SOEN – 6841 Software Project Management

Amin Ranj Bar

Winter, 2021

Overview

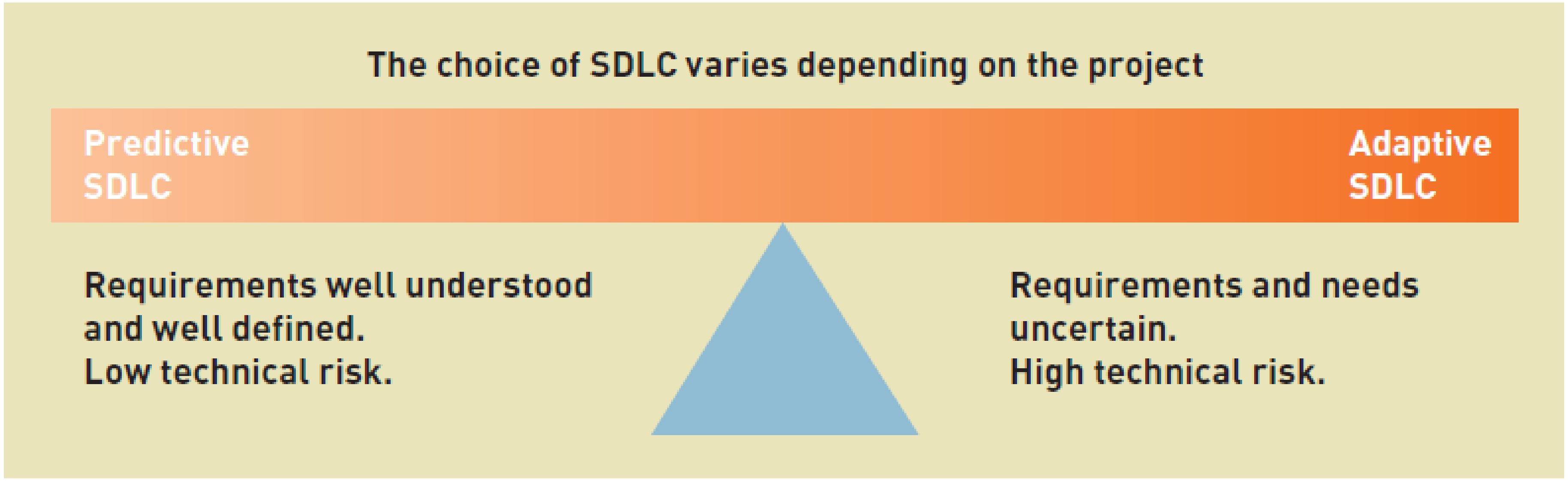
- System development life cycles
- Predictive vs adaptive
- Agile development
- Agile estimating and planning

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 text)
  - 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)

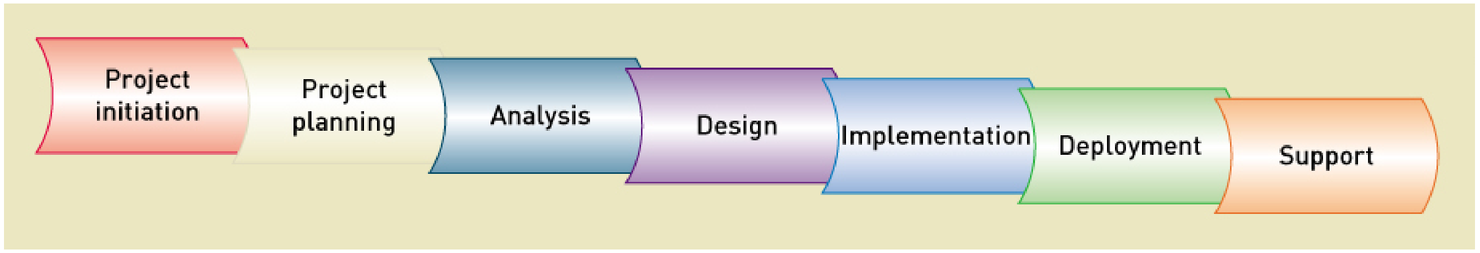
- Most projects fall on a continuum between Predictive and Adaptive


The choice of SDLC varies depending on the project
Predictive SDLC
Adaptive SDLC
Requirements well understood and well defined. Low technical risk.
Requirements and needs uncertain. High technical risk.

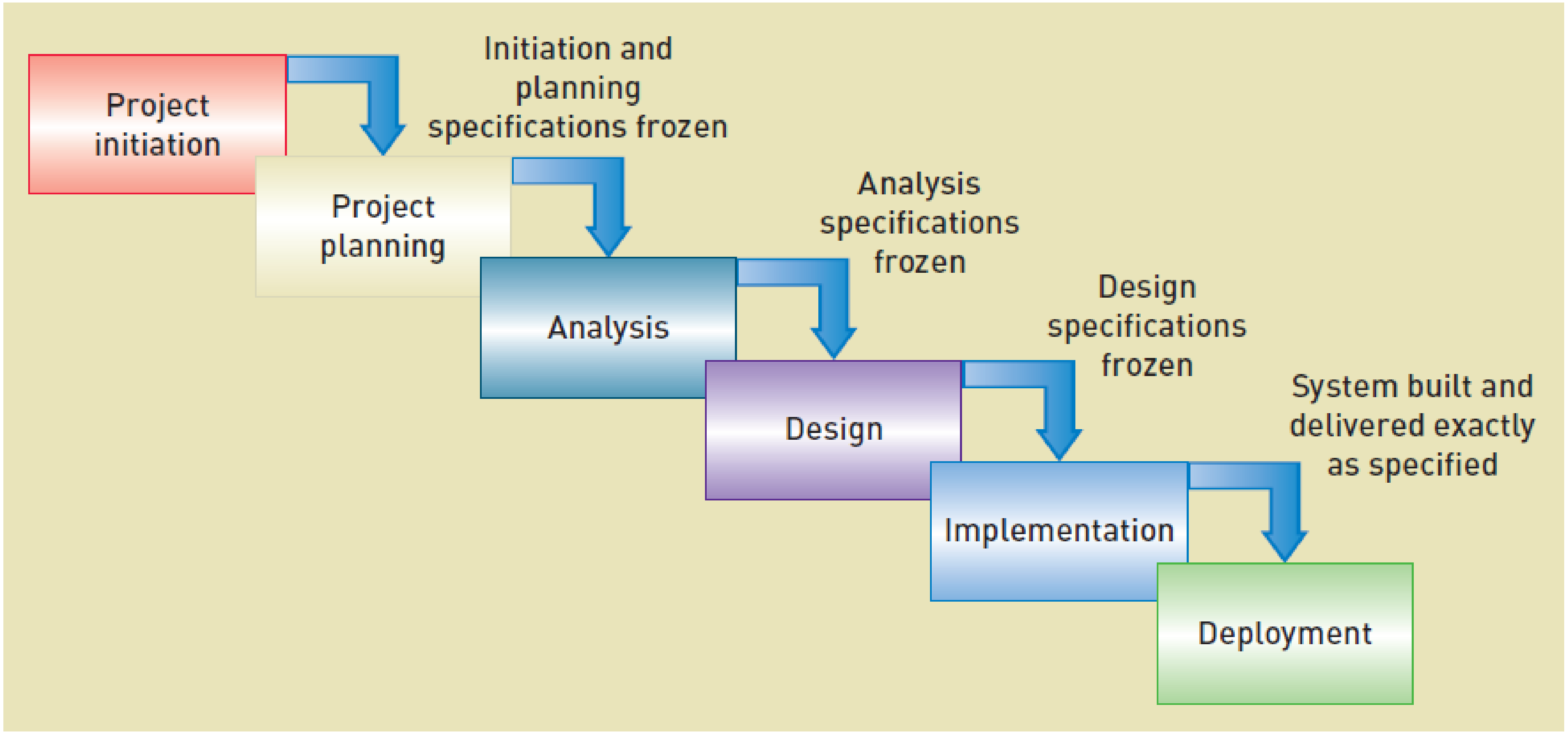
Traditional Predictive SDLC

- 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 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

