

The background is a vibrant blue with stylized white and yellow gears. A large magnifying glass with a black frame and a yellow handle is positioned on the right side, focusing on a dark brown bug with black legs and antennae. The bug is centered within the magnifying glass's lens. The text is overlaid on a dark grey rectangular area on the left side of the slide.

# CISC/CMPE 327 Software Quality Assurance

Queen's University, 2019–fall

Lecture #3

Software Process Models - 2

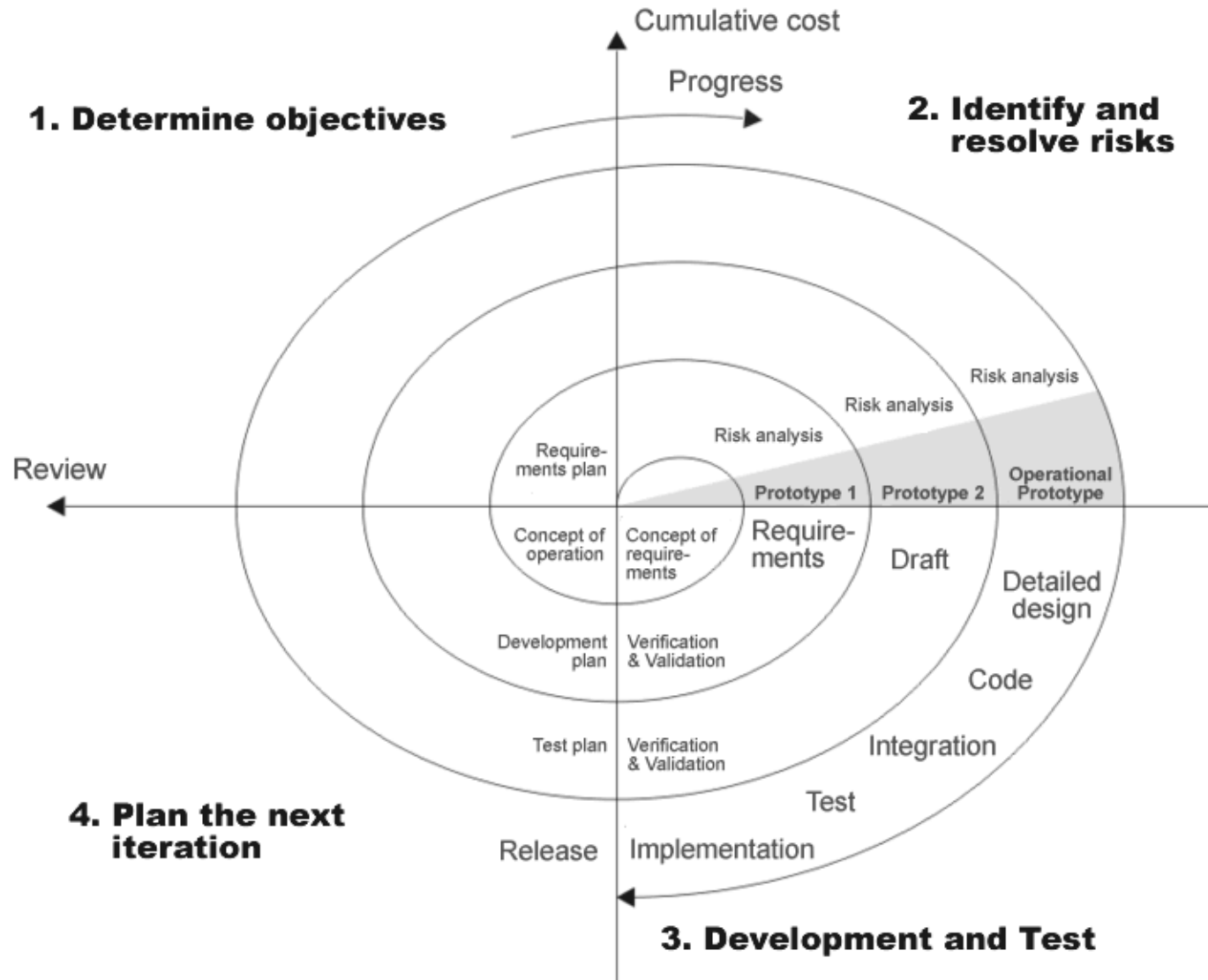
# Software Process Models

- More Process Models
  - We continue today with more **models** of software process
  - Recall that our objective is to understand how **quality control** fits into software processes

# The Spiral Model

- Boehm's Spiral Model
  - The **Spiral Model** is a refinement of the waterfall model designed around continuous **documentation** and evaluation of **risk**
  - Based on **experience** applying the waterfall model to large (U.S.) government software projects
  - Now a **standard** used by many government agencies and software providers

# The Spiral Model



# The Spiral Model

- Spiral Layers

- Roughly, each **layer** of the spiral corresponds to one **phase** of the waterfall (although there are **no fixed phases**)
- For example, the first layer could be the **requirements** phase, second layer the **design** phase, etc.

