



Chapter 10 : Approaches to System Development

Tannaz R.Damavandi
Cal Poly Pomona

Outline

- The System Development Life Cycle
- Methodologies, Models, Tools, and Techniques
- Agile Development
- The Unified Process, Extreme Programming, and Scrum

Overview

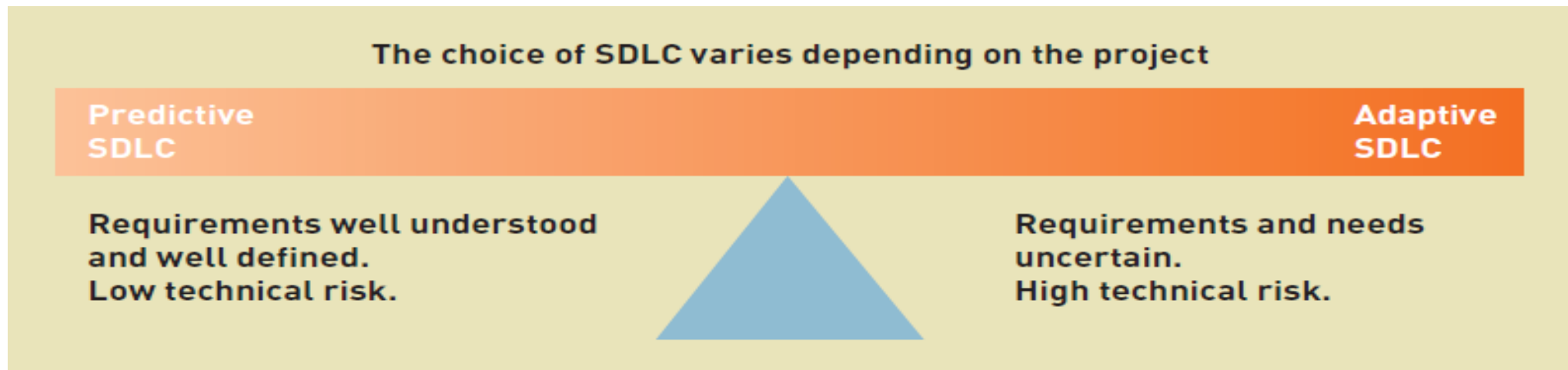
- Chapter 1 demonstrated a system development project that used an iterative and agile system development life cycle (SDLC)
- Later chapters focused on Systems Analysis activities and tasks and some System Design activities and tasks
- Now we return to look at the SDLC and related concepts in more detail
 - Predictive versus Adaptive SDLC variations
 - Activities and Tasks of System Support
 - Models, Methodologies, Tools and Techniques
 - Agile Development
- Specific SDLC Approaches
 - Unified Process (UP)
 - Extreme Programming (XP)
 - Scrum

The System Development Life Cycle (SDLC)

- There are two general approaches to the SDLC
- Predictive Approach to the SDLC
 - Waterfall model
 - Assumes the project can be planned in advance and that the information system can be developed according to the plan
 - Requirements are well understood and/or low technical risk
- Adaptive Approach to the SDLC
 - Iterative model (as see in this course)
 - Assumes the project must be more flexible and adapt to changing needs as the project progresses
 - Requirements and needs are uncertain and/or high technical risk

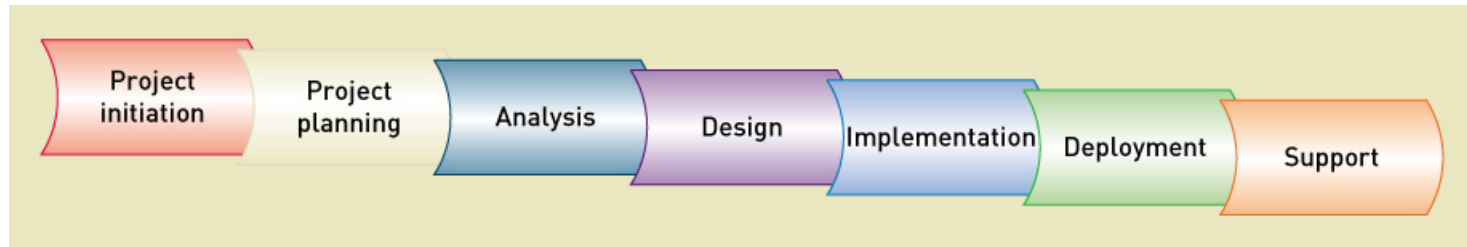
The System Development Life Cycle (SDLC)

