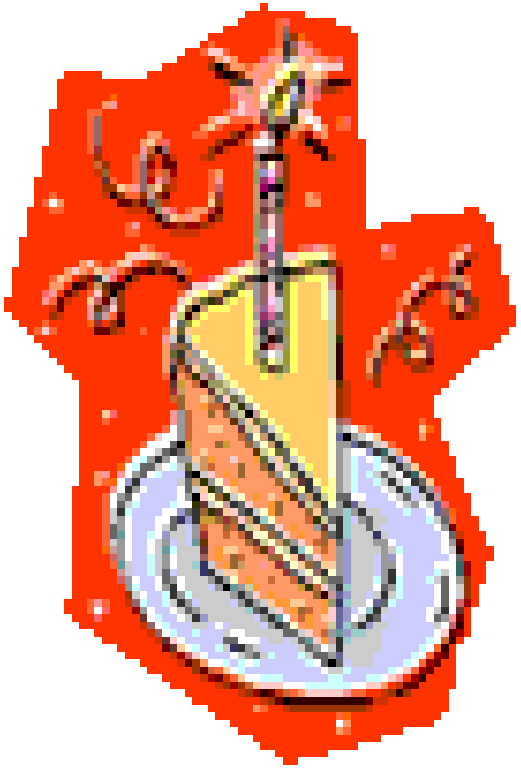
CMPT 276 Class 04: Software Processes

Dr. Jack Thomas
Simon Fraser University
Fall 2020



Fun theory

Lecturers

Practical, technical
Development skills

Today's Topics

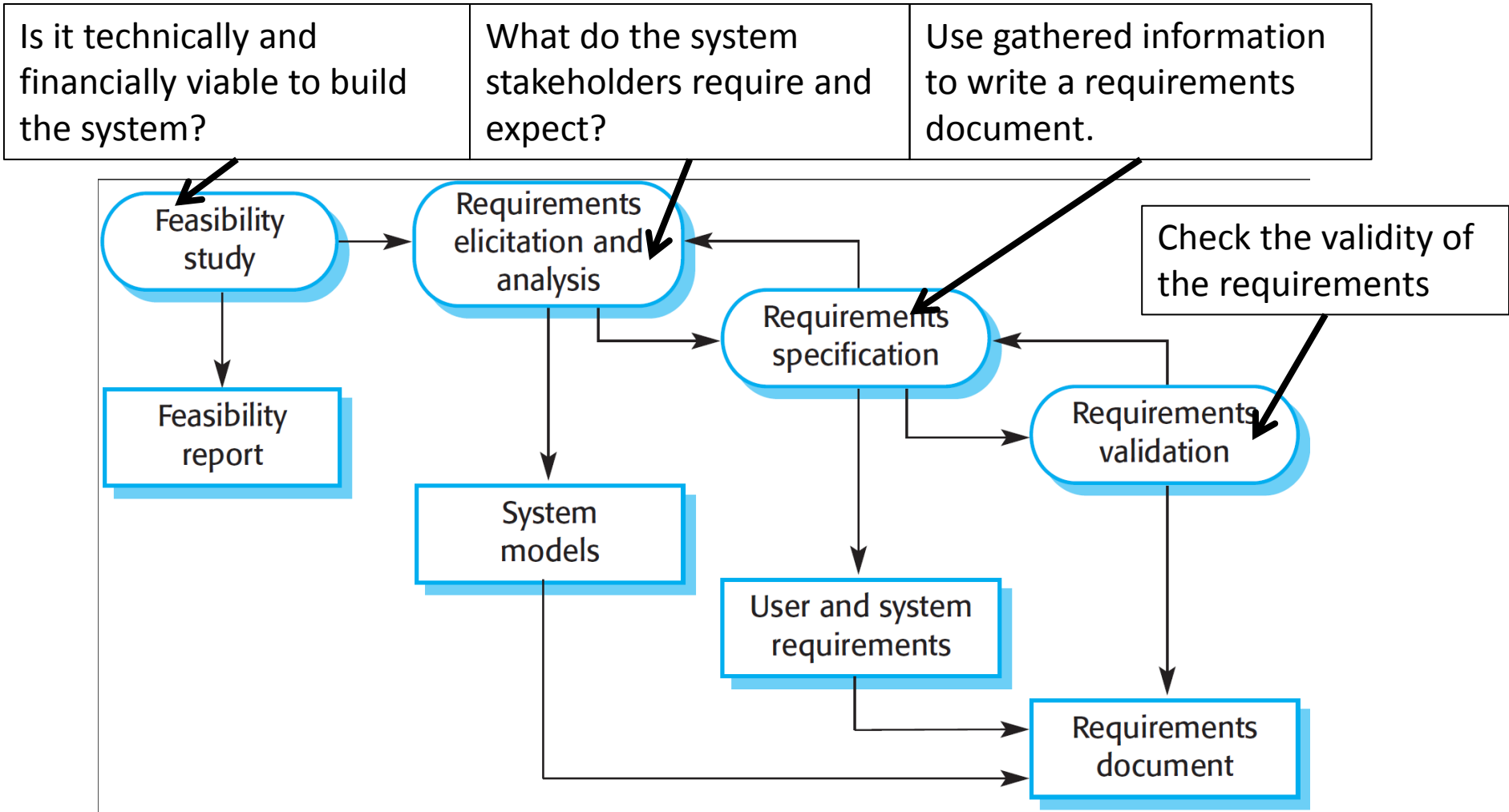
1. What **activities** are part of software development?
2. What are **software process models**?