# The Spiral Model

- **Four Step Cycle**

- In each layer, the same four step cycle is used, consisting of:

- **Determine Objectives**: determine objectives, constraints, risks for next phase
    - **Assess and Reduce Risks**: analyze and reduce identified risks
    - **Develop and Validate**: choose development model, develop and test
    - **Review and Plan**: review status, plan next layer

# The Spiral Model

For each **layer** (phase) of the project:

- **1) Determine Objectives**
  - Specific **objectives** (aims) for the phase of the project are defined
  - **Constraints** on the process and product are identified
  - **Alternatives** for achieving the objectives are identified
  - Potential **risks** associated with each alternative are identified

# The Spiral Model

- 2) Assess and Reduce Risks
  - For each potential risk, a detailed **analysis** is carried out
  - Steps are taken to **reduce risk** (e.g., create prototype to check)
  - Alternatives are chosen to **minimize** risk



# The Spiral Model

- 3) Develop and Validate
  - Based on risk analysis, choose or modify development model
  - For example, to implement and validate,
    - if user interface risks dominate, use evolutionary prototyping
    - if safety risks are the major issue, use formal methods
    - if integration problems are the big risk, use waterfall model

# The Spiral Model

- 4) Review and Plan
  - Review and evaluate results of this phase (layer)
  - Decide whether another layer of the spiral is needed
  - If so, draw up plans for next phase

# Drawbacks of the Spiral Model

- **Heavyweight Process**
  - The spiral model requires a large amount of **overhead**
    - Every layer requires a lot of documentation and many meetings, progress can therefore be **slow**
- **Not really a development model**
  - The spiral model is really more of a "**meta-model**" since it describes the way to carry out stages, not what the stages are
  - But focuses on identifying **potential** problems **early** at every stage, so very good at producing high quality results

# Drawbacks of the Spiral Model

- Depends on Risk Analysis
  - Needs a very experienced team to recognize and analyze risks accurately
  - High dependency on quality of people (itself a risk!)
- Not for Novices
  - Layers of process are flexible and not explicitly laid out
  - Each layer's goals and plan must be decided by team itself, requires experienced people!

# The Iterative Development Model

- Subset Development
  - The Iterative Development Process (IDP) is based on subsets
  - Begin with a subset of the requirements and develop a subset of the software product
  - The subset should:
    - satisfy immediate needs of users
    - serve as a vehicle of training for customers, and learning for developers