Tradition IS Development Phases

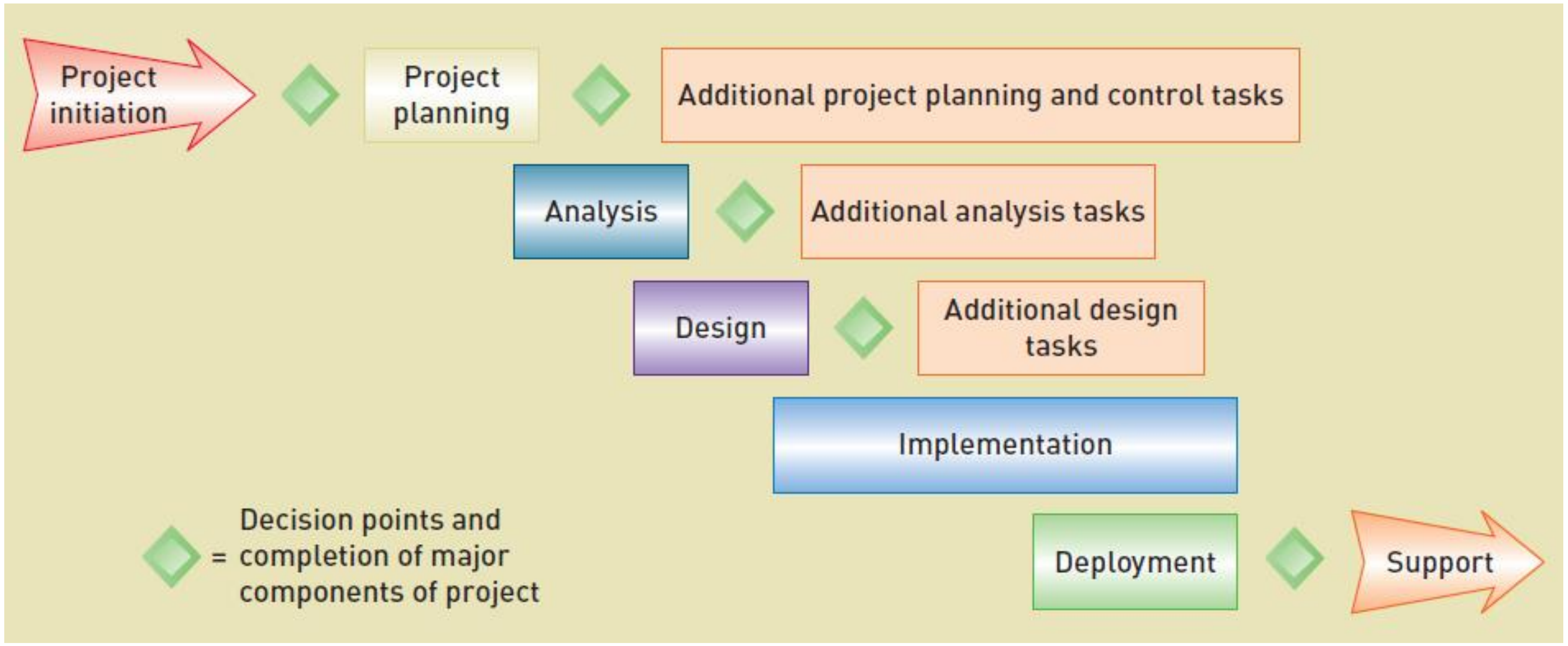


Waterfall Predictive SDLC


Project initiation
Project planning
Analysis
Design
Implementation
Deployment
Initiation and planning specifications frozen
Analysis specifications frozen
Design specifications frozen
System built and delivered exactly as specified

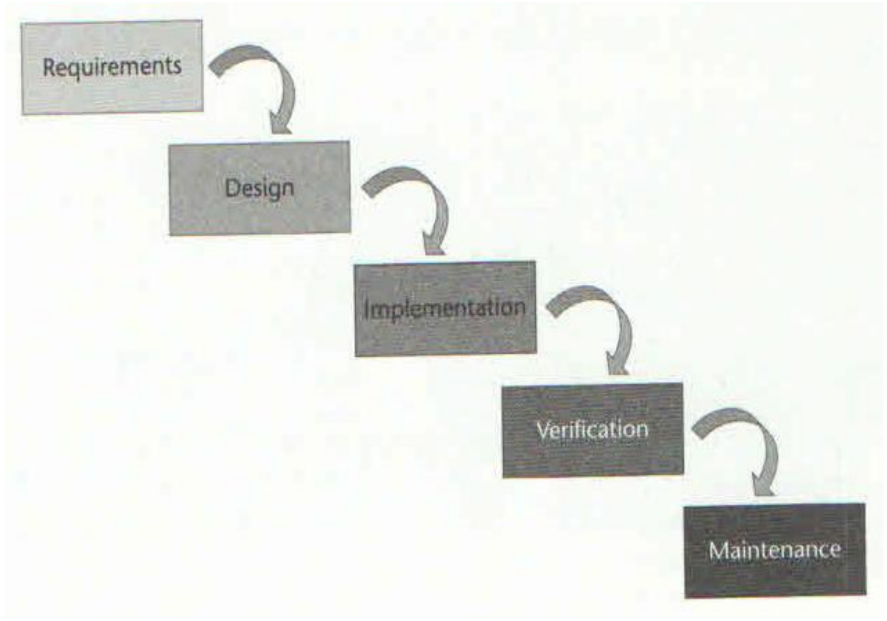
Modified Waterfall with Overlapping Phases

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


Project initiation
Project planning
Additional project planning and control tasks
Analysis
Additional analysis tasks
Design
Additional design tasks
Implementation
Deployment
Support
= Decision points and completion of major components of project

Waterfall approach problems

- The Requirements Problem
- Inflexibility
- Loss of Time-to-Market
- Customer Dissatisfaction