Process Activities: The Software Process

- **Software Process**:
  - A structured set of **activities** required to develop a software system.
- All software processes involve:
  1. **Specification** – What will the system do?
  2. **Design & implementation** – How will it do this? Also, actually making it.
  3. **Validation** – Does it do what the customer wants?
  4. **Evolution** – Change the system to meet the customer's changing needs.
- A **software process model** is an abstract representation of a real process.

1. Software Specification

- Establishes what services are required and what constraints exist on the system's operation and development.



2. Software Design And Implementation

- The process to convert a specification into an executable system.

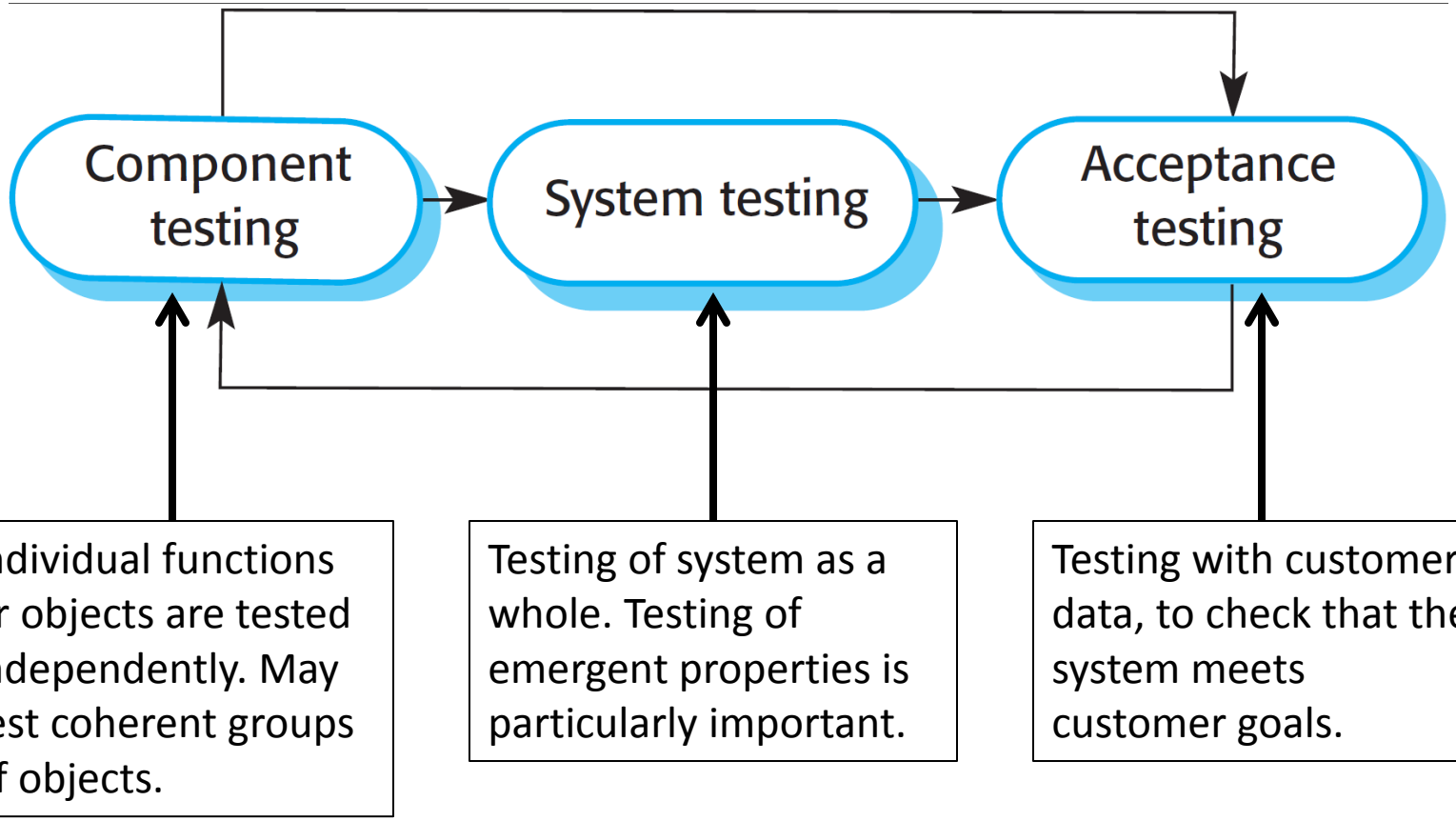
- Design and implementation are closely related and may be interleaved.

