System Development Life Cycle

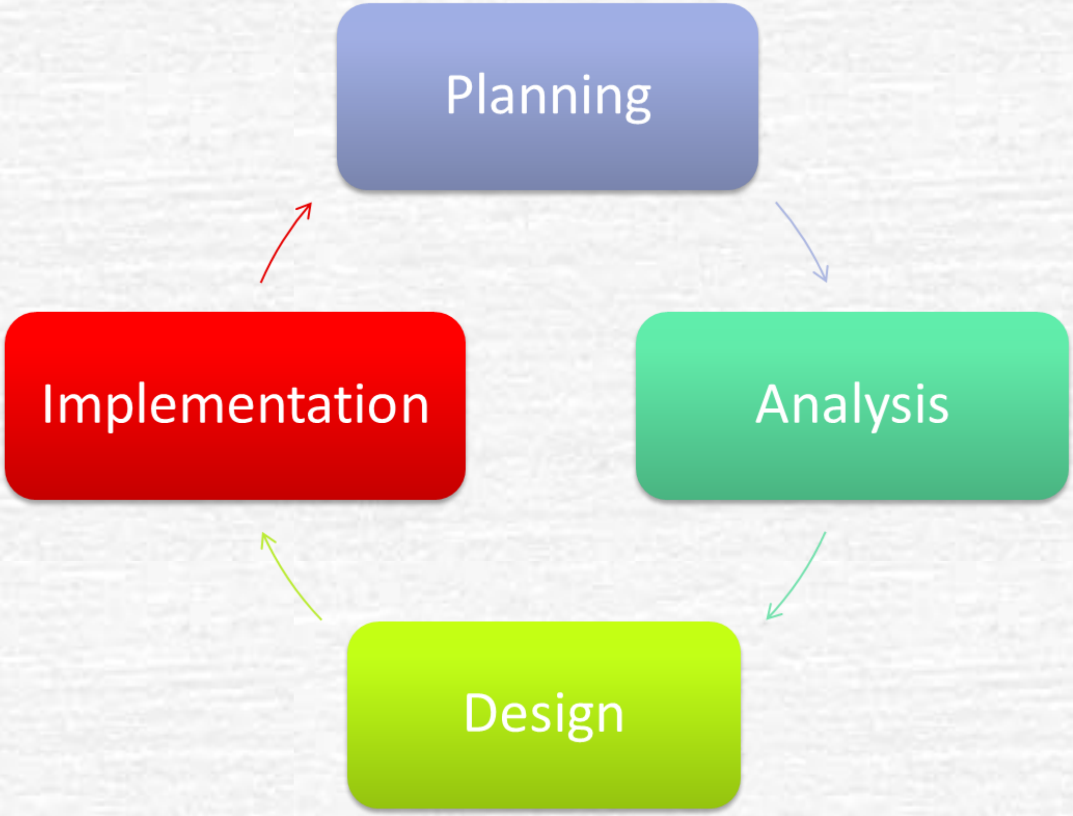
Ratna Wardani

Semester Genap, 2013

Outline

- SDLC
- Project Phase
- System Development Methodologies

SDLC



Project Phase

1. **Planning:** Why build the system?
   - System request, feasibility analysis, project size estimation
2. **Analysis:** Who, what, when, where will the system be?
   - Requirement gathering, business process modeling
3. **Design:** How will the system work?
   - Program design, user interface design, data design
4. **Implementation:** System construction and delivery
   - System construction, testing, documentation and installation

Planning

1. Identifying business value (System Request)
   - Lower costs
   - Increase profits
2. Analyze feasibility
   - Technical Feasibility
   - Economic Feasibility
   - Organizational Feasibility
3. Develop workplan and staffing (WBS)

Analysis

1. Requirement gathering by answering the questions:
   - Who will use the system?
   - What will the system do?
   - When will it be used?
2. Investigate the current system
3. Identify possible improvements
4. Develop a concept for new system

Design

1. **Program Design (UML Diagrams)**
   - What programs need to be written
   - Exactly what each program will do
2. **User Interface Design**
   - How users interact with system
   - Forms / reports used by the system
3. **Data Design (ER Diagrams)**
   - What data is to be stored
   - What format the data will be in
   - Where the data will be stored

Implementation

- **Construction**
  - New system is built and tested
  - Often testing is the longest part
- **Testing**
  - Unit Testing
  - Integration Testing
  - System Testing
  - User Acceptance Test
- **Installation**
  - Old system is turned off
  - New system is turned on

Processes and Deliverables

Process	Product
Planning	System Proposal
Analysis Design	System Specification
Implementation	New System with Testing/Maintenance Plan

System Development Methodologies

What is Methodology

- A formalized approach to implementing the SDLC (series of steps and deliverables)
- Writing code without a well-thought-out
- system request may work for small programs, but rarely works for large ones