Sequential process

Fact: Software development is like 1 of these 3

Shoveling Snow

Assembling an Ikea Desk

Performing Heart Surgery

Newer Adaptive SDLC

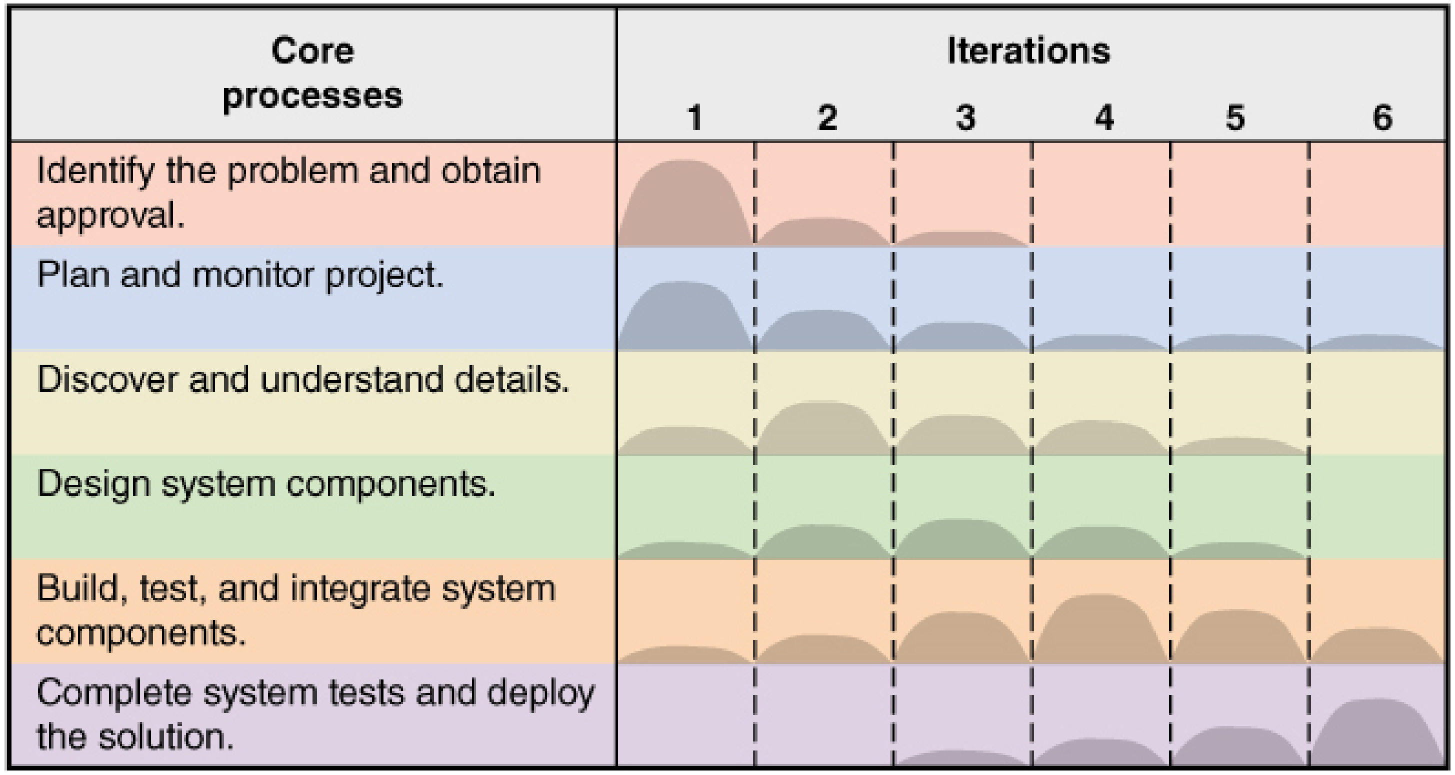
- Emerged in response to increasingly complex requirements and uncertain technological environments
- Always includes iterations where some of design and implementation is done from the beginning
- Many developers claim it is the ***only*** way to develop information systems
- Many IS managers are still 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
- Walking Skeleton
  - The complete system structure is built first, but with bare-bones functionality

A Generic Adaptive Approach

- Six Core Processes go across iterations
- Multiple Iterations as required


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.

Agile Development

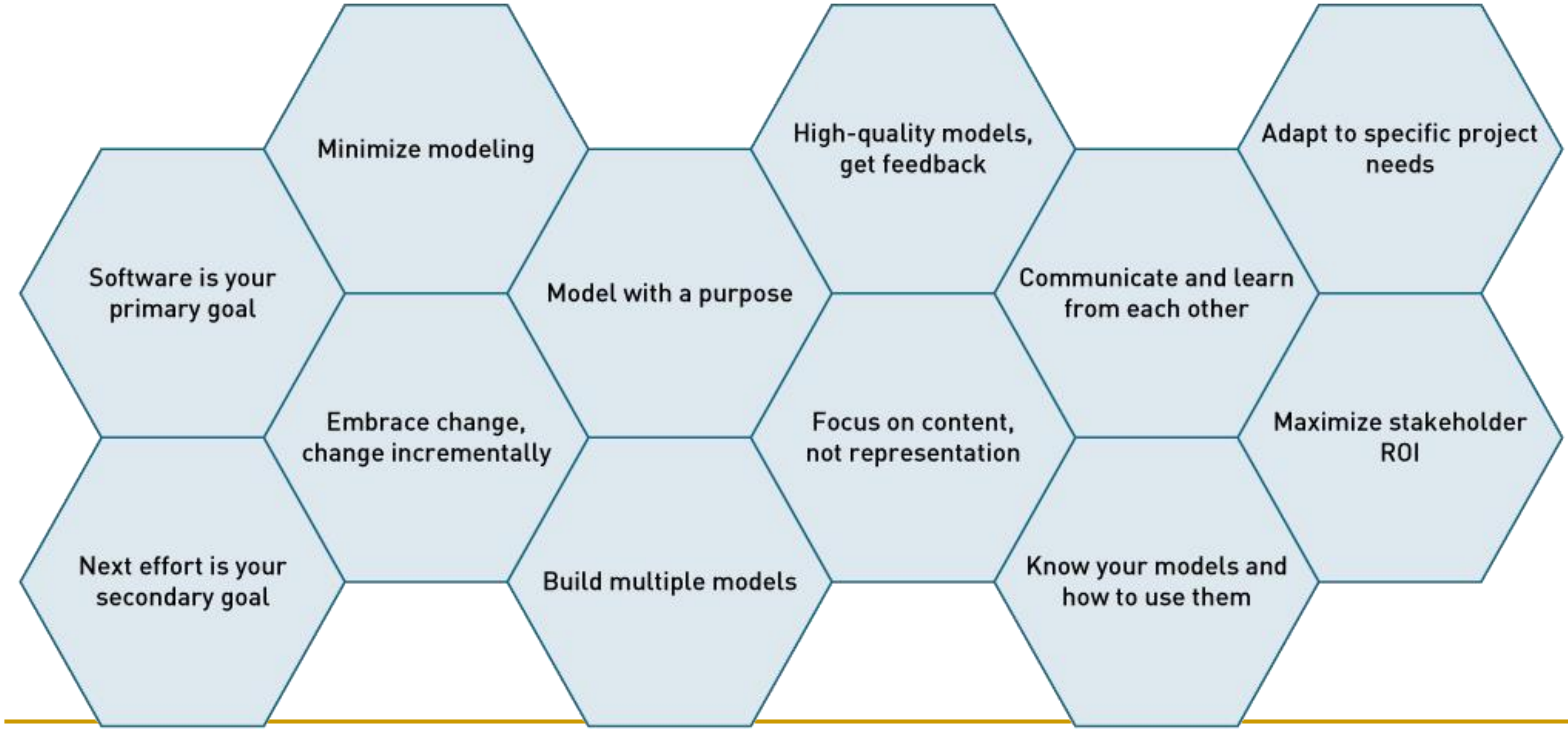
- A guiding philosophy and set of guidelines for developing information systems in an unknown, rapidly changing environment
- Chaordic

