CS 430 - Examples

Collin Roberts

September 21, 2023

Example: The Classical (Waterfall) Life-Cycle

Goals:

1. Describe the state of software development practice before the introduction of the Classical Life-Cycle Model.
2. Describe the implications of the introduction of the Classical Life-Cycle Model.

Example: The Classical (Waterfall) Life-Cycle

1. The year is 1970.
2. You are working as a software developer.
3. All software development work is done on an ad-hoc basis.
4. Project management practice is non-existent.
5. Not surprisingly, the software crisis is in full bloom.

Example: The Classical (Waterfall) Life-Cycle

1. Your manager calls a team meeting to describe the Waterfall Life-Cycle model which she has just adopted for your team to use going forward.
2. The Waterfall Life-Cycle model consists of the **phases** listed on the following slides.
3. The Waterfall Life-Cycle model is so called because each phase must be completed before the next one can start, leading to a waterfall-shaped diagram in which each phase cascades into the next.

Example: The Classical (Waterfall) Life-Cycle

1. Requirements
   1. Elicit client requirements.
   2. Understand client needs.
2. Analysis
   1. Analyze client requirements.
   2. Draft specification document - formal.
   3. Draft Software Project Management Plan (SPMP).
3. Design
   1. Design architecture - divide S/W functionality into components.
   2. Draft detailed design for each component.

Example: The Classical (Waterfall) Life-Cycle

4. Implementation
    1. Coding (development)
    2. Unit testing
    3. Integration (system) testing
    4. Acceptance testing
    5. Deploy to production environment.
5. Post delivery maintenance.
    1. Corrective maintenance
    2. Perfective maintenance
    3. Adaptive maintenance
6. Retirement
    1. Product is removed from service: functionality provided by S/W is no longer useful / further maintenance is no longer economically feasible

Example: The Classical (Waterfall) Life-Cycle

Questions for Discussion:

1. How would each group below react to the introduction of the Waterfall life-cycle model?
    1. your IT team
    2. your business partners
2. Why does the Waterfall life-cycle model not have any of the following phases?
    1. Planning
    2. Testing
    3. Documentation

Example: The Object-Oriented Paradigm

Goals:

1. Describe the implications of the introduction of the Object-Oriented Paradigm.

Example: The Object-Oriented Paradigm

1. It is now the late 1980s.
2. All software development work is done using the Waterfall model, which has improved the situation compared to pre-1970.
3. However as systems become larger and more complex, the Waterfall model scales less and less effectively. It forces developers to focus either on data or on operations on the data, not both equally.
4. The costs of post-delivery maintenance also continue to grow despite everyone's best efforts.

Example: The Object-Oriented Paradigm

1. Your manager calls a team meeting to describe the Object-Oriented paradigm which she has just adopted for your team to use going forward.
2. Under the Object-Oriented paradigm,
    1. data and operations on that data are treated with equal importance,
    2. **encapsulation** means bundling data together with operations on that data to form a module, and
    3. **information hiding** means hiding the implementation details of a module from the outside world.

Example: The Object-Oriented Paradigm

Questions for Discussion:

1. How would each group react to the introduction of the Object-Oriented paradigm?
    1. your IT team
    2. your business partners
2. What would be some advantages and disadvantages of adopting the Object-Oriented paradigm?

Winburg Example

Goals:

1. Reduce traffic congestion in downtown Winburg, Indiana by creating a public transportation (bus) system.
2. Develop software for the scanners that examine the bill the passenger inserts into the fare collection machine.

Winburg Example

1. **Episode 1:** The first version of the software is deployed.
2. **Episode 2:** Tests show that the scanner software is too slow. The first version uses double precision numbers for its computations, which slows things down. A developer starts making the necessary changes.

Winburg Example

3. **Episode 3:** Before the developer can finish, further tests show that this change still will not achieve the desired speed. A new, faster scanning algorithm is needed and is implemented, achieving the desired speed.