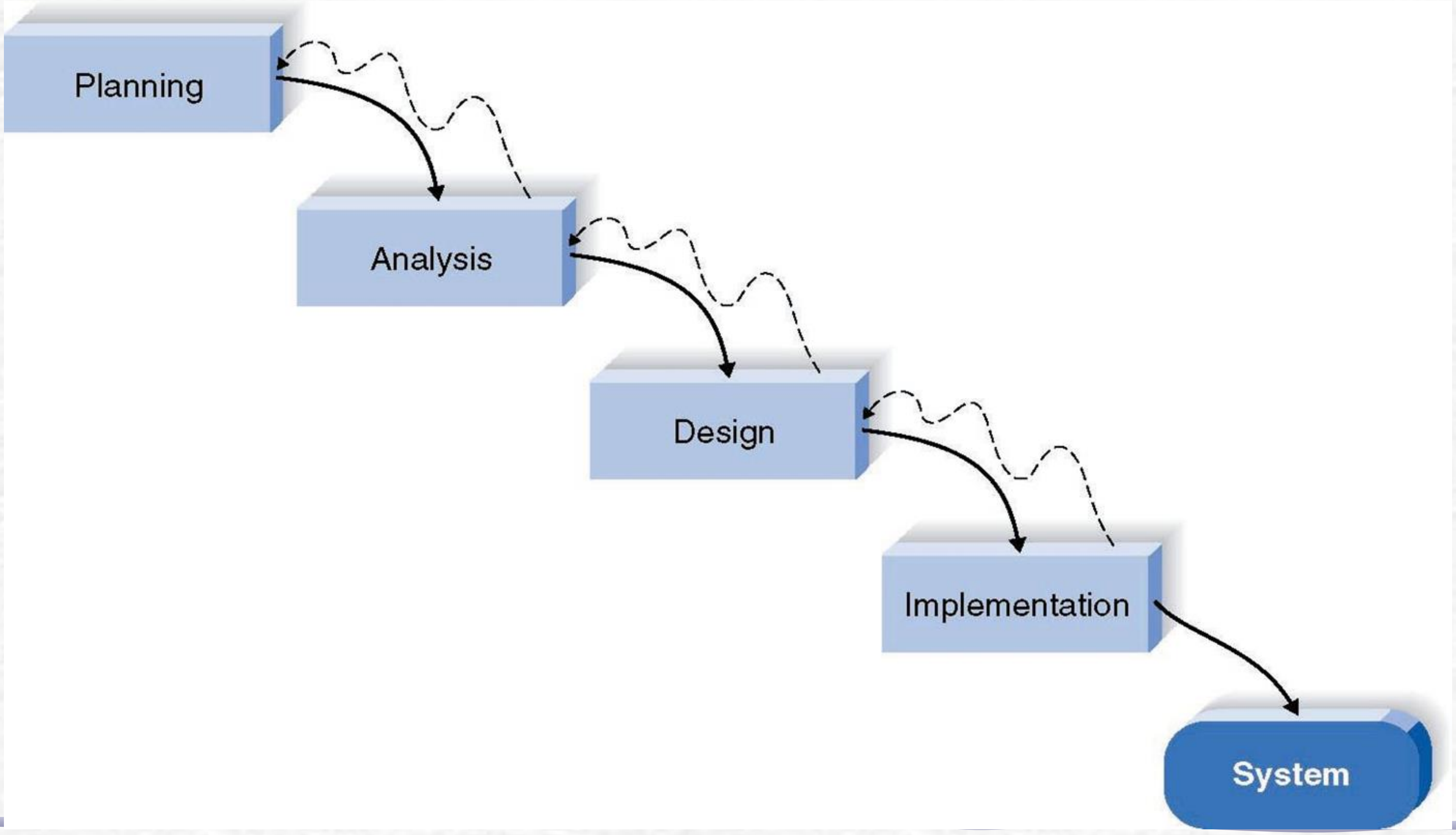
Major Methodologies

1. Structured Design
   - Waterfall method
   - Parallel development
2. RAD Development
   - Phased Development
   - Prototyping
   - Throw-away Prototyping
3. Agile Development
   - Extreme Programming (XP)
   - Scrum

Stuctured Design Methodology

- Projects move methodically from one to the next step
- Generally, a step is finished before the next one begins

Waterfall Method



Pros-Cons Waterfall Method

Pros	Cons
Identifies systems requirements long before programming Begins, it minimizes change to the requirements as the project proceed (mature)	Design must be specified on paper before programming begins
	Long time between system proposal and delivery of new system
	Rework is very hard

Parallel Development

- Addresses problem of time gap between proposal and delivery
- General process:
  1. Breaks project into parallel subproject
  2. Integrates them at the end

