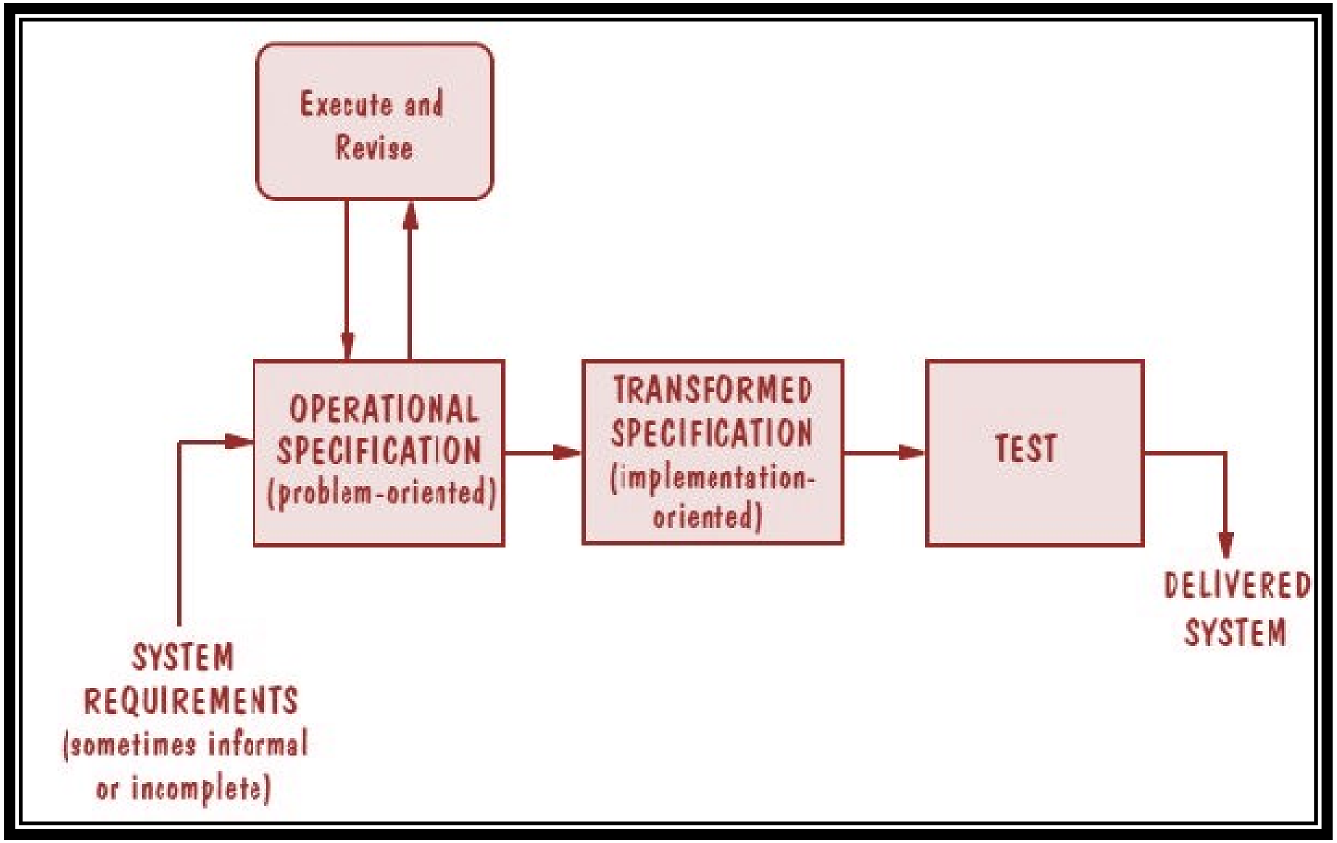
Prototyping Model

- Allows repeated investigation of the requirements or design
- Reduces risk and uncertainty in the development



Operational Specification Model

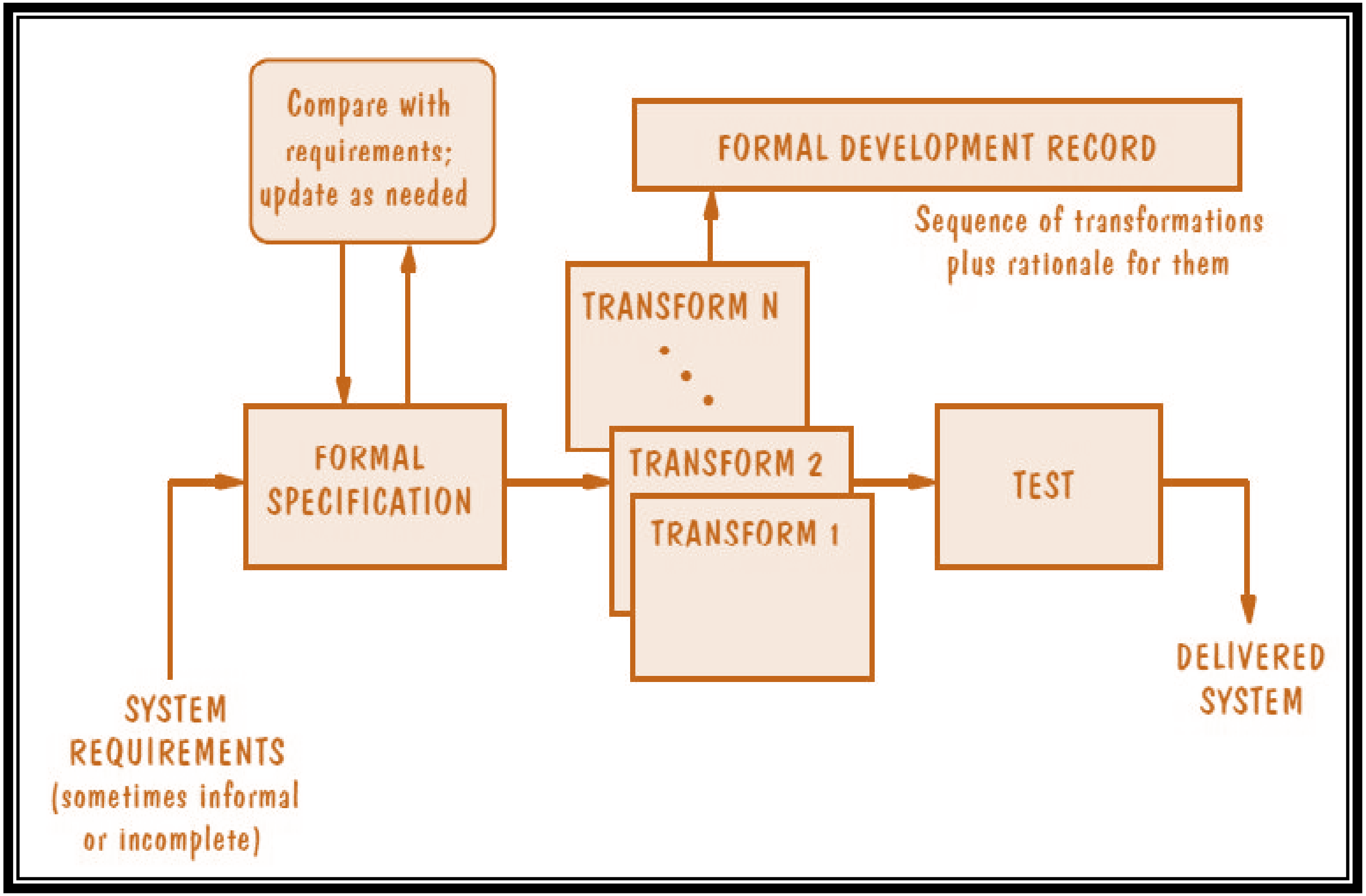
- Requirements are executed (examined) and their implication evaluated early in the development process
- Functionality and the design are allowed to be merged