- Most projects fall on a continuum between Predictive and Adaptive



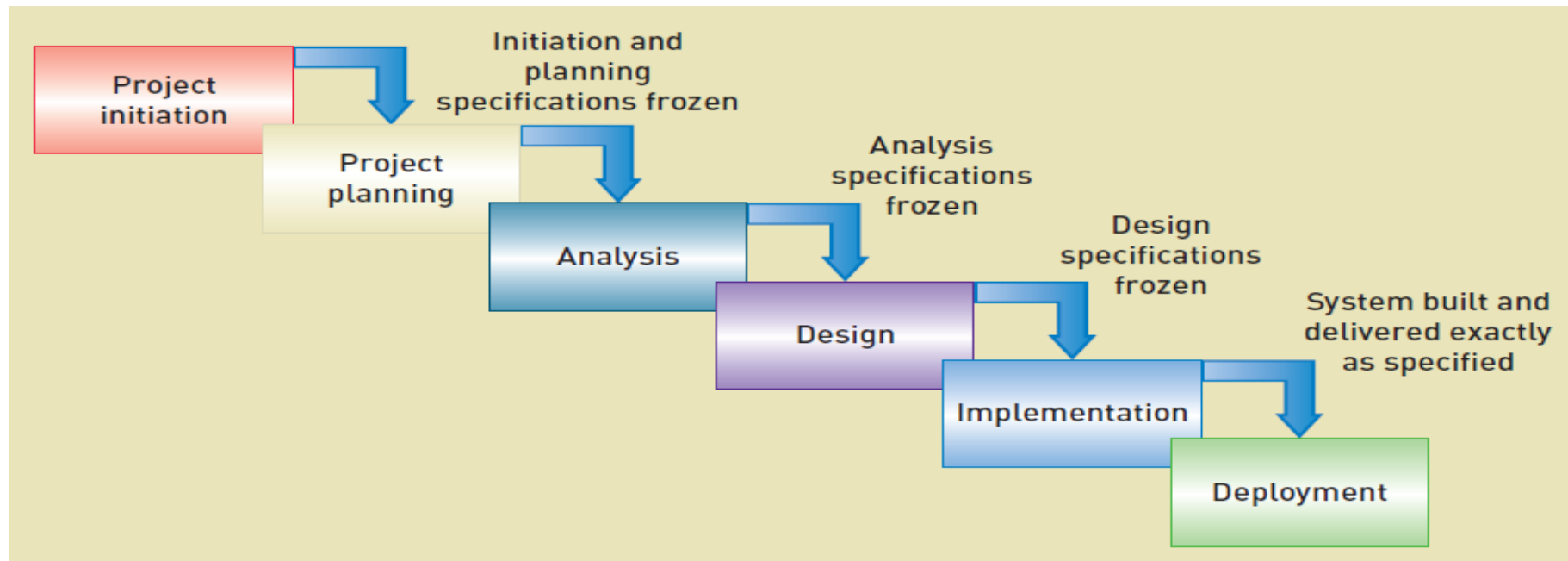
Traditional Predictive SDLC

- Earlier approach based on engineering
- Typically have sequential *Phases*
 - Phases are related groups of development activities, such as planning, analysis, design, implementation, and deployment



