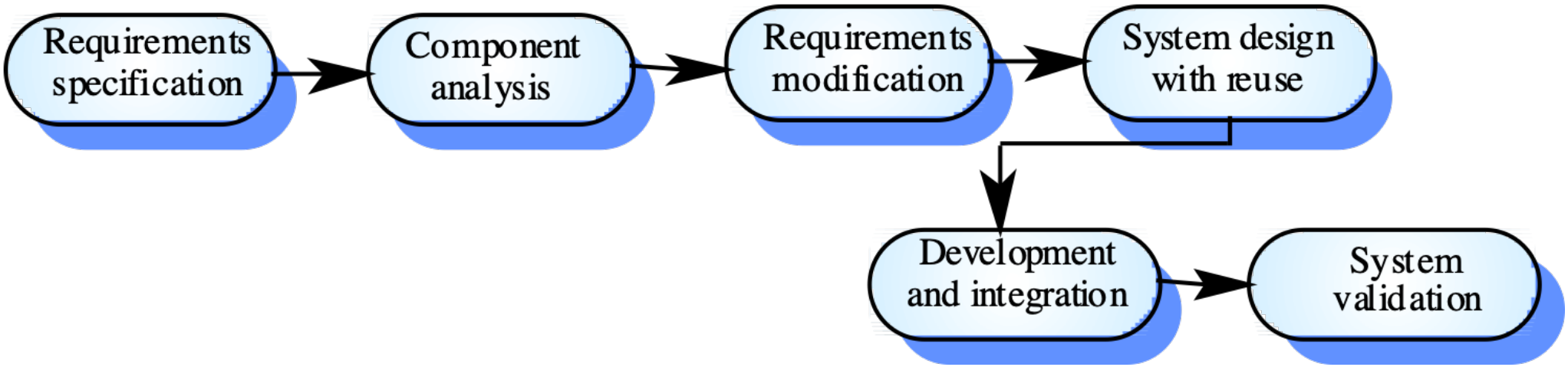
4. Reuse-oriented development

- **Based on systematic reuse** where systems are integrated from existing components or COTS (Commercial-off-the-shelf) or (Component-off-the-shelf) systems

- **Process stages**
  - Component analysis
  - Requirements modification
  - System design with reuse
  - Development and integration

This approach is becoming more important and popular but we still have limited experience with its wide use across different domains.

Reuse-oriented development



Reuse-oriented development

- **Problems**
  - Need for **specialised** (component) analysis and integration skills to ensure appropriate selection of components, for both functionality and quality aspects.
  - Some aspects (or parts) of the system may not be easily reused, such as the user interface
  - Concerns over **maintainability** and support of reused components
  - Concerns over system **evolution** that development is controlled by reused component suppliers.

- **Applicability**
  - **Not critical systems**, that may include common functionality (reusable) components
  - **Large systems**! (components analysis and integration may be too expensive for small and mid-size systems)

Process Iteration

How to develop (or deliver) software?

Modern development processes develop software in iterations (cycles), opposed to one single monolithic cycle.

Process iteration

- **Modern development processes take iteration as fundamental,** and try to provide ways of managing, rather than ignoring, the risk
- **System requirements ALWAYS evolve in the course of a project** so process iteration where earlier stages are reworked is always part of the process for large systems
- **Iteration** can be applied to any of the generic process models
- **Two (related) approaches**

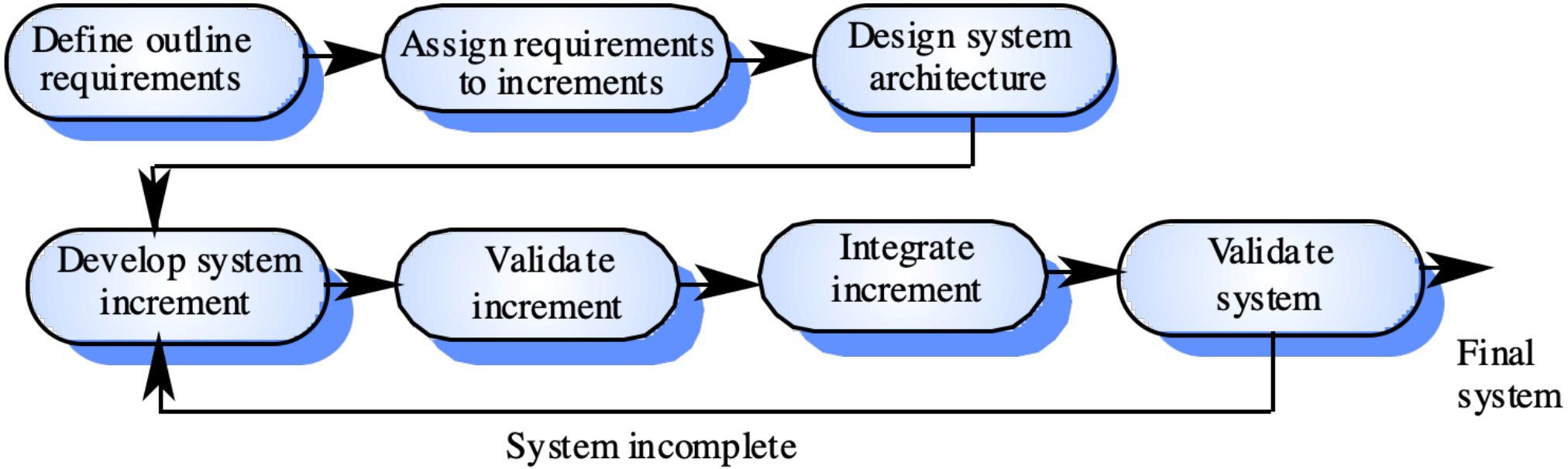
  i. Incremental development

  ii. Spiral development

Incremental development

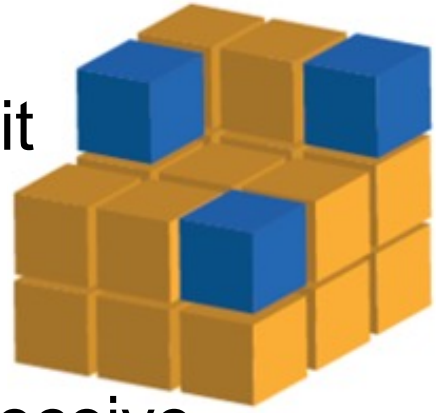
- Rather than deliver the system as a single delivery, **the development and delivery are broken down into increments** with each increment delivering part of the required functionality
- **User requirements are prioritised** and the highest priority requirements are included in early increments
- **Once the development of an increment has started, the requirements are frozen** though requirements for later increments can continue to evolve

i. Incremental development



Incremental development advantages

- **Customer value** can be delivered with each increment so system functionality is available earlier (earlier return on investment)
- **Early increments** act as a prototype to help elicit requirements for later increments
- Has Lower risk of overall project failure
- **The highest priority system services** tend to receive the most testing
- Typical examples of incremental development models:
  - XP
  - Scrum



Extreme programming-XP (Agile)

Developed by
Kent Beck
(published 1999)

- **Incremental approach** to development based on the development and delivery of **very small increments** of functionality (often no longer than two weeks)
- **Relies on constant code improvement**, user involvement in the development team and pairwise programming
- **Design** of the test plan/suites first ! Then you perform **testing** of the system after each small increment

Extreme Programming-XP

- **Work in Pairs**: a Coder and a Reviewer

- **XP practices**: Simple design, test-driven development, refactoring, code convention, strict releases.