Execute and Revise
OPERATIONAL SPECIFICATION (problem-oriented)
TRANSFORMED SPECIFICATION (implementation-oriented)
TEST
DELIVERED SYSTEM
SYSTEM REQUIREMENTS (sometimes informal or incomplete)

Transformational Model

- Fewer major development steps
- Applies a series of transformations to change a specification into a deliverable system
  - Change data representation
  - Select algorithms
  - Optimize
  - Compile
- Relies on formalism
- Requires formal specification (to allow transformations)

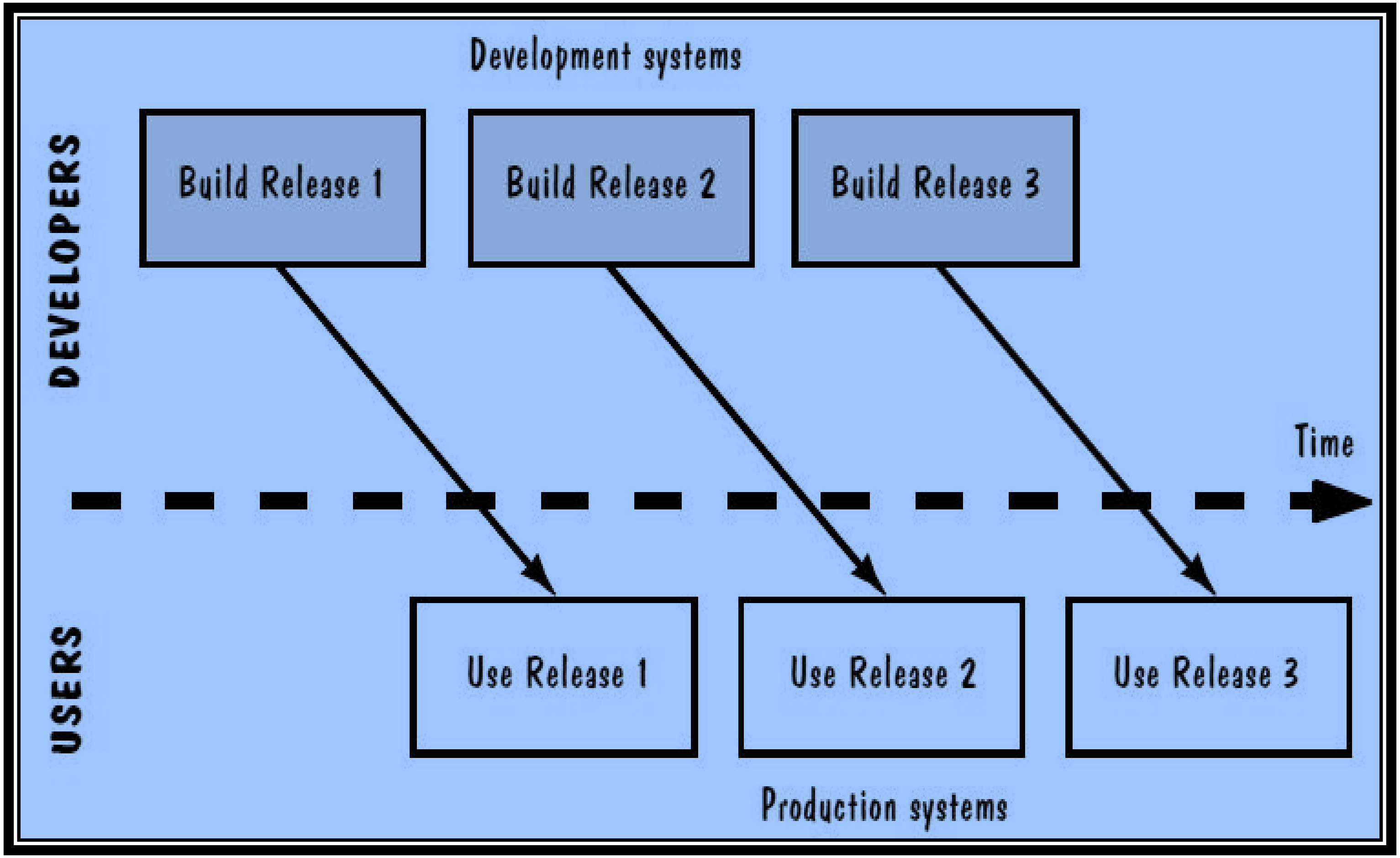
Transformational Model, Cont'd.


