Chapter 2

A Systems View and Systems Methodology

Objectives

- Define the systems approach and its impact on project management
- Define a PMLC and understand how to apply it
- Define several SDLC models and know when to use each different type
- Define the relationship between the PMLC and the SDLC and understand how the two work together

Projects Cannot Be Run In Isolation!

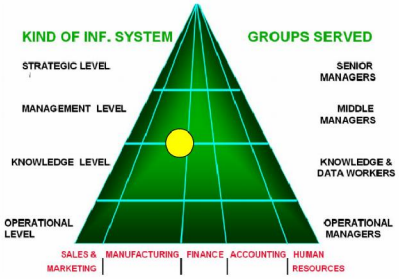
Harold Kerzner wrote:

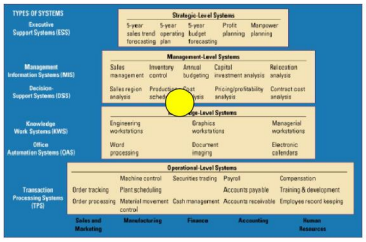
"the ability to analyze the total project, rather than the individual parts is the first prerequisite for successful project management".

- Projects must operate *in a broad organizational environment*
- Project managers need to take a holistic or *systems view or systems thinking approach* of a project and understand how it is situated within the larger organization
- The *Systems Thinking* approach is a process which allows projects to be viewed in the context of the entire environment including both inside and outside of the organization. It is a process that can bring order and discipline to a large, chaotic, unorganized situation
- The ability to examine a problem or issue by first understanding the environment it exists within, before reducing the problem or issue into smaller components and finally managing the resolution of the problem or issue

Conclusion:

Use the "Systems Thinking" approach as often as possible !





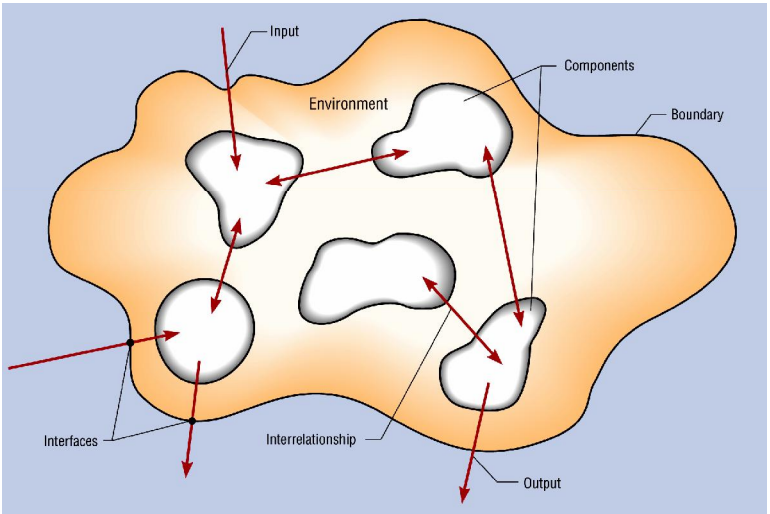
Systems View
or
Systems Thinking Approach

A System and Its Characteristics

- **A System** is an inter-related set of components, with an identifiable boundary, working together for some purpose.
- Any system has **nine (9) characteristics**:

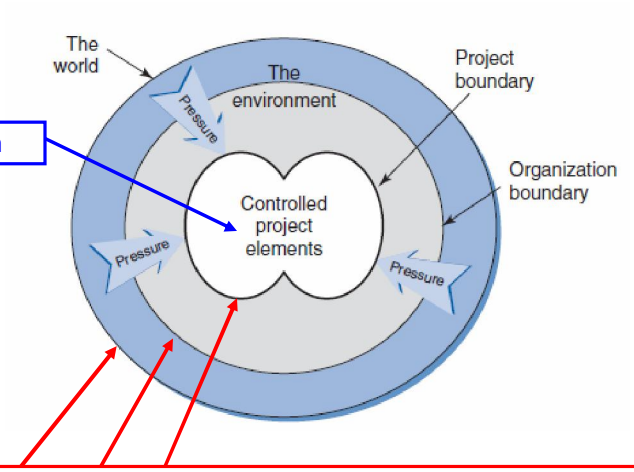
1) **Components** - *an irreducible part or aggregation of parts*
2) **Relations (links)** - *dependence of one subsystem on one or more subsystems*
3) **A Boundary** - *a line that sets off the system from its environment*
4) **A Purpose** - *the overall goal or function of a system*
5) **An Environment** - *everything external to a system that interacts with the system*
6) **Interfaces** - *points of "system-environment" and "subsystem-subsystem" contacts*
7) **Input** - *whatever a system takes from its environment*
8) **Output** - *whatever a system returns to its environment*
9) **Constraint(s)** - *limit(s) to what a system can accomplish*

