

# Object Oriented Software Engineering

Dagmawi Lemma (PhD)

Assistant Professor, AAU

[courses.dag@gmail.com](mailto:courses.dag@gmail.com)

<https://sites.google.com/site/dagmawi2012/>

Dagmawi L.  
([courses.dag@gmail.com](mailto:courses.dag@gmail.com))

Object Oriented Software  
Engineering

1

# Software Design

Dagmawi L.  
([courses.dag@gmail.com](mailto:courses.dag@gmail.com))

Object Oriented Software  
Engineering

91

## Software Design

- Requirements specification was about the WHAT the system will do
  - What the system does
  - What the system needs
- Design is about the HOW the system will perform its functions: How the system is realized
  - provides the overall decomposition of the system
  - allows to split the work among a team of developers
  - also lays down the groundwork for achieving nonfunctional requirements (performance, maintainability, reusability, etc.)
  - takes target technology into account (e.g., kind of middleware, database design, etc.)

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

92

## Software Design: Scope & Objective

- Broadly we have two levels
  1. Architectural design (high-level design)
    - architecture - the overall structure: main modules and their connections
    - design that covers the main use-cases of the system
    - addresses the main non-functional requirements (e.g., throughput, reliability)
    - hard to change
  2. Detailed design (low-level design)
    - the inner structure of the main modules
    - may take the target programming language into account
    - detailed enough to be implemented in the programming language

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

93

## Software Design: Scope & Objective

- The Design Process
  - Study and understand the problem from different viewpoints
  - Identify potential solutions and evaluate the tradeoffs
    - Design experience, reusable artifacts, simplicity of solutions
    - Sub-optimal, but familiar solutions often preferred – well known advantages/disadvantages
    - Design is about making tradeoffs!
  - Develop different models of system at different levels of abstraction and for different perspectives

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

94

## Software Design: Scope & Objective

- The Design Process (continued)
  - Developers
    - define the design goals of the project
    - select strategies for building the system,
      - hardware/software platform on which the system will run,
      - persistent data management strategy,
      - access control policy, and
      - handling of boundary conditions
    - Etc

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

95

## Software Design: Scope & Objective

- Challenges
  - Complexity
    - Often arbitrary, dependent on designer rather than problem ("accidental complexity")
  - Conformity
    - Often expected to conform to other software (e.g., legacy, standards)
  - Changeability
    - Needs to support change due changing requirements, constraints, etc.
  - Invisibility
    - No visible link from between design plans and product

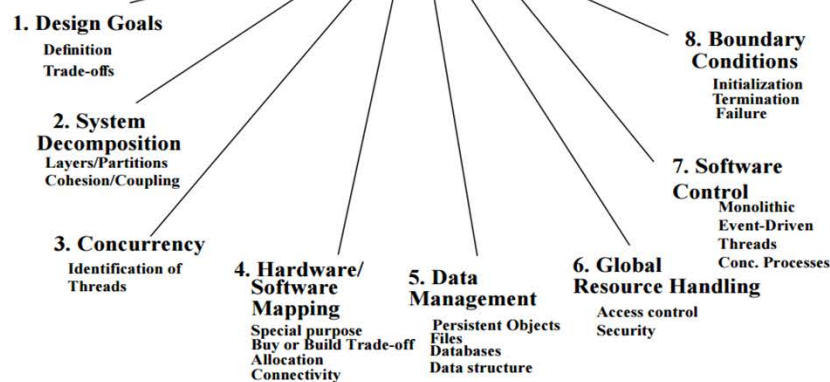
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

96

## Software Design: Scope & Objective

### Complexities of System Design



Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

97

## Software Design: Scope & Objective

- Producing specification for the new system in such a way it can be implemented using existing and selected technologies
- System specification has to be produced for
  - Input and output screen
  - Reports
  - Databases
  - Computer process
  - Deployment environment and infrastructure

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

98

## Design Goals

- Functional requirements are met by the modules/subsystems that are designed with specific service
- Nonfunctional requirements dictate the design process while selecting and as the designers trade-off in the process
  - Some nonfunctional requirements are mandatory, hence pseudo-requirement
  - Some are desired features
- Hence Design Goals need to be set/selected prior to commencing the design process