Compare with requirements; update as needed
FORMAL DEVELOPMENT RECORD
Sequence of transformations plus rationale for them
TRANSFORM N
FORMAL SPECIFICATION
TRANSFORM 2
TRANSFORM 1
TEST
DELIVERED SYSTEM
SYSTEM REQUIREMENTS (sometimes informal or incomplete)

Phased Development: Increments and Iterations

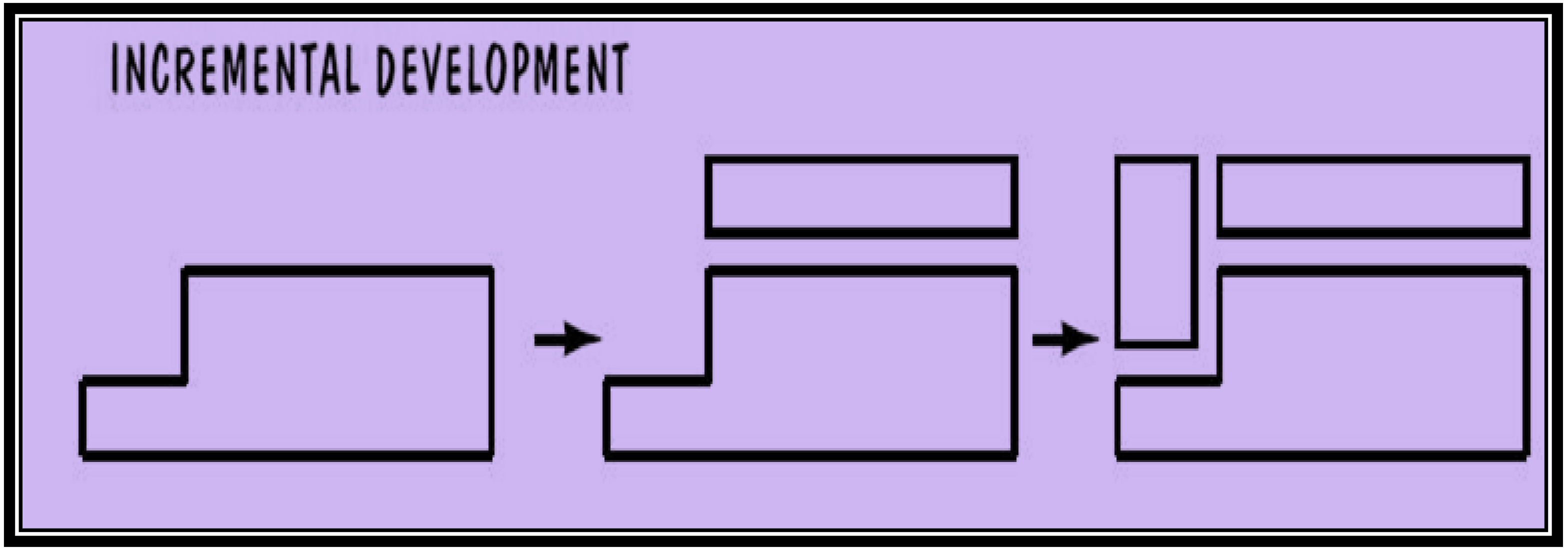
- Shorter cycle time
- System delivered in pieces
  - enables customers to have some functionality while the rest is being developed
- Allows two systems functioning in parallel
  - the production system (release n): currently being used
  - the development system (release n+1): the next version

Phased Development: Increments and Iterations, Cont’d.



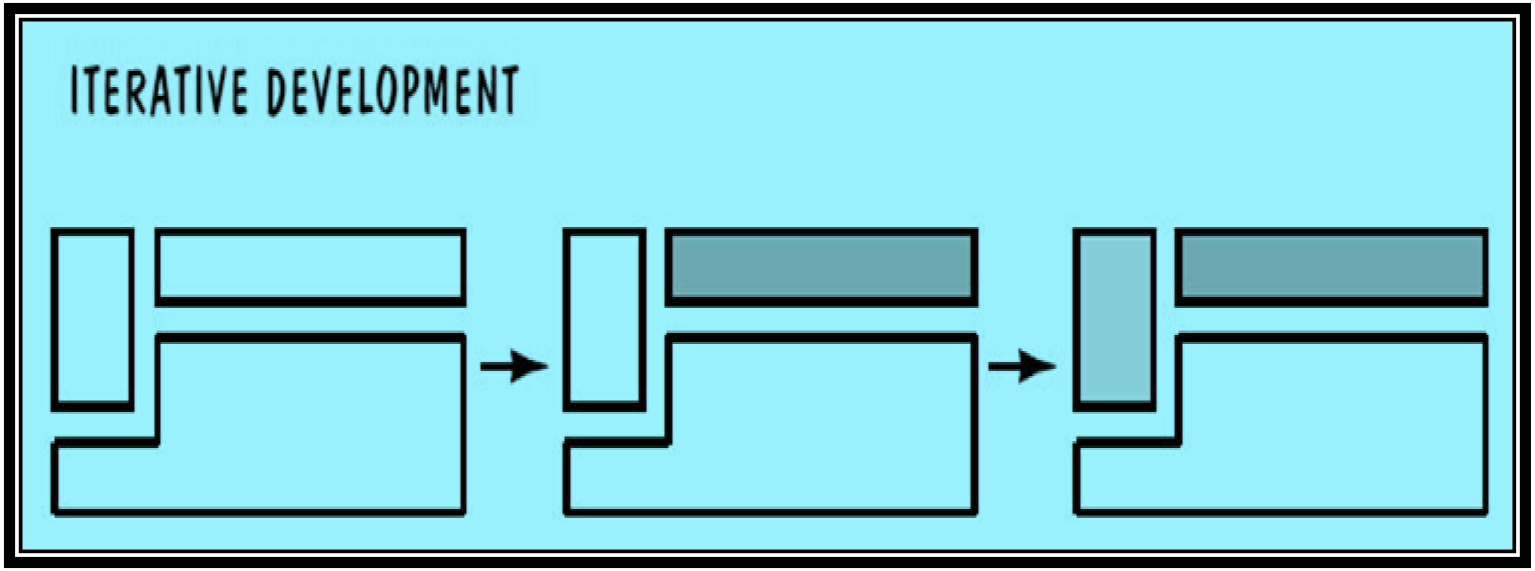
Incremental Development

- Starts with small functional subsystem and adds functionality with each new release



