CMSC 128

Introduction to Software Engineering
Second Semester AY 2009-2010

jachermocilla@uplb.edu.ph

The Process

- What is a software process?
  - framework for the tasks required to build high-quality software
  - approach that is taken as software is engineered
- SE includes methods and automated tools used in the process
- SE is performed by creative and knowledgeable people within defined and matured software process

Software Engineering

"..the establishment and use of <u>sound engineering principles</u> in order to <u>obtain economically</u> software that is <u>reliable</u> and <u>works efficiently on real machines</u>"

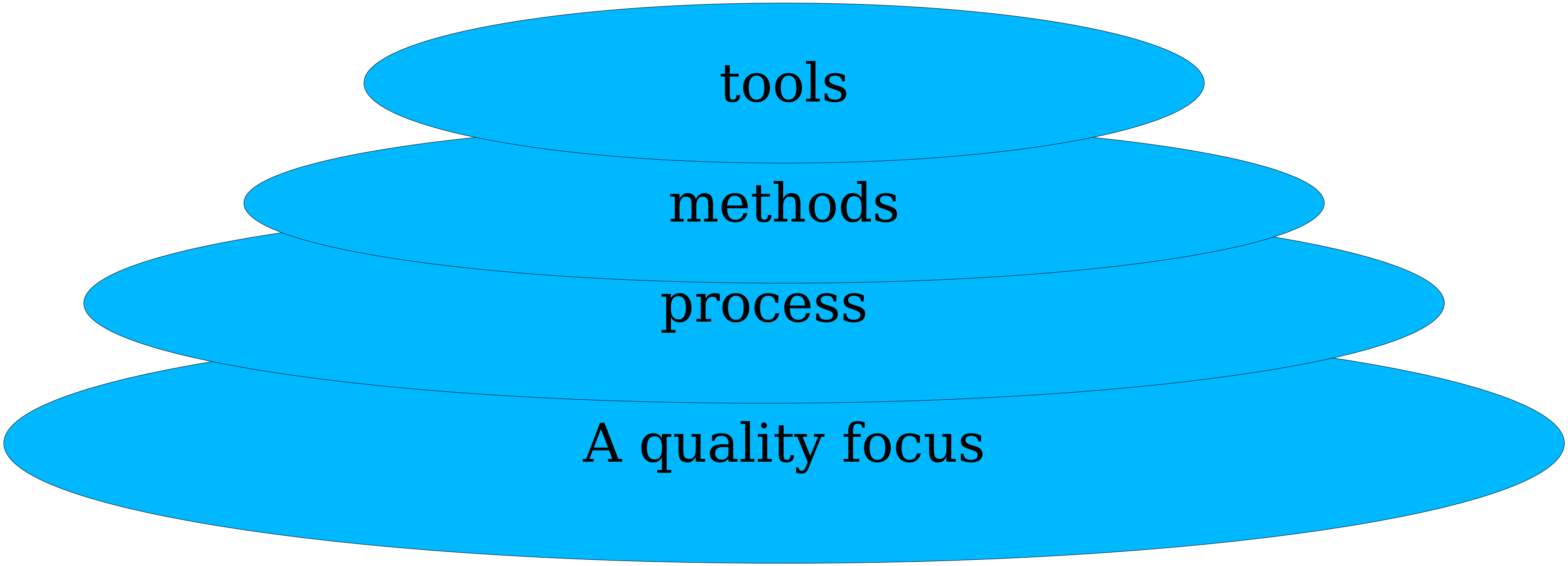
-Fritz Bauer

Software Engineering

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

- IEEE

Layered Technology

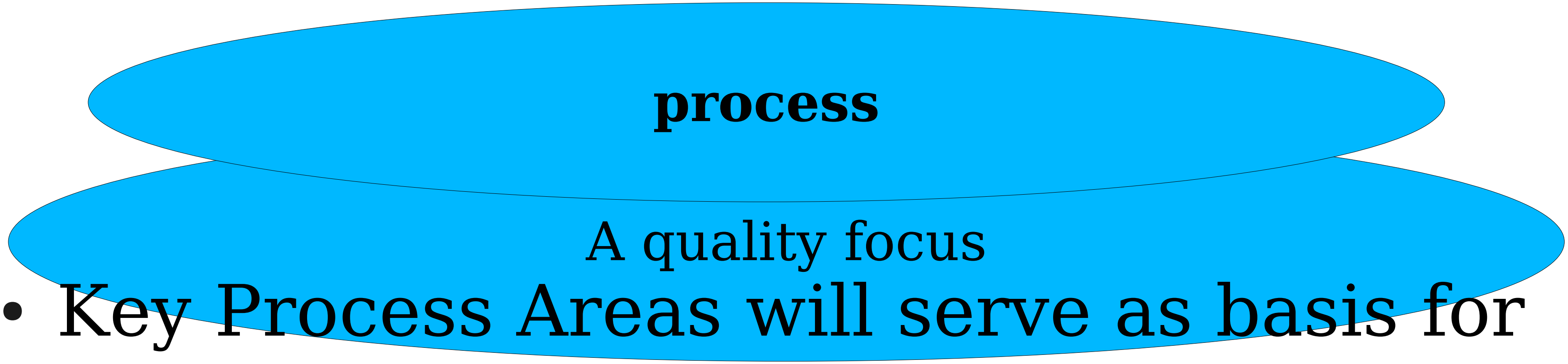


Layered Technology

- Commitment to quality
- Continuous process improvement culture
- The *bedrock* that support SE is a focus on quality