Parallel Development

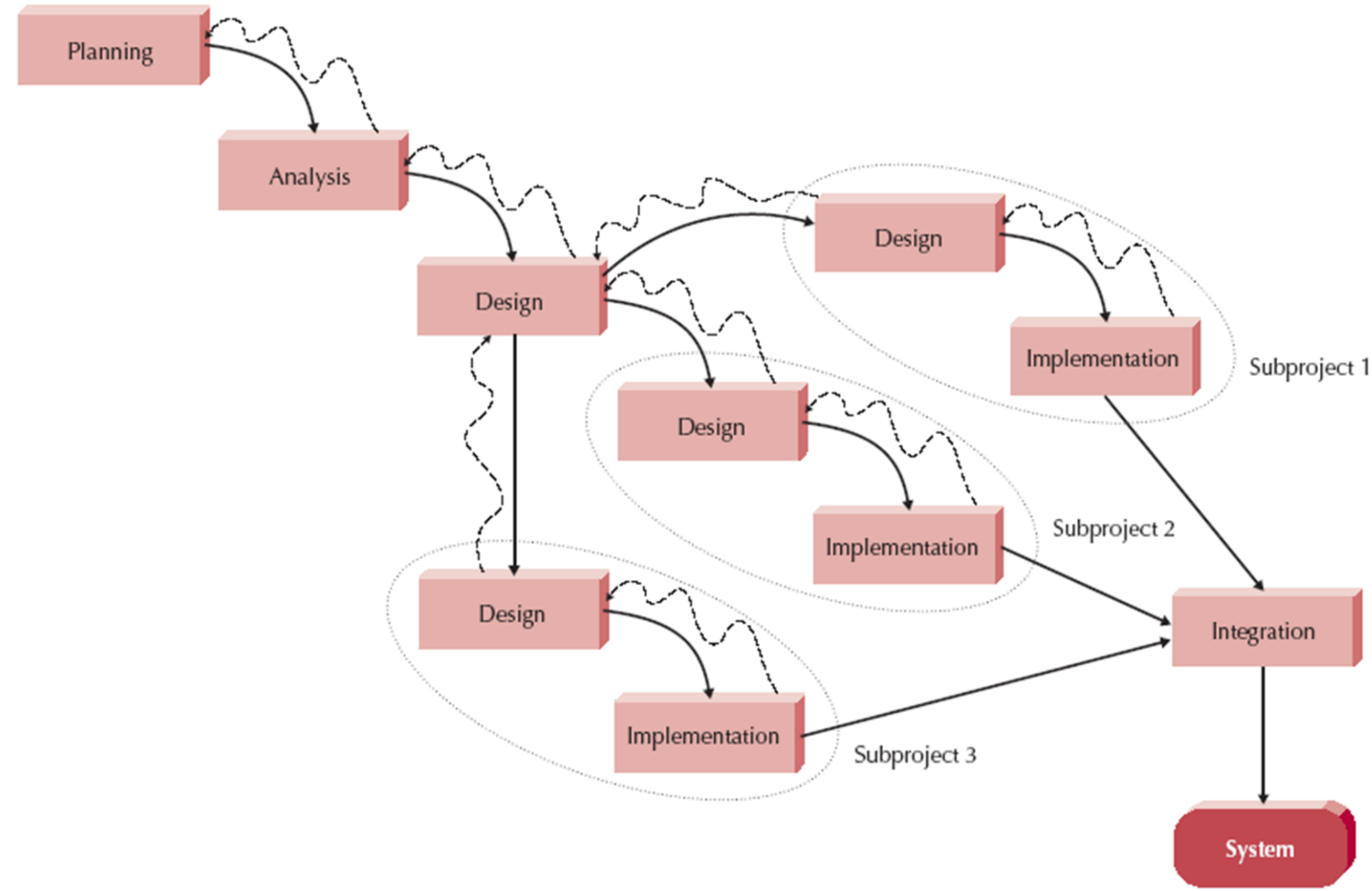


FIGURE 1-3 A Parallel Development-based Methodology

Rapid Application Development

1. Phased development
   - A series of versions
2. Prototyping
   - System prototyping
3. Throw-away prototyping
   - Design prototyping