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

99

## Design Goals

- design goals are selected from a long list of highly desirable qualities
- The possible design criteria are
  - performance,
  - dependability,
  - cost,
  - maintenance, and
  - end user criteria

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

100

## Design Goals

- Performance

| Design criterion | Description                                                                |
|------------------|----------------------------------------------------------------------------|
| Response time    | How soon is a user request acknowledged after the request has been issued? |
| Throughput       | How many tasks can the system accomplish in a fixed period of time?        |
| Memory           | How much space is required for the system to run?                          |

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

101

## Design Goals

- Dependability

| Design criterion | Description                                                                       |
|------------------|-----------------------------------------------------------------------------------|
| Robustness       | Ability to survive invalid user input                                             |
| Reliability      | Difference between specified and observed behavior.                               |
| Availability     | Percentage of time system can be used to accomplish normal tasks.                 |
| Fault tolerance  | Ability to operate under erroneous conditions.                                    |
| Security         | Ability to withstand malicious attacks                                            |
| Safety           | Ability to not endanger human lives, even in the presence of errors and failures. |

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

102

## Design Goals

- Cost

| Design criterion    | Description                                                                                                         |
|---------------------|---------------------------------------------------------------------------------------------------------------------|
| Development cost    | Cost of developing the initial system                                                                               |
| Deployment cost     | Cost of installing install the system and training the users.                                                       |
| Upgrade cost        | Cost of translating data from the previous system.<br>This criteria results in backward compatibility requirements. |
| Maintenance cost    | Cost required for bug fixes and enhancements to the system                                                          |
| Administration cost | Money required to administer the system.                                                                            |

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

103

## Design Goals

- Maintainability

| Design criterion             | Description                                                           |
|------------------------------|-----------------------------------------------------------------------|
| Extensibility                | How easy is it to add the functionality or new classes of the system? |
| Modifiability                | How easy is it to change the functionality of the system?             |
| Adaptability                 | How easy is it to port the system to different application domains?   |
| Portability                  | How easy is it to port the system to different platforms?             |
| Readability                  | How easy is it to understand the system from reading the code?        |
| Traceability of requirements | How easy is it to map the code to specific requirements?              |

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

104

## Design Goals

- End-user criteria

| Design criterion | Description                                            |
|------------------|--------------------------------------------------------|
| Utility          | How well does the system support the work of the user? |
| Usability        | How easy is it for the user to use the system?         |

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

105

## Design Goals

- Generating Alternatives
  - Low-end solution
    - Conservative in terms of the effort, cost, and technology
  - High-end
    - The sky is the limit
  - Solution in the middle

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

106

## Methods and Approaches

- Top-down Vs bottom-up
  - Top-Down
    - Recursively partition problem into smaller sub-problems
    - Continue until tractable solutions found
    - Note: Not practical for large system in its pure form
  - Bottom-Up
    - Assemble, adapt, and extend existing solutions to fit the problem
  - In practice: A combination of both
    - Decompose large problems into smaller, but using previous design knowledge
    - Use existing components and solutions
    - Perhaps tackle problematic portions first

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

107

## Methods and Approaches

- Design methods provide guidance
- Different flavors
  - Heavyweight methods
    - Highly structured and documentation oriented methods
    - Usually generate mega amounts of graphical documentation
  - Agile methods
    - "Travel light"
  - Agile model-based methods
    - Best of both worlds

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

108

## Methods and Approaches

- Design methods provide guidance
- Different flavors
  - Heavyweight methods
    - Highly structured and documentation oriented methods
    - Usually generate mega amounts of graphical documentation
  - Agile methods
    - "Travel light"
  - Agile model-based methods
    - Best of both worlds

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

109

## Methods and Approaches

- Design Notations
  - Taking an abstraction implies that you are making a decision about which details are important and which can be ignored. This decision is based on your viewpoint.
  - Four key viewpoints in software design:
    - Structural - the static properties of the software
    - Behavioral - cause and effect
    - Functional - what tasks the software performs
    - Data modeling - the data objects used
  - UML – takes the lead

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

110

## OOSE – CS446

Thursday May 14, 2020

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

111

## Fundamentals to SD - Modularity, Coupling and Cohesion

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

112

## Modularity

- The concept of modules may vary between programming languages
  - Module can refer to encapsulation of data and functions
  - In some case it might also refers to the collection of several functions/classes, which is also referred as package
- Whichever concept we consider, it is about decomposition of the larger system into smaller and manageable systems, which can be called subsystem