A General Depiction of a System



Source: Management Information Systems, 8th Ed., K. Landon and M. Landon

System's Boundary and Environment

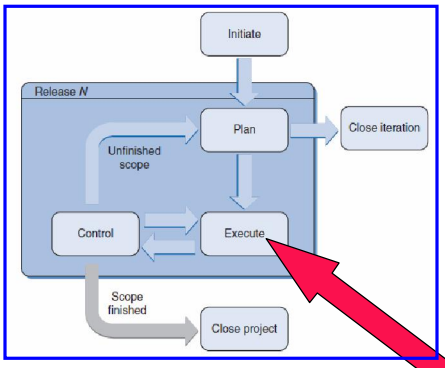


Most of what a Project Manager does exists on the boundaries! This is one more reason that PM must have multiple skills: 1) analytical (problem solving) , 2) technical, 3) managerial, 4) communication.

CIS Project Management (as a process) Life Cycle (CIS PM LC)

PM LC

SW/IT/CIS PM Life Cycle



A SW/CIS project management life cycle is a prescribed order of phases (smaller segments of the entire project) in which each contains a specific deliverable which collectively deliver a result.

- What work (tasks) will (should) be done in each phase.
- A definition of each phase's deliverables and when.
- The change control process for each deliverable
- What resources are involved in each deliverable
- Criteria that needs to be met complete each phase

Software Engineering Framework activities

- Communication (initiation)
- Planning
- Modeling/Simulation/Prototyping
  - **Analysis of requirements**
  - **Design Model**
- Development (construction)
  - **Code generation**
  - **Testing**
- Deployment/Implementation

SW/CIS Devel. Model

5 SE framework activities (for large SW projects, or in large-size companies).

But … in what order? returns/iterations/loops? Feedback from users?

Construction

Modeling

Communication

Deployment

Planning

Classic Models (multiple details – in CS590 course)

Classic System Development Life Cycle SDLC Models

(SE: software engineering prescriptive models)

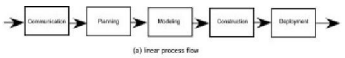
The Systems Development Life Cycle (SDLC) is an approach to building information technology systems consisting of a standard set of phases each producing a prescribed set of deliverables

- **Classic (traditional) SDLC Models/Methodologies (from 1950s)**
  - Waterfall (predictive)
  - Incremental Model (50-50, predictive-adaptive)
  - Evolutionary Prototyping Model (50-50, predictive-adaptive)
  - Spiral Evolutionary (somewhat predictive)
  - Iterative (adaptive)

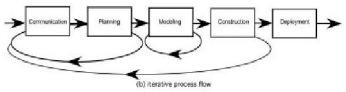
Main Types of SW/IT/CIS Development Process Flow

(various types will have significant impact on project time, cost, human resources, quality, etc.)

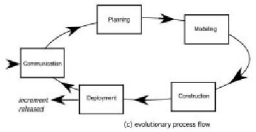
a) Linear process flow (no feedback, no iterations/loops)



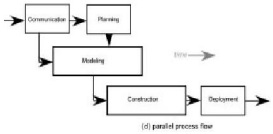
b) Iterative process flow (a circular manner, with a feedback, with iterations)



c) Evolutionary process flow (with various versions or increments released)



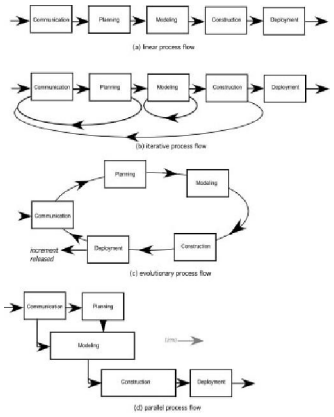
d) Parallel process flow (no feedback, no iterations/loops, with parallel activities)



Software team (project manager, system analyst, software engineers, etc.) MUST identify the best (optimal) process flow or, possibly, a combination of them for a specific software project.