Layered Technology

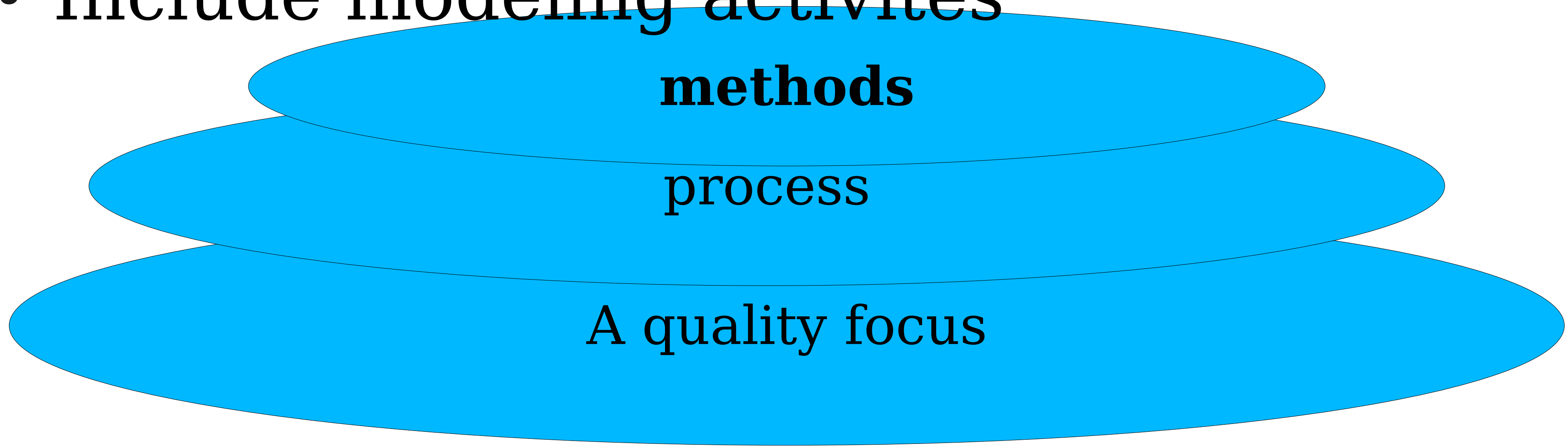
- *foundation* for SE
- Glue that holds technology layers together



- Key Process Areas will serve as basis for management control and establish the context in which technical methods are used

Layered Technology

- Provides technical how-to's for building software
- Include modeling activites



Layered Technology

- Provides automated or semi-automated support for process and methods

- Computer-Aided Software Engineering (CASE)

Engineering

..is the analysis, design, construction, verification, and management of technical (social) entities.

Engineering

- What is the problem to be solved?
- What are the characteristics of the entity that is used to solve the problem?
- How will the entity (and the solution) be realized?
- How will the entity be constructed?
- How will the errors be corrected?
- How will the entity be supported later?

Generic View of SE

- In SE, software is the entity
- Work assosiated with SE can be categorized into three generic phases: *definition*, *development*, and *maintenance*
- Definition Phase
  - answers “what“
  - What information, what function
- *Tasks*:System/Information Engineering, Software Project Planning, Requirements Analysis

Generic View of SE

- Development Phase
  - Focuses on “how“
  - *Tasks*: software design, code generation, and software testing
- Maintenance Phase
  - focuses on “change“
  - *Tasks*: reapplies those from definition and development phases in the context of existing software

Generic View of SE

- Four types of change - maintenance
  - Correction
    - correct defects
  - Adaptation
    - accommodate changes in external environment
  - Enhancement
    - extends software beyond original functions
  - Prevention
    - make the software easy to correct, adapt, and enhance

