MAY, 2012

POWERGATE
THE SYSTEMS DEVELOPMENT LIFECYCLE (SDLC)

THE TALENT POOL IN VIETNAM

PowerGate Software
The Talent Pool in Vietnam

SDLC - CONTENTS

- Duration: 2 hours
- Agenda:
  - The Systems Development Lifecycle (SDLC)
  - SDLC Phases
  - SDLC Models

SDLC

- SDLC stands for
  - Systems
  - Development
  - Life
  - Cycle
- What does it mean?
  - First, SDLC is a Life Cycle.
  - All systems have a life cycle or a series of stages they naturally undergo.
    - The number and name of the stages varies, but the primary stages are conception, development, maturity and decline.
    - The **systems development life cycle** (SDLC) therefore, refers to the development stage of the system's life cycle.

SDLC

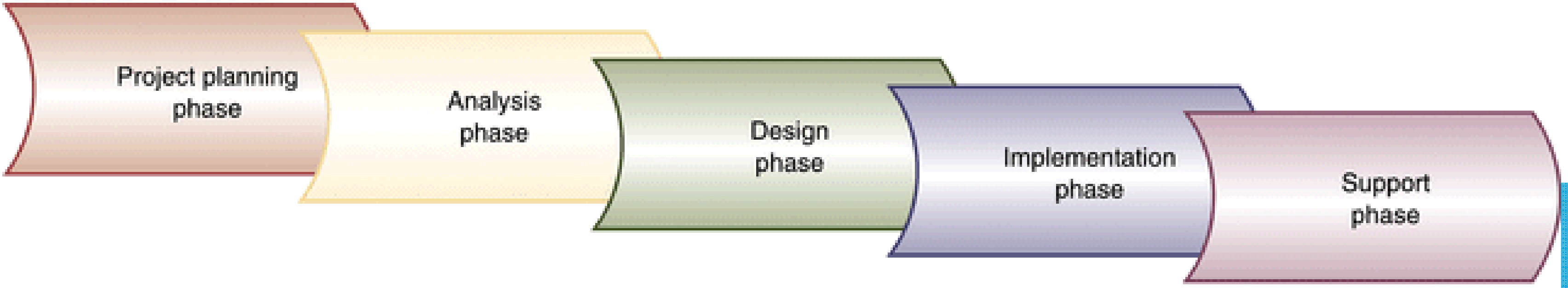
- Project
  - a planned undertaking that has a beginning and an end, and which produces a predetermined result or product.
- Systems development project
  - Planned undertaking
  - Large job
  - Produces new system
- Successful project requirements
  - Detailed plans
  - Organized, methodical sequence of tasks and activities

SDLC

- Systems development life cycle (SDLC)
  - Provides overall framework for managing systems development process
- Two main approaches to SDLC
  - Predictive approach – assumes project can be planned out in advance
  - Adaptive approach – more flexible, assumes project cannot be planned out in advance
- The SDLC is composed of 5 major Phases
  - Grouping of related activities

SDLC

- Three major activities
  - Analysis: understanding business needs
  - Design: conceptualizing computer-system solution
  - Implementation: construction, testing, and installation
- Two additional phases
  - Project planning
  - Support