Iterative Development

- Starts with full system, then changes functionality of each subsystem with each new release



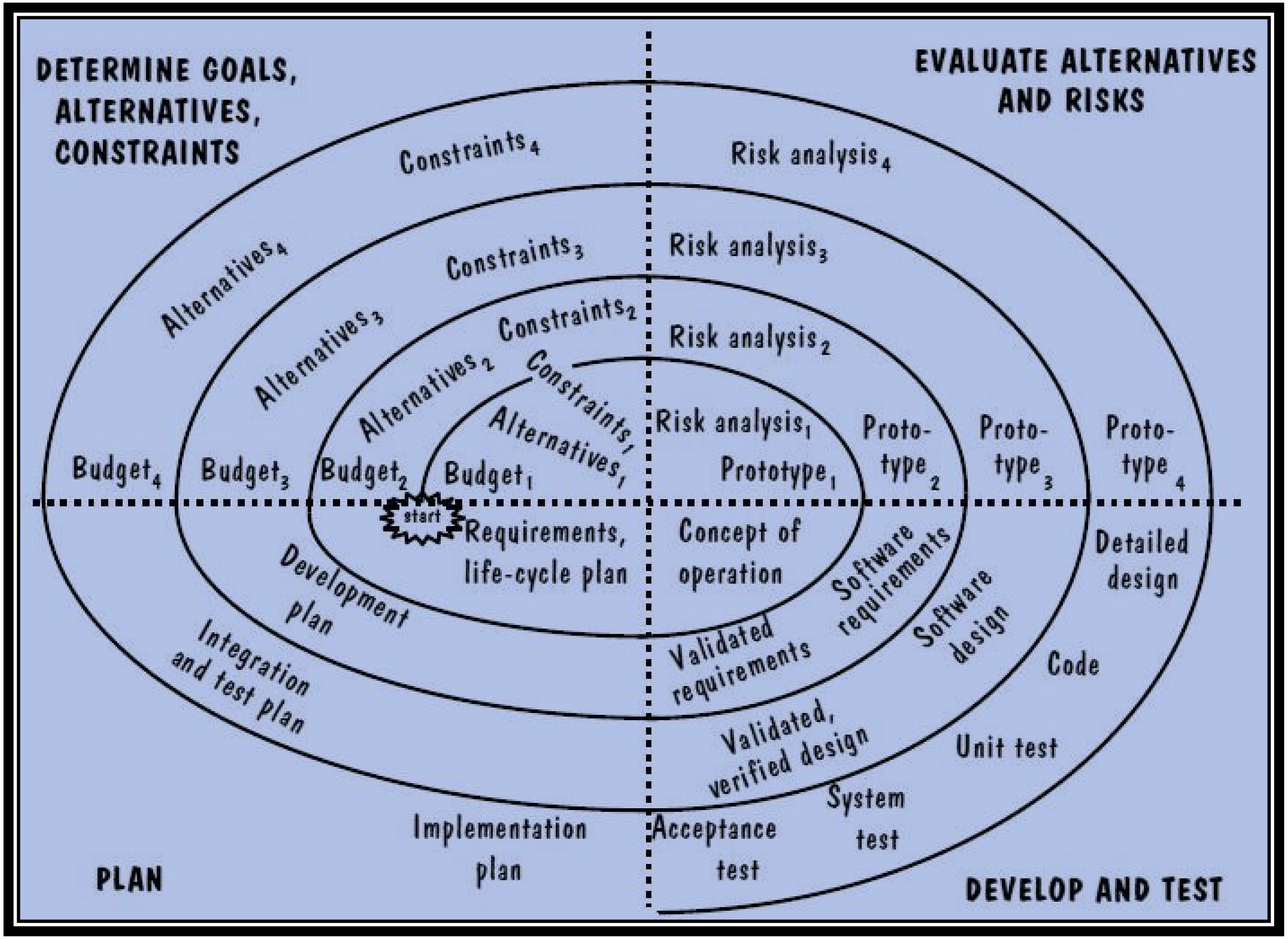
Features of Phased Development

- Phased development is desirable for several reasons
  - Training can begin early, even though some functions are missing
  - Markets can be created early for functionality that has never before been offered
  - Frequent releases allow developers to fix unanticipated problems globally and quickly
  - The development team can focus on different areas of expertise with different releases

Spiral Model

- Suggested by Boehm (1988)
- Combines development activities with risk management to minimize and control risks
- The model is presented as a spiral in which each iteration is represented by a circuit around four major activities
  - Plan
  - Determine goals, alternatives, and constraints
  - Evaluate alternatives and risks
  - Develop and test

Spiral Model, Cont'd.



Agile Methods

- Emphasis on flexibility in producing software quickly and capably
- Agile manifesto
  - Value individuals and interactions over process and tools
  - Prefer to invest time in producing working software rather than in producing comprehensive documentation
  - Focus on customer collaboration rather than contract negotiation
  - Concentrate on responding to change rather than on creating a plan and then following it

Examples of Agile Process

- Extreme programming (XP)
- Crystal: a collection of approaches based on the notion that every project needs a unique set of policies and conventions
- Scrum: 30-day iterations; multiple self-organizing teams; daily scrum coordination
- Adaptive Software Development (ASD)

Extreme Programming

- Emphasis on four characteristics of agility
  - Communication: continual interchange between customers and developers
  - Simplicity: select the simplest design or implementation
  - Courage: commitment to deliver functionality, early and often
  - Feedback: loops built into various activities during the development process

Twelve Facets of XP

- The planning game (customer defines value)
- Small releases
- Metaphor (common vision, common names)
- Simple design
- Writing tests first
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration (small increments)
- Sustainable pace (40 hours/week)
- On-site customer
- Coding standards

When is Extreme too Extreme?