Generic View of SE

- Other umbrella activities
  - Project tracking and control
  - Formal technical reviews
  - Quality assurance
  - Configuration management
  - Document preparation and production
  - Reusability management
  - Measurement
  - Risk management

Software Process

Common Process Framework

Framework Activities

Task Sets

Tasks

Milestones, deliverables

SQA points

Umbrella activities

Software Process

- Process maturity
  - How “mature“ is the software process used in an organization?
  - Different levels exists as described in the model developed by the SEI - Capability Maturity Model (CMM)
  - Each level defines key activities that must be present in the software process adopted by an organization

CMM

- Level 1- Initial
  - Ad hoc, chaotic
  - Success depends on individual effort
- Level 2 – Repeatable
  - Basic project management
  - Process discipline is in place
    - Allows to repeat earlier success on similar projects

CMM (2)

- Level 3 – Defined
  - Software process for mgt and eng'g is documented, standardized, and integrated into org-wide software process
- Level 4 – Managed
  - Measurement of software process and product quality
  - Quantitatively understood and controlled process

CMM (2)

- Level 5 – Optimizing
  - Quantitative feedback from process
  - Testing innovative ideas and technologies

KPA

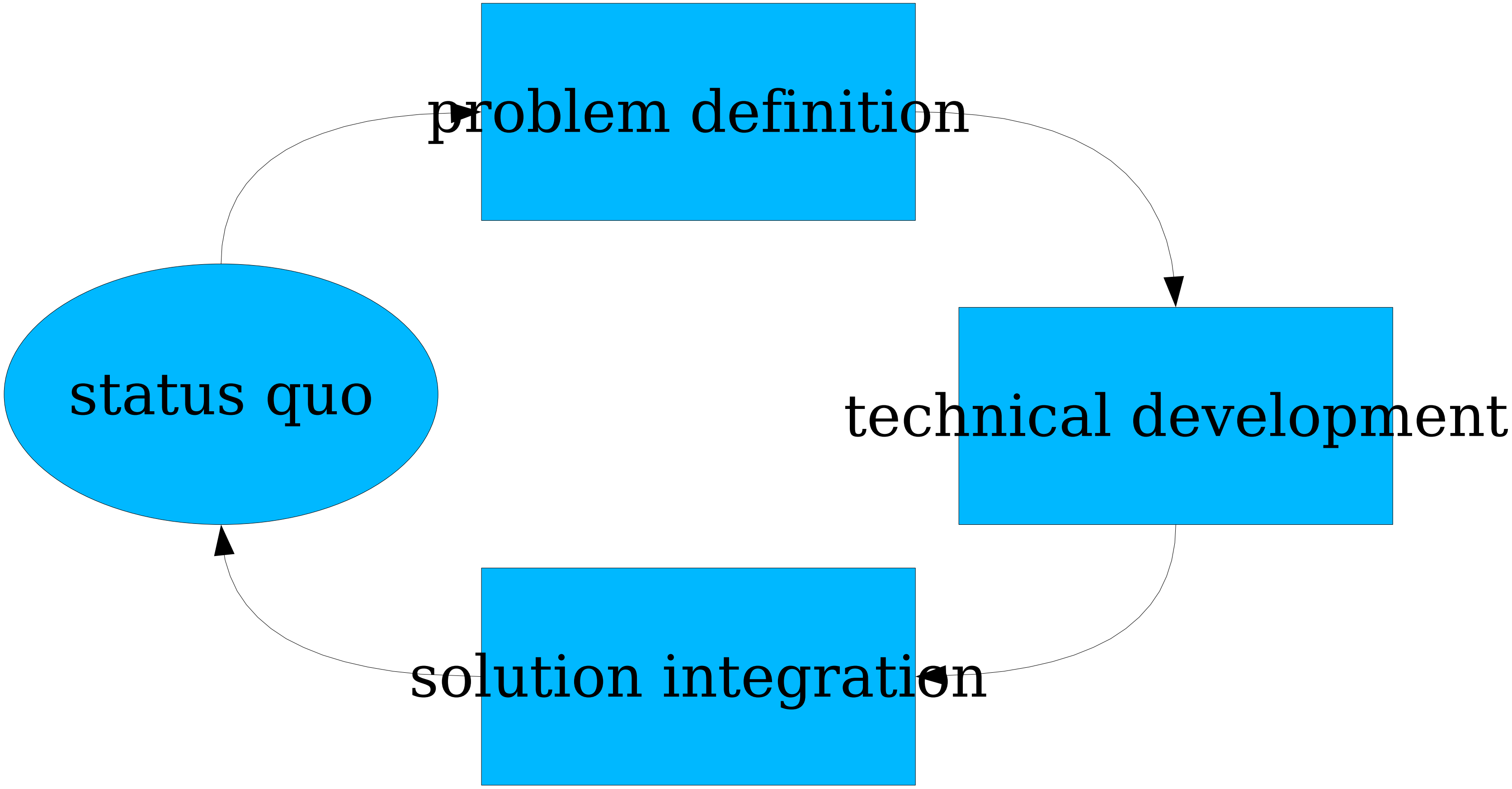
- Knowledge Process Area
  - Software engineering functions that must be present to satisfy good practice at a particular CMM level
  - ex. Software Configuration Management, Software Quality Assurance, Peer Reviews, Training Program, etc.
  - Each KPA is described by its: goals, commitments, abilities, activities, methods for monitoring and verifying implementation