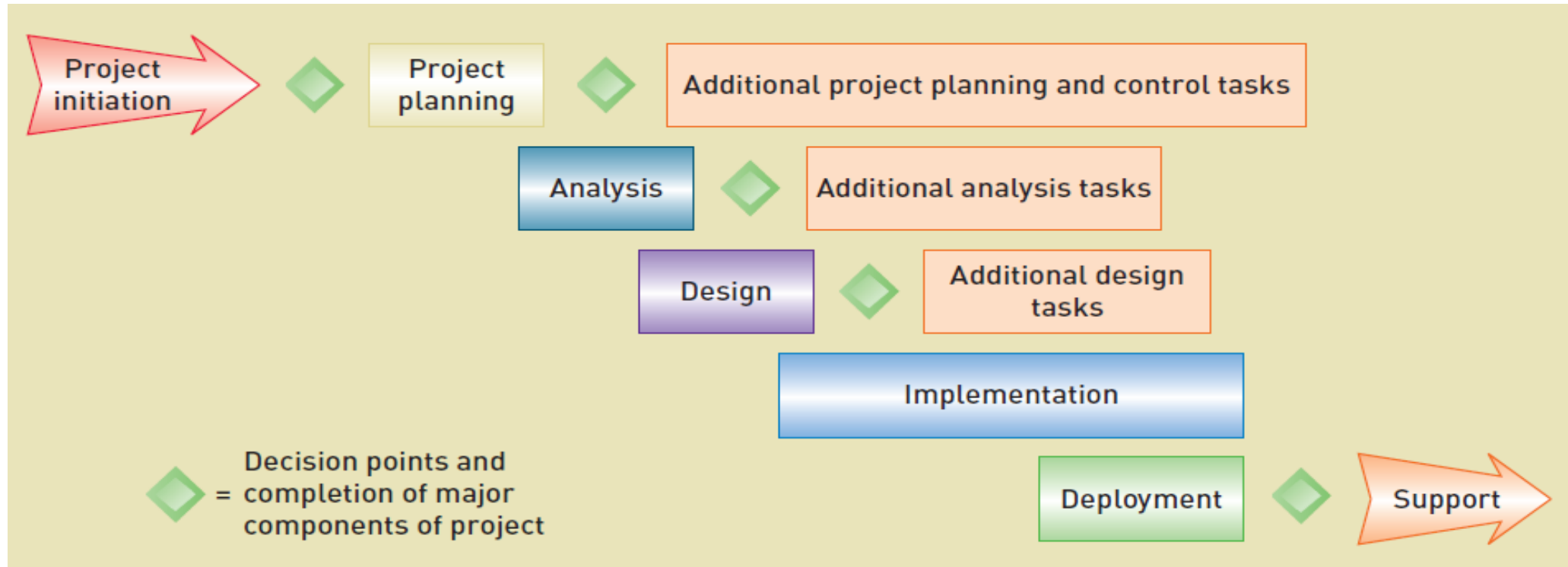
Waterfall Predictive SDLC

- Waterfall model
 - SDLC that assumes phases can be completed sequentially with no *overlap* or *iteration*
 - Once one phase is completed, you fall over the waterfall to the next phase, no going back



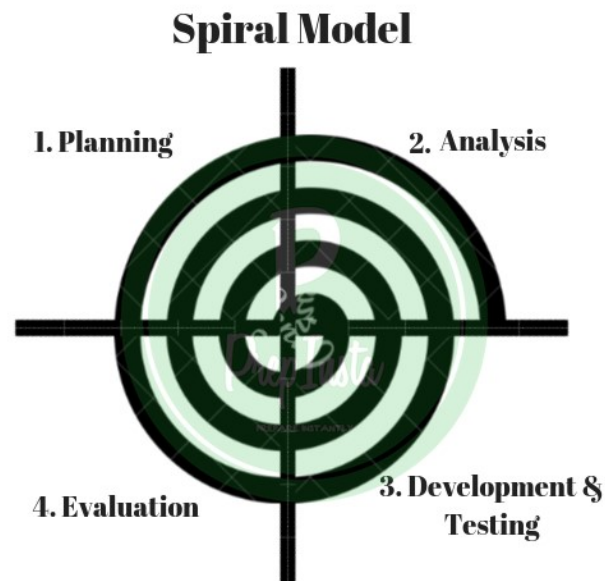
Modified Waterfall with Overlapping Phases

- More flexibility, but still assumes predictive planning and sequential phases







































Newer Adaptive SDLC

- Emerged in response to increasingly complex requirements and uncertain technological environments
- Always includes iterations where some of design and implementation is done from the beginning
- Many developers claim it is the **only** way to develop information systems
- Many IS managers are still sceptical due to apparent lack of overall plan



Adaptive Approaches

- Incremental development
 - Completes portions of the system in small increments and integrated as the project progresses
 - Sometimes considered “growing” a system

Core processes	Iterations					
	1	2	3	4	5	6
Identify the problem and obtain approval.						
Plan and monitor project.						
Discover and understand details.						
Design system components.						
Build, test, and integrate system components.						
Complete system tests and deploy the solution.						