- Extreme programming’s practices are interdependent
  - Vulnerability, if one of them is modified
- Requirements expressed as a set of test cases must be passed by the software
  - System passes the tests, but is not what the customer is paying for
- Refactoring issue
  - Difficult to rework a system without degrading its architecture

Process Modelling Tools and Techniques

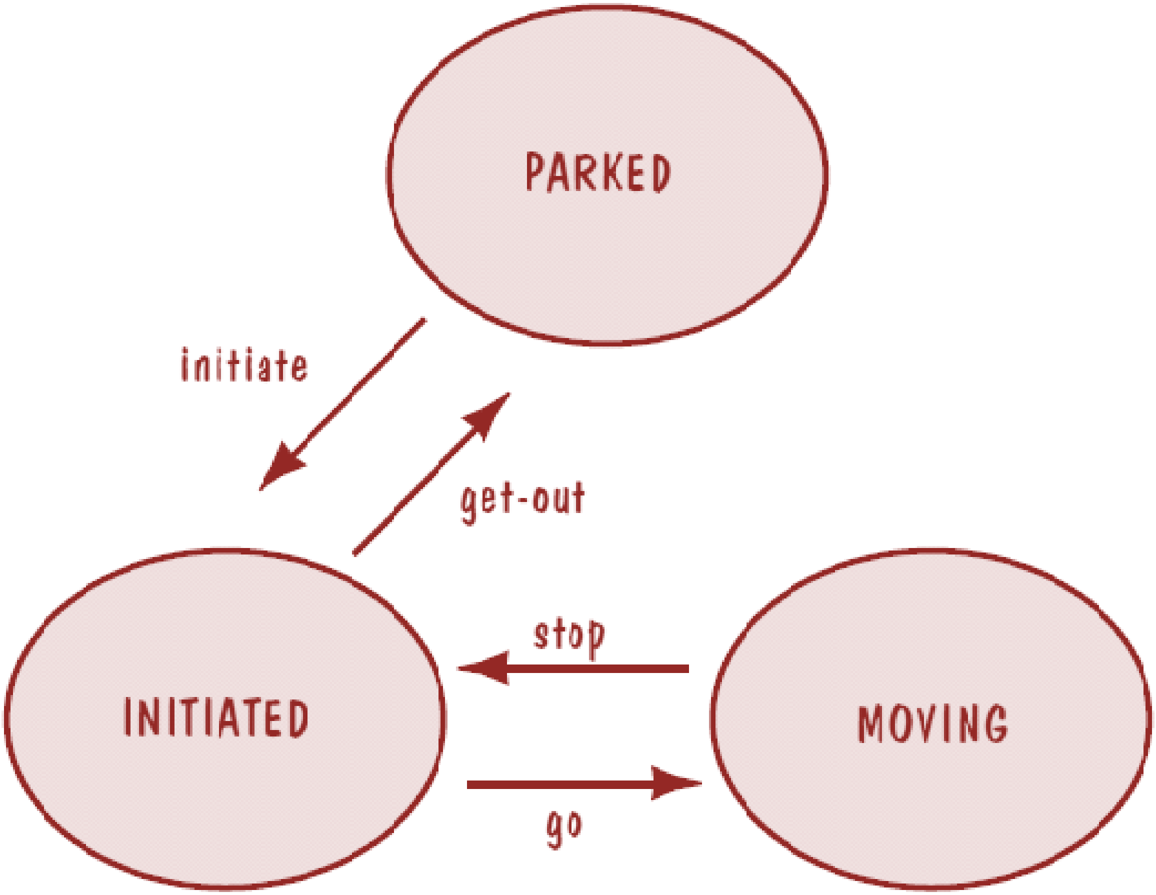
- Notation depends on what we want to capture in the model
- The two major notation categories
  1. Static model: depicts the process
  2. Dynamic model: enacts the process

Static Modelling: Lai Notation

- Several templates, such as an Artifact Definition Template
- Element of a process are viewed in terms of seven types
  - Activity
  - Sequence
  - Process model
  - Resource
  - Control
  - Policy
  - Organization
- Several templates, such as an Artifact Definition Template

Static Modelling: Lai Notation Cont'd.

Static Modelling: Lai Notation: Transition Diagram



Dynamic Modelling

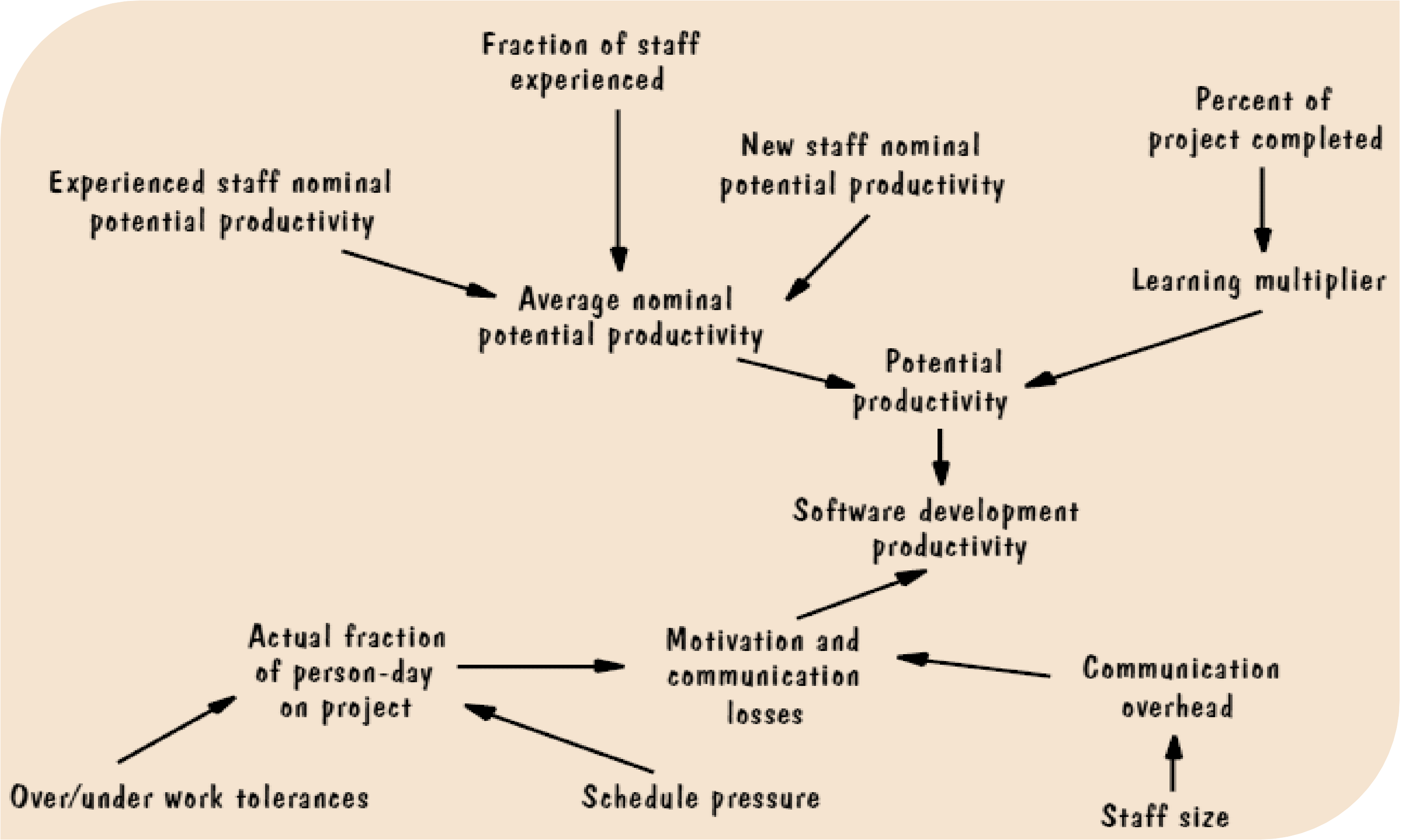
- Enables enaction of process to see what happens to resources and artifacts as activities occur
- Simulate alternatives and make changes to improve the process
- Example: systems dynamics model