Software Process Models

- Strategy that encompasses process and the generic phases for developing software
- Chosen based on the nature of the project and application, methods and tools to be used, and the controls and deliverables that are required

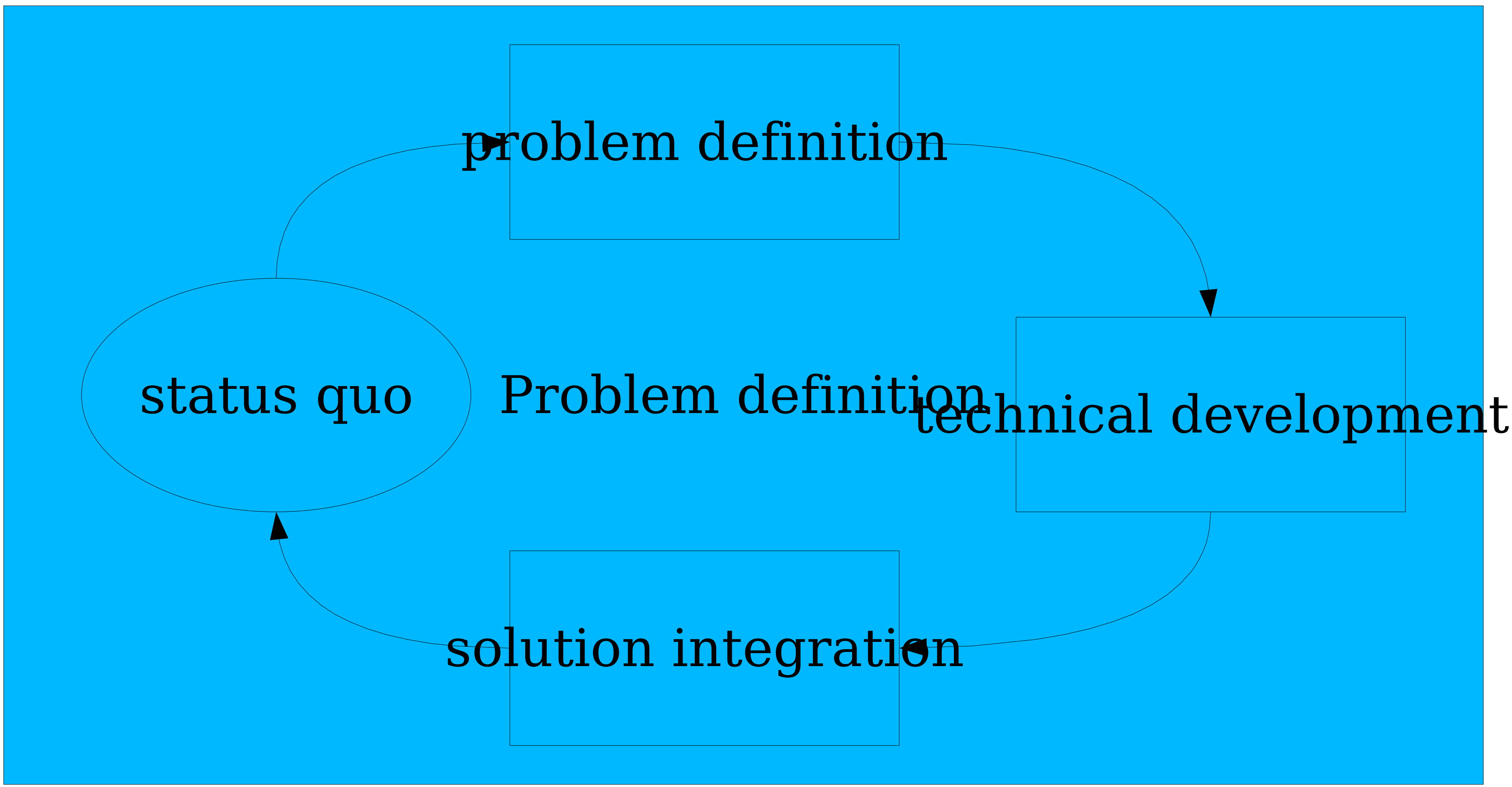
Software Process Models

- Problem solving loop

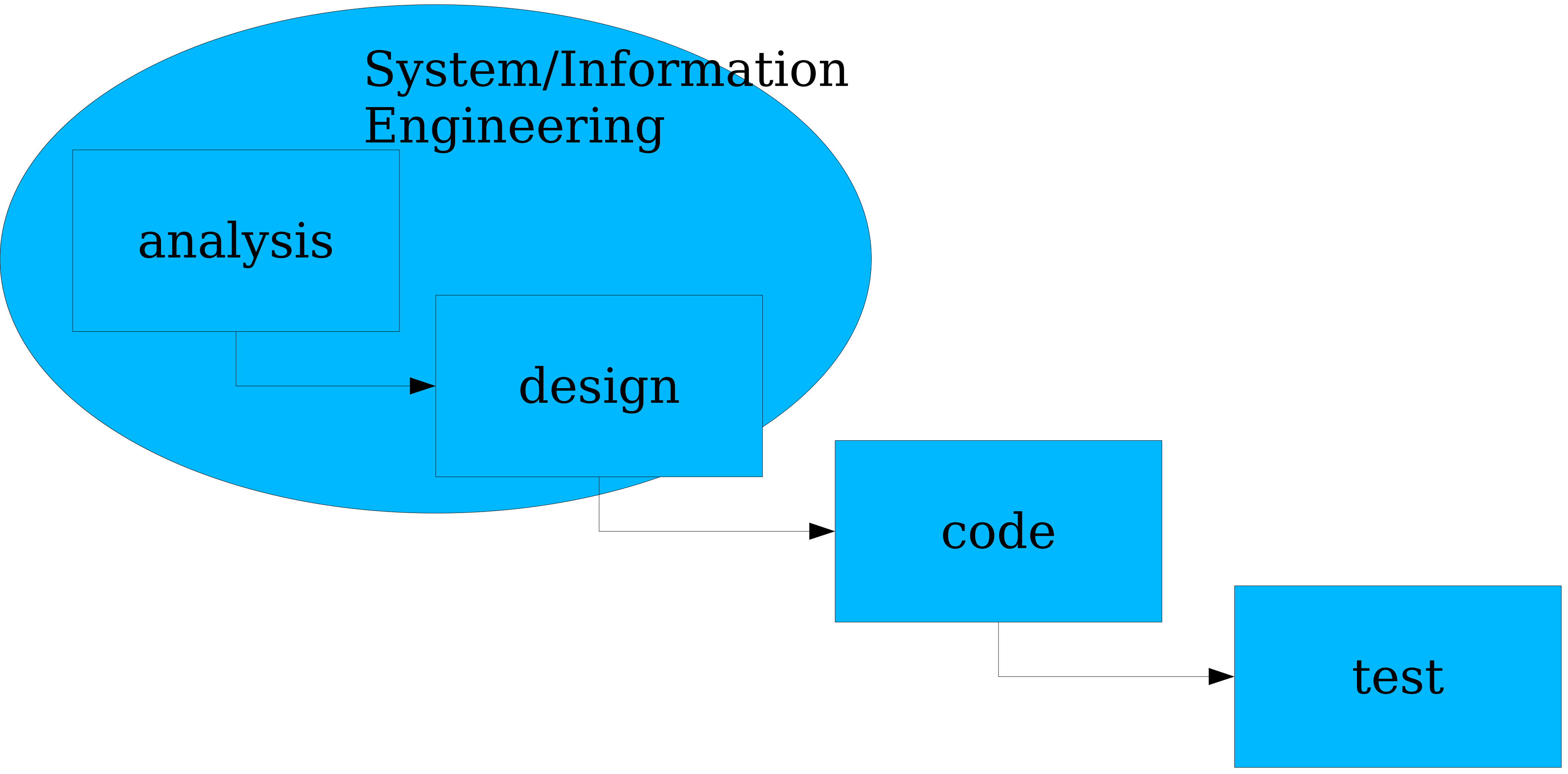


Software Process Models

- Problem solving loop-as fractal



Linear Sequential Model



Linear Sequential Model

- Also known as classic life cycle and waterfall model
- System/Information Engineering
  - Allocate subset of system requirements to software
  - Requirements gathering at the business level
  - Performs top-level analysis and design

Linear Sequential Model

- Software Requirements Analysis
  - Requirements gathering is focused on software
  - Understanding of the information domain, required function, behavior, performance, interfacing
  - Requirements are documented and reviewed with the customer

Linear Sequential Model

- Design
  - Multistep process
  - Focus on: data structures, software architecture, interface representations, procedural (algorithmic) detail
  - Translates requirements into software representation of software that can be assessed for quality
  - Design is documented and reviewed

Linear Sequential Model

- Code Generation
  - Design is translated into machine readable form
  - Can be accomplished mechanistically
- Testing
  - Focus on logical internals and functional externals of system
  - Assures all statements have been tested and correct output for well-defined input