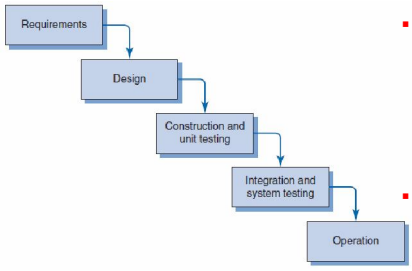
Question: Why do we need to know all those types of SE Process Flow?

Answer: It is about Money, Time, People, and Quality.



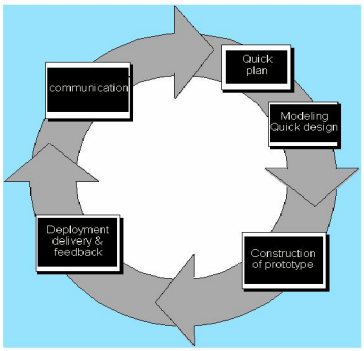
	% of project budget	Waterfall Model	Prototyping Model or Iterative Model	Spiral Model
Communication	5-10%	10	10	10
Planning	15-20%	20	20	20
Modeling	25-30%	30	30	30
Construction/ Development	15-25%	25	25	25
Deployment	10-15%	15	15	15
Notes:			The same budget but for 3…5…??? prototypes (# of prototypes or iterations is unknown at the beginning)	The same budget but for 3…5…??? evolutionary spirals (# of spirals is unknown at the beginning and next spiral is more expansive than the previous one)

Waterfall Model



- **Strengths**
  - Is well understood by most practitioners
  - Easier to manage than the new agile methods
  - When working on large complex applications
  - When teams are distributed geographically
  - When using a less experienced IT resources
- **Weaknesses**
  - Does not accommodate a change to requirements very well
  - All Requirements must be known and defined in the beginning
  - Does not allow a repeat of a phase (iterate)
  - Limited adaptability to different project types
  - Encourages communications gap between users and IT

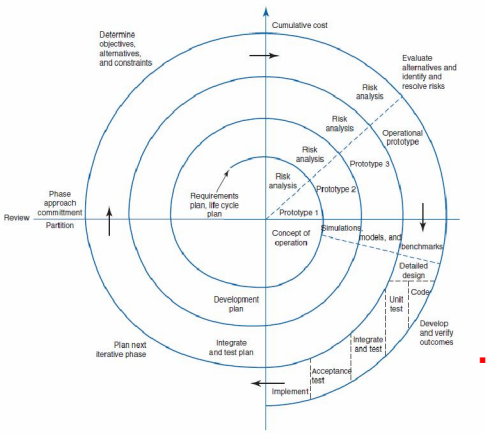
Evolutionary Prototyping Model



- Focuses on gathering correct and consistent requirements and is the approach of building a system incrementally through a series of gradual refinements or prototypes
- Requirements are discovered throughout the process and the system is repeatedly refined based on those discoveries
- Allows developers to learn from each prototype and apply those lessons to future versions
- The prototyping approach is an excellent choice for research and development projects, quickly building mockups of system components for user review allows for timely feedback that can be incorporated in the next design or prototype

- **Strengths**
  - Visibility – customers see steady progress
  - Useful when requirements are changing rapidly or no one fully understands the requirements
- **Weaknesses**
  - It is impossible to know at the beginning of the project how long it will take
  - There is no way to know the number of iterations/phases that will be required
  - Difficult to build an accurate cost estimate

