LECTURE 1

SOFTWARE PROCESS MODELS

TOPICS

Software project lifecycle

Black box and white box models

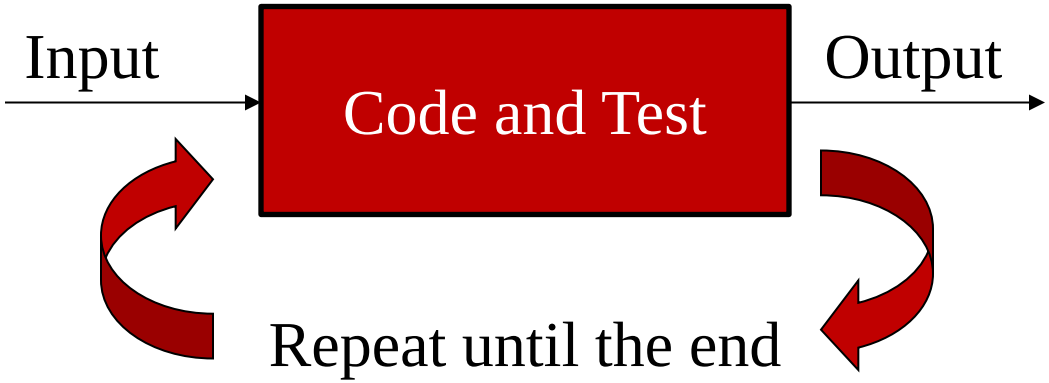
Waterfall model

Iterative model

Agile models

- Scrum

EARLY DAYS OF SOFTWARE DEVELOPMENT



Any problems with this approach?

INTRODUCTION

uOttawa
L'Université canadienne
Canada's university

The life cycle of a software product

- from inception of an idea for a product through
  - domain analysis
  - requirements gathering and modeling
  - architecture design and specification
  - coding and testing
  - delivery and deployment
  - maintenance and evolution
  - retirement

SOFTWARE DEVELOPMENT MODELS ARE NEEDED

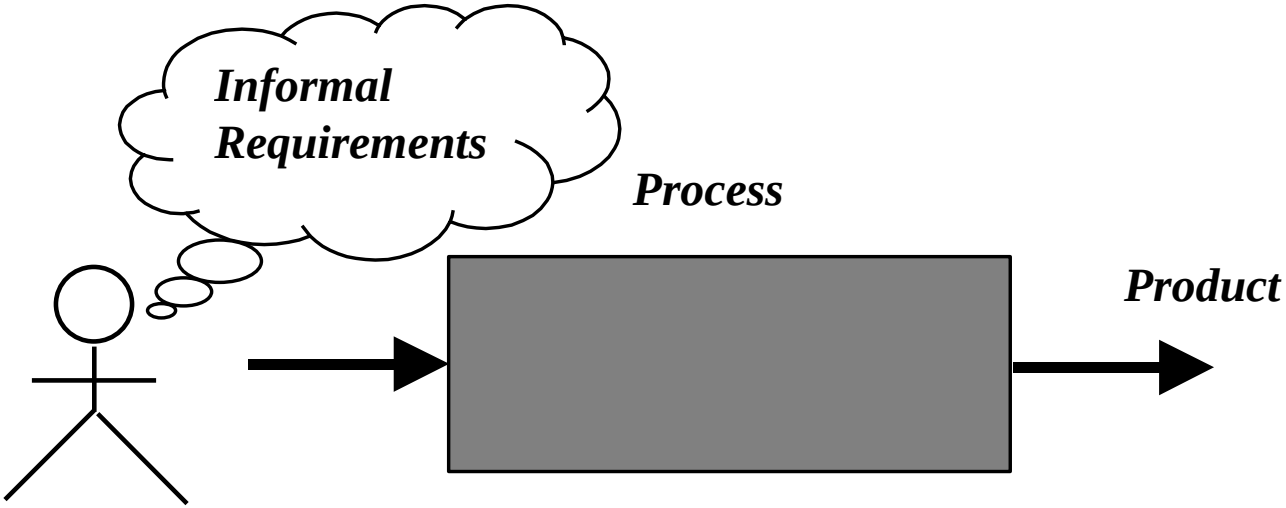
uOttawa
L'Université canadienne
Canada's university

Symptoms of inadequacy: the software crisis

- scheduled time and cost exceeded
- user expectations not met
- poor quality

The size and economic value of software applications required appropriate "process models"

PROCESS AS A "BLACK BOX"