Rapid Application Development

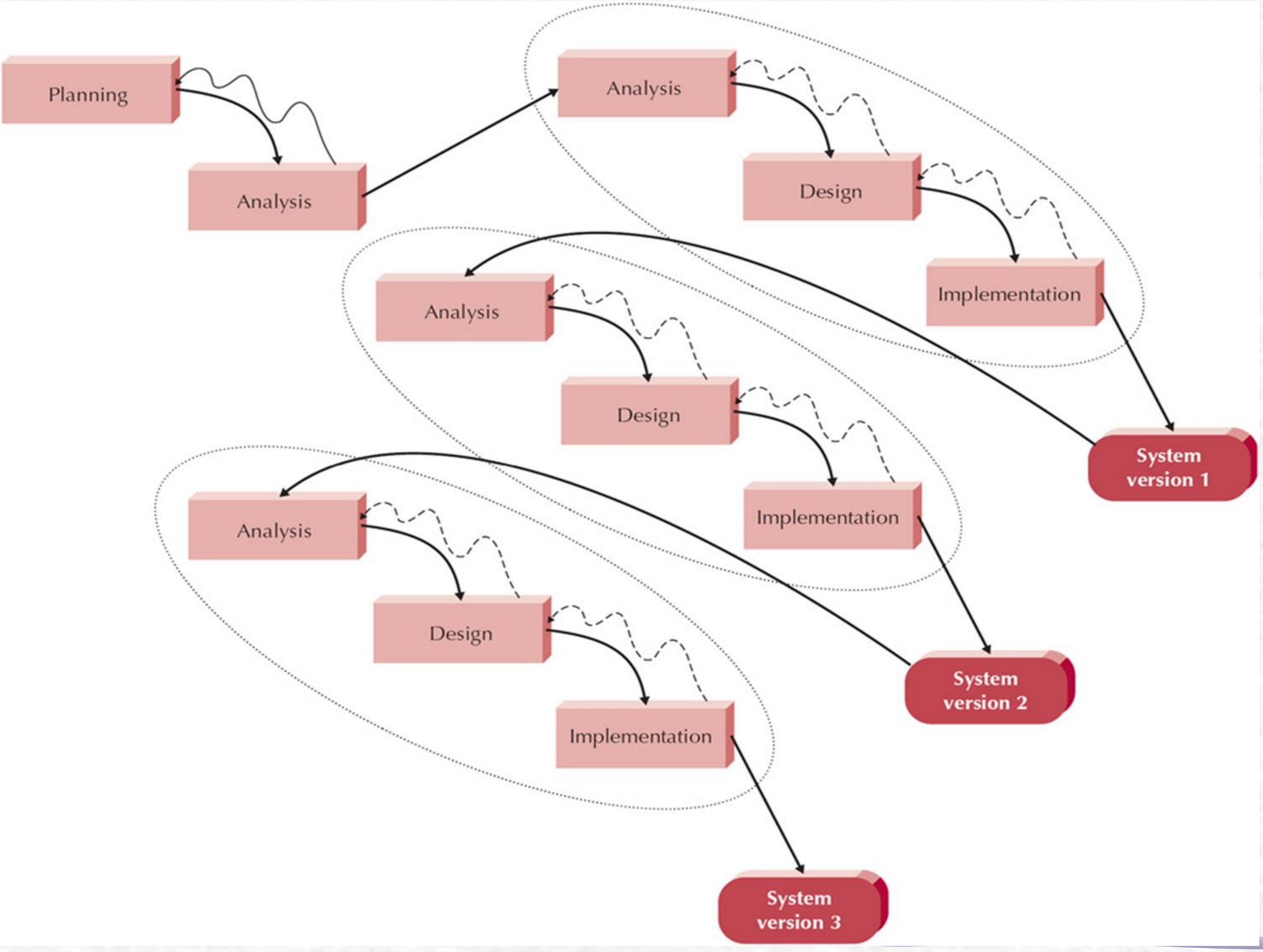
Critical elements to speed up the SDLC:

- CASE tools
- Visual programming languages
- Code generators

RAD: Phased Development

- Break overall system into a series of versions
- Each version has Analysis, Design, and Implementation
- Output from on version is the input to the next
- Incorporate ideas, issues, lessons learned in one version into the next version