Spiral Model



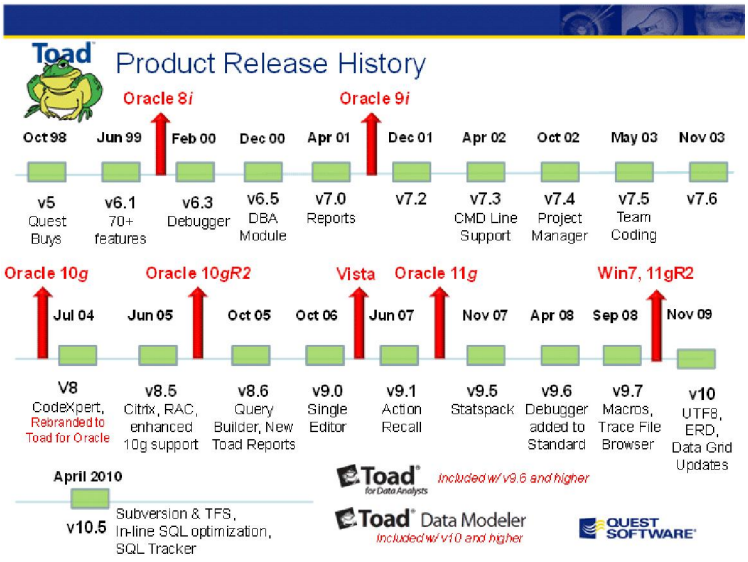
- Similar to the classic waterfall model with the addition of risk analysis and iterations
- Emphasizes the need to go back and reiterate earlier stages a number of times as the project progresses
- It's actually a series of short waterfall cycles, each producing an early prototype representing a part of the entire project

Examples:

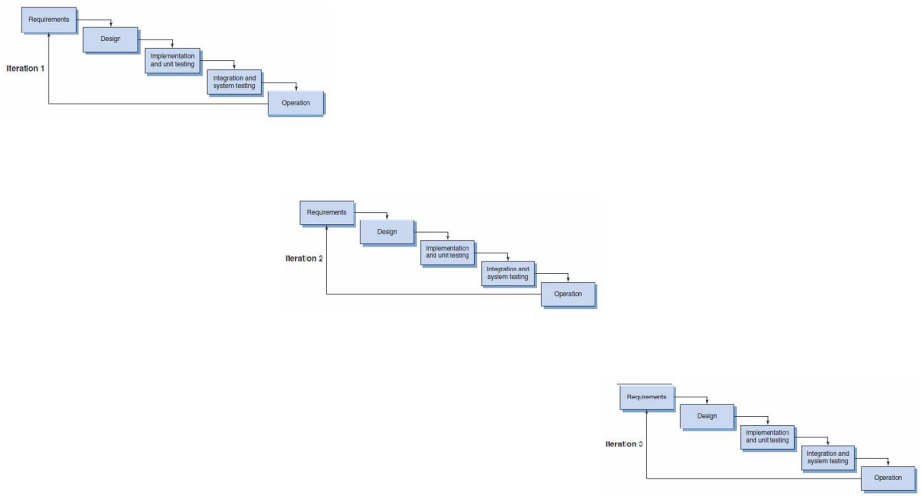
1) **Win'95, Win 98, Win'2000, Win XP (2002), Win Vista, Win 7, Win 8, etc.**
2) **Oracle DB 8, 9, 10, 11, etc.**

- **Strengths**
  - Good for large complex projects
  - Accommodates change well
  - Can react to risks very quickly
  - Software produced early in the life of the project
  - Increased user visibility
- **Weaknesses**
  - Can be a costly model to use
  - Risk analysis requires highly specific expertise
  - Project's success highly dependent on risk analysis
  - Doesn't work well for small projects

