University of
South Australia

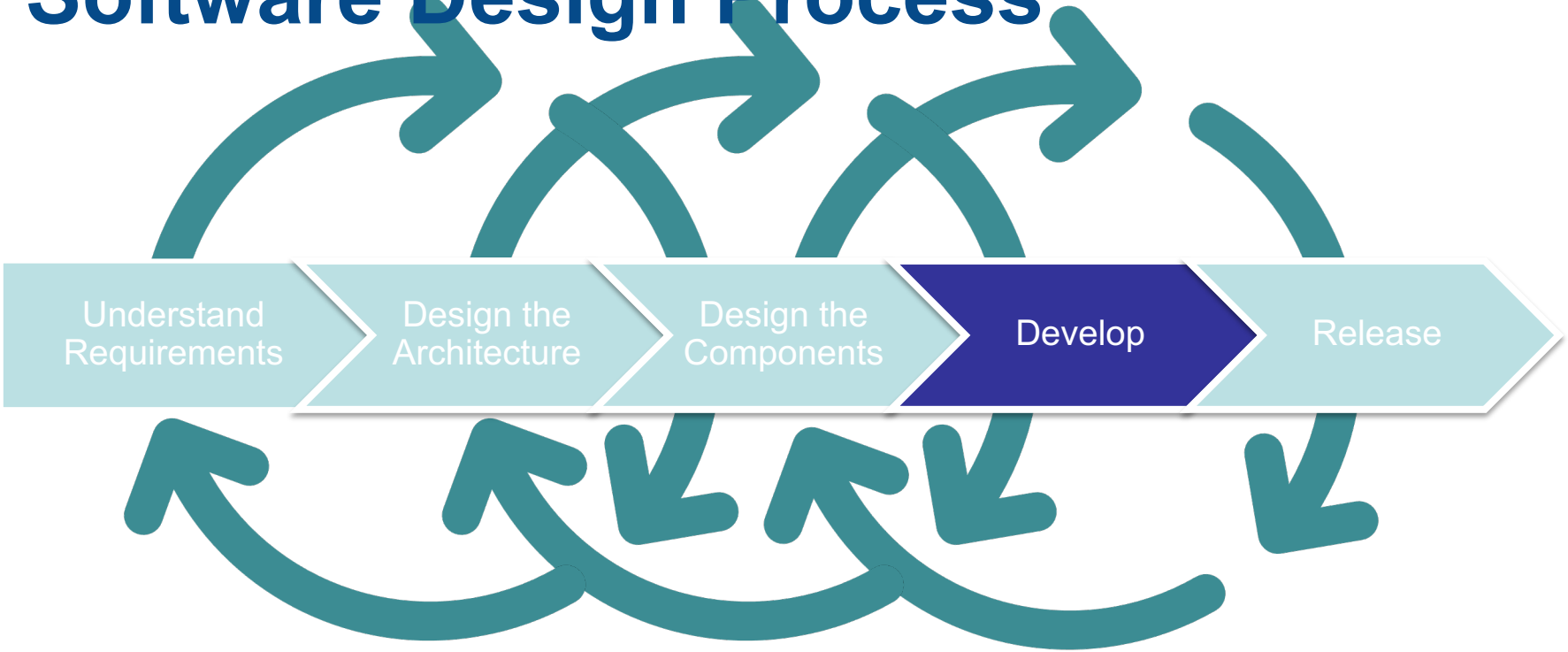
INFS 2044

Week 12
Naming and Commenting

Learning Objectives

- Understand the importance of naming and commenting (CO4)

Software Design Process



Naming Helps Understanding

You know you are working on clean code when each routine turns out to be pretty much what you expected.

(Ward Cunningham)

Intention-Revealing Names

- int d; // elapsed time in days

Better:

- int elapsedTimeInDays;
  int daysSinceCreation;
  int daysSinceModification;
  int fileAgeInDays;

Intention-Revealing Names

```
public List<int[]> getThem() {
    List<int[]> list1 = new ArrayList<int[]>();
    for (int[] x : theList)
      if (x[0] == 4)
        list1.add(x);
    return list1;
  }
```

Intention-Revealing Names

```
public List<int[]> getFlaggedCells() {
    List<int[]> flaggedCells = new ArrayList<int[]>();
    for (int[] cell : gameBoard)
     if (cell[STATUS_VALUE] == FLAGGED)
       flaggedCells.add(cell);
     return flaggedCells;
  }
```