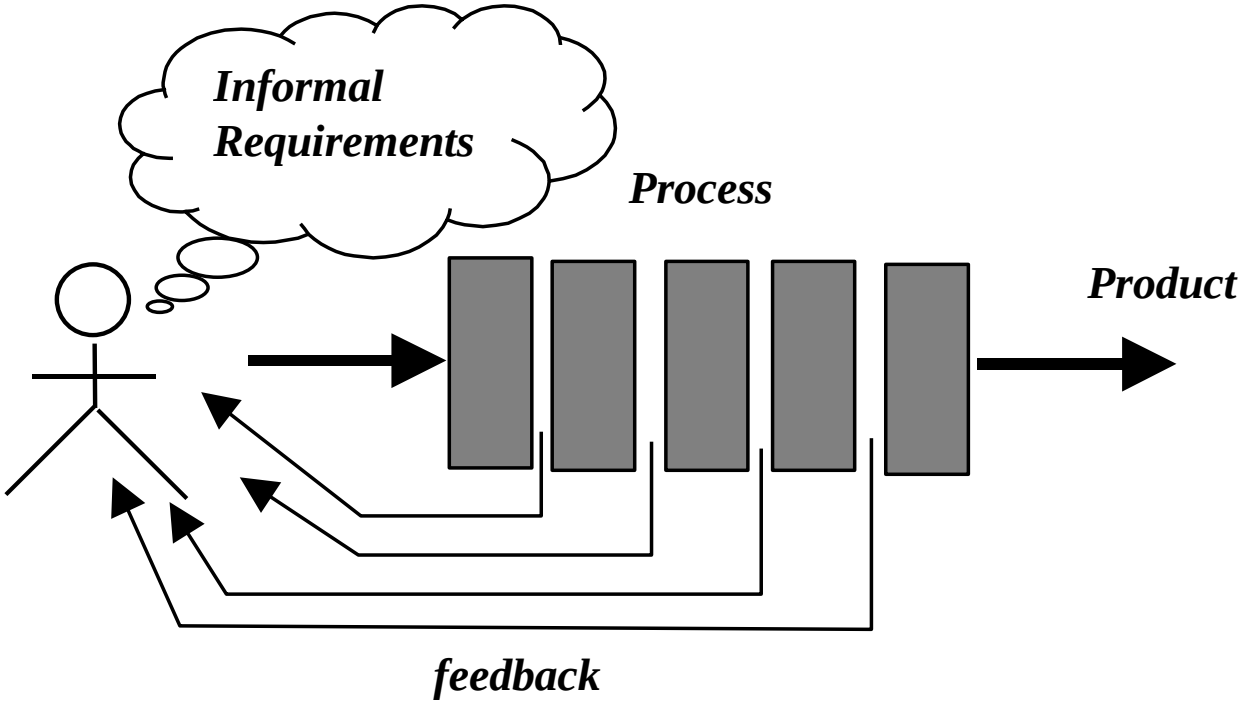
PROBLEMS

The assumption is that requirements can be fully understood prior to development

Unfortunately the assumption almost never holds

Interaction with the customer occurs only at the beginning (requirements) and end (after delivery)

PROCESS AS A "WHITE BOX"