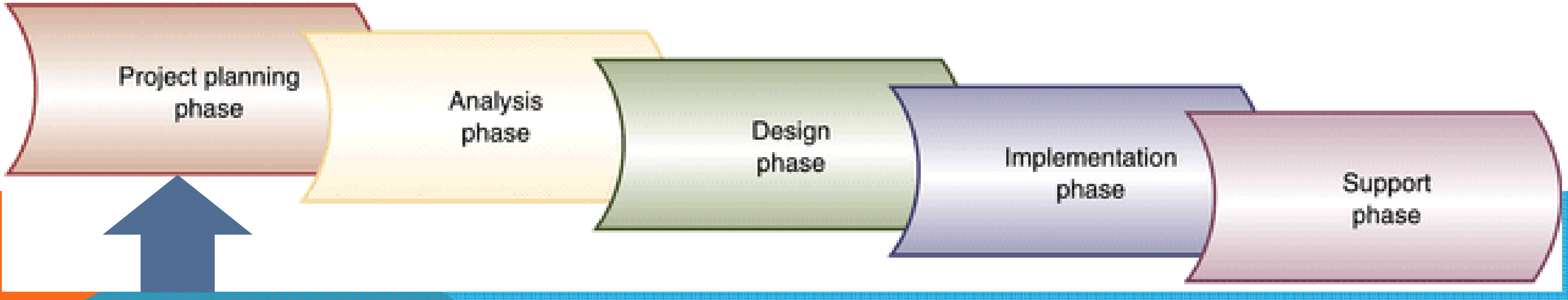


SDLC CONCEPTS

- SDLC is more than phases
  - Principles of management
  - Planning and control
  - Organization and scheduling
  - Problem solving

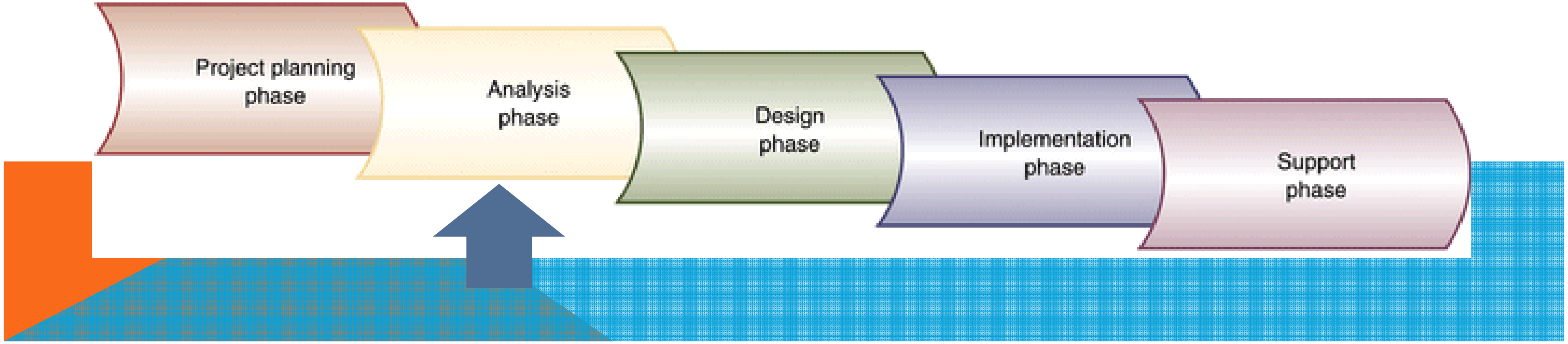
SDLC – PLANNING PHASE

- Define problem
- Confirm project feasibility
- Produce project schedule
- Staff the project
- Launch the project