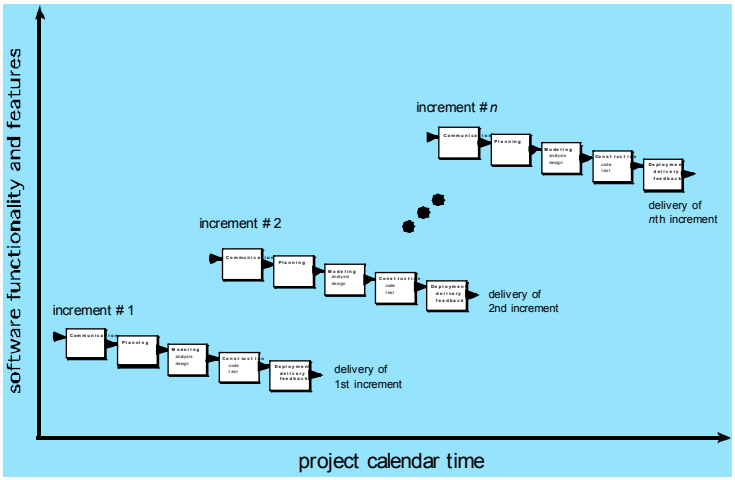
Spiral Model versus Incremental Model



Iterative Model



Incremental Model

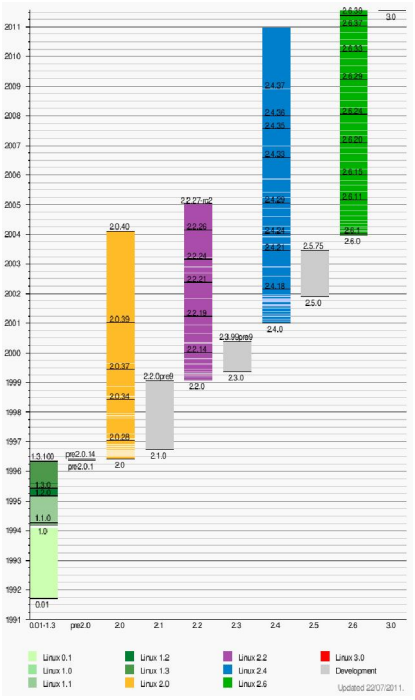


The 1st increment = a *core product*.

Following increments are aimed to better meet customer requirements and deliver additional functionality

Source: Software Engineering, 7th Ed., by Roger Pressman

The Incremental Model: Real World Example (linux operating system)

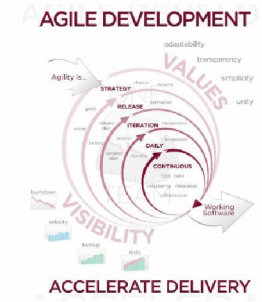


Iterative and Incremental Model

- Repeating a process phase until ultimately meeting the project requirements (iterating the phases) and developing and delivering a system in stages (increments)
- The system grows by adding new and enhanced functionality with each build cycle
- Each cycle tackles a relatively small set of requirements and proceeds until the entire scope of the project is completed

- **Strengths**
  - Generates working software quickly and early
  - Flexibility
  - Ease of testing
  - Ease of risk management
- **Weaknesses**
  - Not easy to manage
  - Must be able to estimate well to plan iterations
  - Hard to determine cost and time estimates early in the process

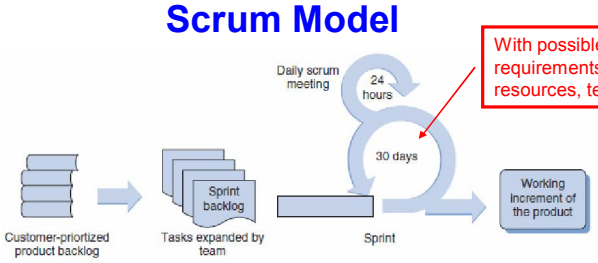
New Models
for product development life cycle
(based on ideas of agility, agile programming)



- New SDLC Models/Methodologies based on ideas of agility or agile programming styles (due to Web-Based SW systems = WebE) – from mid 1990x:
  - Scrum
  - Extreme Programming (XP)
  - Industrial XP (IXP)
  - Adaptive Software Development (ASD)
  - Crystal programming
  - Feature Driven Development (FDD)
  - Lean Software Development (LSD)
  - Dynamic Systems Development Method (DSDM)

Agile software development methodology is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle.