Agile Development

- 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 Principles

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

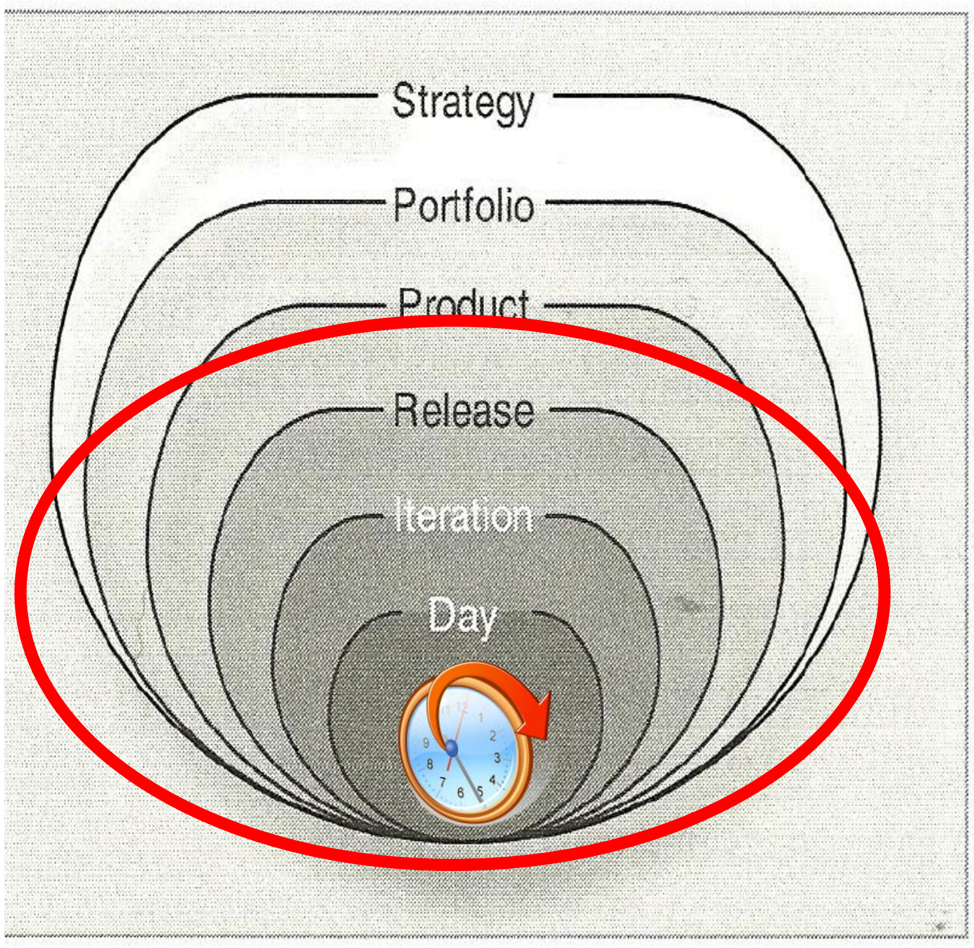

Minimize modeling
High-quality models, get feedback
Adapt to specific project needs
Software is your primary goal
Model with a purpose
Communicate and learn from each other
Embrace change, change incrementally
Focus on content, not representation
Maximize stakeholder ROI
Next effort is your secondary goal
Build multiple models
Know your models and how to use them

The purpose of Agile management

- **Goal** - to arrive iteratively to an optimal answer to the overall product development question of what to build:
  - Features, Resources, Schedule
- **Planning process = quest for value**
  - reduce risk, reduce uncertainly, support reliable decision making, convey information, establish trust

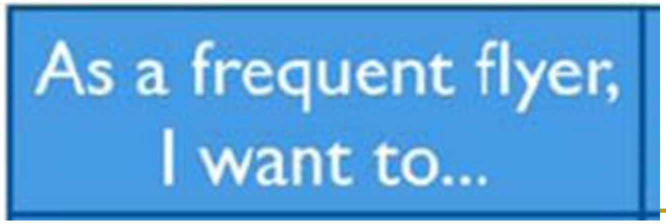
An Agile approach: the planning onion

- Agile planning:
  - encourages change,
  - results in plans that are easily changed,
  - is spread throughout the project.
- Changing the plan does not mean we change the dates



Agile Methodology Principles

- Work as one team
  - **Team:** product owner, customer, developer, project manager
- Work in short iterations
  - **timeboxed**
- Deliver user-valued features each iteration
  - Focus on business priorities
  - **User stories**



As a frequent flyer, I want to...

What is a User Story?

As a frequent flyer,
I want to...

- Short description of a feature written from the perspective of a user or customer.
- Template and example:
  - *As a <role of user>, I can <capability> so that <benefit>*
  - *As a user, I want to be able to cancel a reservation so that...*

Conditions of satisfaction (== user story acceptance tests)

  - *A user who cancels more than 24 hours in advance gets a complete refund*
  - *A user who cancels less than 24 hours in advance is refunded all but a $25 cancellation fee*
  - *A cancellation code is displayed on the site and is emailed to the user*

Personas & “Narrative Scenarios”

- Persona
  - Imaginary person represents a type or category of user
  - Can be based on a real person or set of people
- Narrative Scenario
  - Also known as “use scenario”, “task description”
  - Take key tasks your system supports and then create the scenario that your personas are likely to be in when they engage in these tasks
  - **Tell a story** that describes how each persona would be thinking and feeling and how they would behave in that situation
- The purpose for inventing personas and writing scenarios
  - Allows you to think and talk about concrete people using your system in the real world

Example: Leaf View App

- … "*rather involved and a bit too specific about user interface elements but worked well for the team*" …

  Persona: Joe is a 30 year old botanist that works at the Smithsonian Institution. He uses Microsoft Word in his daily work and a little bit of Excel, but isn't always comfortable using computers. He often carries a backpack and makes weekly trips to Plummers Island to collect and identify leaf samples. He wears glasses.

Example: Leaf View App

Use Scenario: Joe starts up LeafView and sees the browsing view, which enables him to look around at various plants. He knows he needs to start collecting today, so he closes the laptop and walks along the path looking for a specific plant. He sees what might be the right plant and cuts a leaf off the stalk, placing it on the gray background of the laptop. He snaps a picture and places the leaf in a sample bag. When he opens the laptop, he sees the acquired leaf image and can decide to search based on the image, save the image, or cancel. He searches for the leaf and the top 15 images are returned. He wants to see a comparison, so he brings the comparison image alongside the results. He zooms in on the Catalpa bignonioides species, and after reading the text and looking at the veins and edges, decides that this is the correct selection. He presses the select button and LeafView shows the information that will be saved. He clicks the save button and continues working.

How to develop good scenarios

1. Says what the user wants to do without saying how they would do it
   - no assumptions made about the interface
   - can be used to compare design alternatives in a fair way
2. Are very specific
   - says exactly what the user wants to do
   - specifies actual items the user would somehow want to input/output

Developing good scenarios (c.2)

3. Describes a complete job

- forces designer to consider how interface features work together
- contrasts how information input / output flows through the dialog
  - where does information come from?
  - where does it go?
  - what has to happen next?