Winburg Example

4. **Episode 4:** The project is now well behind schedule and over budget. The mayor asks the team to improve the scanning portion, to facilitate selling this piece to vending machine manufacturers. Accuracy is improved to 99.5%, and this version is installed in the fare machines. S/W development is now complete. Selling the scanning portion defrays about $\frac{1}{3}$ of the cost overrun.

Winburg Example

5. **Epilogue:** When the sensors wear out and have to be replaced, new software is required. It was decided to write the new software in a new programming language. When the text was written, this new project was 6 months behind schedule and 25% over budget.

Winburg Example

Questions for Discussion:

1. What observations / reactions do you have about this case study?
2. In particular, was this case study similar to or different from your own experiences working in IT?

Example: Problems with Requirements

1. **Fact:** Your instructor is a CS Academic Advisor for the BBA/BCS Double Degree Program with Wilfrid Laurier University.
2. Consider the following draft requirements for a report your instructor might run for determining **Academic Progression at the end of the S23 term**.
3. Document all the problems that you can find with these draft requirements.

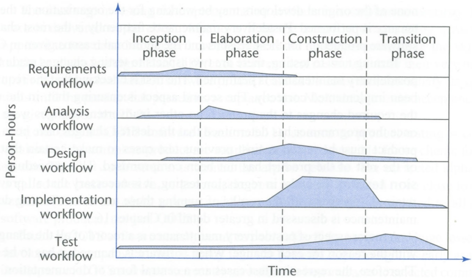
Example: Problems with Requirements

1. Criteria for Student Inclusion
    1. MAV < 60
    2. Standing has changed since the end of the S23 term
2. Student Data to Include
    1. Student Number (N.B. sorted ascending)
    2. LoginID
    3. Name
    4. Current Standing
    5. CAV
3. Report Organization and Formatting
    1. Students on the rows
    2. Data items on the columns
    3. Sort Students by LoginID, ascending

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. For each of the following artifacts,
   1. State to **which workflow** it belongs, and
   2. State during **which phase** it will most likely be delivered.

FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Business Model: First Draft

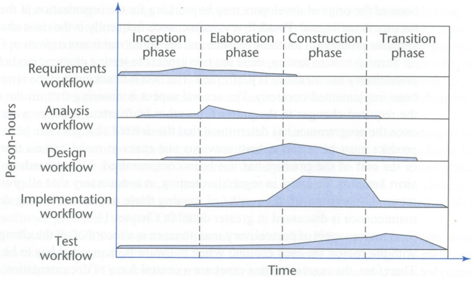
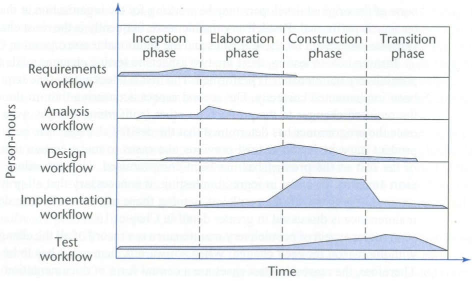


FIGURE 3.1 The core workflows and the phases of the Unified Process.

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Business Model: Completed

FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. User Requirements: First Draft

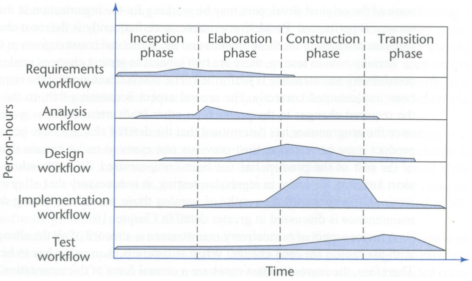
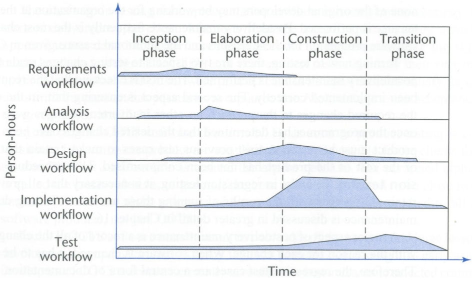


FIGURE 3.1 The core workflows and the phases of the Unified Process.