Scrum Model



- Based on the concept that software development is not a defined process, but an empirical process with complex input to output transformations that may or may not be repeated under differing circumstances
- The main idea of ***Scrum*** is that systems development involves several environmental and technical variables that are likely to change during the process (for example, requirements, time frame, resources, technology)
- The name ***Scrum*** is essentially derived from the game of rugby. In rugby, a play where two opposing teams attempt to move against each other in large, brute-force groups is called a scrum. Each group must be quick to counter the other's thrust and adjust and exploit any perceived weakness, without the luxury of long term planning
- This makes the development process unpredictable and complex, requiring flexibility of the systems development process in order to respond to these changes

SDLC (software development life cycle) Models. Do we need to know it?

(With written permission from Mr. Sunnihith Bojedla)

December 7, 2011

Hello Dr. Uskov,

I got placed in SME (Insurance sector).

Thanks for all your dedication towards students, which helped us to learn lot of technical aspects, to face the interview confidently, whatever the position may be.

Every interviewer is concerned about whether we had the knowledge of SDLC or not. When we answered the whole process in brief and explained some examples which we done as part of projects they were satisfied more than enough.

Thanks for all your help and I wish every other student should benefit in the same way from your dedicated professional approach.

Cordially,
Sunnihith Bojedla

What SDLC Model to Use? Factors to be Considered

Classic (traditional) SDLC Models/Methodologies:

- Waterfall (predictive)
- Incremental Model (50-50, predictive-adaptive)
- Evolutionary Prototyping Model (50-50, predictive-adaptive)
- Spiral Evolutionary (somewhat predictive)
- Iterative (adaptive)

???

New SDLC Models/Methodologies based on ideas of agility or agile programming styles:

- Scrum
- Extreme Programming (XP)
- Industrial XP (IXP)
- Adaptive Software Development (ASD)
- Crystal programming
- Feature Driven Development (FDD)
- Lean Software Development (LSD)
- Dynamic Systems Development Method (DSDM)

Factors to be considered:

1) What is company's size, philosophy, experience, management, etc:
   - large (>2001 workers)
   - medium size: 51-2000 workers,
   - small-size: 1-50 workers;

synchronization of SW development activities of 10+ persons is an issue !

2) What is project's goal ? Top quality (after a lot tests) + focus on thousands/millions of users
   OR customer satisfaction (usually, fewer number of customers) + time

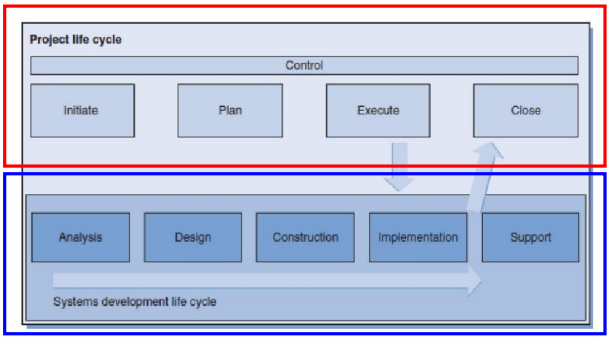
3) Financial issues (constraints, money available, etc.), etc.

Integration of

Project Management Life Cycle

and

Software Development Life Cycle



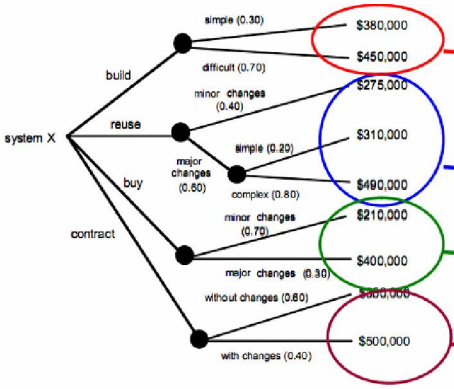
- The project life cycle applies to all projects, regardless of the products being produced
- Product life cycle models vary considerably based on the nature of the product
- Most large software products are developed as a series of projects
- Project management is conducted during all of the product life cycle phases

1) The overlap occurs from project management life cycle (PMCL) to software development product life cycle (SDLC) during Analysis and Design.

2) During project Execution the bulk of the product is built.

Systems View in Action:
1) Development (make by yourself) is NOT the only option for Project manager.

2) PM must consider other options – Reuse, Buy, Contract, etc., and find the optimal project solution (cost M, time M, HR M., etc.)