RAD: Phased Development



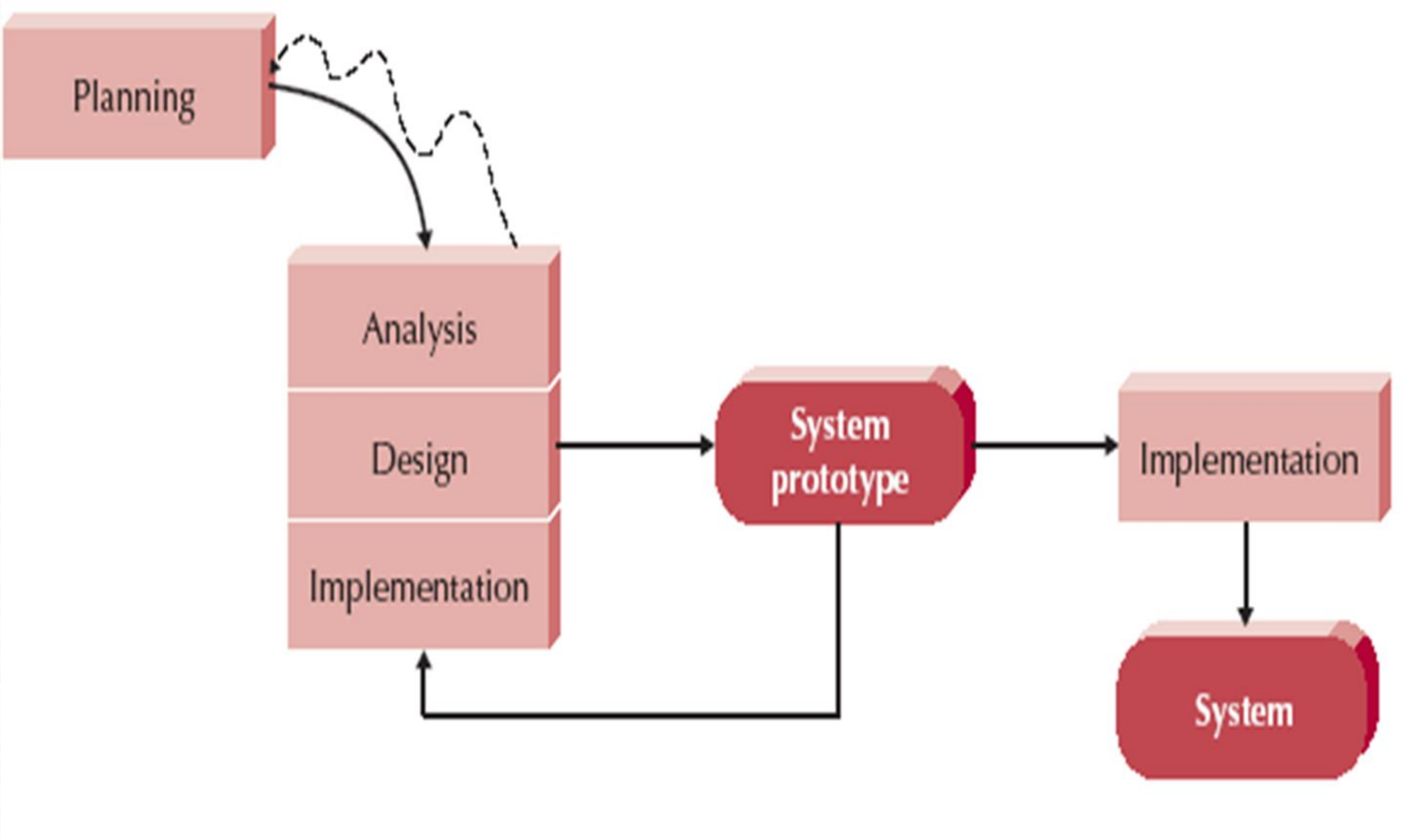
RAD: Phased Development

Pros	Cons
Gets useful system to users quickly	Initial system is intentionally incomplete
Most important functions tested most	System requirements expand as users see versions