- Walking Skeleton
 - The complete system structure is built first, but with bare-bones functionality

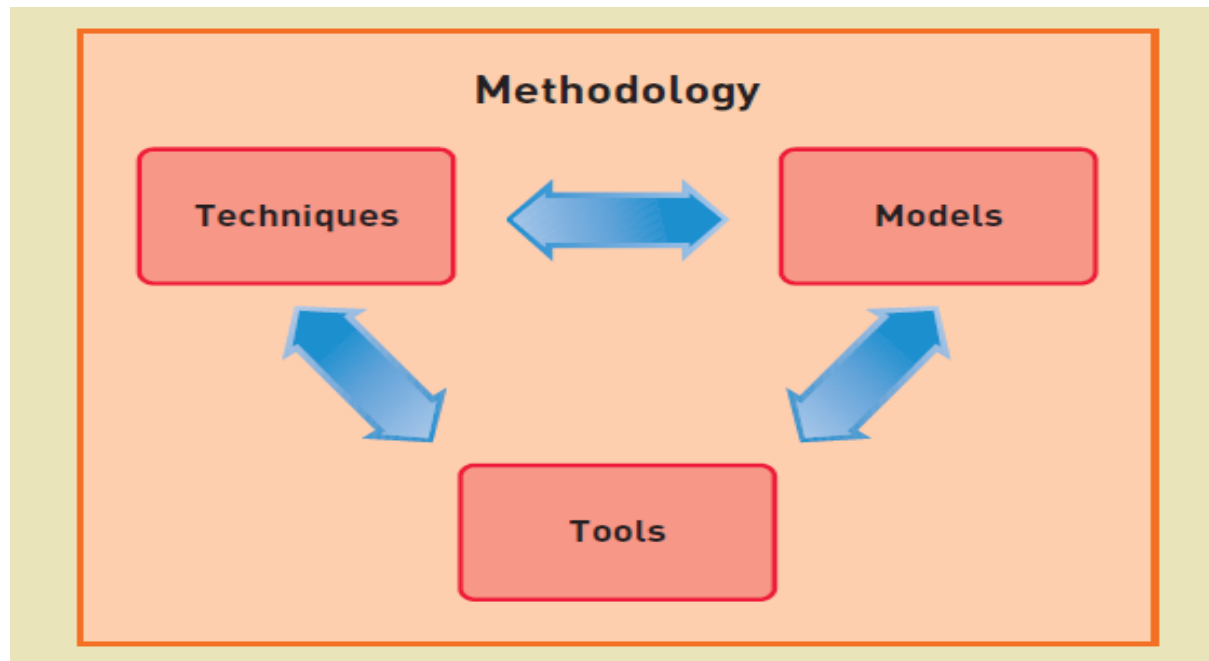
SDLC Support Phase

- All information systems need to be supported once completed.
 - Predictive SDLCs typically include support as a project phase.
 - Adaptive SDLCs treat support as a separate project.
- Support activities
 - Activities whose objectives is to maintain and enhance the system after it is installed and in use.
 - Maintaining the system
 - Fix problems/ Error
 - Make minor adjustments
 - Update for changes in operating system or environments
 - Enhancing the system
 - Add desired functionality
 - Add or change functionality to comply with regulations or legislation
 - Supporting the users
 - Ongoing user training
 - Help desk support

Methodologies

A **Methodology** includes a collection of techniques that are used to complete activities and tasks, including modeling, for every aspect of the project. (aka. *System development process*)

It provides guidelines for every facet of system development: what , when and how to do.



Model, Tools and Techniques

Model

- An abstraction of an important aspect of the real world.
- Makes it possible to understand a complex concept by focusing only on a relevant part
- Each model shows a different aspect of the concept
- Crucial for communicating project information
 - i.e : Flow charts, ERD, DFD, Use case diagram, Class diagram, SSD, Database schema ,...