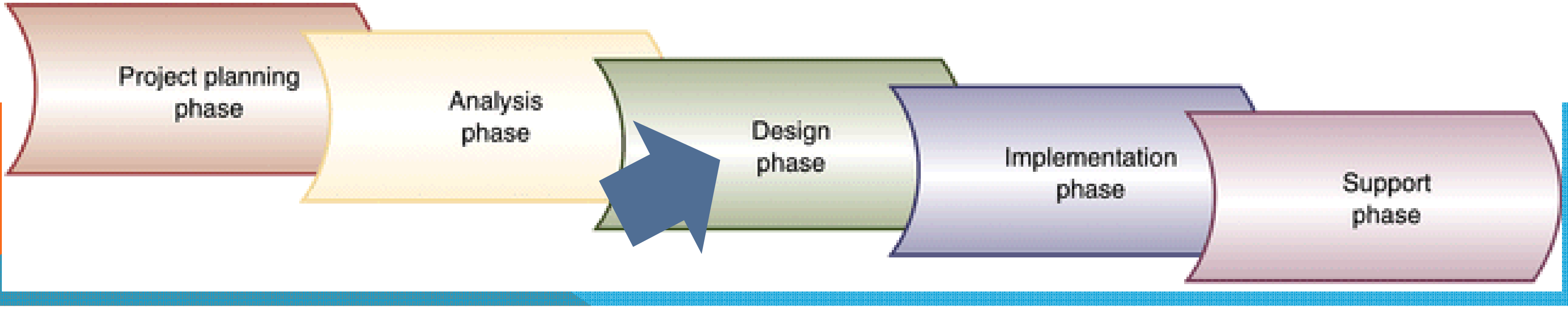
SDLC – ANALYSIS PHASE

- Gather information
- Define system requirements
- Build prototypes for discovery of requirements
- Prioritize requirements
- Generate and evaluate alternatives
- Review recommendations with management



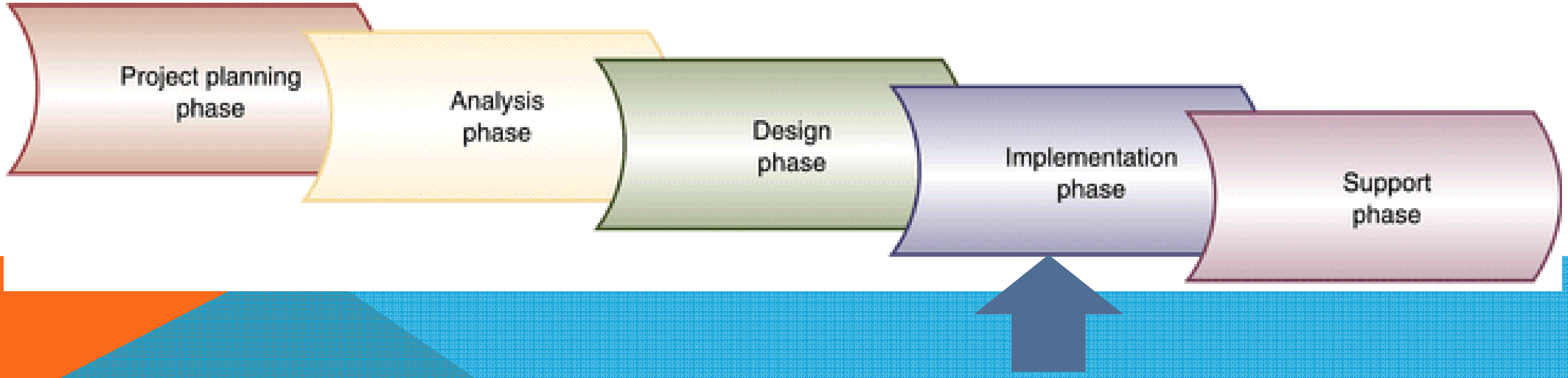
SDLC – DESIGN PHASE

- High Level Design (Architecture)
  - Design and integrate the network
  - Design the application architecture
- Low Level Design
  - Design the user interfaces
  - Design the system interfaces
  - Design and integrate the database
  - Prototype for design details
  - Design and integrate the system controls