- Do not
  - create a list of simple things the system should do
  - present a sub-goal independent of other sub-goals

Developing good scenarios (c.3)

- 4. Says who the users are
  - state names, if possible
  - says what they know

  - Why?
    - design success strongly influenced by what users know
    - can go back and ask them questions later
    - reflects real interests of real users
    - helps you find tasks that illustrate functionality in that person’s real world context

Developing good scenarios (c.3)

5. Are evaluated
   - Circulate descriptions among team members, users – and rewrite if needed
   - Ask users for omissions, corrections, clarifications, suggestions

6. As a set, identifies a broad coverage of users and task types

the typical ‘expected’ user,	typical routine tasks
the occasional but important user,	infrequent but important tasks
the unusual user,	unexpected or odd tasks

The Cheap Shop Catalog Store

In Cheap Shop, people shop by browsing paper catalogs scattered around the store.

When people see an item they want, they enter its item code from the catalog onto a form.

People give this form to a clerk, who brings the item(s) from the back room to the front counter.

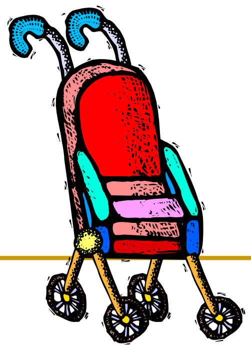
People then pay for the items they want.

SAVE $10
Century
Eddie Bauer
new
EDDIE BAUER™ BY COSCO® ALL-TERRAIN STROLLER
323 066 570 – Each
NEW! CENTURY 'EURO ADVENTURE' STROLLER
169.99

Item code	Amount
323066 697	1

Developing scenarios: Cheap Shop

- Scenario 1
  - Fred Johnson, who is taking care of his demanding toddler son, wants a good quality umbrella stroller (red is preferred, but blue is acceptable)
  - He browses the catalog and chooses the JPG stroller (cost $98, item code 323 066 697)
  - He pays for it in cash, and uses it immediately
  - Fred is a first-time customer to this store, has little computer experience, and says he types very slowly with one finger. He lives nearby on Dear Bottom Avenue NW



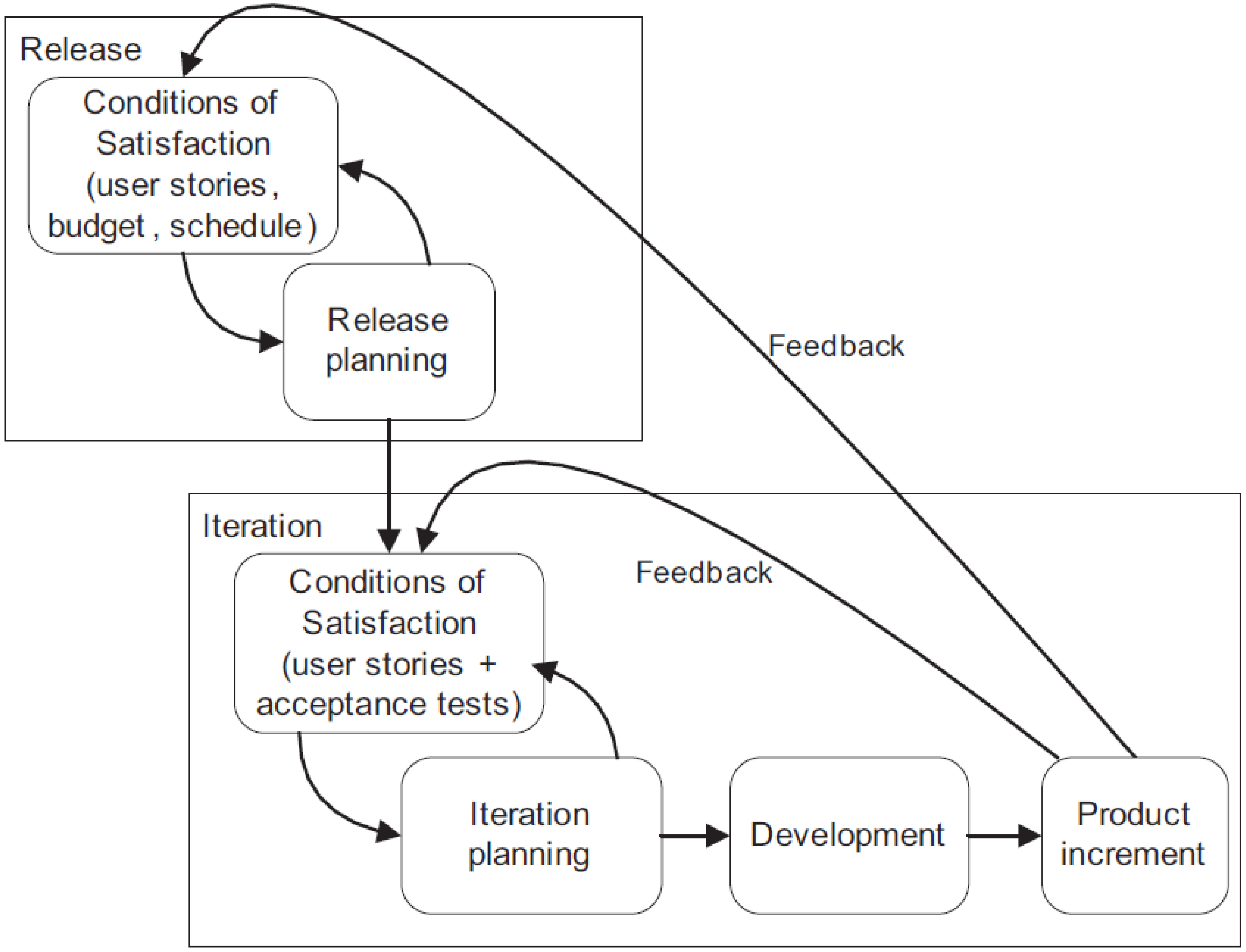
JPG Stroller. This well made but affordable Canadian stroller fits children between 1-3 years old. Its wheels roll well in light snow and mud.
…**$98.**