Design Activity	Description
Architectural Design	Identify overall structure of the system & principle components:.. sub-systems or modules.
UI design	Layout initial ideas for user interface (UI).
Component design	Design each system component
Database design	Design the system's data structures and database

3. Software Validation

- Checks the system conforms to its specification and meets customer's requirements.
- Involves **testing**:
  - Create test cases which ensure the system behaves correctly for some component/feature.
  - Works best if using real-world data.
- Can involve **Formal Verification**, logically proving that a system operates correctly.
  - Hard in practice; often restricted to critical components of life-critical components.

Testing Stages



4. Software Evolution

- Software is inherently flexible and can change.
- Software must change to meet new business needs.
  - Most of a project's **time** and **cost** associated with **maintenance**
- The programming stereotype is that **development is creative and interesting**, but **maintenance is dull**.
- This is increasingly irrelevant as most **new systems** are built on **existing components**.
- Line between old and new is blurring.

Software Processes

- Describe each process by:
  - The **activities** in the process, such as designing how data is stored, or the user interface, etc
  - The **ordering** of these activities.
- All processes involve the four basic activities of specification, development, validation and evolution.
- Two big questions:
  - **Planning**: Done up front? Or as you go?
  - **Delivery**: Done at the end? Or multiple times?

Planning Paradigms

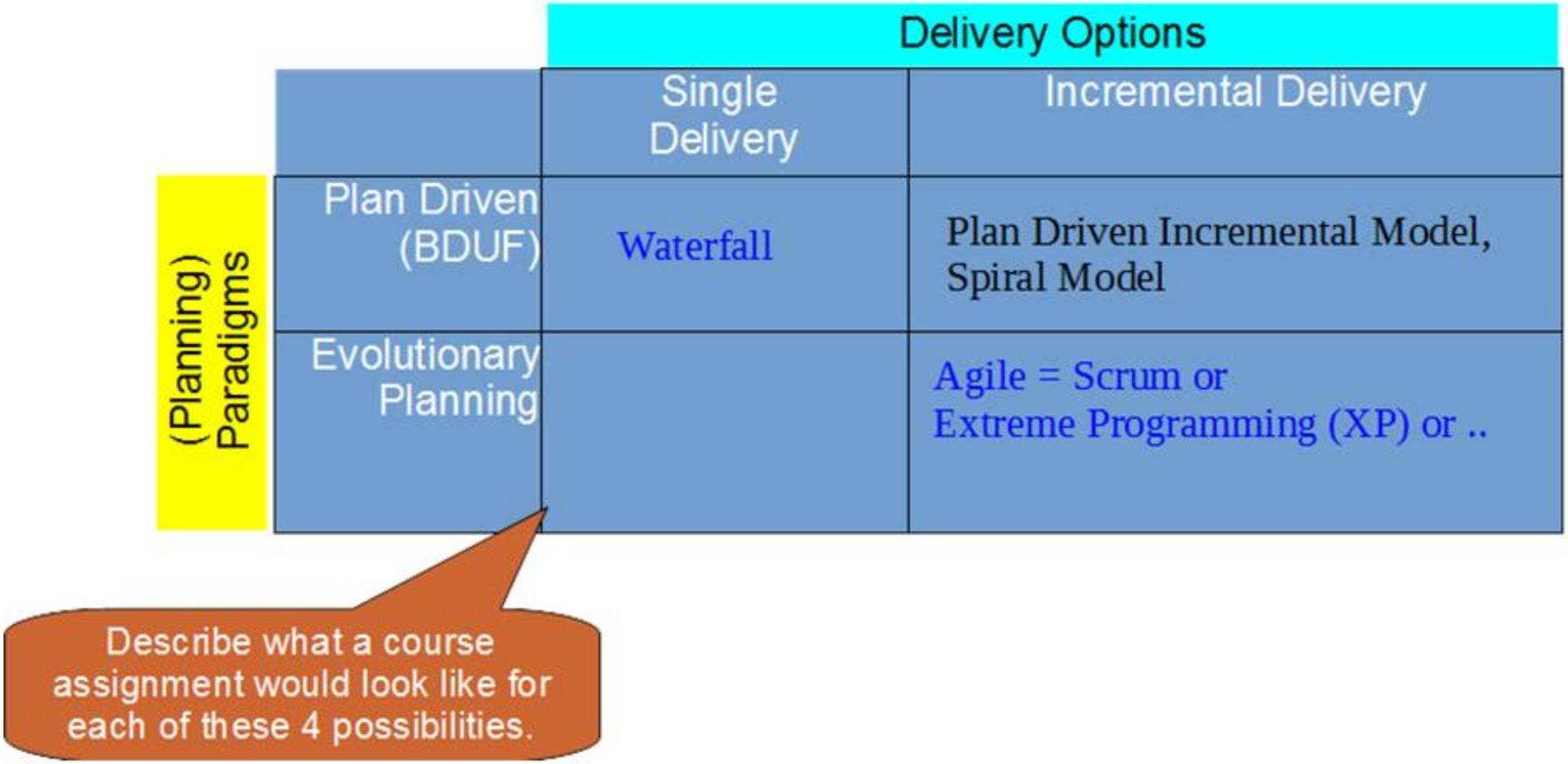
- **Plan-driven** processes:
  - All process activities are planned in advance.
  - Progress is measured against this plan.
  - Also called Big Design Up Front (**BDUF**).
- **Agile** processes:
  - Planning is incremental.
  - Easier to change the process to reflect changing customer requirements.
- Most practical processes include elements of both plan-driven and agile approaches. **There's no right or wrong software process**

Delivery

- **Single Delivery** (at end)
  - The software is only delivered to the customer once it's fully completed.

- **Incremental Delivery**
  - The customer is given incomplete versions of the software throughout development.