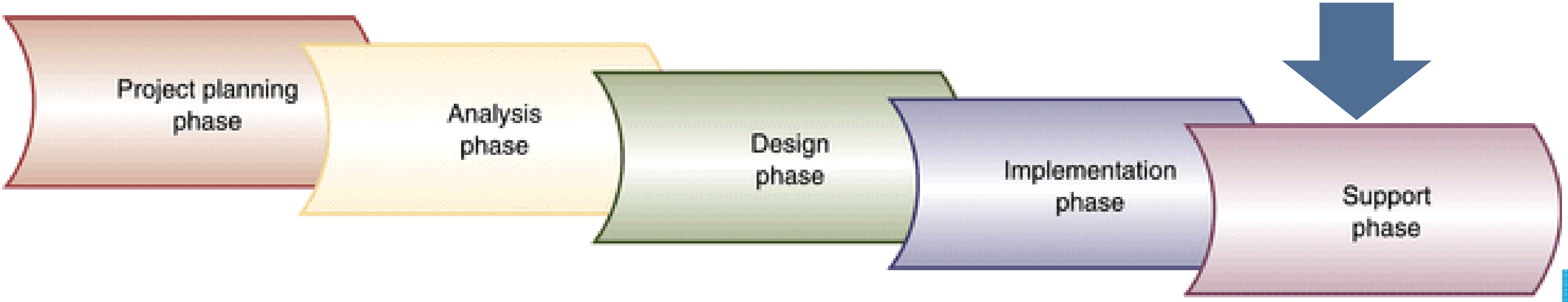
SDLC – IMPLEMENTATION PHASE

- Construct software components
- Verify and test
- Convert data
- Train users and document the system
- Install the system



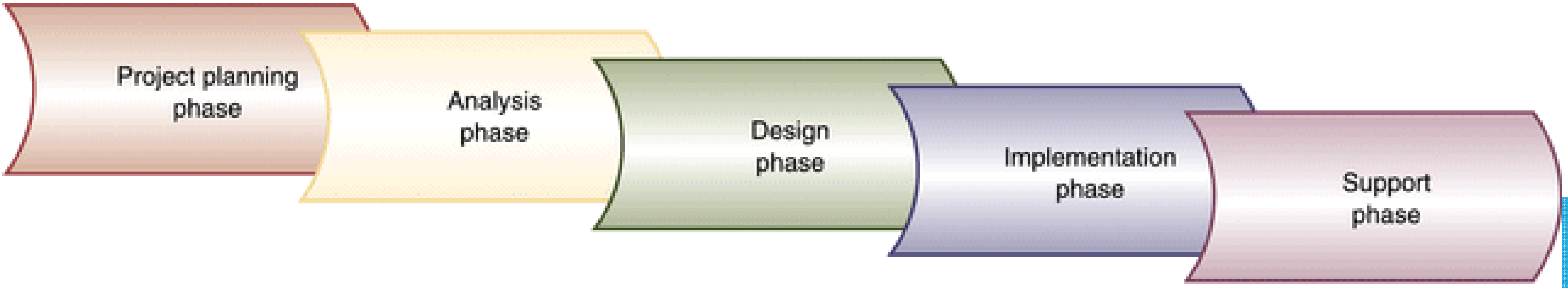
SDLC – SUPPORT PHASE

- Maintain the system
- Enhance the system
- Support the users
  - Help desk



SDLC – TRADITIONAL "WATERFALL"

- Waterfall Strengths
  - Easy to understand, easy to use
  - Provides structure to inexperienced staff
  - Milestones are well understood
  - Sets requirements stability
  - Good for management control (plan, staff, track)
  - Works well when quality is more important than cost or schedule