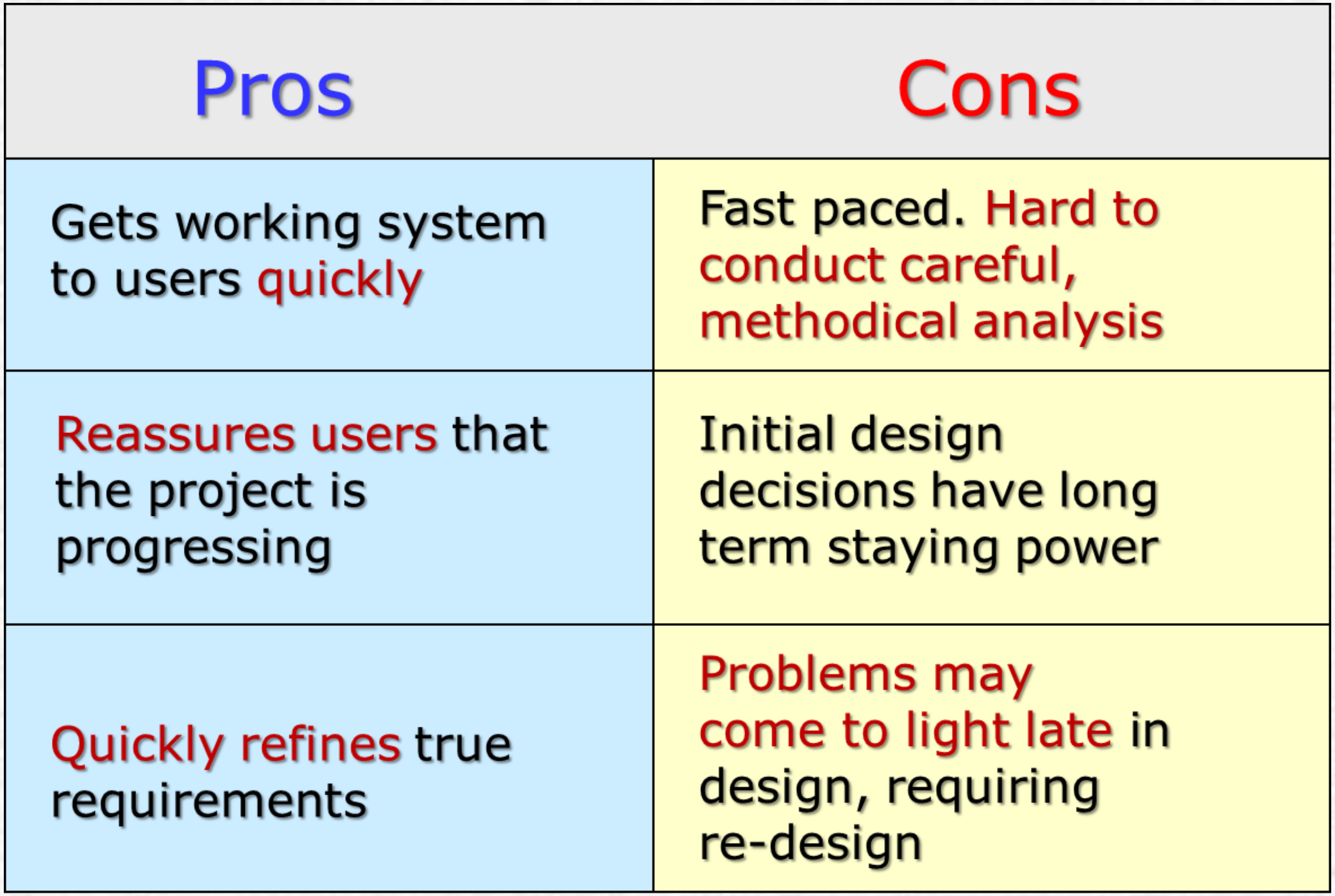
RAD: Prototyping

- Analysis, Design, Implementation are performed concurrently
- Start with a "quick-and-dirty" prototype
  - Provides minimal functionality
- Repeat process, refining the prototype each time
- Stop when prototype is a working system

RAD: Prototyping



RAD: Prototyping

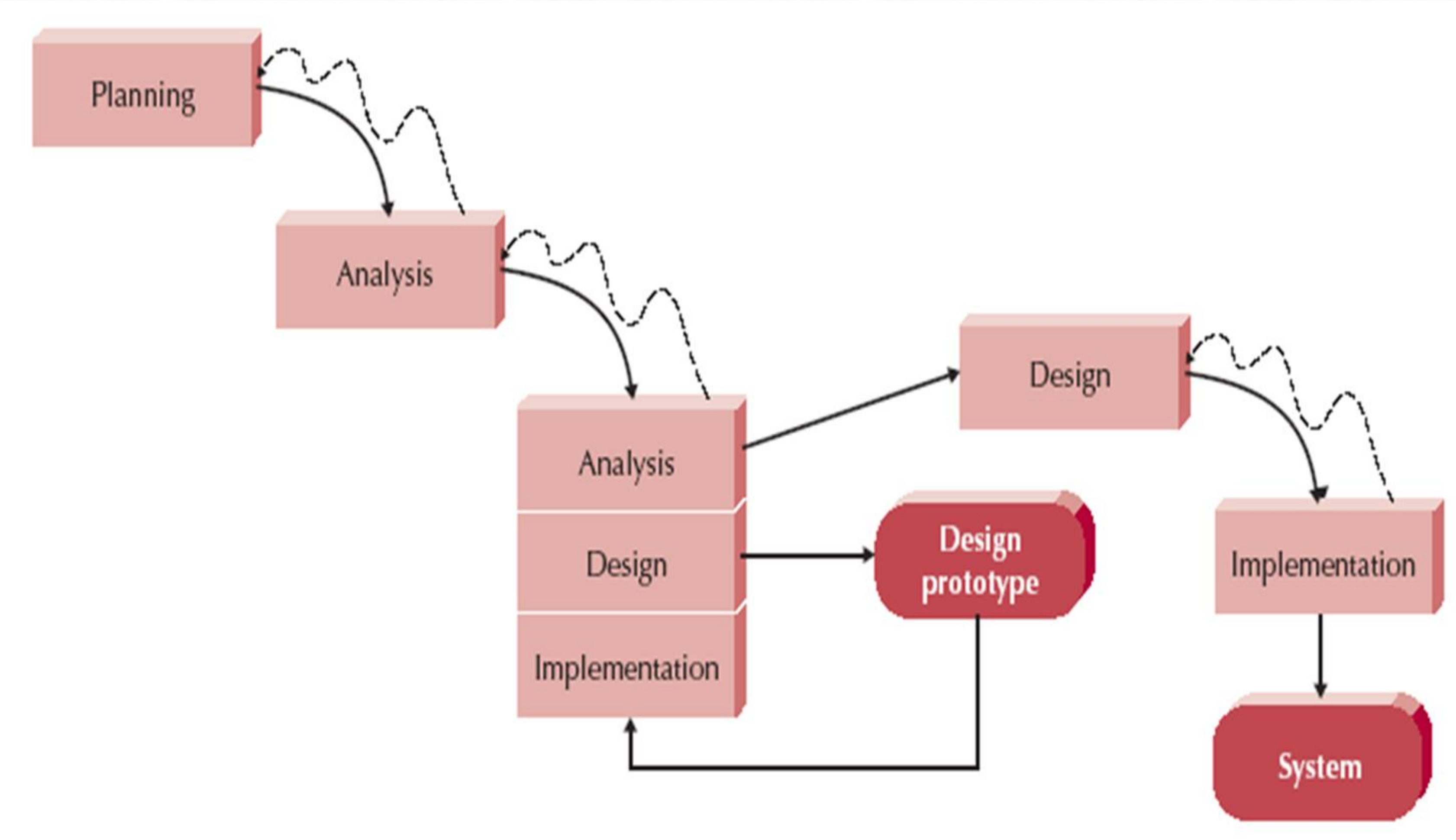


Pros	Cons
Gets working system to users quickly	Fast paced. Hard to conduct careful, methodical analysis
Reassures users that the project is progressing	Initial design decisions have long term staying power
Quickly refines true requirements	Problems may come to light late in design, requiring re-design