Tools

- Software applications that assists developers in creating models or other components required for a project
 - i.e : code generator tool, IDE, Visual modeling tool, Data base management application, reverse engineering tool ,code generation, ...

Technique

- A collection of guidelines that help an analyst complete an activity or task
 - i.e : Object Oriented Programming technique, RDB technique, Use case modeling, UI design, User interviewing , data modeling , ...

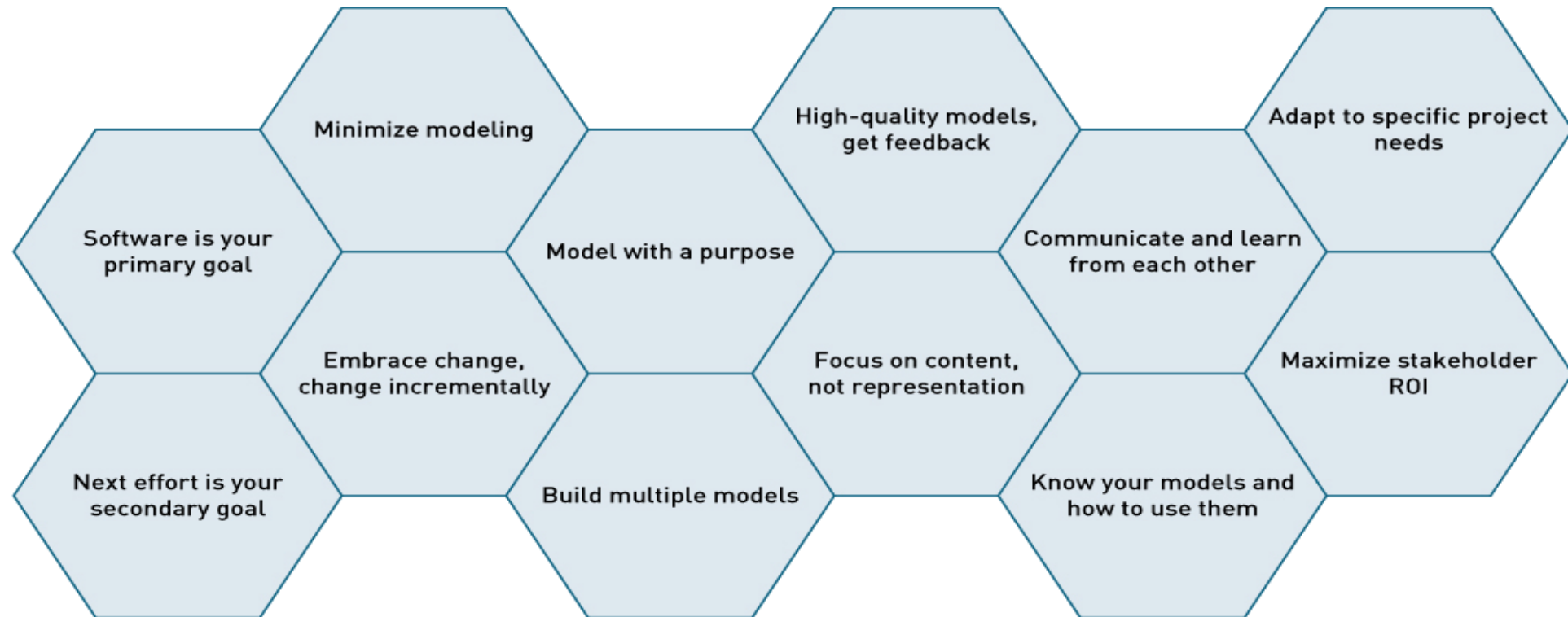
Agile Development

A guiding philosophy and set of guidelines for developing information systems in an unknown, rapidly changing environment.