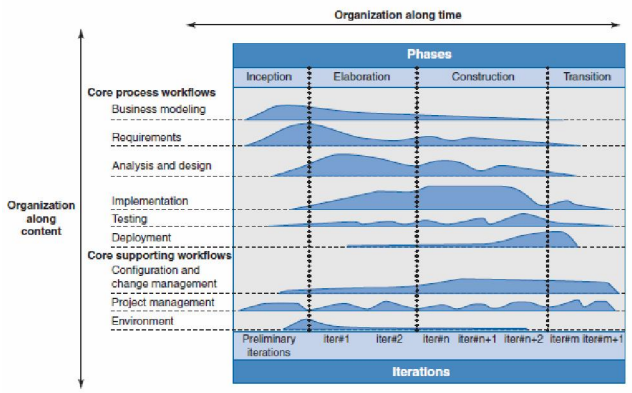


	A	B	C	D	E	F	G	H
1	Make-Buy Decision Tree: Calculations							
2								
3		Build	Simple	0.3			380,000	$ 114,000
4			Difficult	0.7			450,000	$ 315,000
5							Total:	$429,000
6								
7								
8		Reuse	Minor changes	0.4		1	275,000	$ 110,000
9			Major changes	0.6	Simple	0.2	310,000	$ 37,200
10					Complex	0.8	490,000	$ 235,200
11							Total:	$382,400
12								
13								
14		Buy	Minor changes	0.7			210,000	$ 147,000
15			Major changes	0.3			400,000	$ 120,000
16							Total:	$267,000
17								
18		Contract	Without changes	0.6			350,000	$ 210,000
19			With changes	0.4			500,000	$ 200,000
20							Total:	$410,000

Chapter 2

A Systems View and Systems Methodology (additional information)

Rational Unified Process (RUP) Model



- Developed as a process complement to the Unified Modeling Language (UML)
- Adaptable process framework which can be used for both heavy and light projects
- Based on iterative development paradigm with Four Phases
  - Inception, Elaboration, Construction and Transition
  - Each phase contains one or more iterations

Rational Unified Process (RUP) Model: Strengths and Weaknesses

- **RUP Strengths**
  - Risks are mitigated (diminished) earlier
  - Change is more manageable
  - Higher level of reuse
  - The project team can learn along the way
  - Better overall quality
  - Enhances team productivity, by providing every team member with easy access to a knowledge base with guidelines, templates and tool mentors for all critical development activities.
- **RUP Weaknesses**
  - Not easy to tailor to smaller projects
  - Has a large volume of process guidelines and is detail heavy

eXtreme Programming (XP)

- Basic approach includes short development cycles, frequent updates, dividing business and technical priorities, and assigning user stories
- Four key values: communication, feedback, simplicity, and courage
- Designed to allow small development teams to deliver quickly, change quickly, and change often

- **XP Strengths**
  - The project is more manageable
  - Progress is made, even when requirements are not stable
  - Everything is visible to everyone
  - Team communication improves
  - The team shares successes along the way and at the end
  - Customers see on-time delivery of increments
  - Customers obtain frequent feedback on how the product actually works
- **XP Weaknesses**
  - Doesn't work well with large teams/projects
  - Requires very experienced team members

Scrum Model Strengths

SCRUM Model strengths:

- The project is more manageable
- Progress is made, even when requirements are not stable
- Everything is visible to everyone
- Team communication improves
- The team shares successes along the way and at the end
- Customers see on-time delivery of increments
- Customers obtain frequent feedback on how the product actually works

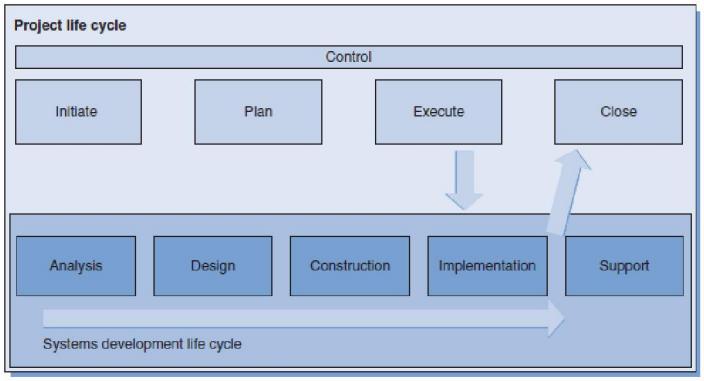
SCRUM Model weaknesses:

- **Doesn't work well with large teams**
- **Requires experienced developers**
- **Not good for mission or life critical systems.**
- **Requires hands-on management, but not micromanagement**
- **Requires constant monitoring both quantitatively and qualitatively**

Integration PROJECT Life Cycle (Phases) and PRODUCT Life Cycles

- The project life cycle applies to all projects, regardless of the products being produced
- Product life cycle models vary considerably based on the nature of the product
- Most large software products are developed as a series of projects (or, a Program)
- Project management is conducted during all of the product life cycle phases

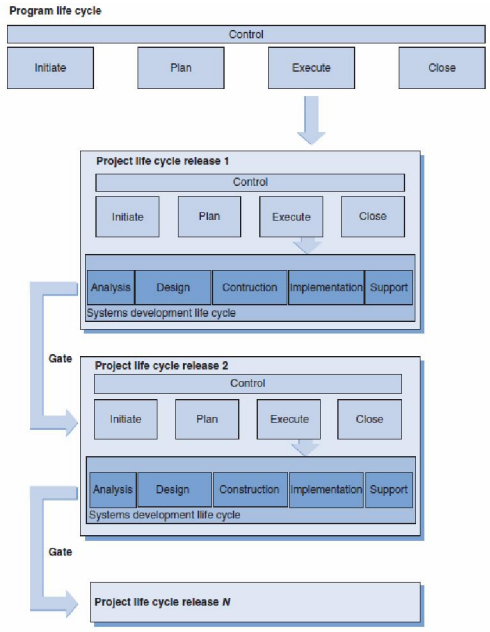
Integration Project to Product Life Cycles



1) The overlap occurs from project life cycle to product life cycle during Analysis and Design.
2) During project Execution the bulk of the product is built.

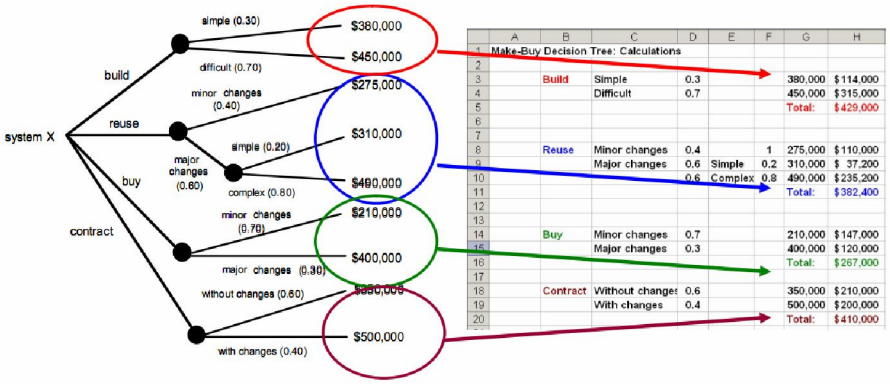
Integration of Project to Product Life Cycles – Multiple Phases/Projects

(like nested loops in programming)



Software Development by Yourself is not the only option !

SPM should investigate all other options ("Make-Buy Decision"), and find optimal solution



	A	B	C	D	E	F	G	H
1	Make-Buy Decision Tree: Calculations							
2								
3		Build	Simple	0.3			380,000	$114,000
4			Difficult	0.7			450,000	$315,000
5							Total:	$429,000
6								
7								
8		Reuse	Minor changes	0.4		1	275,000	$110,000
9			Major changes	0.6	Simple	0.2	310,000	$ 37,200
10				0.6	Complex	0.8	490,000	$235,200
11							Total:	$382,400
12								
13								
14		Buy	Minor changes	0.7			210,000	$147,000
15			Major changes	0.3			400,000	$120,000
16							Total:	$267,000
17								
18		Contract	Without changes	0.6			350,000	$210,000
19			With changes	0.4			500,000	$200,000
20							Total:	$410,000