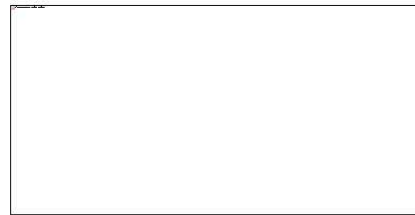
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

113

## Modularity

- Modularity is about a logical partitioning of a system considered as a technique for systematically slicing complex software to be manageable for the purpose of implementation and maintenance
- The logic of partitioning, often, is based on the functional dependency or data dependency
  - Note: implementation considerations are also used as a criterion to decompose a system



Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

114

## Coupling and Cohesion

- A system need to be modularized. Yet the modules need to interdependent.
- System is required to be
  - loosely coupled
  - Highly cohesive
- **Coupling:** the extent to which modules are interdependent
- **Cohesion:** the extent to which all instructions in a module related to a single function

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

115

## Coupling and Cohesion

- Types of Coupling
  1. **Data coupling**
    - Parameter = Unit data
  2. *Stamp coupling*
    - Parameter = Data structure
  3. Control coupling
    - Based on control flags passed between modules
  4. Common coupling
    - Global variable
  5. *Content coupling*
    - E.g. passing by reference

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

116

## Coupling and Cohesion

- Types of Cohesion
  1. Functional
  2. Sequential
    - Instructions within are related on data
  3. Communicational
    - Sequence does not matter
  4. Procedural
    - Instructions sequence matter + flow of control
    - Instructions within the module can be related to other modules

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

117

## Coupling and Cohesion

- Types of Cohesion (continued)
  5. Temporal
    - Instructions are related through flow of control but the order doesn't matter
    - Instructions are not functionally related but appear at the same time
  6. Logical
    - Consists of several sets of instructions, but the particular set that is executed is determined from outside the module
    - Flag that specifies what to do is passed in from outside
  7. Coincidental
    - Instructions have no relationship
    - The worst type of cohesion
    - Result of disorganized programming

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

118

## Designing a Software

- Structured approach (modules and the spec.)
- OO approach
  - How your requirement helps your design
  - use-case to Subsystem mapping
  - Subsystem specification
  - Subsystem Decomposition – how to
- Components vs. Subsystem
  - Component description
  - Organizing subsystem/component in your project

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

119

## Structured Approach

- Structured approaches creates the design specifications, independently for the data and the process logic
- Designing Internals of the software leads to decomposing the software and specifying those decomposed parts, which we call modules (i.e. for the process logics)
  - Turning system description into a blueprint for the physical information system that will be programmed
  - Deliverables
    - Structure Chart
    - Module Specification

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

120

## Designing Internal

- Structured Chart
  - Used to show how the various program parts (modules) are physically organized
- Module
  - A module is a relatively small unit of a system that is defined by its functions.
    - Self-contained system component
  - Often, single point of entry and single point of exit