Dynamic Modelling: System Dynamics

- Introduced by Forrester in the 1950's
- Abdel-Hamid and Madnick applied it to software development
- One way to understand system dynamics is by exploring how software development process affects productivity

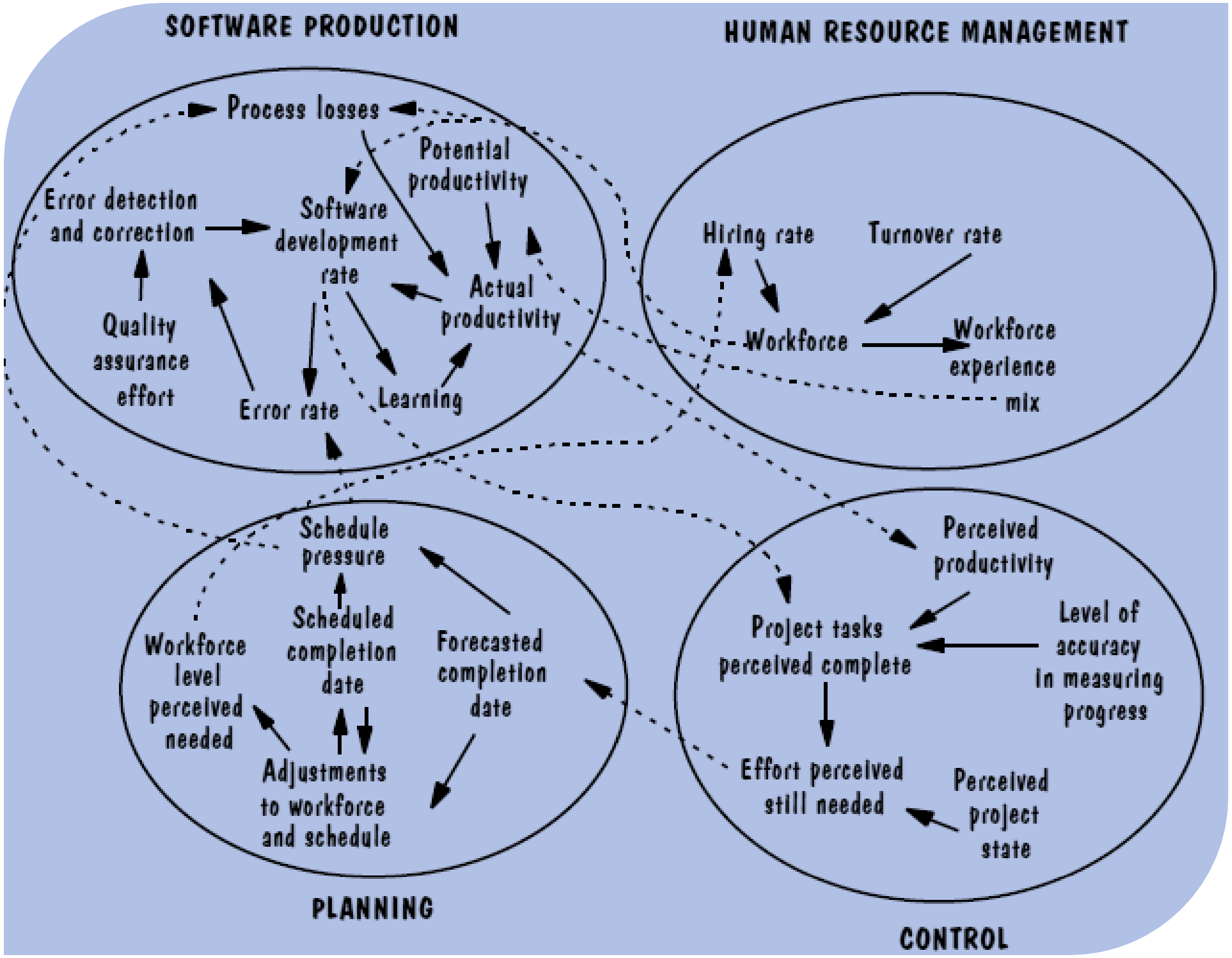
Systems Dynamics Model

- Pictorial presentation of factors affecting productivity
- Arrows indicate how changes in one factor change another


Fraction of staff experienced
Percent of project completed
New staff nominal potential productivity
Experienced staff nominal potential productivity
Average nominal potential productivity
Learning multiplier
Potential productivity
Software development productivity
Actual fraction of person-day on project
Motivation and communication losses
Communication overhead
Over/under work tolerances
Schedule pressure
Staff size

Systems Dynamics Model, Cont'd.