- Agile Values in “Manifesto for Agile Software Development”
 - Value responding to change over following a plan
 - Value individuals and interactions over processes and tools
 - Value working software over comprehensive documentation
 - Value customer collaboration over contract negotiation

Agile Modeling Principles

- Agile Modeling 12 Principles
 - A philosophy – build only necessary models that are useful and at the right level of detail



Agile Modeling Principles

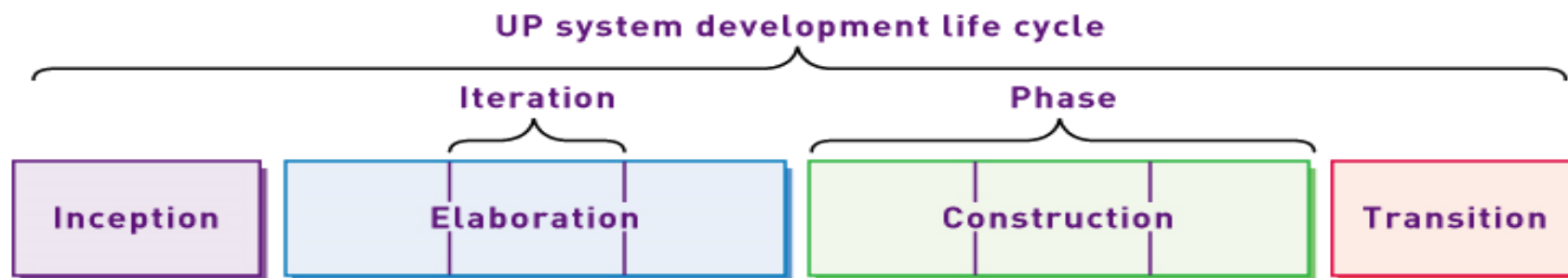
- Develop software as the primary goal
 - Don't get distracted by documentation or models
- Enable the next effort as your secondary goal
 - Be aware of next step versions or revisions
- Minimize your modeling activity
 - Only build what helps move the project forward
- Embrace change and change incrementally
 - Take small steps that keep you on-track and that can be reversed if necessary
- Model with a purpose
 - Model to understand
 - Model to communicate

Agile Modeling Principles

- Build multiple models
 - Look at problems from different perspectives
- Build high-quality models and get feedback
- Focus on content rather than representation
 - Always focus on stakeholder needs
- Learn from each other with open communication
- Know your models and how to use them
- Adapt to specific project needs
- Maximize stakeholder Return On Investment (ROI)

The Unified Process (UP)

- The UP was the early leader in adaptive approaches
- UP and UML (Unified Modeling Language) were developed together
- UP Phases organize iterations into four primary areas of focus during a project
 - Inception phase – getting the project started
 - Elaboration – understanding the system requirements
 - Construction – building the system
 - Transitions – preparing for and moving to deploying the new system



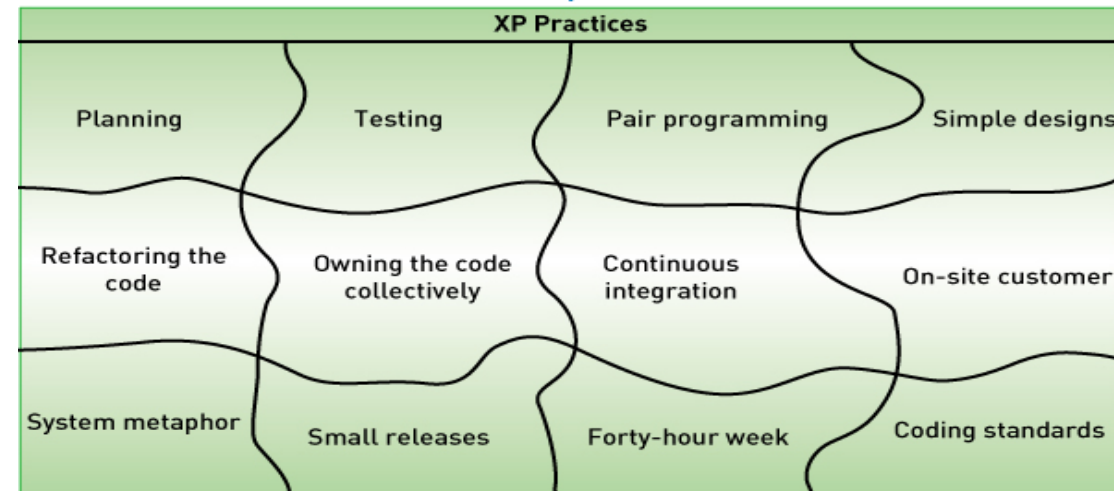
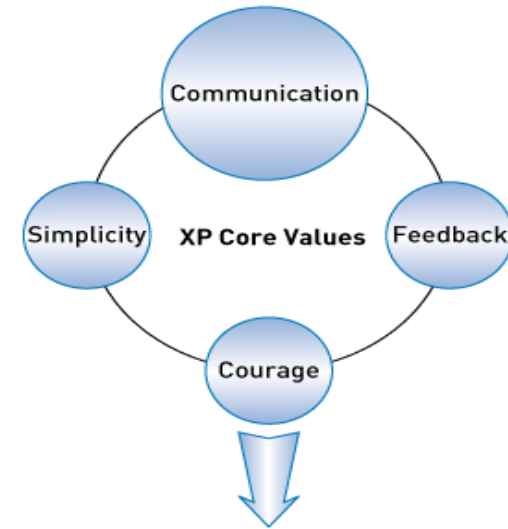
The Unified Process (UP)