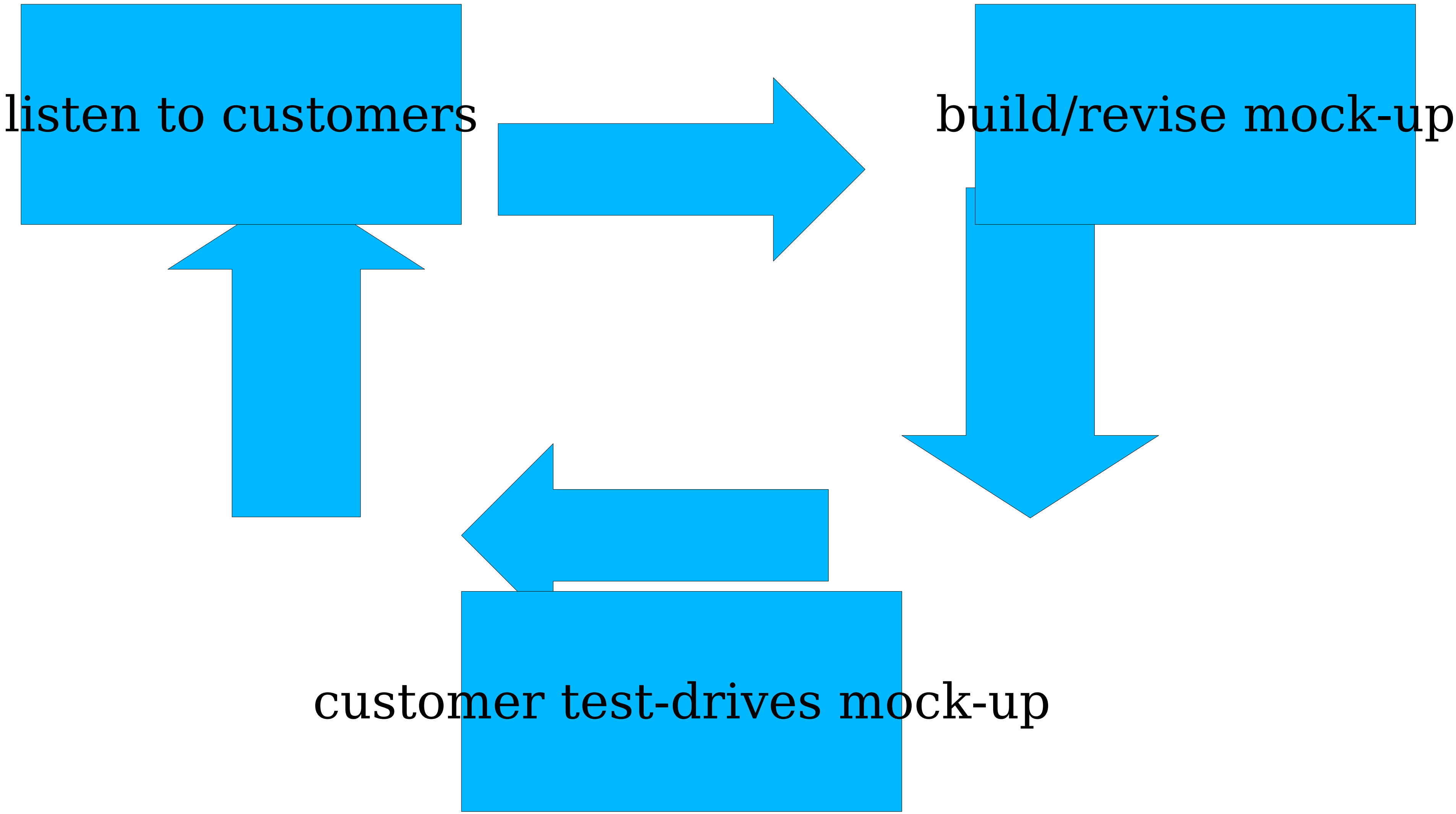
Linear Sequential Model

- Maintenance
  - Reapplies preceding phases to existing software to accomodate change
- Criticisms
  - Real projects don't follow sequential flow
  - Difficult for the customer to state requirements explicitly
  - No working product is delivered until late in the project
  - Developers are delayed-because of “waiting“

Prototyping Model

- No detailed requirements
- Unsure of the efficiency of the algorithm, adaptability of OS, or form of HCI
- A prototype is developed – aspects of software visible to customers
- Prototype used to refine requirements for the software to be developed
- Throw-away prototyping – prototype is discarded

Prototyping Model



Prototyping Model

- Criticisms
  - Customers think that the prototype will evolve into the final product
    - no guarantee for quality in the prototype
  - Developers make implementation compromises to quickly develop prototype
    - inefficient algorithm
    - developers forget decisions

RAD

- Rapid Application Development
- Linear sequential model that emphasizes short development life cycle
- Uses component-based construction
- Used primarily for information system applications

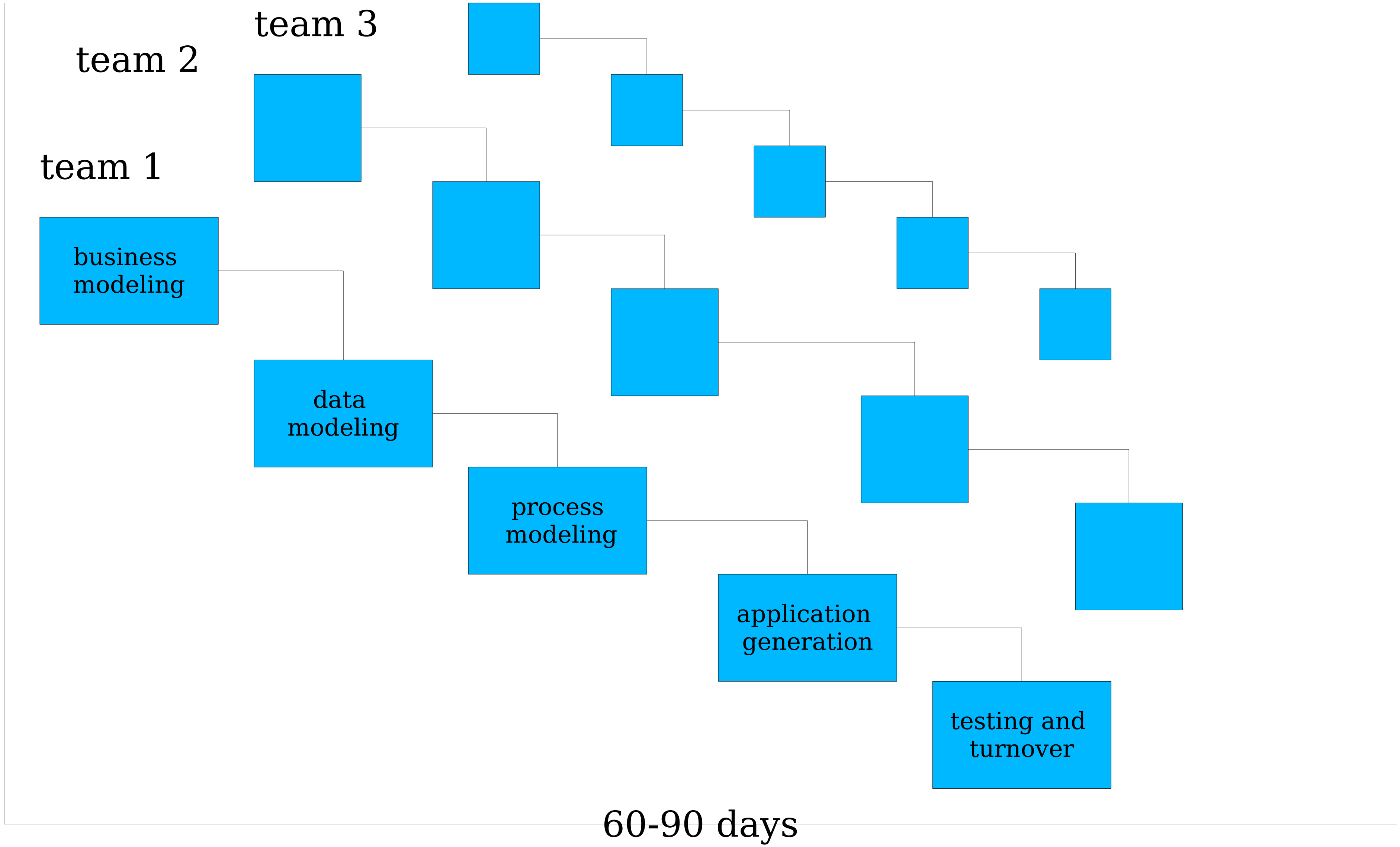
RAD

- Business Modeling
  - information flow among business functions
- Data Modeling
  - data objects are modeled to support business model
- Process Modeling
  - processing descriptions