- A system dynamic model containing four major areas, affecting productivity


SOFTWARE PRODUCTION
HUMAN RESOURCE MANAGEMENT
Process losses
Potential productivity
Error detection and correction
Software development rate
Actual productivity
Quality assurance effort
Error rate
Learning
Hiring rate
Turnover rate
Workforce
Workforce experience mix
Schedule pressure
Scheduled completion date
Forecasted completion date
Workforce level perceived needed
Adjustments to workforce and schedule
PLANNING
Perceived productivity
Project tasks perceived complete
Level of accuracy in measuring progress
Effort perceived still needed
Perceived project state
CONTROL

Information System Example

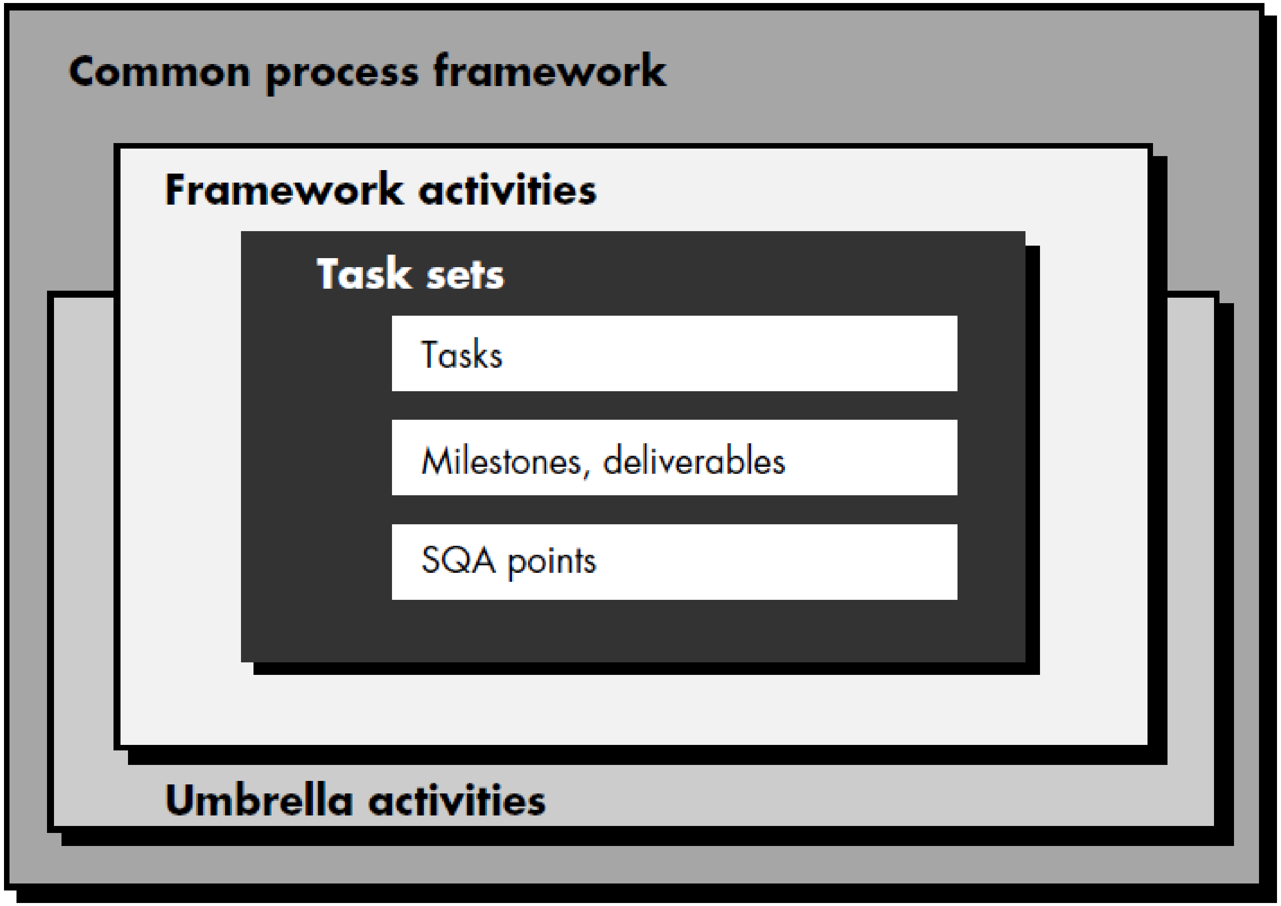
Piccadilly Television advertising program

- Software system should be easily maintained and changed
  - Most of the uncertainty is in advertising regulations and business constraints
- Waterfall model may be too rigid
- Prototyping included for building the user interface
- Spiral model is a good candidate
  - Allows us to revisit assumptions, analyse our risks and prototype system characteristics
- For high level Boehm's representation; but at finer level use techniques such as Lai's notation
  - Characterisation of "Risk" using Artifact Definition Form
  - Similarly, other aspects of the s/w development

A Process Framework

- A process framework
  - Establishes the foundation for a complete software engineering process
- Framework activities
  - Are applicable to all software projects, regardless of their size or complexity
- Umbrella activities
  - Are applicable across the entire software process

A Process Framework


Common process framework

Framework activities

Task sets

Tasks

Milestones, deliverables

SQA points

Umbrella activities

The software
process

Generic Framework Activities

A generic process framework for software engineering encompasses **five** activities

1. Communication
   - Communicate and collaborate with the customer (and other stakeholders)
   - Understand stakeholders, objectives for the project and to gather requirements
2. Planning
   - Planning activity creates a "map" (a software project plan) that helps to guide the team as it makes the journey
   - Technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule

Generic Framework Activities contd.