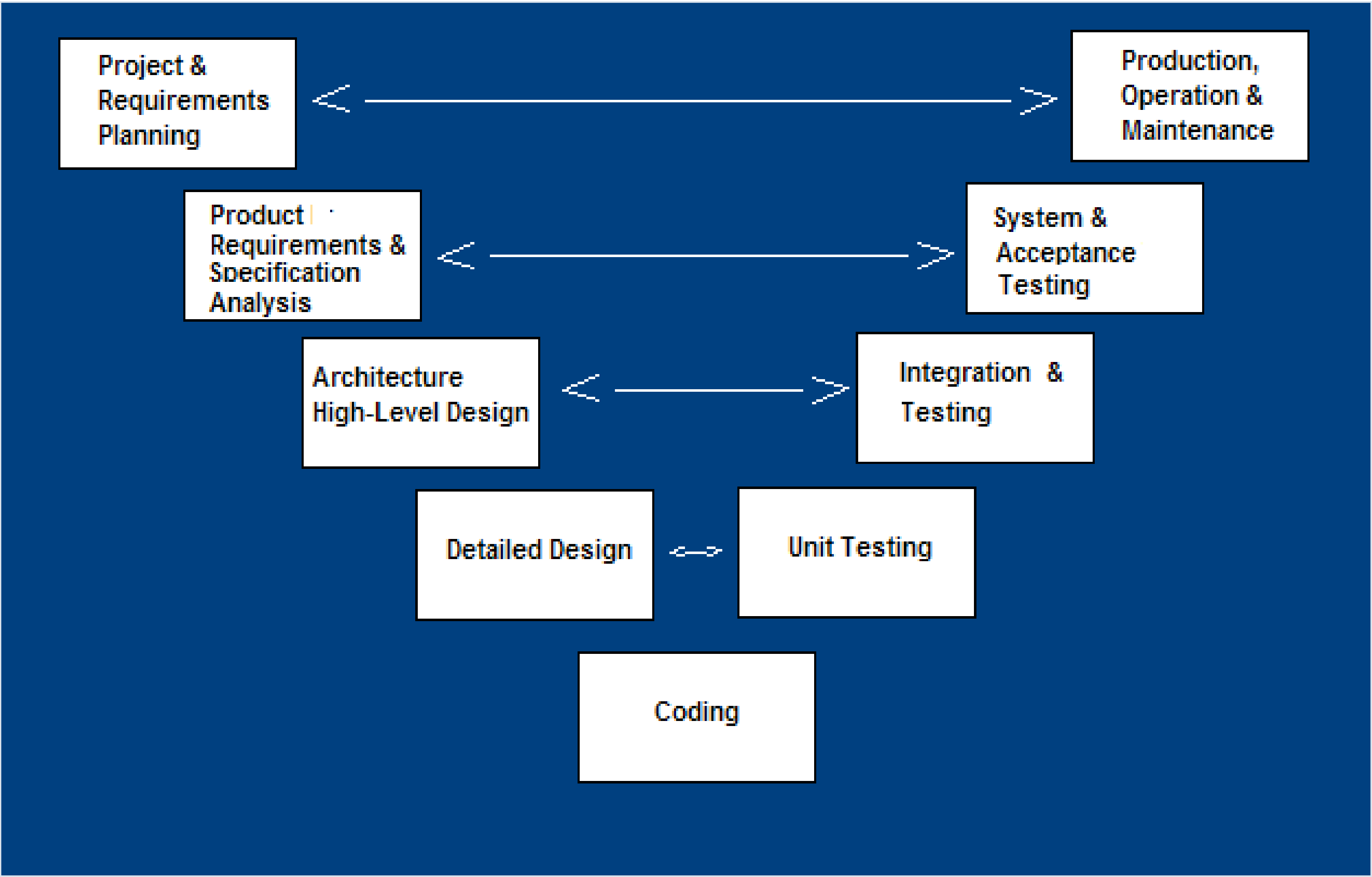
SDLC – TRADITIONAL “WATERFALL”

- Waterfall Deficiencies
  - All requirements must be known upfront
  - Deliverables created for each phase are considered frozen – inhibits flexibility
  - Can give a false impression of progress
  - Does not reflect problem-solving nature of software development – iterations of phases
  - Integration is one big bang at the end
  - Little opportunity for customer to preview the system (until it may be too late)

SDLC – TRADITIONAL "WATERFALL"

- When to use the Waterfall Model
  - Requirements are very well known
  - Product definition is stable
  - Technology is understood
  - New version of an existing product
  - Porting an existing product to a new platform

SDLC – V-SHAPED


Project & Requirements Planning
Production, Operation & Maintenance
Product Requirements & Specification Analysis
System & Acceptance Testing
Architecture High-Level Design
Integration & Testing
Detailed Design
Unit Testing
Coding

SDLC – V-SHAPED

- V-Shaped Strengths
  - Emphasize planning for verification and validation of the product in early stages of product development
  - Each deliverable must be testable
  - Project management can track progress by milestones
  - Easy to use
- V-Shaped Weaknesses
  - Does not easily handle concurrent events
  - Does not handle iterations or phases
  - Does not easily handle dynamic changes in requirements
  - Does not contain risk analysis activities