RAD

- Application Generation
  - RAD assumed 4G techniques (component-based)
  - Automated tools are used to generate code
- Testing and Turnover
  - Assumes components have been tested
  - Testing of interfaces (connection points)

RAD


team 3
team 2
team 1
business modeling
data modeling
process modeling
application generation
testing and turnover
60-90 days

RAD

- Drawbacks/Criticisms
  - Requires sufficient human resources to create RAD teams
  - Requires commitment from both developers and customers for rapid-fire development
  - Target system can be modularized
  - Not appropriate for projects with high technical risks-use of new technology breaks interfaces

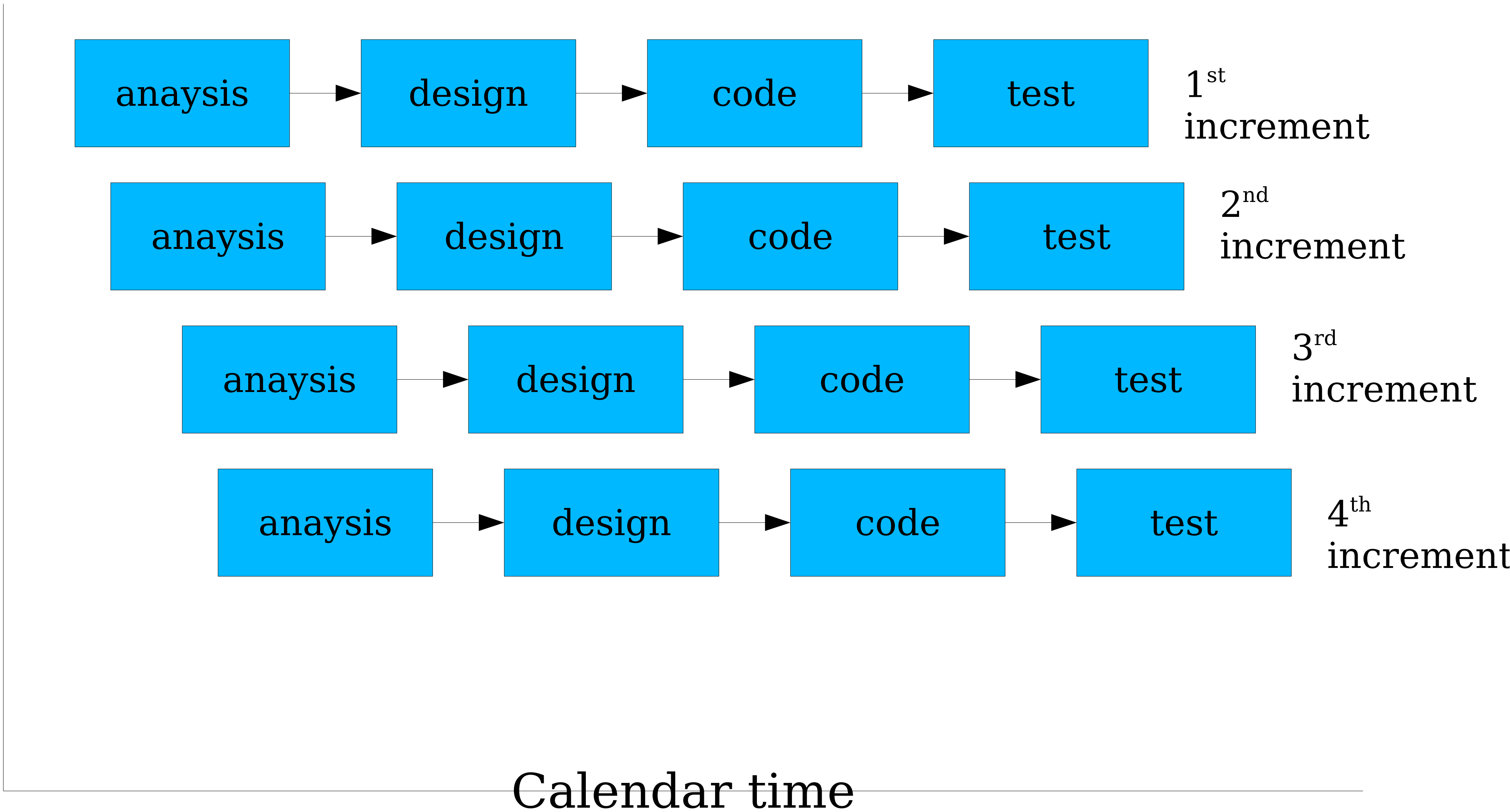
Evolutionary Models

- Software evolves over time
- Limited version must be delivered early to meet business competition and pressure
- Generally iterative
- Examples: Incremental, Spiral, Component Assembly,Concurrent

Incremental

- Linear sequential + iterative prototyping
- Produces increments at each iteration
- Increment vs. Prototype?
  - Increments are working products(stripped down)
  - Protoype – thrown away
- First increment is a core product
  - Future increments are built on top of the core
- Used when staffing is unavailable, hardware