Red:	**323 066 697**
Blue:	**323 066 698**

Developing scenarios: Cheap Shop

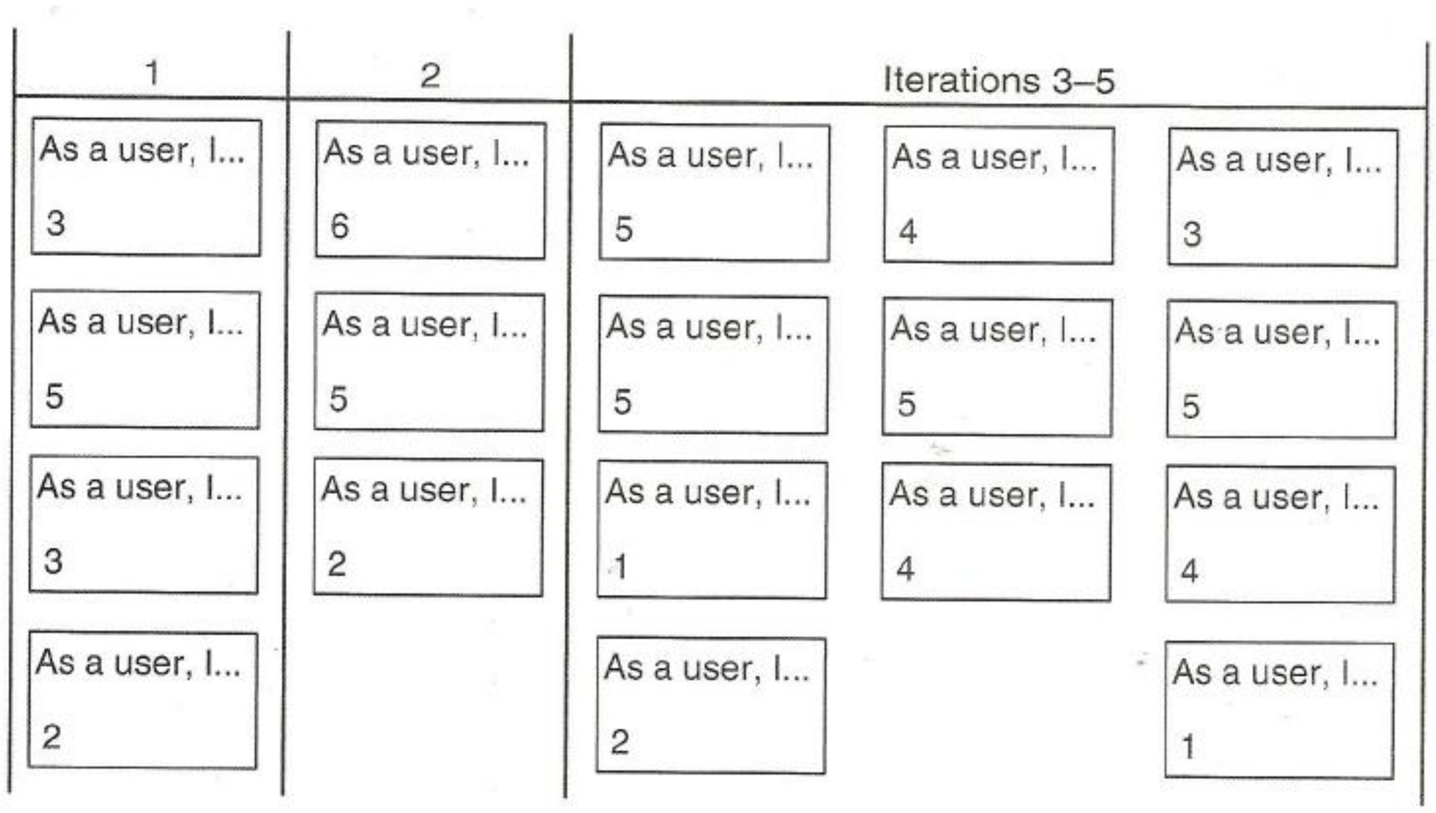
- Scenario 2
  - Mary Vornushia is price-comparing the costs of a child’s bedroom set, consisting of a wooden desk, a chair, a single bed, a mattress, a bedspread, and a pillow all made by Furnons Inc.
  - She takes the description and total cost away with her to compare with other stores.
  - Three hours later, she returns and decides to buy everything but the chair.
  - She pays by credit card
  - She asks for the items to be delivered to her daughter’s home at 31247 Lucinda Drive, in the basement suite at the back of the house.
  - Mary is elderly and arthritic.

Conditions of satisfaction drive both release and iteration planning


Release
Conditions of Satisfaction (user stories, budget, schedule)
Release planning
Feedback
Iteration
Conditions of Satisfaction (user stories + acceptance tests)
Feedback
Iteration planning
Development
Product increment

Release planning: update the release plan after each iteration!

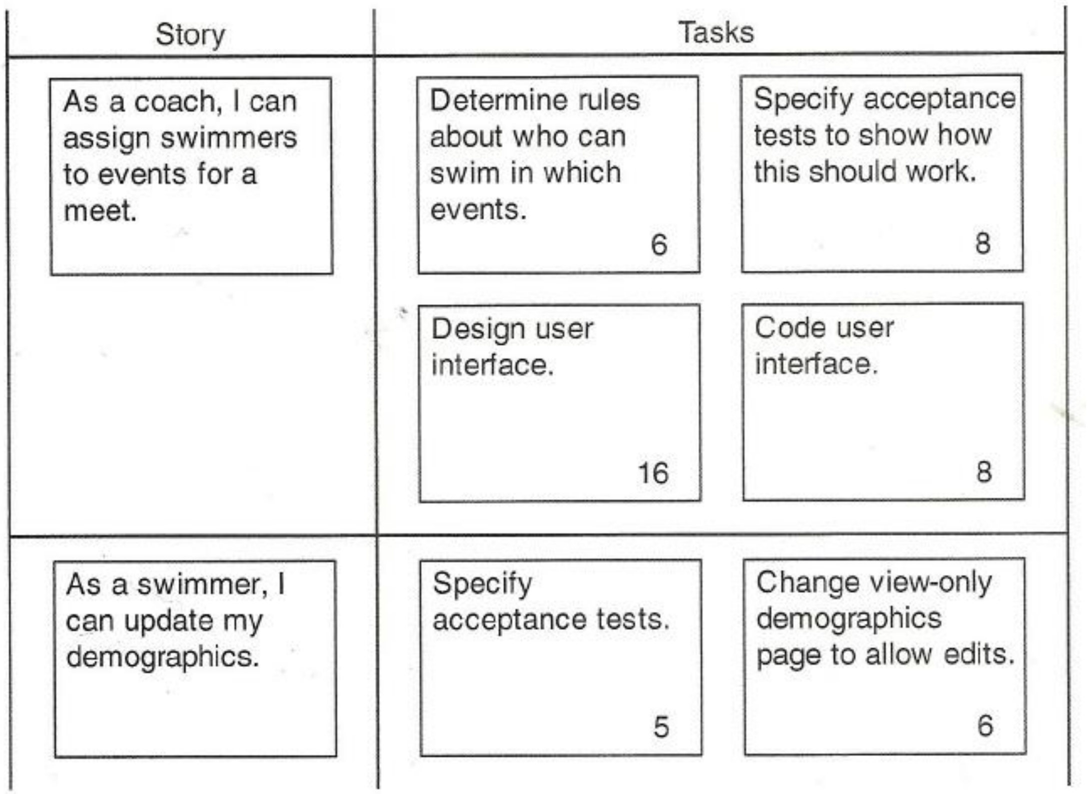
- Arranging iterations: cards are stacked to indicate which stories comprise each iteration



1	2	Iterations 3–5		
As a user, I... 3	As a user, I... 6	As a user, I... 5	As a user, I... 4	As a user, I... 3
As a user, I... 5	As a user, I... 5	As a user, I... 5	As a user, I... 5	As a user, I... 5
As a user, I... 3	As a user, I... 2	As a user, I... 1	As a user, I... 4	As a user, I... 4
As a user, I... 2		As a user, I... 2		As a user, I... 1