SDLC – V-SHAPED

- When to use the V-Shaped Model
  - Excellent choice for systems requiring high reliability – hospital patient control applications
  - All requirements are known up-front
  - When it can be modified to handle changing requirements beyond analysis phase
  - Solution and technology are known

SDLC - STRUCTURED EVOLUTIONARY PROTOTYPING MODEL

- Developers build a prototype during the requirements phase
- Prototype is evaluated by end users
- Users give corrective feedback
- Developers further refine the prototype
- When the user is satisfied, the prototype code is brought up to the standards needed for a final product.

STRUCTURED EVOLUTIONARY PROTOTYPING STRENGTHS

- Customers can "see" the system requirements as they are being gathered
- Developers learn from customers
- A more accurate end product
- Unexpected requirements accommodated
- Allows for flexible design and development
- Steady, visible signs of progress produced
- Interaction with the prototype stimulates awareness of additional needed functionality

STRUCTURED EVOLUTIONARY PROTOTYPING WEAKNESSES

- Tendency to abandon structured program development for “code-and-fix” development
- Bad reputation for “quick-and-dirty” methods
- Overall maintainability may be overlooked
- The customer may want the prototype delivered.
- Process may continue forever (scope creep)

WHEN TO USE STRUCTURED EVOLUTIONARY PROTOTYPING

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development
- With the analysis and design portions of object-oriented development.

SDLC - RAPID APPLICATION MODEL (RAD)

- Requirements planning phase (a workshop utilizing structured discussion of business problems)
- User description phase – automated tools capture information from users
- Construction phase – productivity tools, such as code generators, screen generators, etc. inside a time-box. (“Do until done”)
- Cutover phase -- installation of the system, user acceptance testing and user training

RAD STRENGTHS

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
- Focus moves from documentation to code (WYSIWYG).
- Uses modeling concepts to capture information about business, data, and processes.