- Objectives of each phase

UP Phase	Objective
Inception	Develop an approximate vision of the system, make the business case, define the scope, and produce rough estimates for cost and schedule.
Elaboration	Define the vision, identify and describe all requirements, finalize the scope, design and implement the core architecture and functions, resolve high risks, and produce realistic estimates for cost and schedule.
Construction	Iteratively implement the remaining lower-risk, predictable, and easier elements and prepare for deployment.
Transition	Complete the beta test and deployment so users have a working system and are ready to benefit as expected.

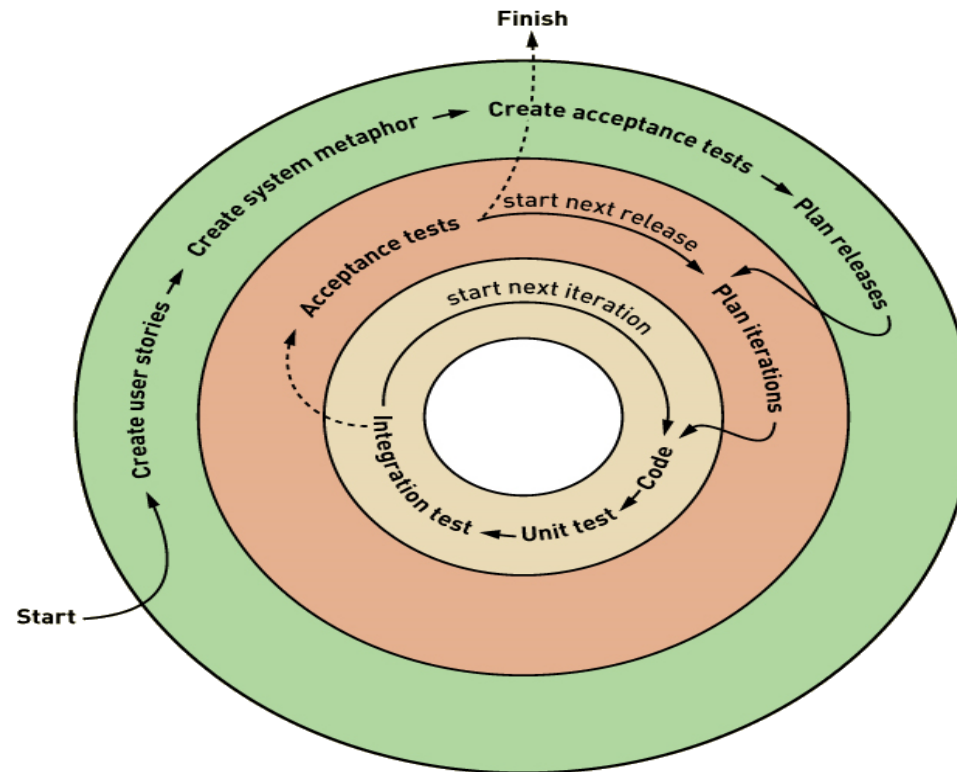
Extreme Programming (XP)

- Takes the best practices of software development and extends them “to the extreme”
 - Focus intensely on proven industry practices
 - Combine them in unique ways to get better
- XP Core Values
 - Communication
 - Ensures that open, frequent communication occur
 - Simplicity
 - Feedback
 - User: on Functionality and requirements
 - Developers: design and code
- Courage
 - Stick with too-tight schedule and “do it right”