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. User Requirements: Completed

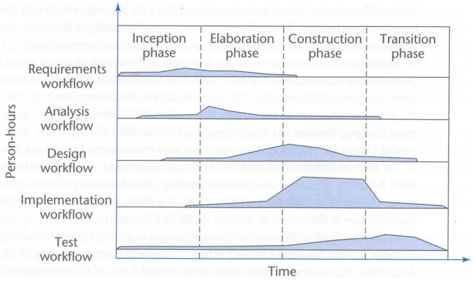
FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Inspection Report: User Requirements

FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. SPMP: First Draft

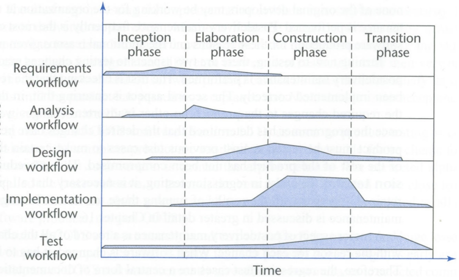


FIGURE 3.1 The core workflows and the phases of the Unified Process.

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. SPMP: Completed

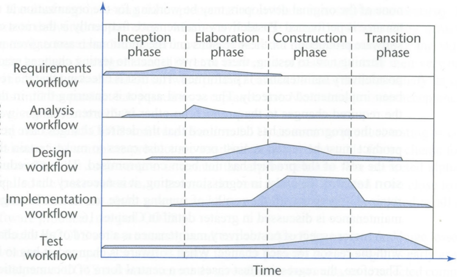
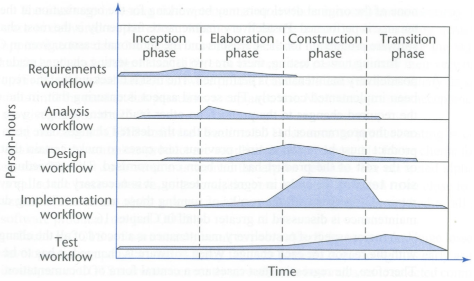


FIGURE 3.1 The core workflows and the phases of the Unified Process.

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. System Architecture Design: First Draft

FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. System Architecture Design: Completed

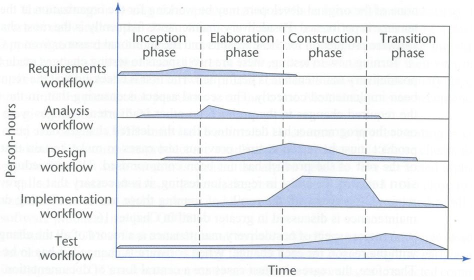


FIGURE 3.1 The core workflows and the phases of the Unified Process.

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Code: First Draft

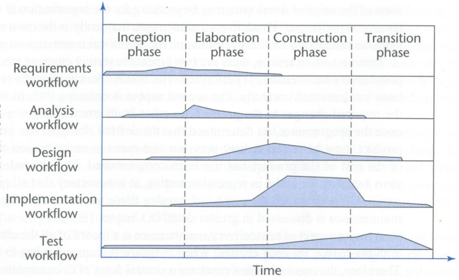
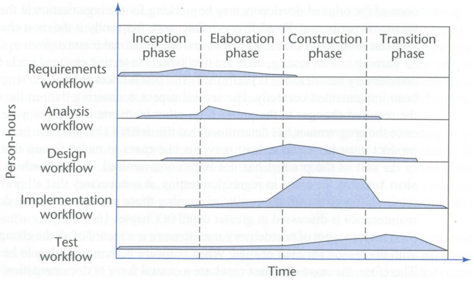


FIGURE 3.1 The core workflows and the phases of the Unified Process.

Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Code: Ready for User Acceptance Testing

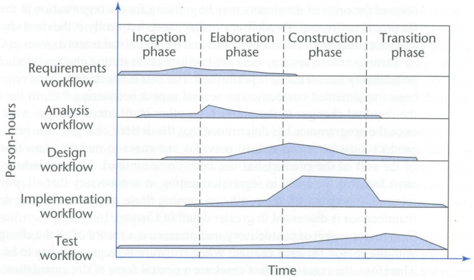
FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Code Test Report: First Draft

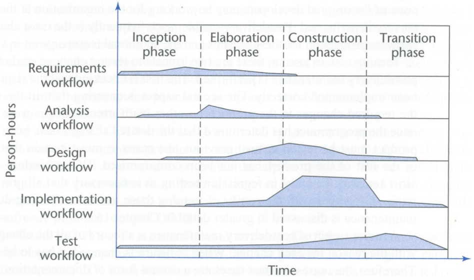
FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Code Test Report: Completed

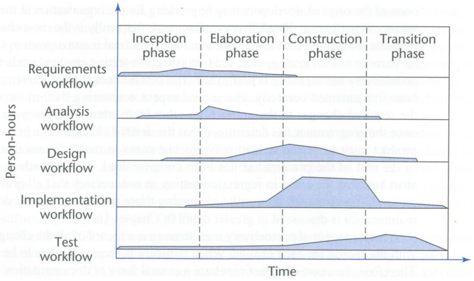
FIGURE 3.1
The core workflows and the phases of the Unified Process.



Example: The Interaction Between the Workflows and the Phases of the Unified Process

1. Code: Ready for Deployment

FIGURE 3.1
The core workflows and the phases of the Unified Process.



Correctness Proving Mini Example

- **Program:** solve a **line-editing** problem, i.e. a **word-processing** problem.
- Format text according to these rules:
  - Line breaks must be made only when the given text contains a `blank` or `newline`.
  - Each line must be filled as far as possible, provided
  - No line will contain more than `maxpos` characters.

Naur, 1969

- The first program consisted of about 25 lines of code.
- Naur presented an informal proof of program correctness.

Leavenworth, 1970

- The first word of the first line is preceded by a blank unless the first word is exactly `maxpos` characters long.
- This (minor) fault could have been detected easily by basic testing.

London, 1971

- Three additional faults:
    - One was that the procedure does not terminate unless a word longer than `maxpos` characters is encountered.
- These faults could also have been detected by testing.
- London presented a corrected procedure with a formal proof of its correctness.
- The original correctness proof had been informal.

Goodenough / Gerhart, 1975

- Three additional faults:
  - One was that the last word is not output unless it is followed by a `blank` or `newline`.
- These faults could also have been detected by testing.

Key Facts from the Example:

1. Proofs of correctness can vary in their level of rigour. E.g. in the case study, the first proof was informal, and some obvious faults were missed.
2. Even formal proofs of correctness depend on choices of assertions (including invariants) made by the prover. So it is possible to have a correct proof of the wrong assertions. Often this situation could be detected easily by basic testing.

Key Facts from the Example:

3. Proving the correctness of even a short program can be extremely tedious. Hence it can be prone to errors. An erroneous correctness proof provides us nothing useful.