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

121

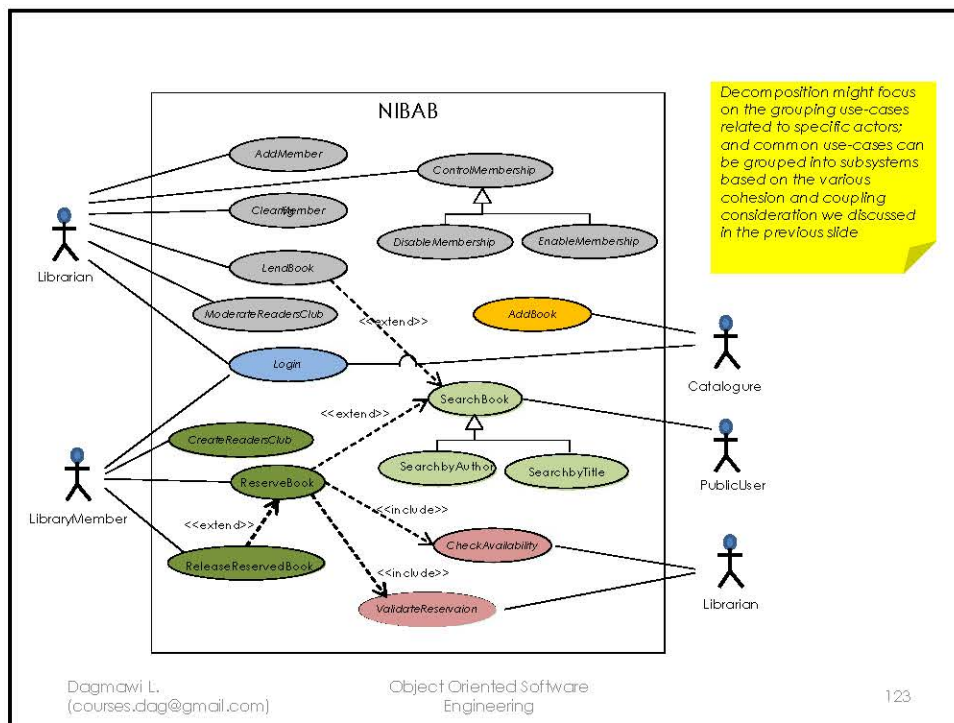
# Object Oriented Approach

- OO approaches creates the design specifications by collecting classes into subsystems
- In the OO approach, the persistent part of the software is created by collecting the entity class types
  - visibility is determined for each method/interface and attribute of the class
  - datatype of the attributes, returns and parameters are defined (this could be programming language specific)
- UML used to model the internal structure as well as architecture of the system
  - Package diagram
  - Component diagram
  - Deployment diagram
- Subsystems can be created by
  - grouping related classes,
  - grouping use-cases, where
    - the focus can be the relationship between the use-cases or
    - the actor view can be considered for deciding the decomposition

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

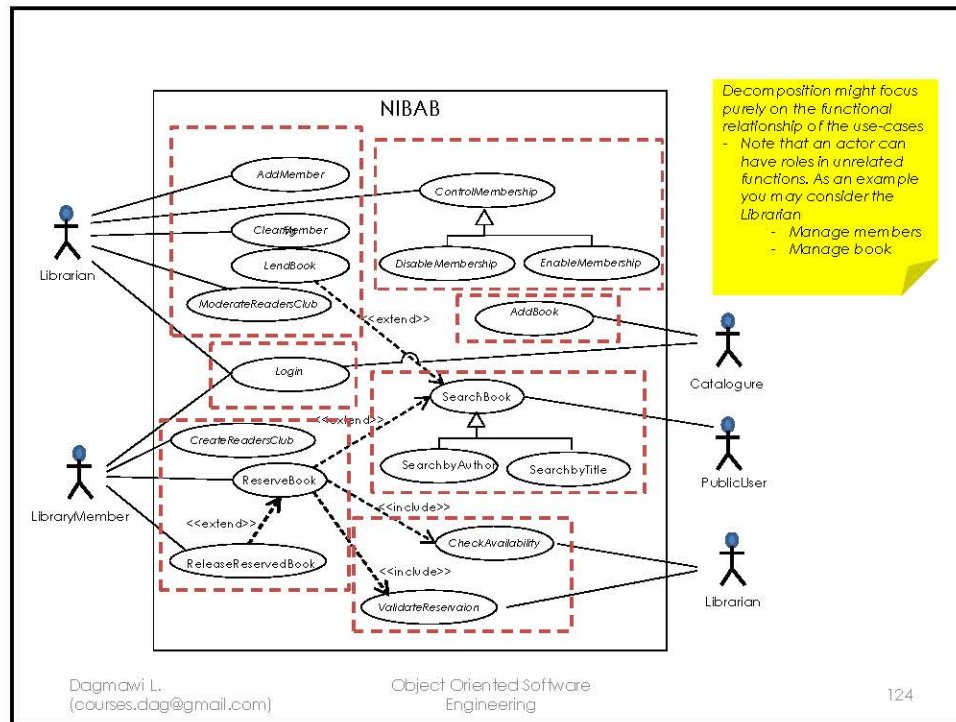
122



Dagmawi L.  
(courses.dag@gmail.com)