Iteration planning

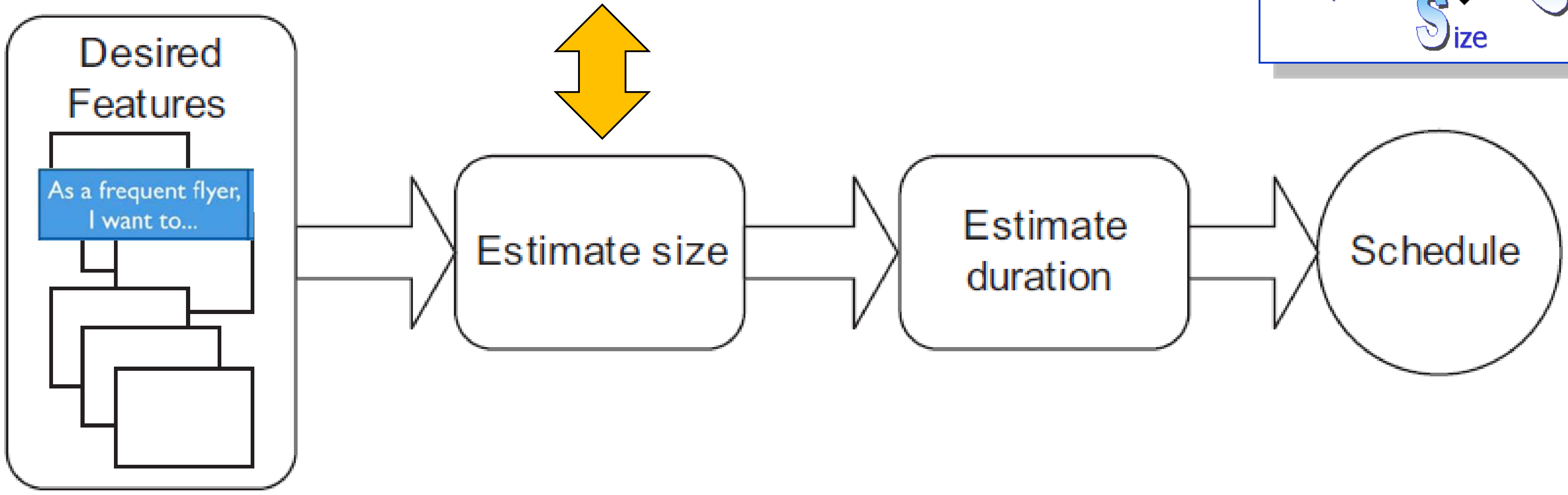
- Iteration planning meeting
- Rule of thumb”: do not allocate tasks during the iteration planning!

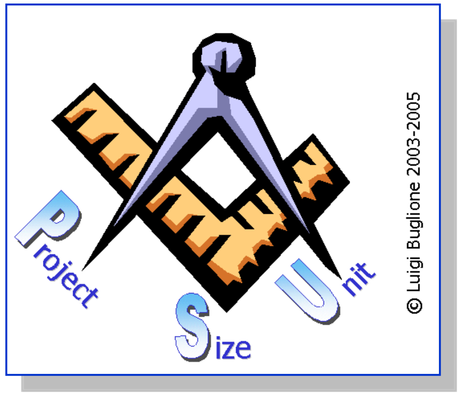


Story	Tasks	
As a coach, I can assign swimmers to events for a meet.	Determine rules about who can swim in which events. 6	Specify acceptance tests to show how this should work. 8
	Design user interface. 16	Code user interface. 8
As a swimmer, I can update my demographics.	Specify acceptance tests. 5	Change view-only demographics page to allow edits. 6

Estimating the duration of a project begins with estimating its size.

Story points





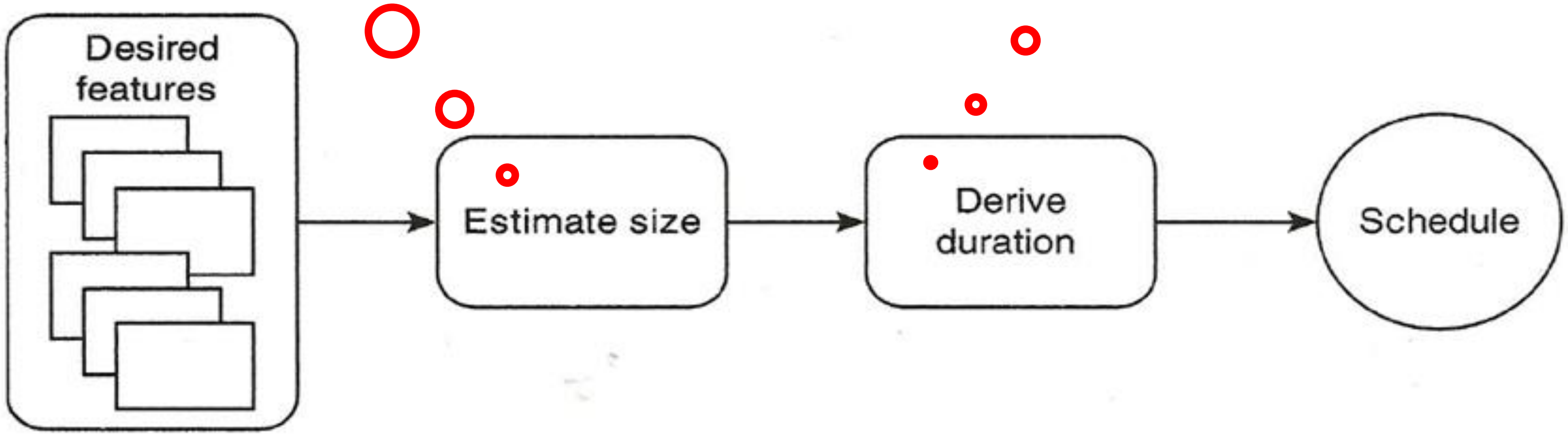
Estimating in story points: size v.s. velocity & duration

Do not give partial credit for partially completed user stories

Velocity=te

points per iteration]

10 story points/[5 story points per iteration] = **2** iterations

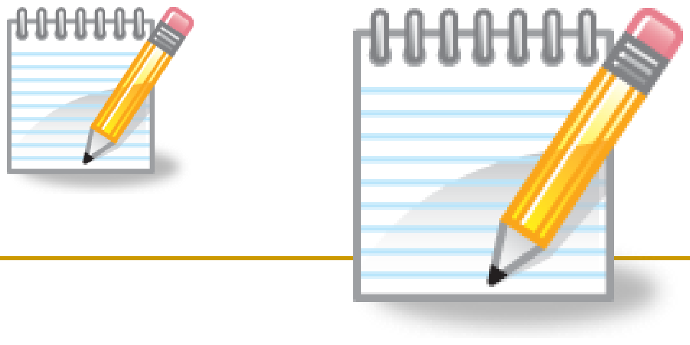


Estimating in story points: Release planning

- Purpose
  - To answer questions such as:
    - How much will be done by December 1?
    - When can we ship with this set of features?
    - How many people or teams should be on this project?
- Inputs
  - Velocity
  - The length of the project
  - Prioritized product backlog

Estimating size with story points

- Probably the most commonly used estimating unit among agile teams today
- Name is derived from agile teams commonly expressing requirements as “user stories”
- Based on a combination of the size and complexity of the work
- A 10-point user story is expected to take twice as long as a 5-point user story



Techniques for estimating

- Be aware of the law of diminishing returns on time spent estimating
- **estimation scales:**
  1. Fibonacci: 1, 2, 3, 5, 8
  2. 1, 2, 4, 8

Deriving an estimate - **Planning poker** technique

- An iterative approach to estimating, loosely based on wideband Delphi

Steps

1. Each estimator is given a deck of cards, each card has a valid estimate written on it
2. Customer/Product owner reads a story and it’s discussed briefly
3. Each estimator selects a card that’s his or her estimate
4. Cards are turned over so all can see them
5. Discuss differences (especially outliers)
6. Re-estimate until estimates converge