ADVANTAGES

Reduce risks by improving visibility

Allow project changes as the project progresses

- based on feedback from the customer

MAIN ACTIVITIES

The main activities must be performed independently of the model

The model simply affects the flow among activities

WATERFALL MODELS

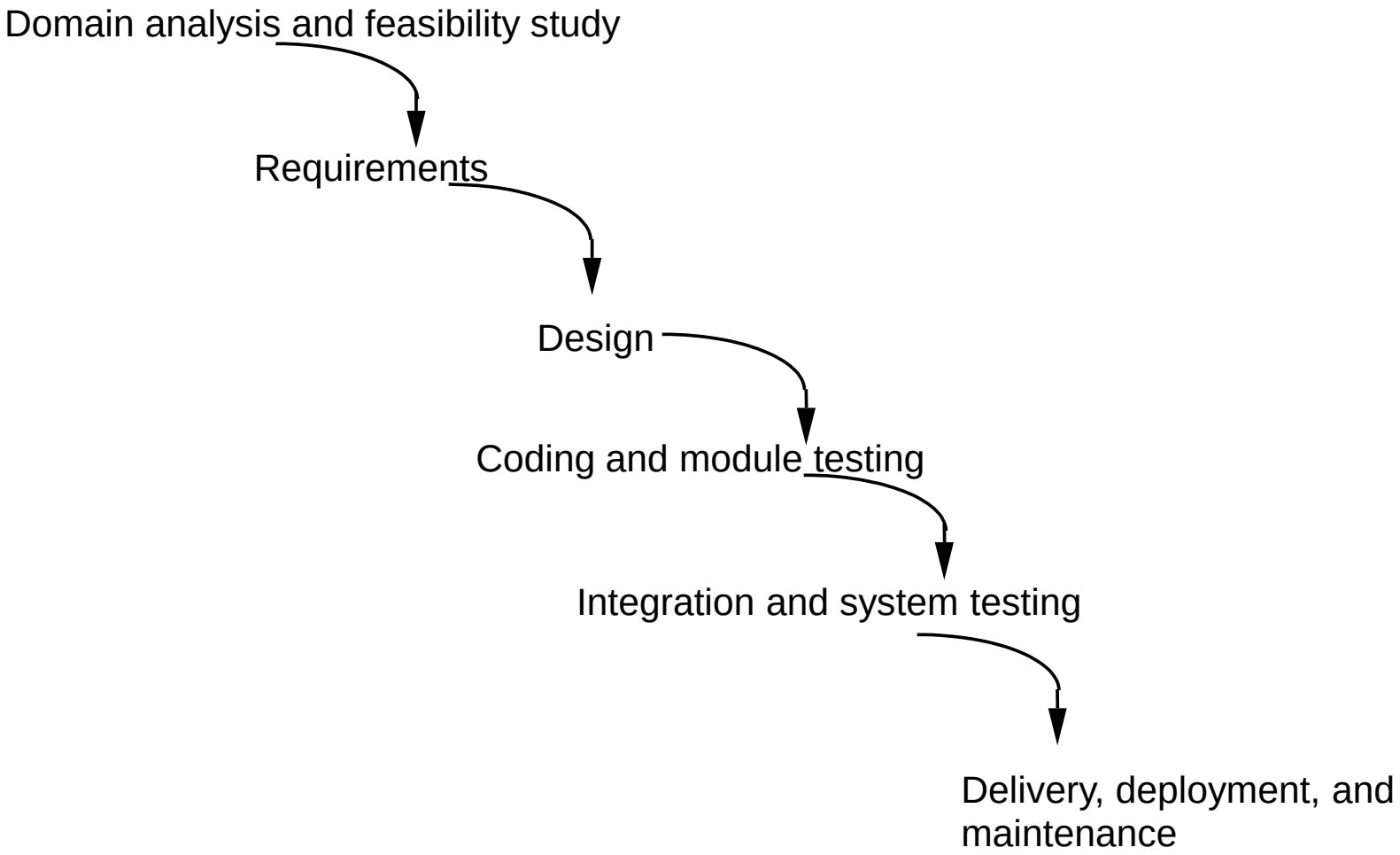
Invented in the late 1950s for large air defense systems, popularized in the 1970s

They organize activities in a sequential flow

- Standardize the outputs of the various activities (*deliverables*)