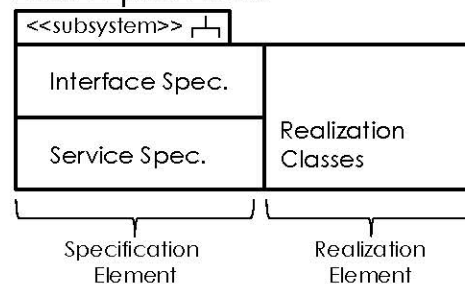
Object Oriented Software  
Engineering

123



## Subsystem

- How do we create subsystem?
  - Grouping classes can be one approach
  - Mainly subsystem is characterized by the service it provides



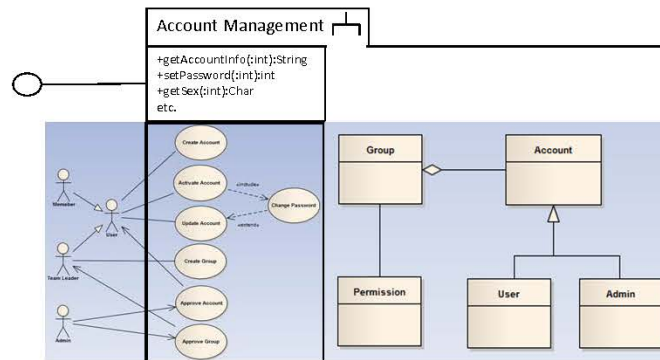
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software Engineering

125

## Subsystem

- What does a subsystem that deliver multiple services implies?



Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

126

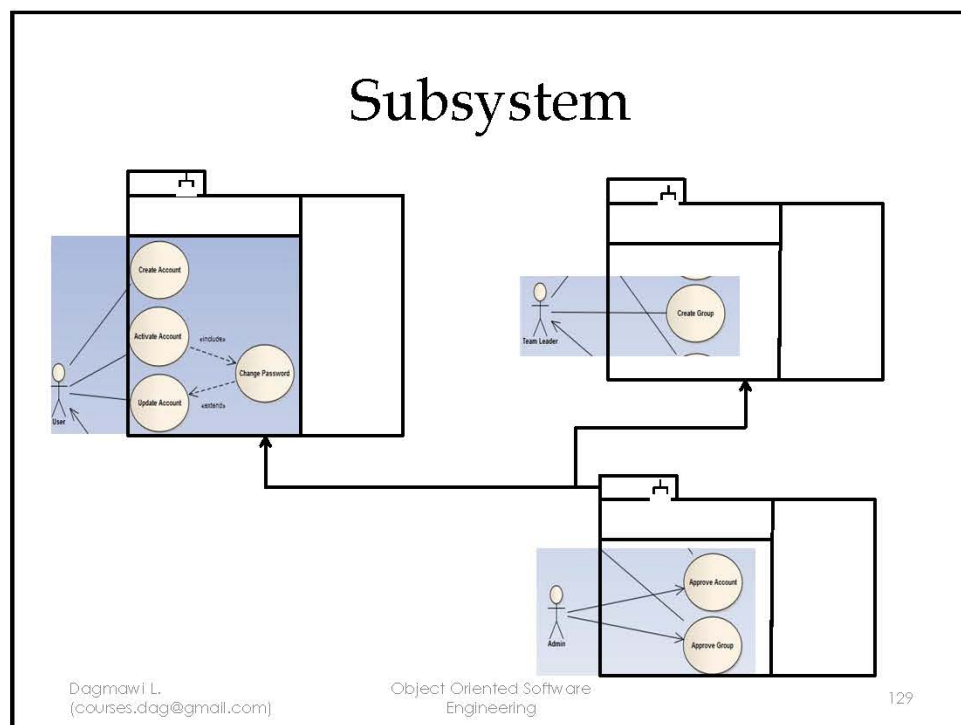
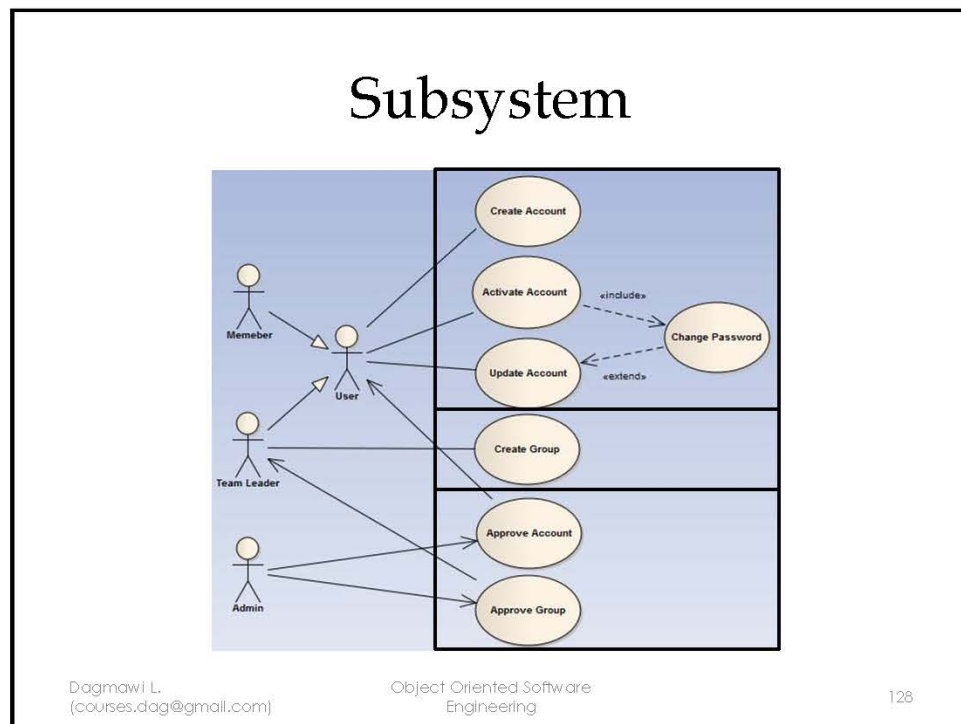
## Subsystem