Planning poker - an example

Estimator	Round 1	Round 2
Susan	3	5
Vadim	8	5
Ann	2	5
Chris	5	8

Estimating in story points: Three key advantages

1. Forces the use of relative estimating
   - Studies have shown we’re better at this
2. Puts estimates in units that we can add together
3. Completely separates the *estimation of effort* from the *estimation of duration*

   We derive duration empirically by seeing how much we complete per iteration

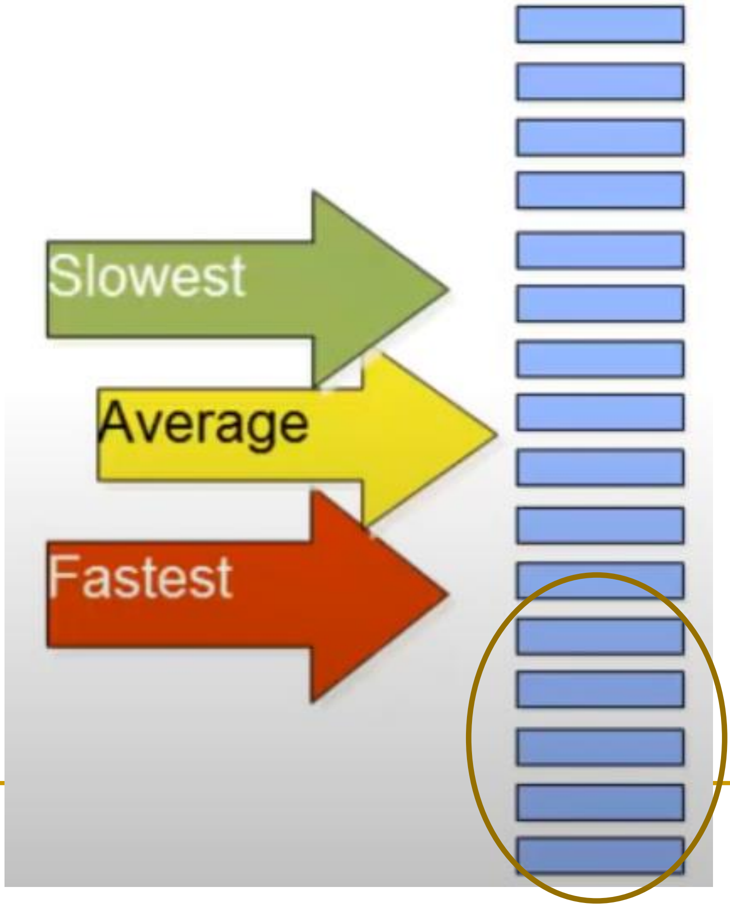
Exercises

- Suppose a project has 90 story points in total.
- Over the first 4 iterations the team has achieved a velocity of 6, 20, 10, and 17.
- You are asked when the project will be done.

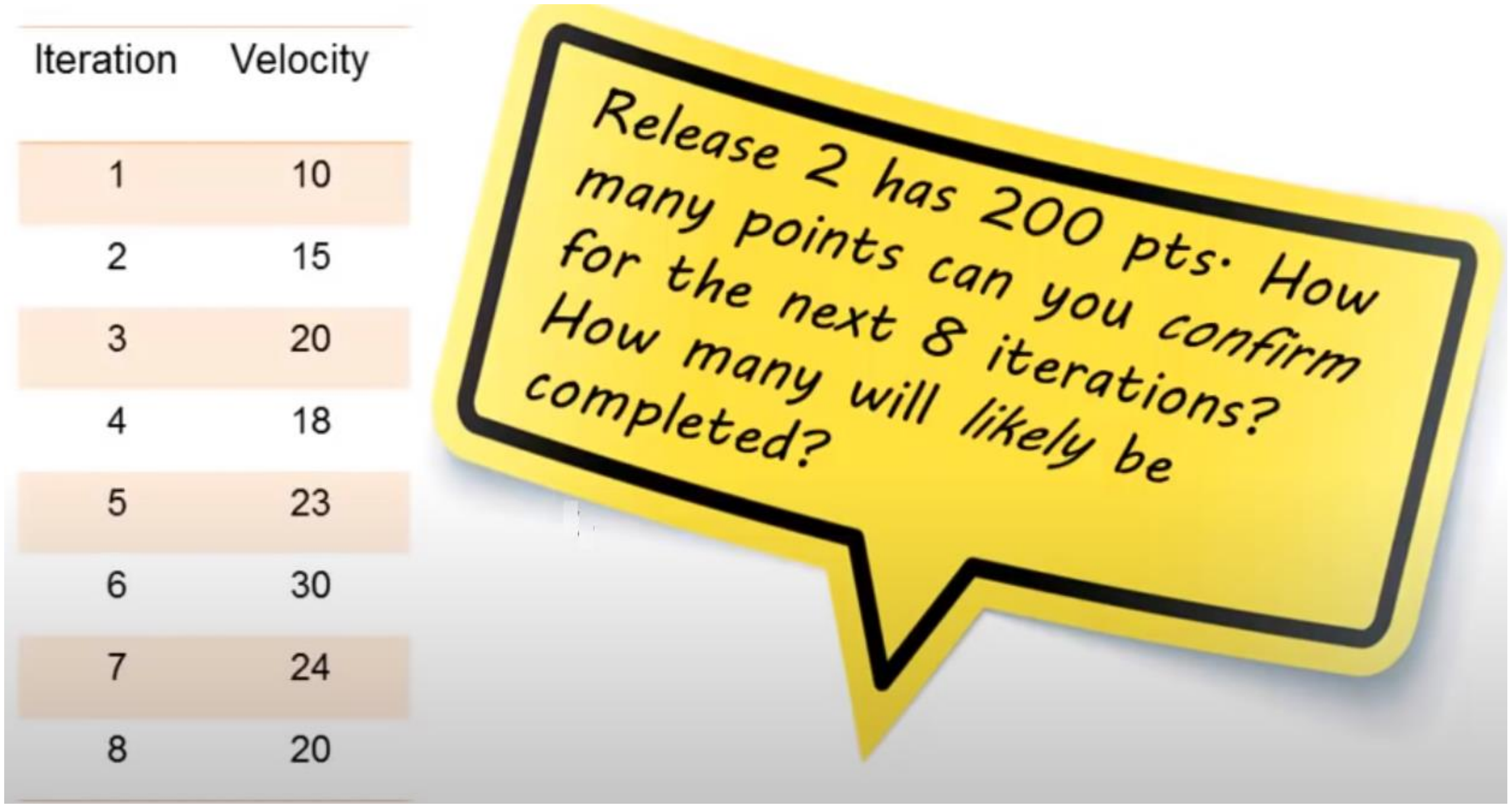
What do you say?

How velocity helps with planning?

- Understating the team’s velocity will enable better release planning and estimation.



Exercises



Iteration	Velocity
1	10
2	15
3	20
4	18
5	23
6	30
7	24
8	20

Release 2 has 200 pts. How many points can you confirm for the next 8 iterations? How many will likely be completed?

Summary

- System development life cycles
- Predictive vs adaptive
- Agile development
- Estimating and planning in agile