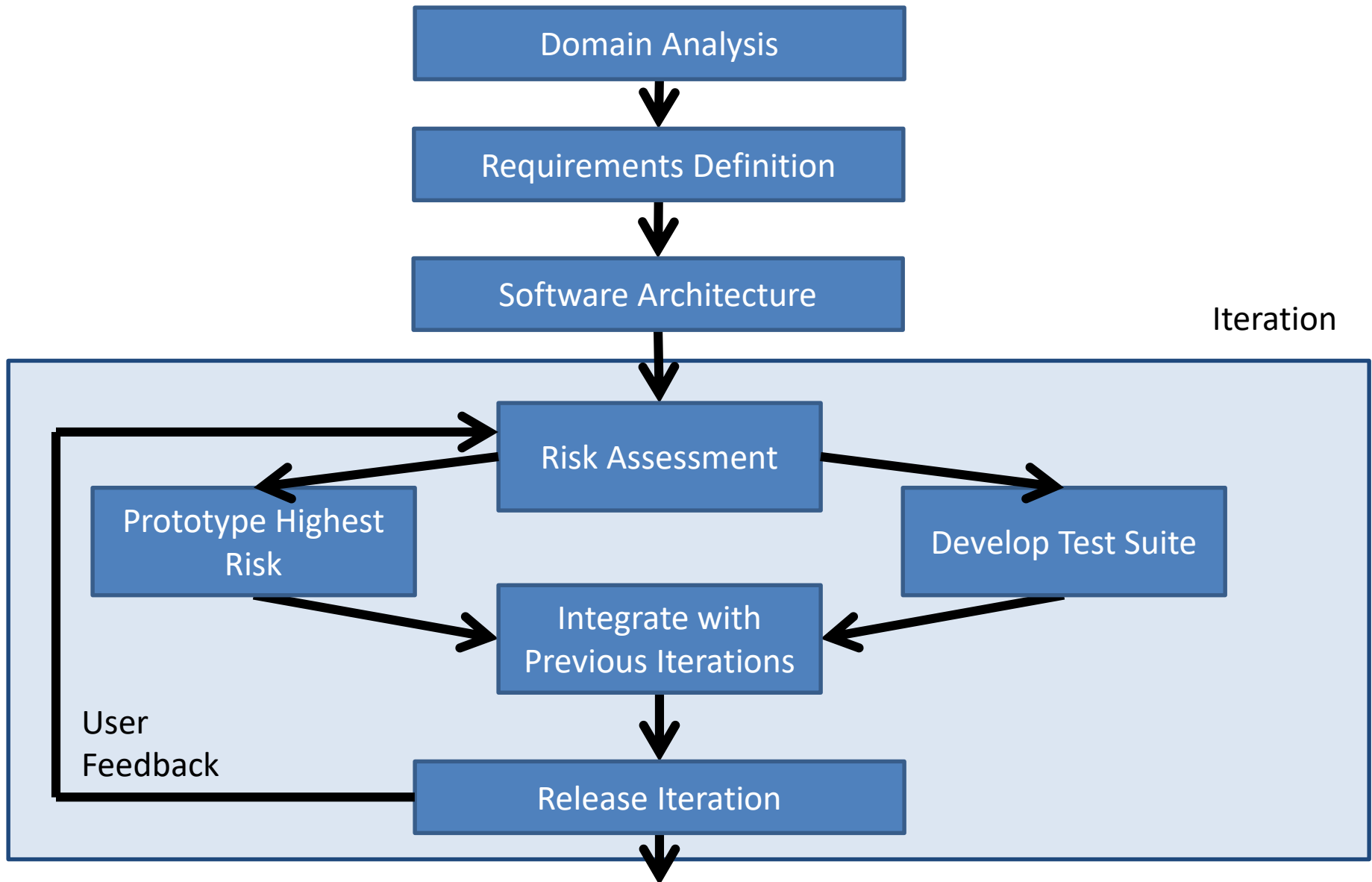
# The Iterative Development Model

- Sequence of Intermediate Products
  - Analysis of the subset product leads to modifications to the design and requirements, from which we build a new (hopefully larger) subset product
  - Design and requirements refined over a series of iterations to provide a system that meets evolving customer needs with improved design based on feedback and learning

# The Iterative Development Process

- Iterative Development Process
  - Analysis of the problem domain and definition of requirements begins process as usual
  - Need initial architecture design to begin
  - Add most critical remaining features each cycle
  - Quality control, development of test suite for new features on each iteration

# The Iterative Development Process





# Drawbacks of the Iterative Process

- Needs Small Team

- Process does not allow for large scale parallel development, depends on focussing on **one** remaining risk at a time
- Works best with relatively **small** teams