Avoid Ambiguity

- date.add(5)
- Does it add 5 weeks, days, hours,… ?
  - Better: date.addDaysTo(5) or date.increaseByDays(5)
- Does it mutate the receiver (date) or return a copy?
  - better: date.daysLater(5)

Avoid False Clues

- accountList should be a List object, otherwise rename the variable
  - accountGroup, bunchOfAccounts, accounts
- Avoid lower-case L or uppercase O as variable names

Meaningful Distinctions

```
public static void copyChars(char a1[], char a2[]) {
    for (int i = 0; i < a1.length; i++) {
      a2[i] = a1[i];
    }
  }
```

Pronounceable Names

```
class DtaRcrd102 {
    private Date genymdhms;
    private Date modymdhms;
    private final String pszqint = ”102”;
    /* … */
  };
```

```
class Customer {
    private Date generationTimestamp;
    private Date modificationTimestamp
    private final String recordId = ”102”;
    /* … */
  };
```

Searchable Names

- Choose names that are easy to search for
  - WORK_DAYS_PER_WEEK vs “d” vs “5”
- The length of a name should correspond to the size of its scope
  - Single-letter variables should be confined to local scope

Named Constants

- Use named constant symbols instead of hard-coded strings or numbers
  - DAYS_IN_WEEK instead of 7
- Descriptive name, all CAPS
- Consider using an enum/class/etc for constants
- Easier to read, search, change value, test

Class Names

- Use noun or noun phrase names
  - Customer
  - WikiPage
  - Account
- Avoid general terms
  - Data, Info, Manager, Processor

Method Names

- Use verb or verb phrase
  - convey what the method does, not how it is implemented
- Accessors, mutators, predicates should be named according to coding conventions
  - getXXX, setXXX, isXXX, hasXXX, includesXXX, etc

Consistency

- Pick one term per concept
  - Fetch, retrieve, or get (not all of them)
  - Controller, manager, driver
- One concept per term
  - Don’t reuse the same term for multiple meanings
  - Choose the word that makes most sense: add, insert, or append

Comments

- **Comments should describe things that are not obvious from the code**
  - What & Why, not HOW

- Comments and code may not match
- Redundant comments add noise

- **Improve your code rather than adding comments**!

Self-Explanatory Code

- // Check to see if the employee is eligible for full benefits
  if ((employee.flags & HOURLY_FLAG) &&
  (employee.age > 65))

Better:

- if (employee.isEligibleForFullBenefits())

Functions/Variables over Comments

```
// does the module from the global list <mod> depend on the
// subsystem we are part of?
if (smodule.getDependSubsystems().contains(subSysMod.getSubSystem()))

ArrayList moduleDependees = smodule.getDependSubsystems();
String ourSubSystem = subSysMod.getSubSystem();
if (moduleDependees.contains(ourSubSystem))
```

Acceptable Comments

- Legal comments, copyright, etc
- Explanation of intent
- Warning of consequences
- Amplification
- TODOs
- Self Documentation – Javadocs

Explanation of Intent

```
/**
    * This class Generates prime numbers up to a user specified
    * maximum.  The algorithm used is the Sieve of Eratosthenes.
    * Given an array of integers starting at 2:
    * Find the first uncrossed integer, and cross out all its
    * multiples.  Repeat until there are no more multiples
    * in the array.
    */
```

Unacceptable Comments

- Redundant (delete)
- Misleading (change or delete)
- Change logs, journals (use version control for this)
- Position and section markers (break up files)
- Closing brace markers (use shorter functions)
- Commented out code (delete, use version control)
- Too much information (relegate to a separate file)

Redundant Comments

```
// The day of the month.
private int dayOfMonth;


// Default constructor.
protected AnnualDateRule() {
```

Write the Comments First

1. Write class interface comments first
2. Write signatures and method comments for public interface methods, leaving the body empty
3. Iterate this process until it seems right
4. Then code the implementation

- This ensures that the comments are free of implementation details (how) and no comments are missing

Summary

- Meaningful names are critical for human understanding of software
- Consistent naming can speed up code understanding and reduce the potential for errors
- Good comments convey important information that is not obvious from the code.

Activities this Week

- Read the required readings
- Participate in Practical 3
- Complete Quiz 8

University of
South Australia