Cohesion and Coupling Example

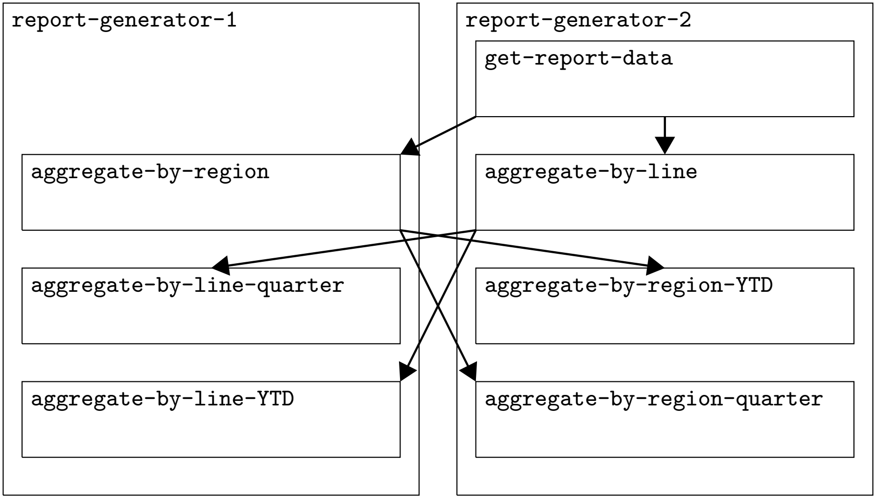
On the following slides you will be give two solutions, each of which computes data for a quarterly sales report. The data points for the report are

1. total sales by geographic region
    1. for the quarter, and
    2. YTD
2. total sales by business line
    1. for the quarter, and
    2. YTD

Cohesion and Coupling Example

1. This solution uses two modules, with the given procedures. Assume that the functionality of each procedure is described by its name.
    1. `report-generator-1`
        1. `aggregate-by-region`
        2. `aggregate-by-line-quarter`
        3. `aggregate-by-line-YTD`
    2. `report-generator-2`
        1. `get-report-data`
        2. `aggregate-by-line`
        3. `aggregate-by-region-YTD`
        4. `aggregate-by-region-quarter`

Cohesion and Coupling Example

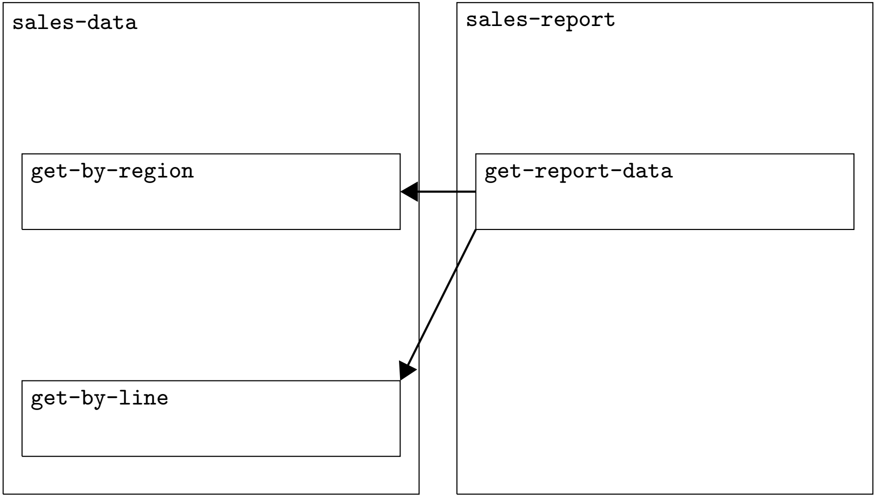


Cohesion and Coupling Example

2. This solution uses two classes, with the given methods. Assume that the functionality of each method is described by its name and accompanying comments.
    1. `sales-data`
        1. `get-by-region(type) // type=0 Q, type=1 YTD`
        2. `get-by-line(type) // type=0 Q, type=1 YTD`
    2. `sales-report(my-sales-data)`
        1. `get-report-data`

```
/* call my-sales-data.
* get-by-region(0) - by region Q
* get-by-region(1) - by region YTD
* get-by-line(0) - by line Q
* get-by-line(1) - by line YTD
*/
```

Cohesion and Coupling Example



Cohesion and Coupling Example

Questions For Each Solution:

1. Do the modules in the solution have high or low cohesion?
2. Does the pair of modules in the solution have loose or tight coupling?

Cohesion and Coupling Example

Answers:

Solution	Cohesion of Modules	Coupling of Pairs of Modules
1	low	tight
2	high	loose