Exist in many variants, all sharing sequential flow style

A WATERFALL MODELS



WATERFALL STRENGTHS

Easy to understand, easy to use

Provides structure to inexperienced staff

Milestones are well understood

Sets requirements stability

WATERFALL WEAKNESSES

uOttawa
L'Université canadienne
Canada's university

All requirements must be known upfront

Deliverables created for each phase are considered frozen – inhibits flexibility

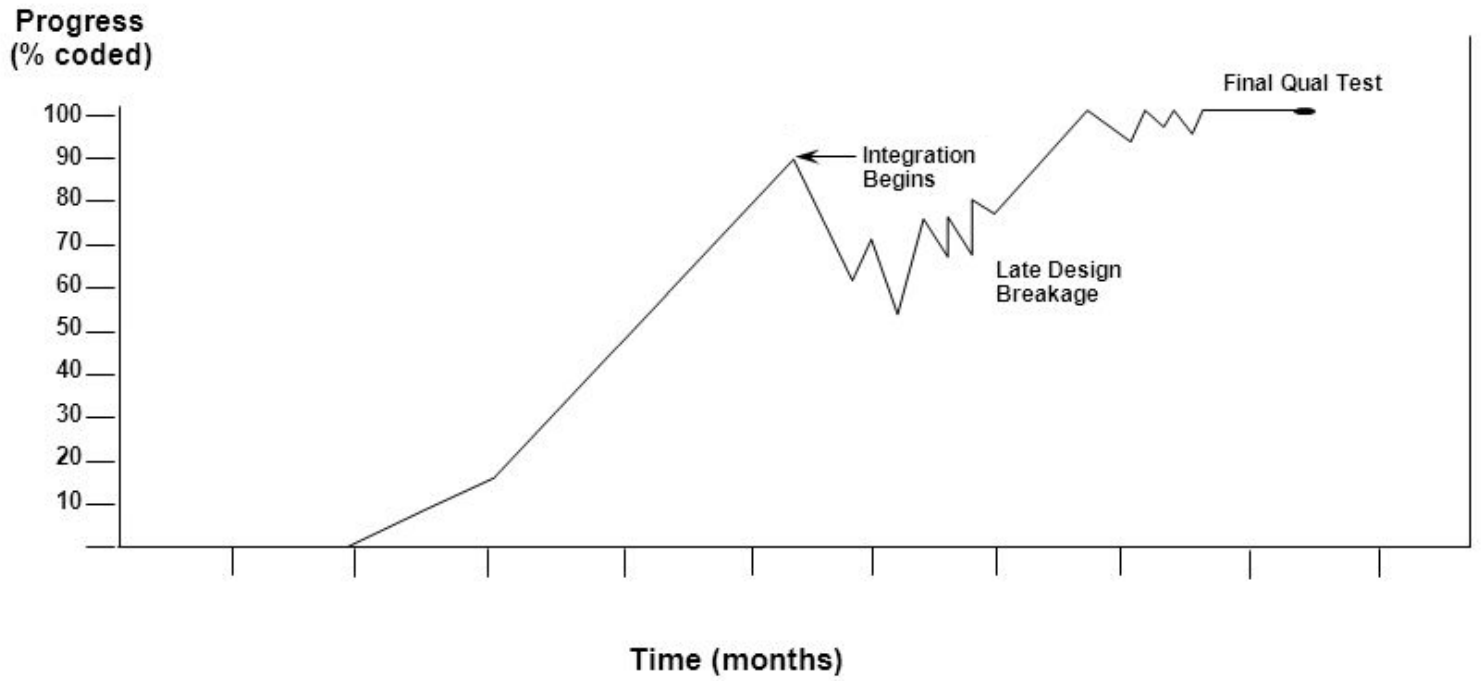
Can give a false impression of progress

Does not reflect problem-solving nature of software development – iterations of phases

Integration is one *big bang* at the end

Little opportunity for customer to preview the system (until it may be too late)

LATE DESIGN BREAKAGE



WHEN TO USE WATERFALL

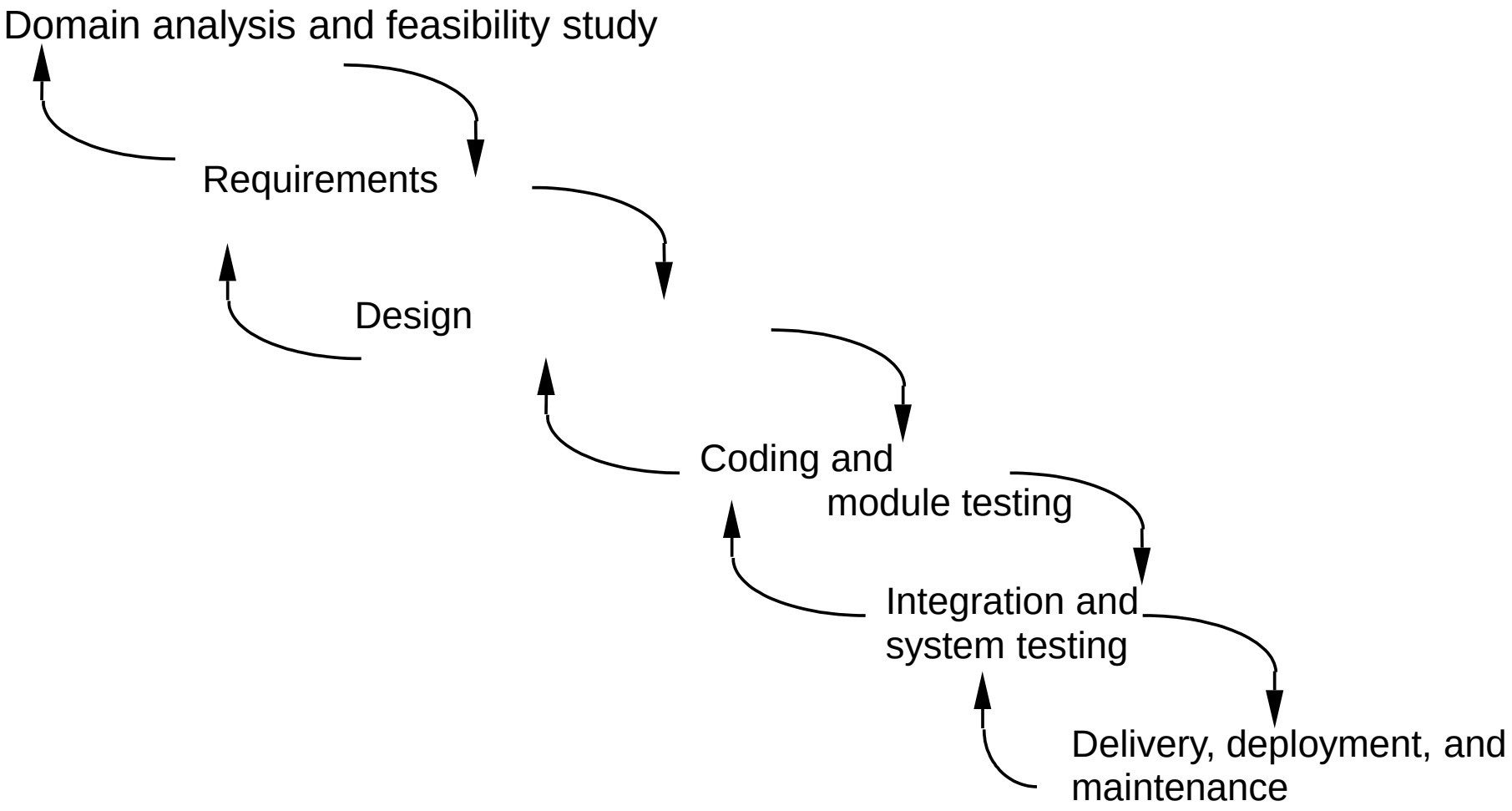
uOttawa
L'Université canadienne
Canada's university