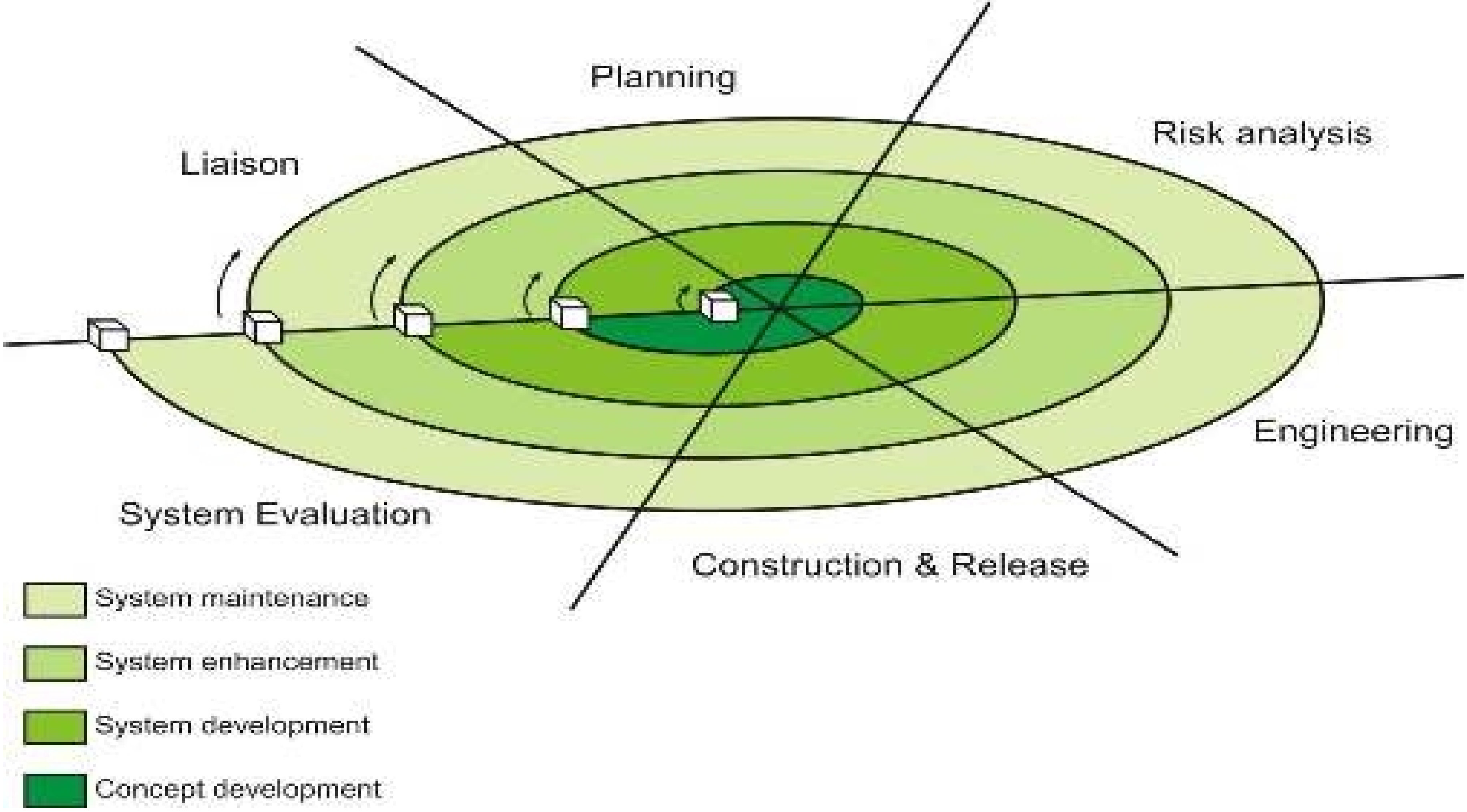
Incremental



Spiral Model

- Proposed by Boehm
- Software is developed in incremental releases
- Model is divided into framework activities or task regions

Spiral Model



http://www.srl.cam.ac.uk/genepi/images/Web_spiral_model2.jpg

Spiral Model

- Customer Communication/Liason
  - Task to establish effective communication between developer and customer
- Planning
  - Tasks to define resources, timelines, other project related information
- Risk Analysis
  - Tasks to assess technical and management risks

Spiral Model

- Engineering
  - Tasks to build one or more representations of the application
- Construction and Release
  - Tasks required to construct, test, install, and provide user support
- Customer Evaluation
  - Task to obtain customer feedback

Spiral Model

- Can be applied throughout the life of software, from concept to finished product
- More realistic as software evolves as the process progresses
- Risks are considered-through prototyping

Spiral Model

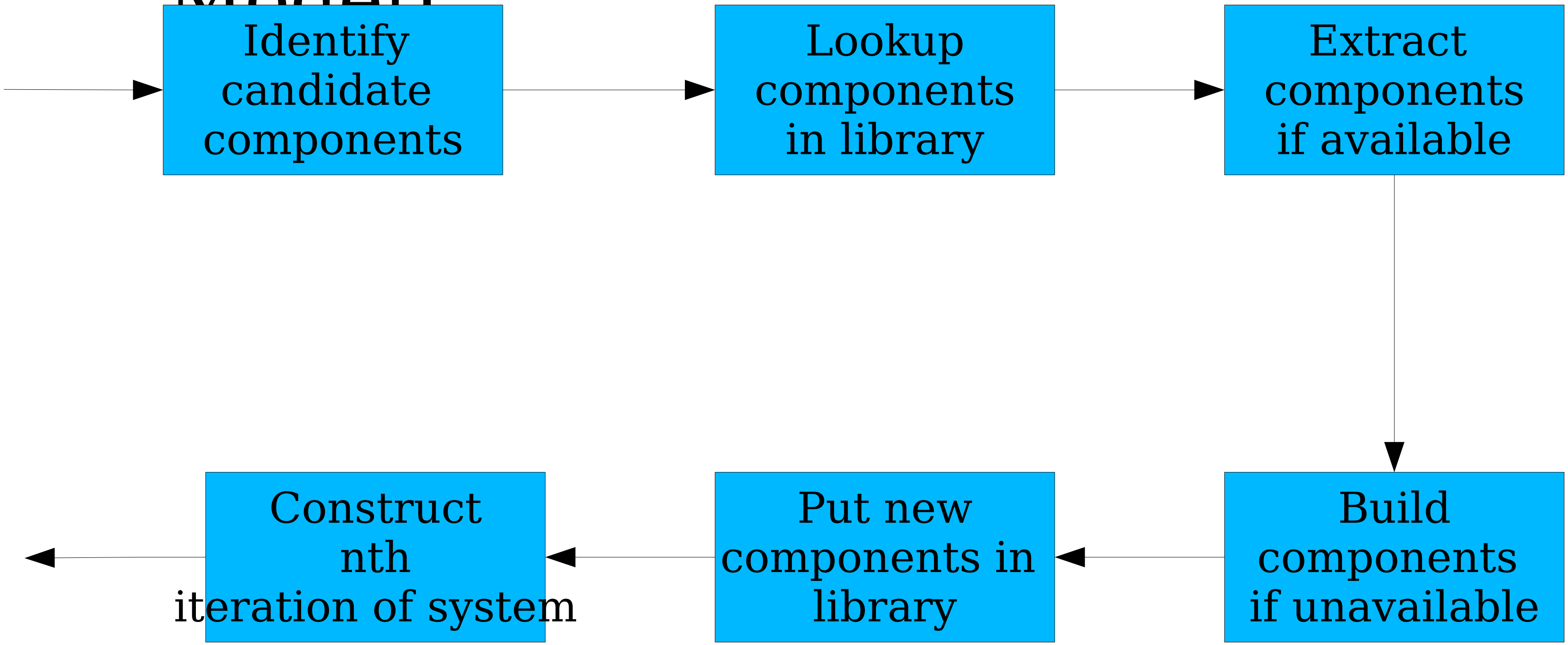
- Drawbacks
  - Difficult to convince customers that project is easy to control
  - Relies heavily on risk management
  - Model is relatively new – not used widely compared to waterfall or prototyping