3. Modelling
   - Create a "sketch" of the thing so that you'll understand the big picture
   - To better understand software requirements and the design that will achieve those requirements
4. Construction
   - Combines code generation and the testing that is required to uncover errors in the code
5. Deployment
   - The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation

Umbrella Activities

- Software engineering process framework activities are complemented by a number of umbrella activities
- Applied throughout a software project and help a software team manage and control progress, quality, change, and risk

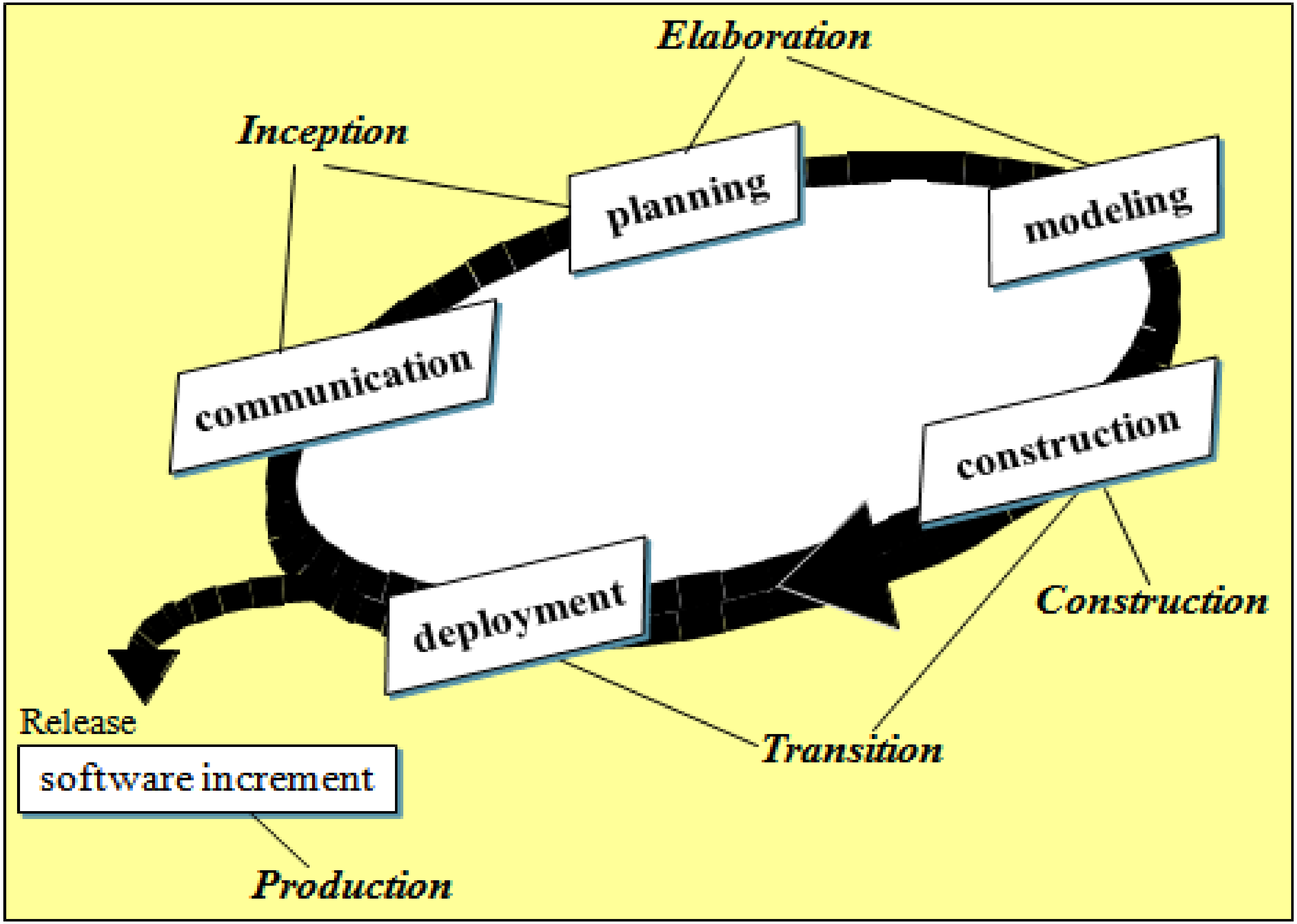
Typical umbrella activities include:

- Software project tracking and control
- Risk management
- Software quality assurance
- Technical reviews
- Software configuration management

Unified Process

- It is a use-case driven, architecture-centric, iterative and incremental software process
- An attempt to draw on the best features and characteristics of conventional software process models
- Phases are more concerned to businesses rather than technical concerns
- A framework for object-oriented software engineering using UML (Unified Modeling Language)

Phases of the Unified Process



Inception

- Encompasses both customer communication and planning activities
- Fundamental business requirements are described through a set of preliminary use-cases
  - A use-case describes a sequence of actions that are performed by an actor (e.g., a person, a machine, another system) as the actor interacts with the software
- A rough architecture for the system is also proposed

Elaboration

- Encompasses customer communication and modeling activities
- Refines and expands the preliminary use-cases
- The goals are to
  - Develop an understanding of the problem domain
  - Establish an architectural framework for the system
  - Develop the project plan
  - Identify key project risks

Construction

- The construction phase involves
  - System design
  - Programming
  - Testing
- Makes each use-case operational for end-users
- Parts of the system are developed in parallel and integrated during this phase
- Result
  - A working software system and associated documentation that is ready for delivery to users

Transition

- The final phase is concerned with moving the system from the development community to the user community and making it work in a real environment
- The software team creates the necessary support information
  - User manuals
  - Trouble-shooting guides
  - Installation procedures
- At the conclusion of the transition phase, the software increment becomes a usable software release
- Result
  - A documented software system that is working correctly in its operational environment

Production

- Coincides with the deployment activity of the generic process
- The on-going use of the software is monitored
- Support for the operating environment (infrastructure) is provided
- Defect reports and requests for changes are submitted and evaluated

Summary

- Process development involves activities, resources, and product
- Process model includes organizational, functional, behavioural, and other perspectives
- A process model is useful for guiding team behaviour, coordination and collaboration
- Process model should not only describe series of tasks, but also should detail factors that contribute to a project's inherent uncertainty and risk
- Dynamic Modelling enables enaction of process to see what happens to resources and artifacts as activities occur