To Recap… many SDLCs exist

- Many SDLCs exist, each with own strengths, weaknesses and suitability
  - Build-and-fix model
  - Waterfall model
  - Rapid prototyping model
  - Spiral model
  - Prototyping Model
  - Phased Development Model
    - incremental development model
    - iterative development model
  - Formal Systems Development
  - Agile models (model or method)
    - Extreme programming
    - Scrum
    - Lean

Are you reading?

See books like Ian Sommerville, Software Engineering or almost any SE book for intro. chapter on process models

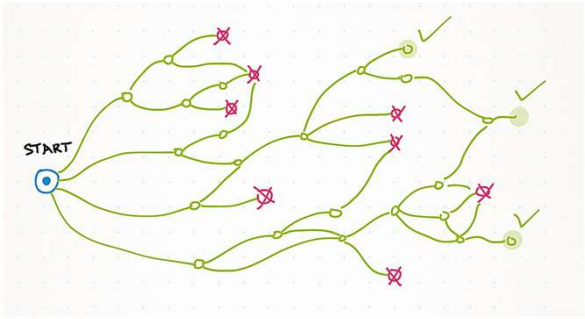
Need to know:
- What each one is
- When to use it (strengths)
- When **not** to use it (weaknesses)
- How to choose???

How do you choose an SDLC?

Do you choose a single SDLC?

Reality… many paths

Reality…
Usually many "good" paths…



Choosing an SDLC Strategy

- Choosing an appropriate SDLC strategy is **important** to the **success** of the project
- Choose the **wrong** approach and you will add time to the development cycle
  - Adding extra time to the development cycle will increase your budget and very likely prevent you from delivering the project on time
- Choosing the **wrong** methodology can also hamper your effective management of the project
  - And may also interfere with the delivery of some of the project's goals and objectives

Choosing an SDLC

- Selecting an appropriate SDLC is a **difficult** and **important** process
- It is sometimes an emotive experience
- One possibility…
  - **STEP 1:**
    - Understand the differences between (and the pros and cons of) each SDLC model.
  - **STEP 2:**
    - Assess the needs of your organization.
  - **STEP 3:**
    - Apply your criteria against your context.

Sounds simple?

- "Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple. But it's worth it in the end because once you get there, you can move mountains."

  — Steve Jobs

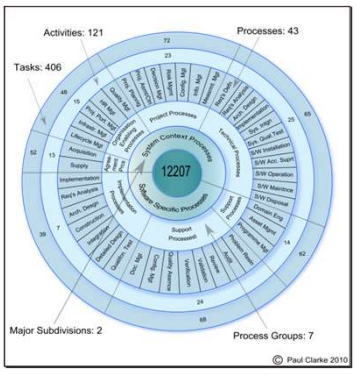
Theory Vs Reality

- So how to identify an SDLC for a real project?
- One of the issues facing organizations today is to select the right SDLC and avoid the pitfalls of "trying" something new without understanding the implications of their choice.

Recap…

- There is a **lifecycle** associated with software development – s/w components and systems from birth to death.
- Many activities in the software lifecycle: requirements, analysis, design, etc.
- Many people can be involved (often this is the case)
- => A **process** is required to structure the numerous activities and people, to ensure:
  - Adherence to budget & schedule, correct functionality delivered, appropriate quality levels achieved, etc.

There are a great many activities…



A further word Project V Product

- These definitions are not sacrosanct but often in practice, they are used in this way:
  - **Projects** can be thought of as once-off/bespoke solutions
  - **Products** attempt to isolate and componentise commonalities required across many solutions, why?
    - Helps to reduce costs (sometimes significantly: think of the increased efficiencies associated with reduced testing, maintenance & support)
    - Reduced costs = increased profits
    - Increased profits = increased company valuation

What is the best process?