Component Assembly

- Similar to spiral-iterative
- Relies on object-oriented technologies
- Composes applications from prepackaged components
- Leads to software reuse

Component Assembly

- Engineering task region(see Spiral Model)

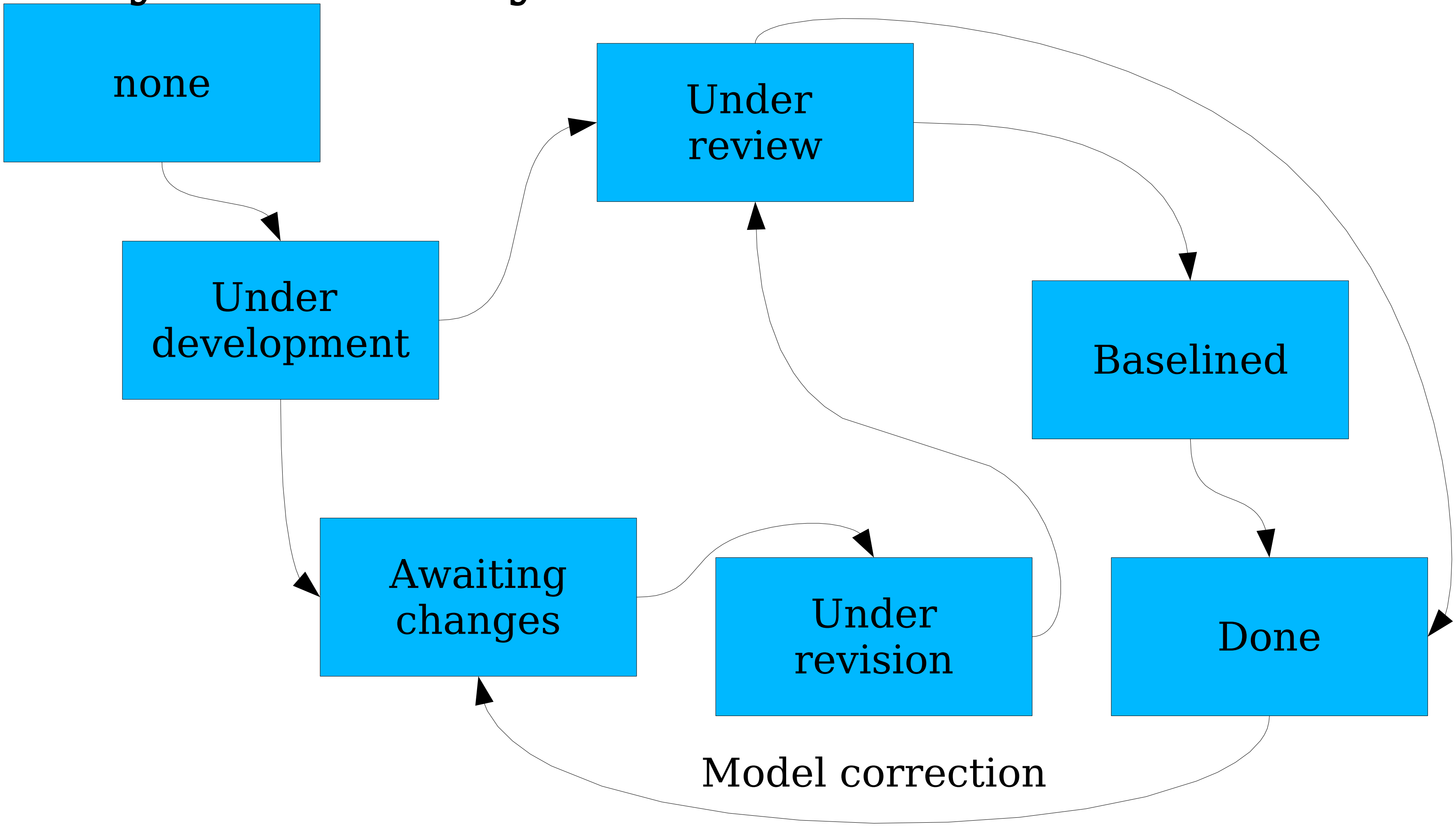


Concurrent Development

- Concurrent activities occur in each phase of the classic life cycle
- Each activity/task is represented with associated states
- State change happens based on events
- Used as paradigm in client-server applications

Concurrent Development

- Analysis activity



Formal Methods

- Leads to mathematical specification of software, for safety critical
- Eliminates ambiguity, incompleteness, and inconsistency
- Basis for verification
- Drawbacks
  - Time consuming and expensive
  - Requires extensive training
  - Difficult to communicate models to customers

4G

- Encompass tools that allows software engineers to specify some characteristics of software at a high level
  - Code generation
- Non-procedural languages for database query, report generation, screen manipulation
- Works side by side with component assembly

Product and Process

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

Summary

- SE is an engineering discipline that integrates process, methods, and tools
- Different models exists, each having its own strengths and weaknesses, all having a series of generic phases in common

Reference

- Roger S. Pressman.Software Engineering: A Practitioner's Approach, 4th Ed.McGraw-Hill,1997. Chapter 2