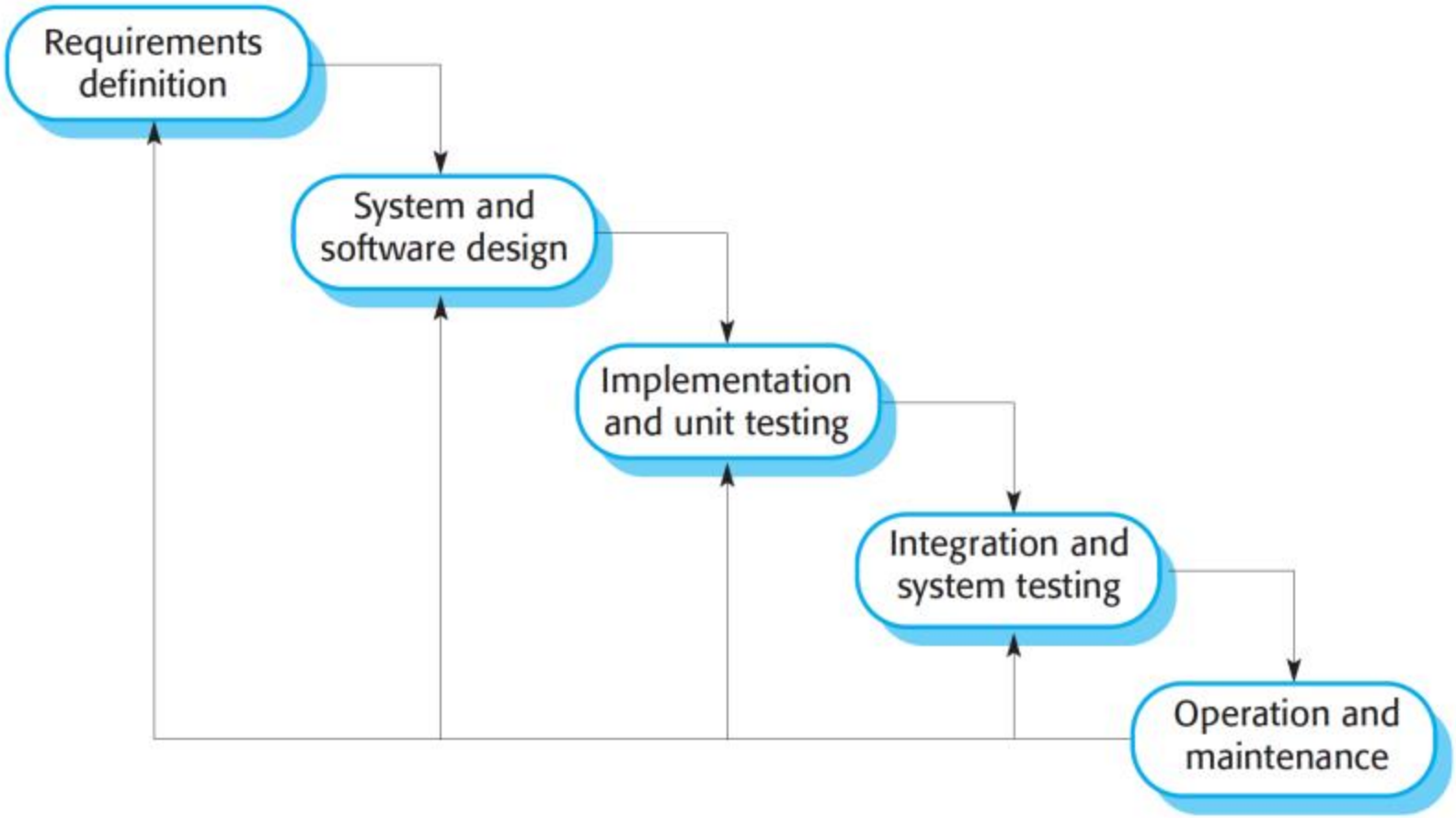
High-Level View of Software Processes



Software Process Models

- **The Waterfall Model**
  - Plan-driven model – **separate and distinct phases** of specification and development.
- **Incremental Development**
  - Specification, development and validation are **interleaved**.
- **Agile**
  - Lightweight process to adapt to changing requirements.
- Most large systems developed using a process that incorporates elements from multiple models.