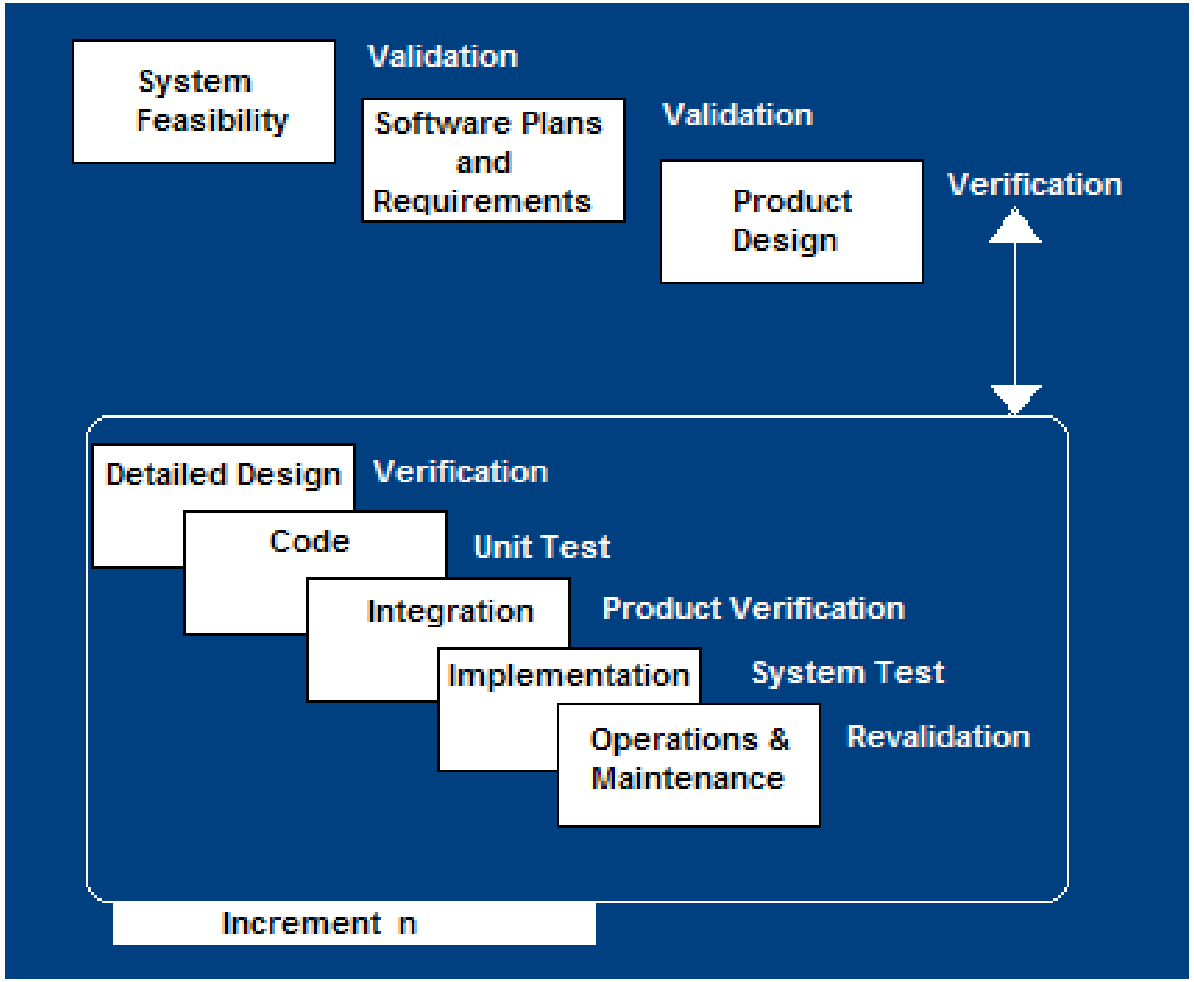
RAD WEAKNESSES

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

WHEN TO USE RAD

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
- High performance not required
- Low technical risks
- System can be modularized

INCREMENTAL SDLC MODEL



INCREMENTAL MODEL STRENGTHS

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses "divide and conquer" breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