RAD: Throw-Away Prototyping

- Use prototypes only to understand requirements
  - Example: use html to show UI
- Prototype is **not** a working design
- Once requirements are understood, the prototypes are thrown away
- The system is then built using SDLC

RAD: Throw-Away Prototyping



Agile Development

- Just a few rules that are easy to learn and follow
- Streamline the SDLC
  - Eliminate much of the modeling and documentation
  - Emphasize simple, iterative application development
- Examples include:
  - Extreme Programming (XP)
  - Scrum
  - Dynamic Systems Development Model (DSDM)

Extreem Programming (XP)

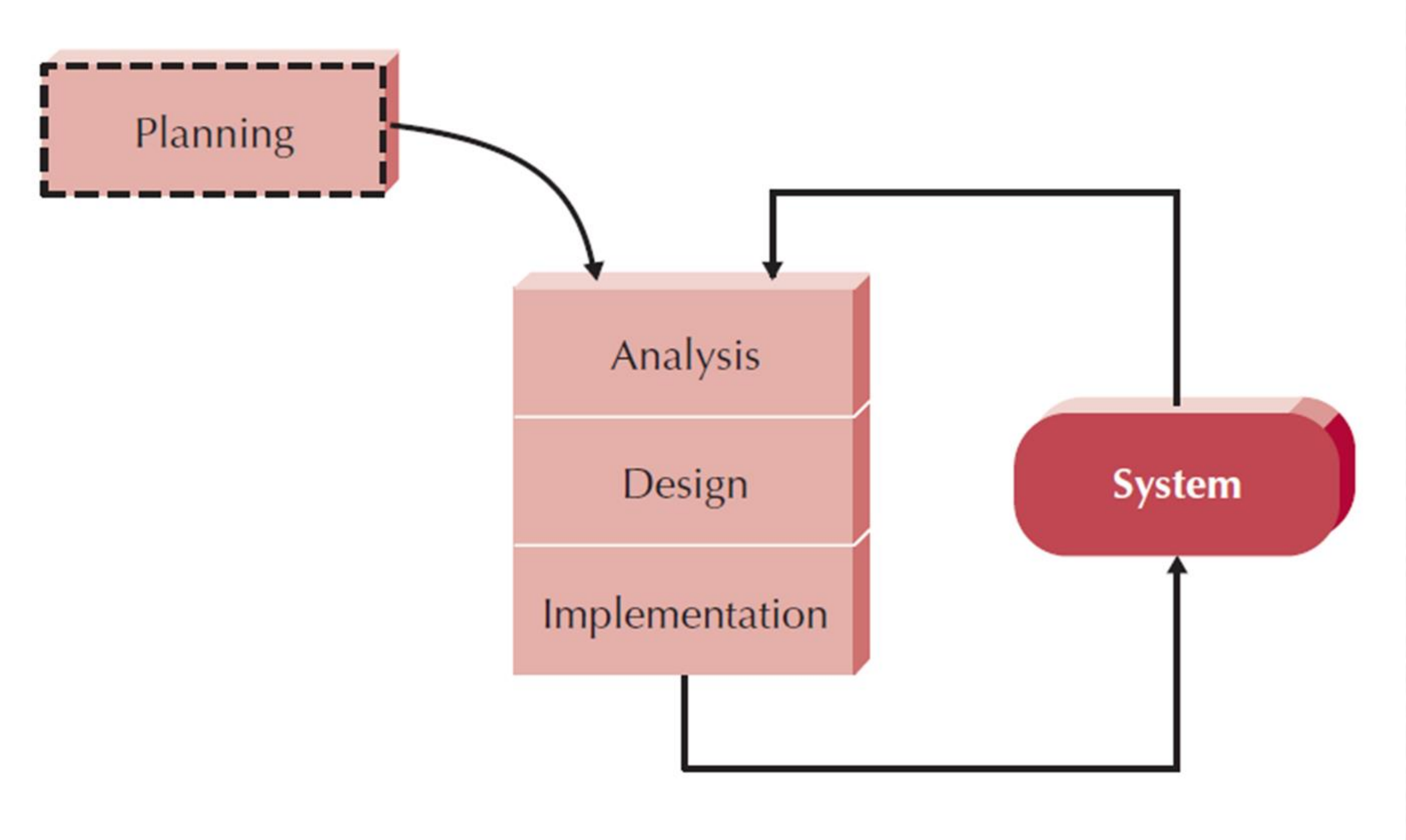
"Core Values" of XP

1. Communication – All to All
2. Simplicity – KISS, refactoring
3. Feedback – Embrace Change
4. Courage – Quality First, test and efficient coding

Extreem Programming (XP)

1. User Stories about system do
2. Code small program using defined standards
   - Naming conventions
   - Coding practices
3. User Feedback
4. Repeat