Waterfall Model Phases



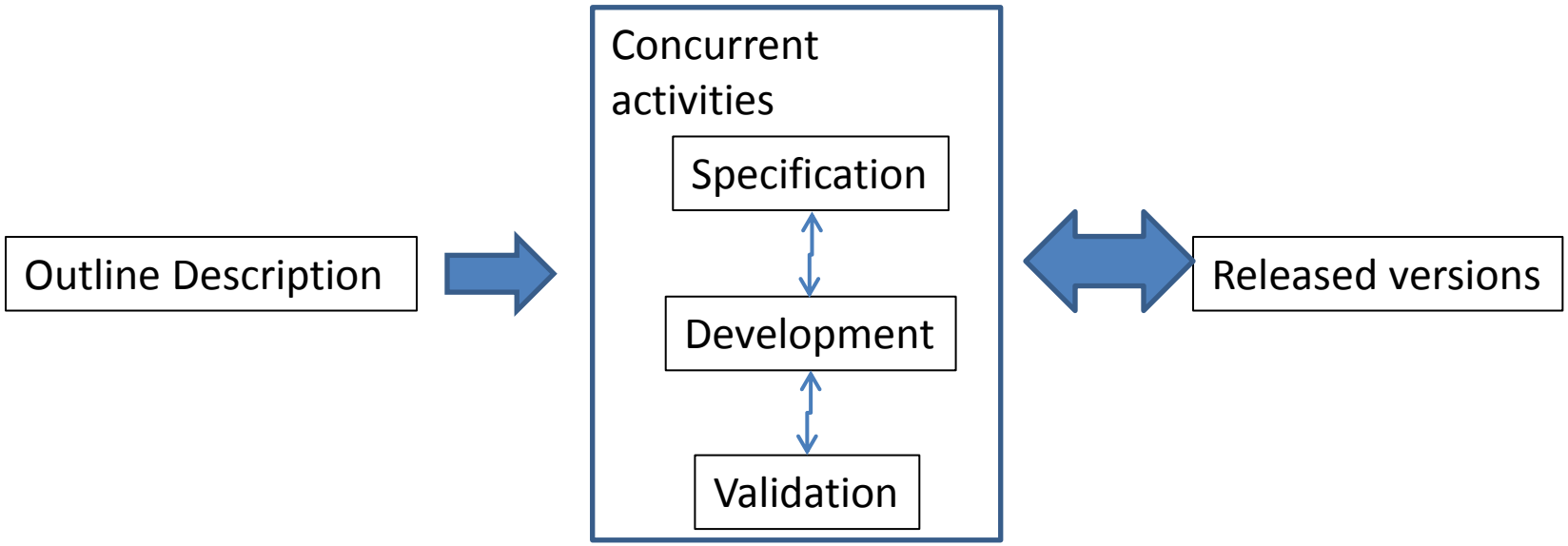
Separate and distinct phases in the process.

Waterfall Model Problems

- **Inflexible** stages make it difficult to meet changing customer requirements.
  - "Must complete phase N before starting phase N+1."
- Waterfall model is (somewhat) appropriate when requirements are well understood and changes are limited.
  - **Few business systems** have stable requirements.
- Plan-driven nature of the waterfall model **helps coordinate** the work.
- However waterfall is so rigid it is virtually **never used** as a full methodology.

Incremental Development

- The waterfall model delivers the **full system** to user at the end of the process.
- Incremental development delivers **incomplete intermediate versions**.



Incrementalism And Its Benefits

- Incremental development usable by either paradigm
  - **Plan Driven Models**: Functionality of increments are planned in advance.
  - **Agile Models**: Functionality of early increments are planned, later increments driven by customer needs.
- Reduced cost from changing customer requirements.
  - Not as much code (plan?) written that must change.
- Quick delivery of useful software.
  - Easier to get customer feedback on working software rather than paper designs.
  - Customer uses and gains value from the software earlier than with a single end delivery process.

Incremental Problems

- **Code Rot**:
  - Regular changes tend to corrupt a system's structure.
  - Incorporating code changes becomes increasingly difficult and costly.
  - Time and money must be spent refactoring to improve the software.

Refactoring

- A fancy word for making the code better without adding new features.
- **Refactoring Examples**:
  - Rename a poorly named variable.
  - Split huge function into smaller ones.
  - Improve the Object Oriented Design.
  - Fixing parts of the code which have poor code quality or poor readability.

Agile

- Agile methodologies are lightweight, they try to **reduce process overhead**.
  - Ex: Only as much documentation and planning as needed.
- Develop application in **short iterations**
  - ~1-3 weeks long.
  - Select features at the start of each iteration.
  - Deliver working software at end of each iteration.
- Very common in industry
  - Whole slide-deck on it soon!