- There is no ***best*** process or process model but there is a most appropriate process for any given context
- **What is context?** These are the circumstances that a software development effort are faced with
- The generic models presented earlier are appropriate only in a generalised type of way
- No two software development contexts are identical
- Some degree of *synthesis* / *tailoring* / *adaptation* of a generic model (or models) is required to address the contextual factors

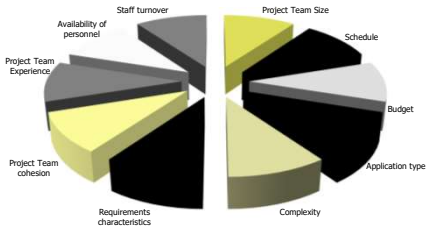
Selection of a lifecycle model

- Many **contextual factors** affect the suitability of a lifecycle model

See research paper on Loop…
The situational factors that affect the software development process: Towards a comprehensive reference framework

- The purpose of the following slides is:
  - Identification of some of the more significant factors
  - Explanation of why they are important
  - Explanation of the impact the factors have on the lifecycle model selection

Contextual factors



Team size

Team Size

1
2
5
15
50
250+

Increasing potential for breakdown of comms & task handovers = Increasing cost of lack of process efficiency = Increasing need for greater process awareness & explicit process managment

Schedule

Delivery date

No fixed delivery date
↓
Explicit delivery date
↓
Explicit delivery date with late penalties applying

Increasing need for certainty in relation to requirements (and **earlier** in the S/W development effort) = Increasing need for process to elicit requirements upfront

Application type

Application

Small web-based ads
↓
Financial data processing
↓
Aircraft flight control

Increasing cost of software failure = Increasing need for process to reduce/contain failure events

Application complexity

complexity

Low ↓ Medium ↓ High

Increasing probability of S/W failures

Increasing need for process to prevent failures

Requirements characteristics

Rerquirements

Not understood / Innovating ↓ Reasonably well understood ↓ Explicitly defined & fixed

Increasing probability / frequency of changing / emerging requirements

Increasing need for process to accommodate reqs changes

Team experience with...

Tech.	Domain	Each other	Company
Low	Low	Low	Low
↓	↓	↓	↓
Medium	Medium	Medium	Medium
↓	↓	↓	↓
High	High	High	High

Increasing need for process to focus on planning, progress & qual. Control (and other concerns?)

Staff availability

Availability

Highly volatile ↓ Medium risk of outages ↓ 100% Committed

Increasing probability of staff shortages

Increasing need for process to accommodate staffing volatility

Many other contextual factors...

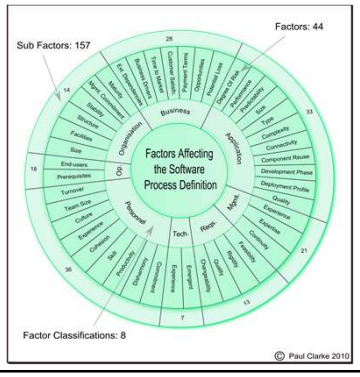
Category	Factors
Project	**Project size**
	Schedule
	Budget
	Type
	Complexity
	Requirements characteristics
	Project drivers
Team	**Team size**
	Cohesion
	Turnover of personnel
	Experience
	Resistance to changes
	Availability of personnel
External stakeholders	Number of stakeholders
	Stakeholders' background
	Access to stakeholders
Organisation	Organisation policy
	Process maturity
	Senior management commitment
	Technical Support

From: Xu & ramesh (2007)

Classification	Risk Factor
User Risk	Users resistant to change
	Conflict between users
	Users with negative attitudes toward the project
	Users not committed to the project
Requirements Risk	Lack of cooperation from users
	Continually changing system requirements
	System requirements not adequately identified
	Unclear system requirements
	Incorrect system requirements
Project Complexity Risk	Project involves the use of new technology
	High level of technical complexity
	Immature technology
	Project involves use of technology that has no been used in prior projects
Planning & Control Risk	Lack of an effective project management methodology
	Project progress not monitored closely enough
	Inadequate estimation of required resources
	Poor project planning
	Project milestones not clearly defined
	Inexperienced project manager
	Ineffective communication
Team Risk	Inadequately trained development team members
	Inexperienced team members
	Team members lack specialized skills required by the project
Organizational Environment Risk	Change in organizational management during the project
	Corporate politics with negative effect on project
	Unstable organizational environment
	Organization undergoing restructuring during the project

Six dimensions of project risk (incl. factors) (Wallace, Keil and Rai 2004)

Many other contextual factors...



Questions?