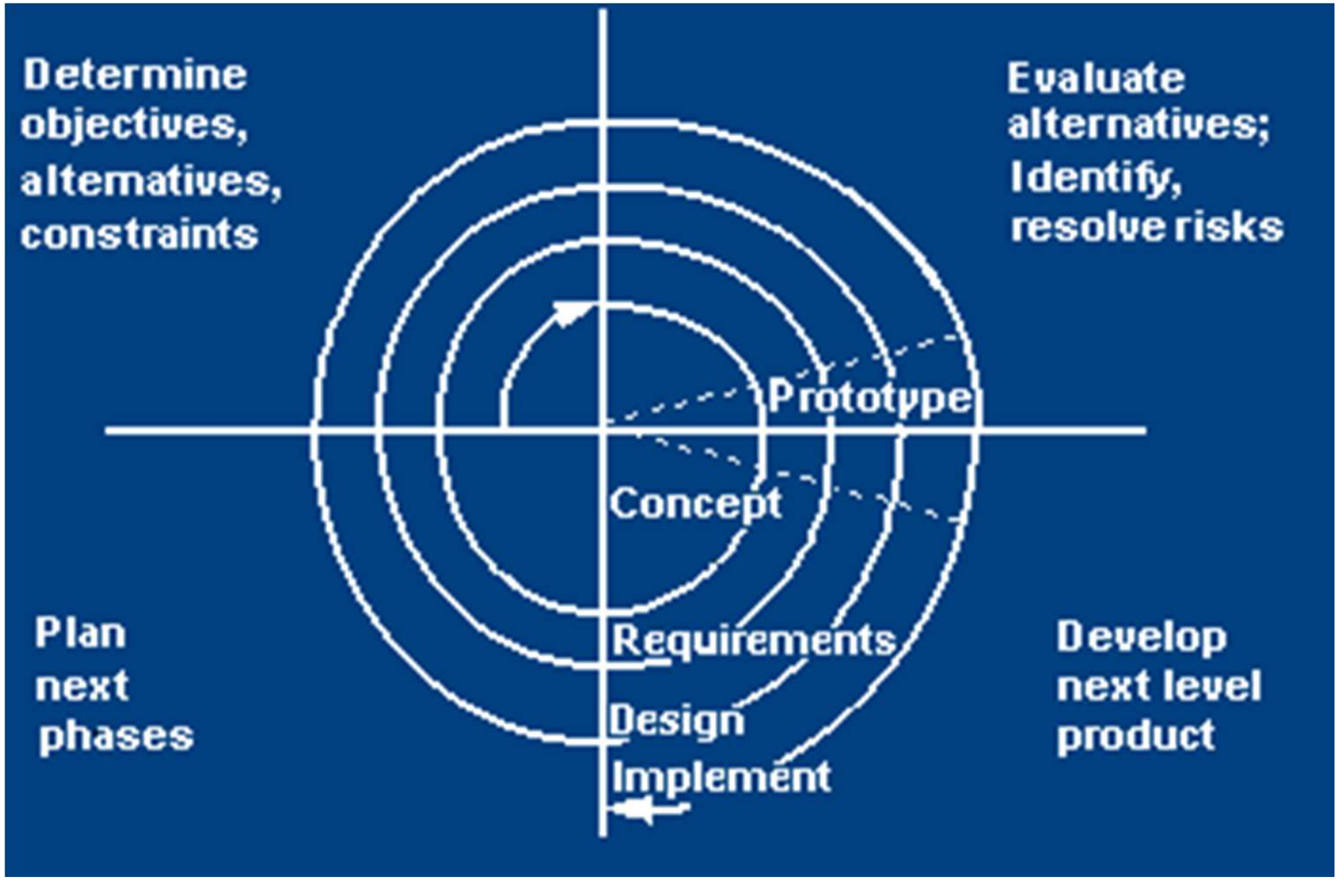
INCREMENTAL MODEL WEAKNESSES

- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

WHEN TO USE THE INCREMENTAL MODEL

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

SPIRAL SDLC MODEL


Determine objectives, alternatives, constraints
Evaluate alternatives; Identify, resolve risks
Prototype
Concept
Requirements
Design
Implement
Plan next phases
Develop next level product

SPIRAL MODEL STRENGTHS

- Provides early indication of insurmountable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

SPIRAL MODEL WEAKNESSES

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

WHEN TO USE SPIRAL MODEL

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

AGILE SDLC'S

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

SOME AGILE METHODS

- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Rapid Application Development (RAD)
- Scrum
- Extreme Programming (XP)
- Rational Unify Process (RUP)

EXTREME PROGRAMMING - XP

For small-to-medium-sized teams developing software with vague or rapidly changing requirements

Coding is the key activity throughout a software project

- Communication among teammates is done with code
- Life cycle and behavior of complex objects defined in test cases – again in code

FEATURE DRIVEN DESIGN (FDD)

Five FDD process activities

1. Develop an overall model – Produce class and sequence diagrams from chief architect meeting with domain experts and developers.
2. Build a features list – Identify all the features that support requirements. The features are functionally decomposed into Business Activities steps within Subject Areas.

   Features are functions that can be developed in two weeks and expressed in client terms with the template: <action> <result> <object>

   i.e. Calculate the total of a sale
3. Plan by feature -- the development staff plans the development sequence of features
4. Design by feature -- the team produces sequence diagrams for the selected features
5. Build by feature – the team writes and tests the code

TAILORED SDLC MODELS

- Any one model does not fit all projects
- If there is nothing that fits a particular project, pick a model that comes close and modify it for your needs.
- Project should consider risk but complete spiral too much – start with spiral & pare it done
- Project delivered in increments but there are serious reliability issues – combine incremental model with the V-shaped model
- Each team must pick or customize a SDLC model to fit its project