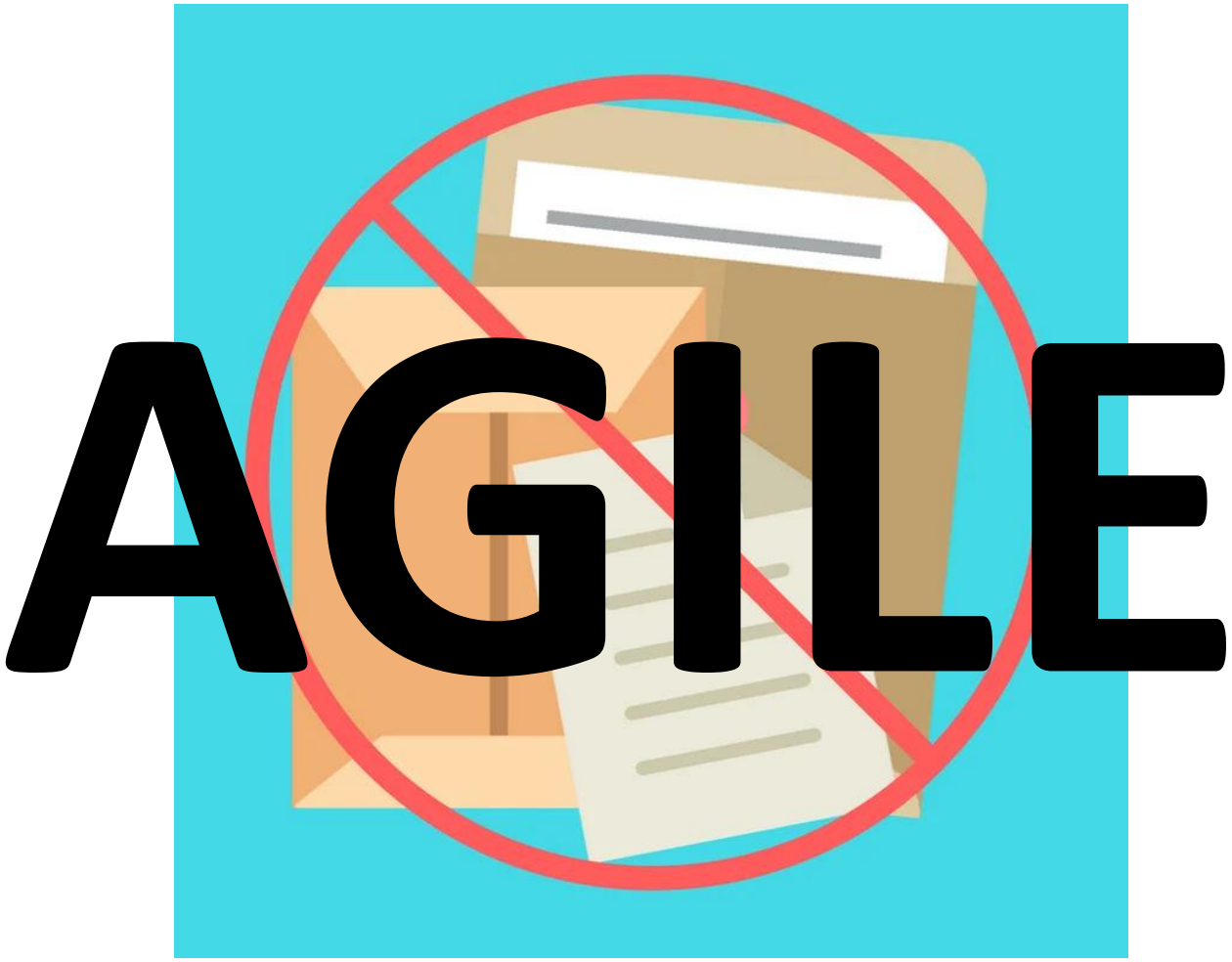


Image credit: https://loanscanada.ca/loans/no-and-low-documentation-personal-loans-in-canada/