Today, almost never!!

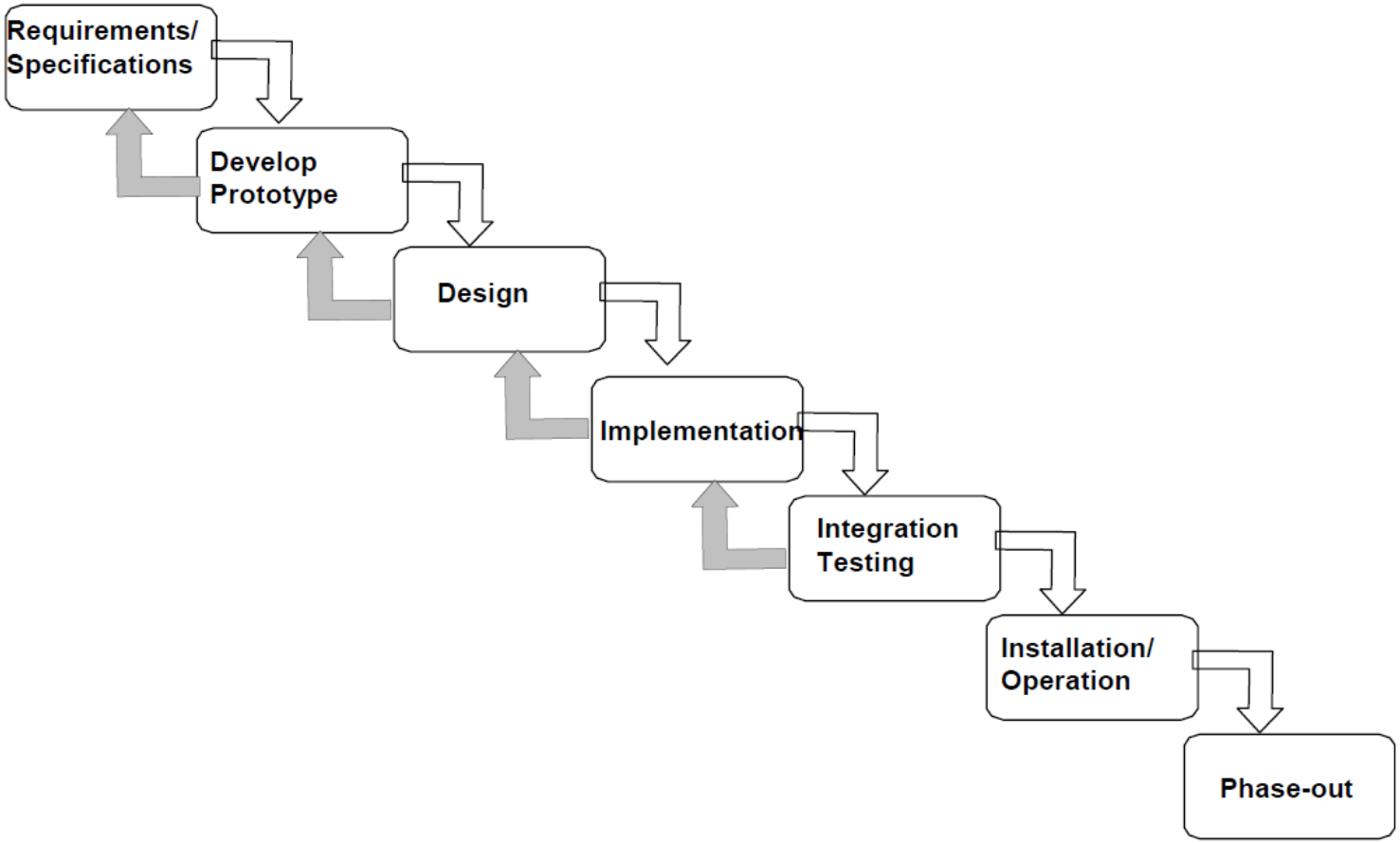
However, rarely when:

- Requirements are very well known
- Product definition is stable
- Technology is *very well* understood
- New version of an existing product *(maybe!)*
- Porting an existing product to a new platform

WATERFALL – WITH FEEDBACK



RAPID PROTOTYPING



ITERATIVE DEVELOPMENT PROCESS

uOttawa
L'Université canadienne
Canada's university

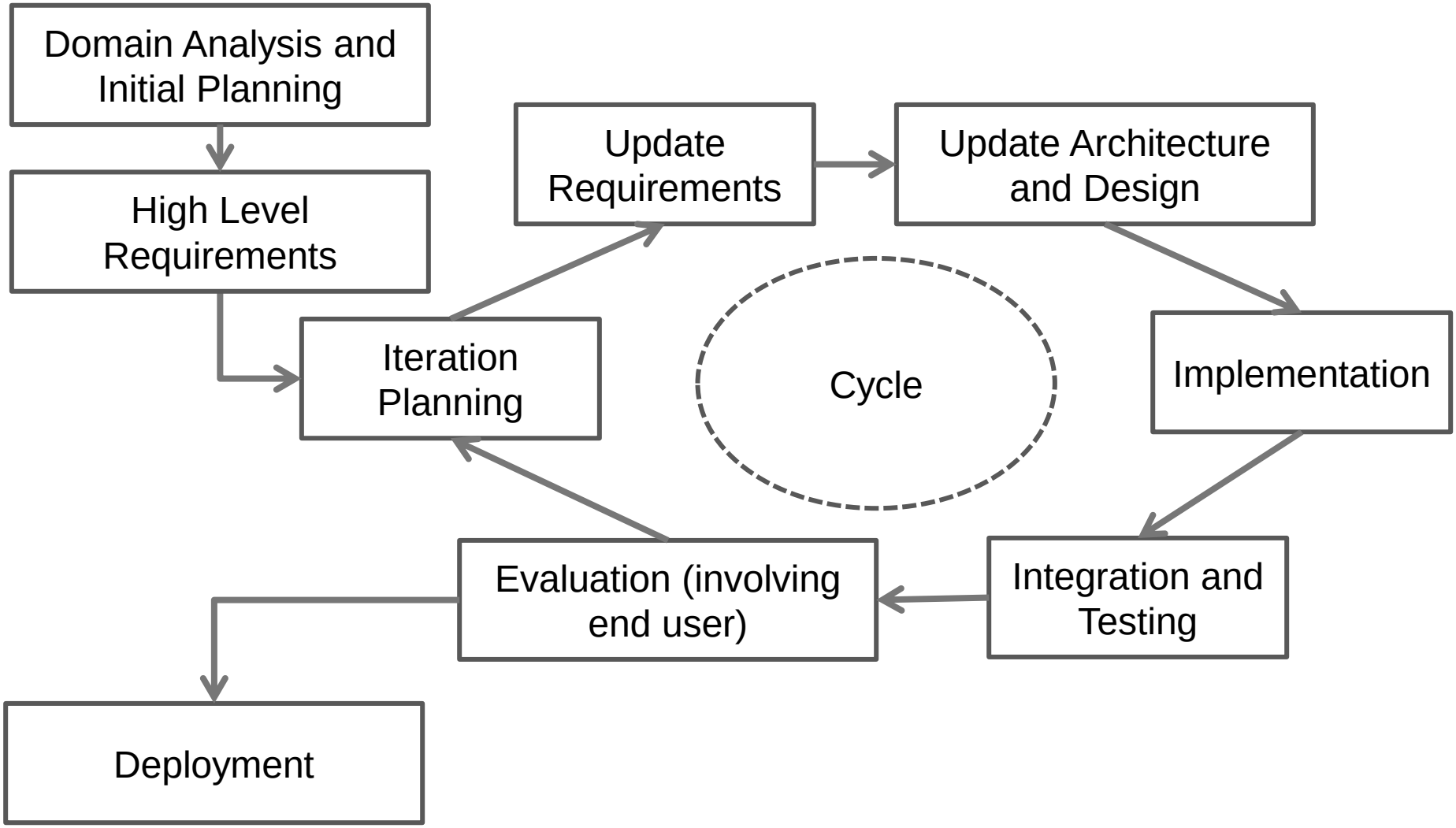
Develop system through repeated cycle (iterations)