Extreme Programming (XP)

- Three level approach (three rings)
 - Outside ring – create user stories and define acceptance tests
 - Middle ring – conduct tests and do overall planning
 - Inside ring – iterations of coding and testing

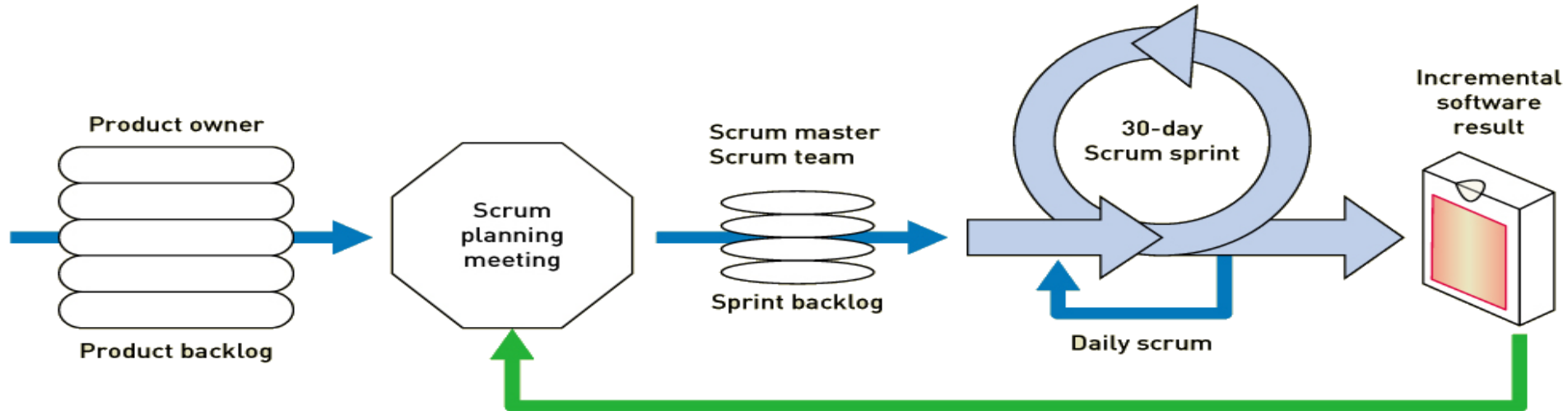


SCRUM

- A philosophy based on Agile development principles
 - Responsive to a highly changing, dynamic environment in which users might not know exactly what is needed and might also change priorities frequently
 - Intense effort involving the entire team for a defined period of time
 - Focuses primarily on the team level (social engineering)
- Product backlog
 - Prioritized list of user requirements
- Product owner
 - The client stakeholder who controls backlog
- Scrum master
 - Scrum project manager, facilitator
- Scrum Team
 - A small group of developers
 - Sets its own goal for what it can accomplish in a specific period of time
 - Organizes itself and parcels out the work to members

SCRUM Sprint

- Scrum sprint
 - A time-controlled mini-project to implement part of the system



SCRUM Practices

- Scope of each sprint is frozen (but can be reduced if necessary)
- Time period is kept constant
- Daily Scrum meeting
 - What have you done since the last daily Scrum (during the last 24 hours)?
 - What will you do by the next daily Scrum?
 - What kept you or is keeping you from completing your work?

Summary

- This chapter covers approaches to system development in more detail
- There are two approaches to the SDLC: Predictive and Adaptive
- A predictive SDLC, also known as the waterfall model, is used when it is possible to plan the project completely in advance
- An Adaptive SDLC, which uses iteration, is used when the requirements are less certain and the project will need to react to changes
- All new System development project use a methodology (or development process) and many are available. A methodology includes an SDLC and tools, techniques, and models
- Agile development is the current trend in system development
- Unified Process is a formal iterative approach which uses UML models (with Agile philosophy) and UP disciplines
- Extreme Programming (XP) is an iterative approach which takes good practices to the extreme
- Scrum is an iterative approach using a Scrum Sprint