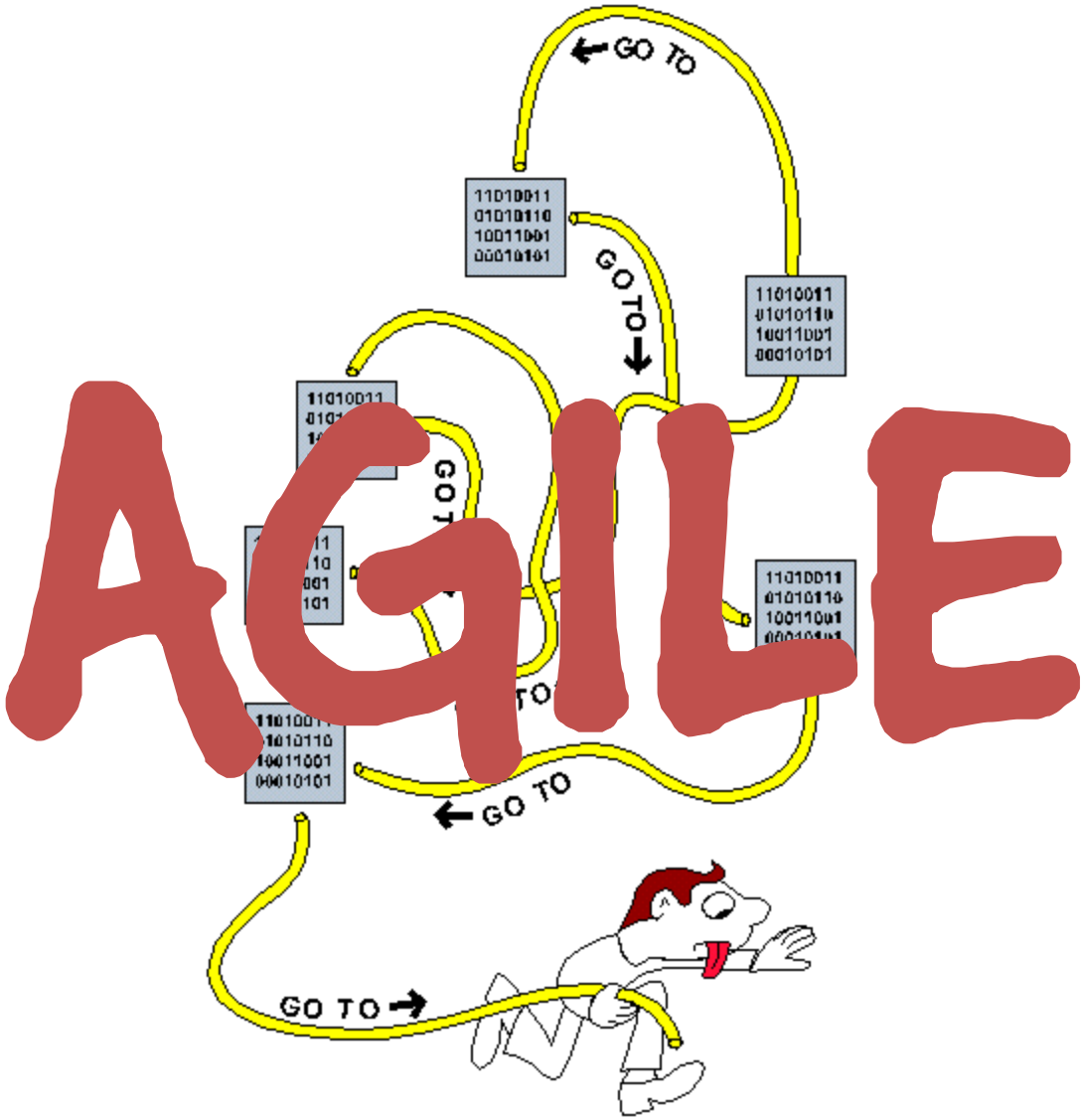


Image credit: https://www.pcmag.com/encyclopedia/term/spaghetti-code

Image credit: https://www.datanumen.com/blogs/5-common-signs-computer-going-crash/

DOC JACK'S CYNICAL REALISM CORNER

- Many of these activities and models were developed to describe how people already worked on software, not the other way around.
- Often used to justify or cover up flaws in the process.
- This goes both ways – both managers and programmers use buzz words to try and deflect blame.
- Creating a chain of accountability is more important than improving the final product.

Recap – The Process Of Summarization

- **Software processes** are the **activities** involved in producing a software system.
  - **Requirements engineering**: develop the **specification**.
  - **Design and implementation**: transform requirements specification into an executable software system.
  - **Software validation**: check the system conforms to its specification and meets the needs of its users.
  - **Software evolution**: change existing software systems to meet new requirements.
- **Process models** describe a sequence of activities: '**waterfall' model**, **incremental development**, and **agile development**.