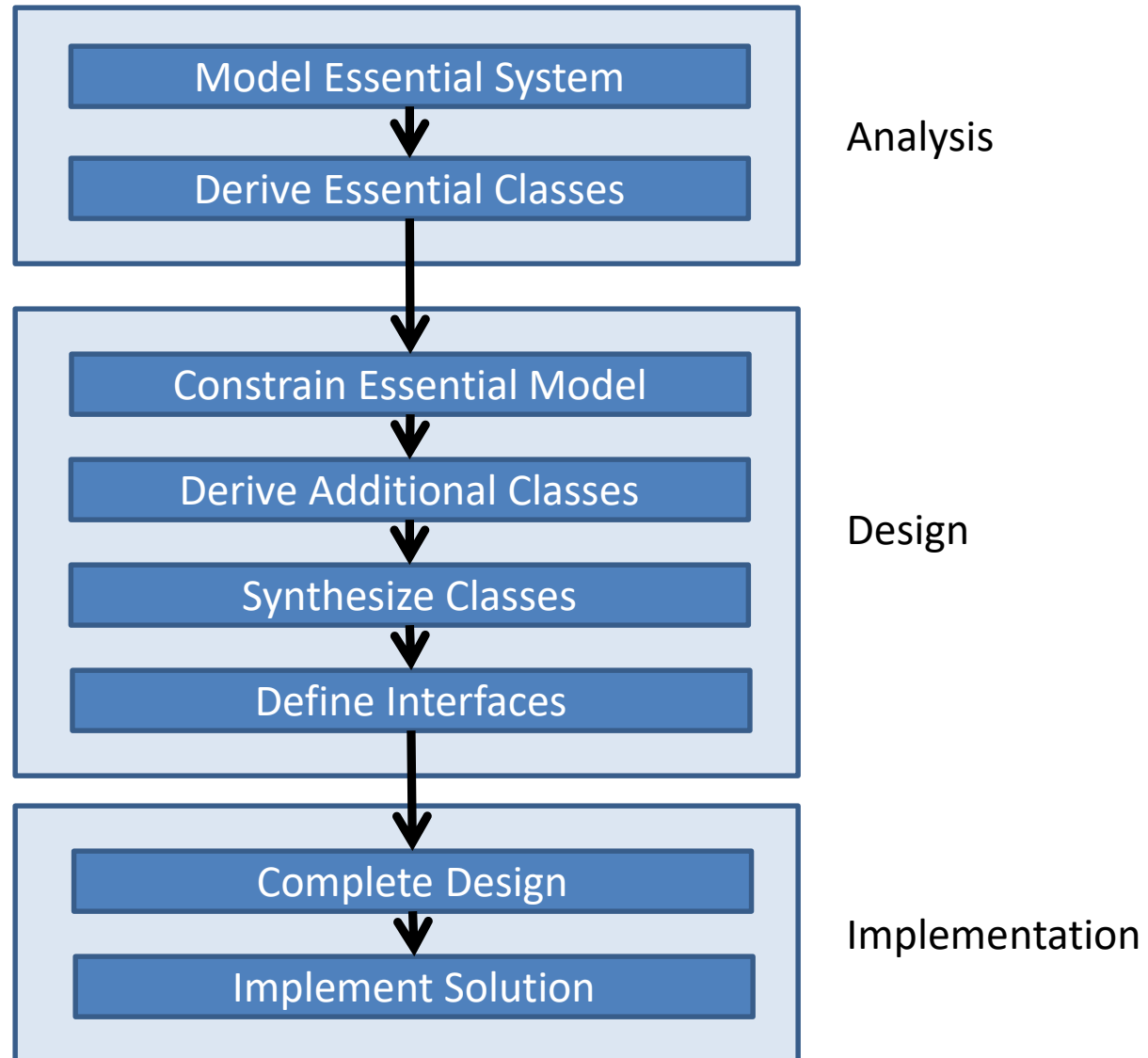
- Needs Early Architecture

- Requires early design of overall **architecture**, difficult to change later
- But when architecture can be settled early, has been very successful at producing significant, very **high quality** products, e.g., IBM's **OS/2** system

# Object-Oriented Development

- OO Process Models
  - Many OO process models are proposed, based on OOAD ("Object-Oriented Analysis and Design")
  - All have three major phases:
    - Analysis: model the "essential system" to represent user requirements, and design implementation-independent "essential classes"
    - Design: constrain and refine essential classes to be implemented on particular implementation environment, derive additional classes
    - Implementation: define class interfaces and implementation methods, then code and unit test all classes

# Object-Oriented Development



# OO Development Process - Analysis

- 1) Model the Essential System
  - Create a "user view" of the system
  - Model essential activities, essential solution data, and how they are related
  - Quality control: requirements reviews (inspection)

# OO Development Process - Analysis

- 2) Derive Essential Classes
  - "Carve" out candidate **essential classes** from the essential model using data-flow diagrams, process, and data specifications
  - Quality control: **design reviews** (inspection)

# OO Development Process - Design

- 3) Constrain the Essential Model
  - Modify essential model to fit within constraints of target implementation environment
  - Map essential activities and data to implementation processors (hardware/software) and containers (memory/files)

# OO Development Process - Design

- 4) Derive Additional Classes
  - Additional **classes** and **methods** specific to implementation environment added to support additional activities added while constraining the essential model

# OO Development Process - Design

- 5) Synthesize Classes
  - Essential classes and additional classes **refined** and **organized** into a class hierarchy
  - Final classes chosen to maximize **reuse**
  - Quality control: **design review** (inspection)
- 6) Define Interfaces
  - Class definitions written for final classes



# OO Development - Implementation

- 7) Complete Design

- Design of "implementation module" completed
- Implementation module specifies methods such that each provides a single cohesive function
- Quality control: design review

- 8) Implement Solution

- Implementation of classes and methods is coded and validated
- Quality control: unit testing (class-wise)

# Drawbacks of the OO Process

- **Delayed Testing**
  - Development process missing intermediate results
  - Most testing **delayed** to final implementation stage
- **Architectural Inflexibility**
  - Process assumes that overall architecture can be designed in the requirements phase
  - Allows little **architectural flexibility** in design and implementation steps
  - Can lead to spaghetti results

# Summary

- **Software Process**
  - **Spiral Model** organizes and generalizes the waterfall model
  - **Iterative Development Process** is based on product subsets
  - **Object-Oriented Development** is (was?) a currently popular model with drawbacks

# Review Questions

- Waterfall model
  - Drawbacks?
  - When would it be appropriate?
- Prototyping model
  - Drawbacks?
  - Advantages?

# Review Questions

- Spiral model
  - Strengths?
  - Drawbacks?
- Iterative development
  - Strengths?
  - Drawbacks?
- Object-oriented development
  - Strengths?
  - Drawbacks?

# Summary

- Today's References
  - Sommerville, *Software Engineering*, ch. 2
  - Kan, *Metrics and Models in Software Quality Engineering*, ch. 2
- Next Time
  - Quality standards and assessment of software processes
- Then...
  - The *eXtreme Programming* software process