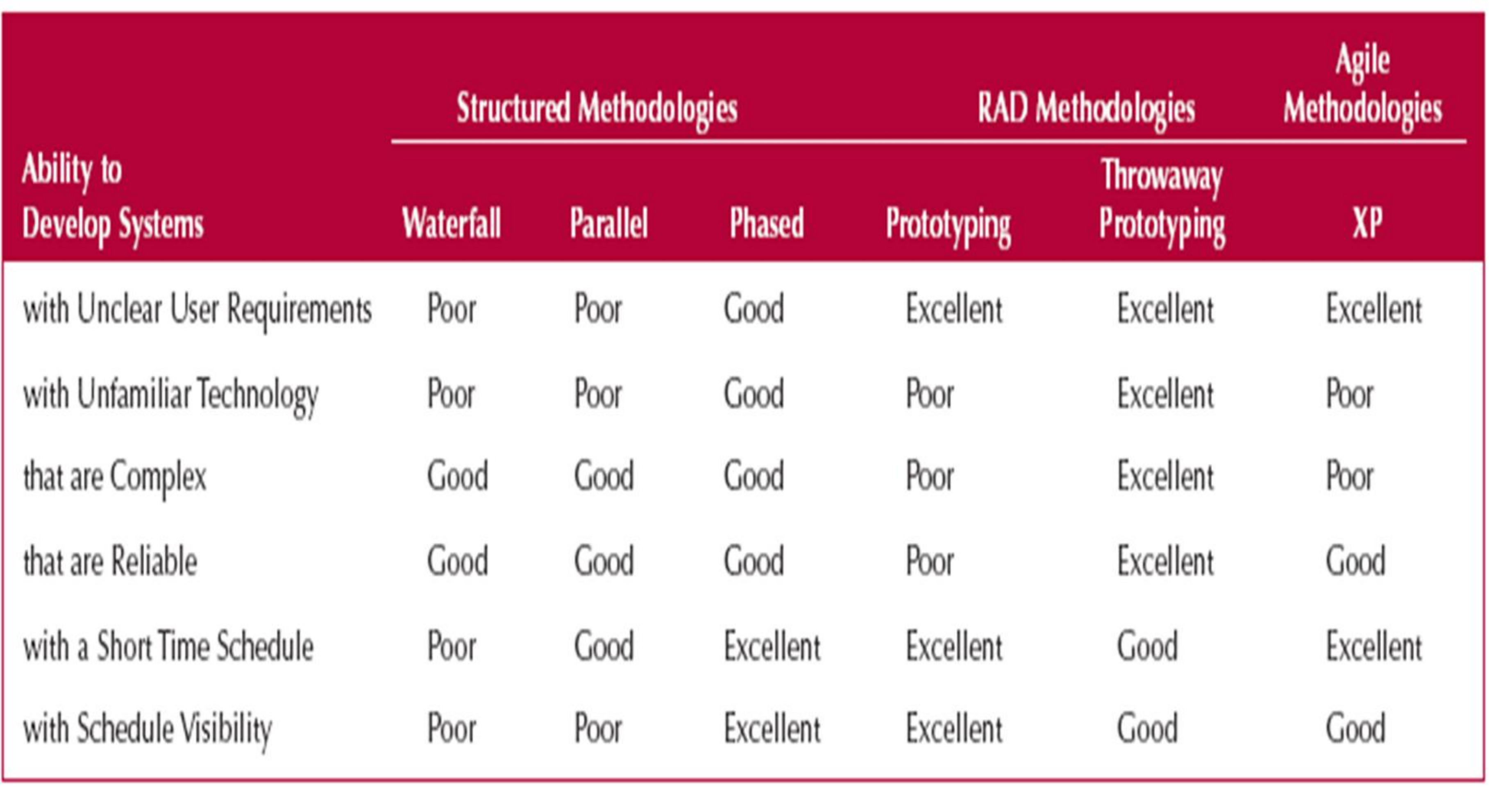
Extreem Programming (XP)



Selecting the Appropriate Methology

1. Clarity of User Requirements
2. Familiarity with Technology
3. System Complexity
4. System Reliability
5. Short Time Schedules
6. Schedule Visibility

Selecting the Right Methology



	Structured Methodologies			RAD Methodologies		Agile Methodologies
Ability to Develop Systems	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP
with Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent
with Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Poor
that are Complex	Good	Good	Good	Poor	Excellent	Poor
that are Reliable	Good	Good	Good	Poor	Excellent	Good
with a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent
with Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Good

Exercise: Selecting Methology

- Suppose you are an analyst for the Roanoke Software Consulting Company (RSCC), a large consulting firm with offices around the world. The company wants to build a new knowledge management system that can identify and track the expertise of individual consultants anywhere in the world based on their education and the various consulting projects on which they have worked. Assume that this is a new idea that never done before been attempted in RSCC or elsewhere. RSCC has an international network, but the offices in each country may use somewhat different hardware and software. RSCC management wants the system up and running within a year.