- Each cycle is responsible for the development of a small portion of the solution (slice of functionality)

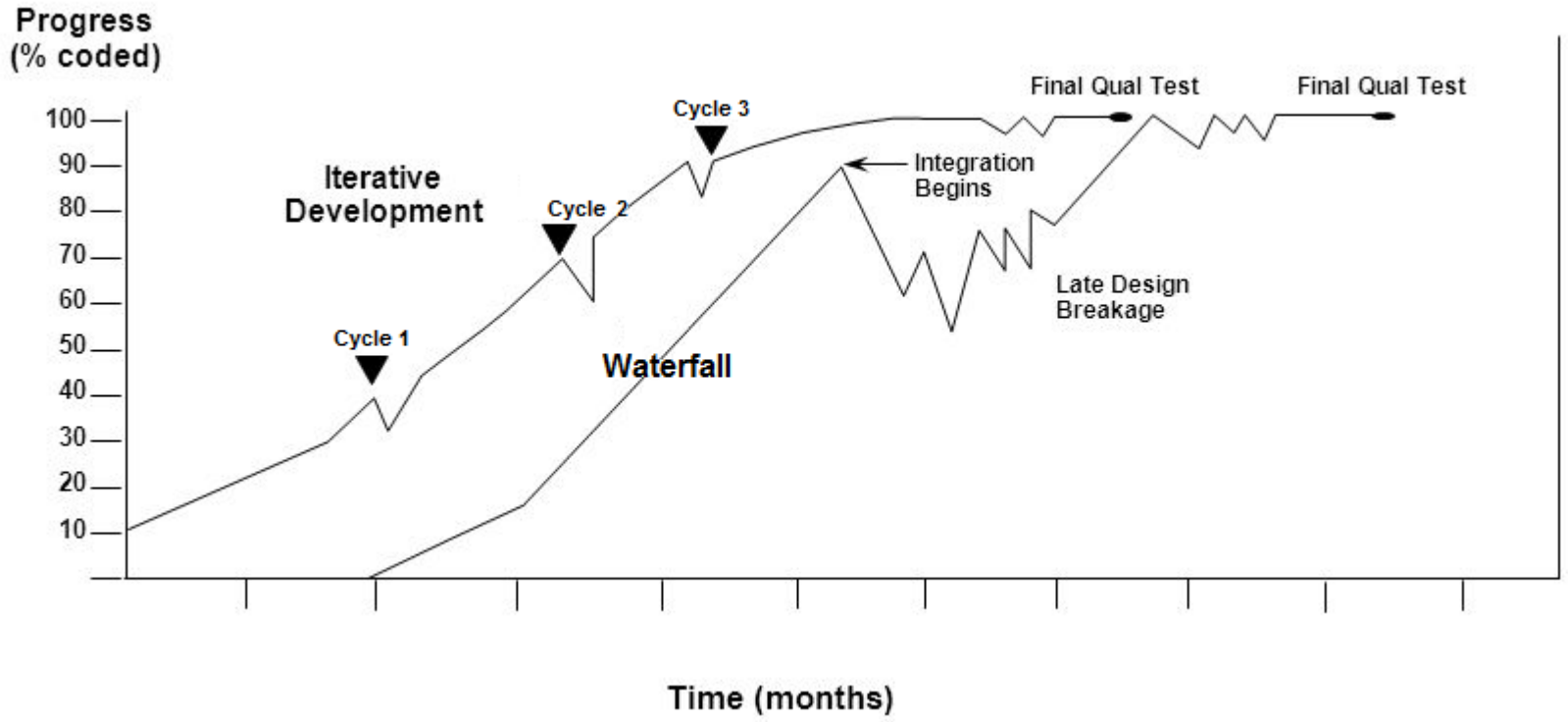
Contrast with waterfall:

- Water fall is a special iterative process with only one cycle

ITERATIVE DEVELOPMENT PROCESS



CONTINUOUS INTEGRATION



AGILE METHODS

Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods

These methods:

- Focus on the code rather than the design
- Are based on an iterative approach to software development
- Are intended to deliver working software quickly

The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework

AGILE MANIFESTO

We are uncovering better ways of developing software by doing it and helping others do it

Through this work we have come to value:

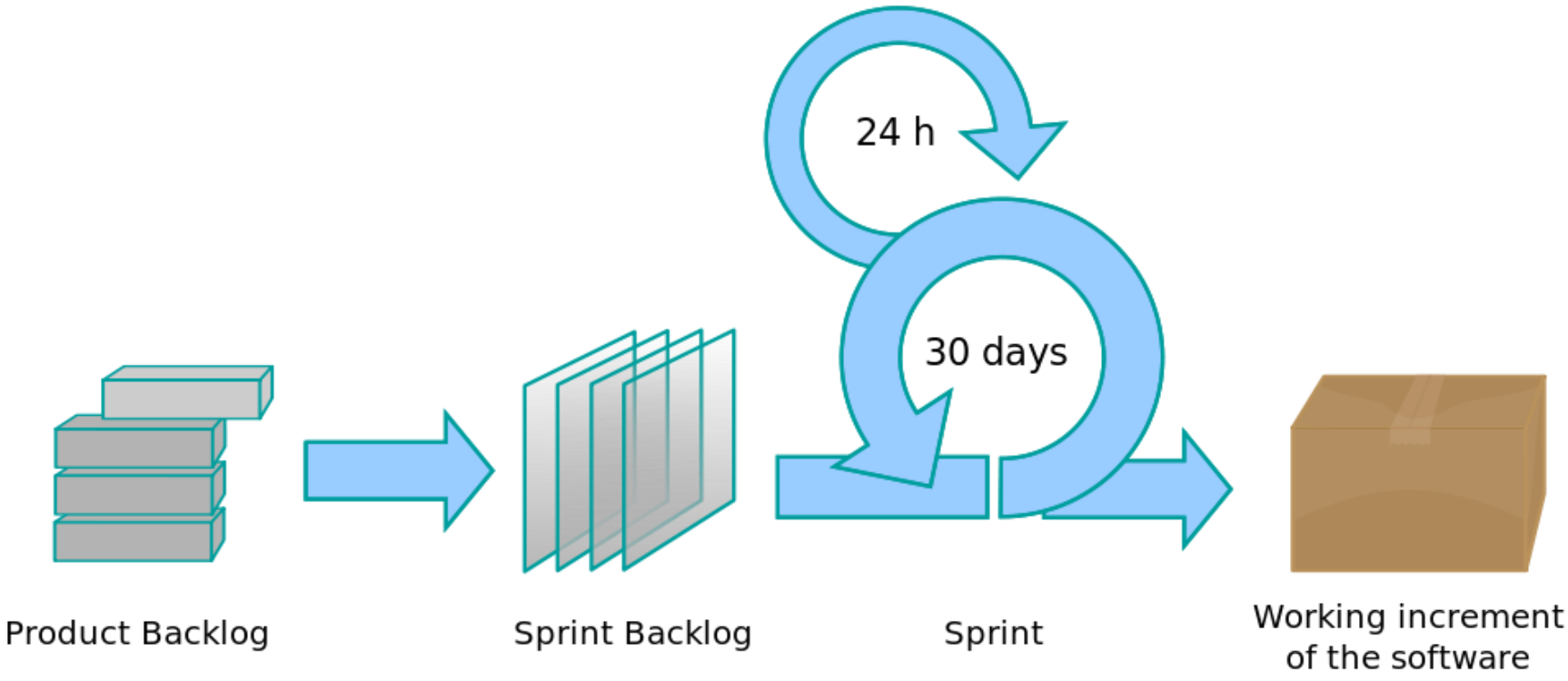
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
- Maintain simplicity

THE PRINCIPLES OF AGILE METHODS

uOttawa
L'Université canadienne
Canada's university

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is to provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

SCRUM PROCESS



PROBLEMS WITH AGILE METHODS

It can be difficult to keep the interest of customers who are involved in the process

Team members may be unsuited to the intense involvement that characterizes agile methods

Prioritizing changes can be difficult where there are multiple stakeholders

Minimizing documentation: almost nothing is captured, the code is the only authority

THANK YOU!

QUESTIONS?