- What does a subsystem that deliver multiple services implies?
  - Coupling is high (or may not exist)
  - Indicates the need for decomposition

Dagmawi L.  
(courses.dag@gmail.com)

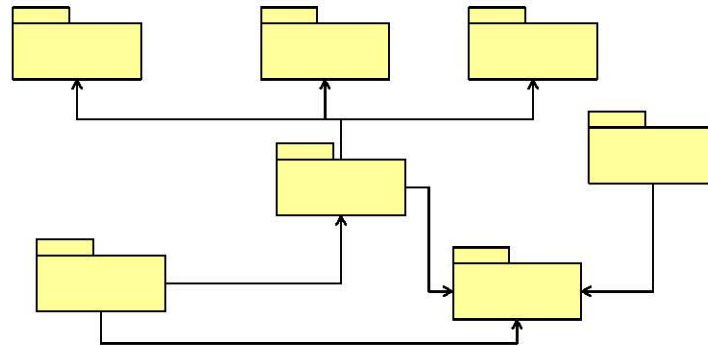
Object Oriented Software  
Engineering

127



## Subsystem

- What does a subsystem that contains single class or service implies?



Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

130

## Subsystem

- What does a subsystem that contains single class or service implies?
  - Cohesiveness is low
  - Indicates the need for composition

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

131

## Subsystem Specification

- Pseudocode
  - Helps an analyst think in structured way about the task a module is designed to perform
  - Acts as a communication tool between the development team
    - The analyst who understands what the module needs to do
    - The programmer who must write the code to make the module perform appropriately

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

132

## OOSE - CS446

Thursday May 21, 2020

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

133

## Subsystem Specification

- Subsystem specification shall be clearly written
  - Name of the module
  - Modules function/purpose
  - Parameters and returns description (including the type)
  - The type of coupling and cohesion it has with other modules

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

134

## Subsystem Representation

- The UML **package diagram** is used to represent each subsystem and the dependency between them
- At the analysis phase, we have used class diagram to show the structure of the system
  - Class diagram can be complex as the system gets larger (i.e., involves many classes)
- Also, now we have started to decompose the system
  - Hence we need to show the decomposition and specify the high level structure of the system

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

135

## Subsystem Representation

- Package diagram used to structure high level systems while it is also useful
  - to group elements
  - to show how the system is decomposed
  - Associate each use case as a service hence missing requirements can be unlikely or reduced
  - to provide a namespace for the grouped elements
  - A package may contain other packages, thus providing for a hierarchical organization of packages.
  - UML elements can be grouped into packages.

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

136

## Components

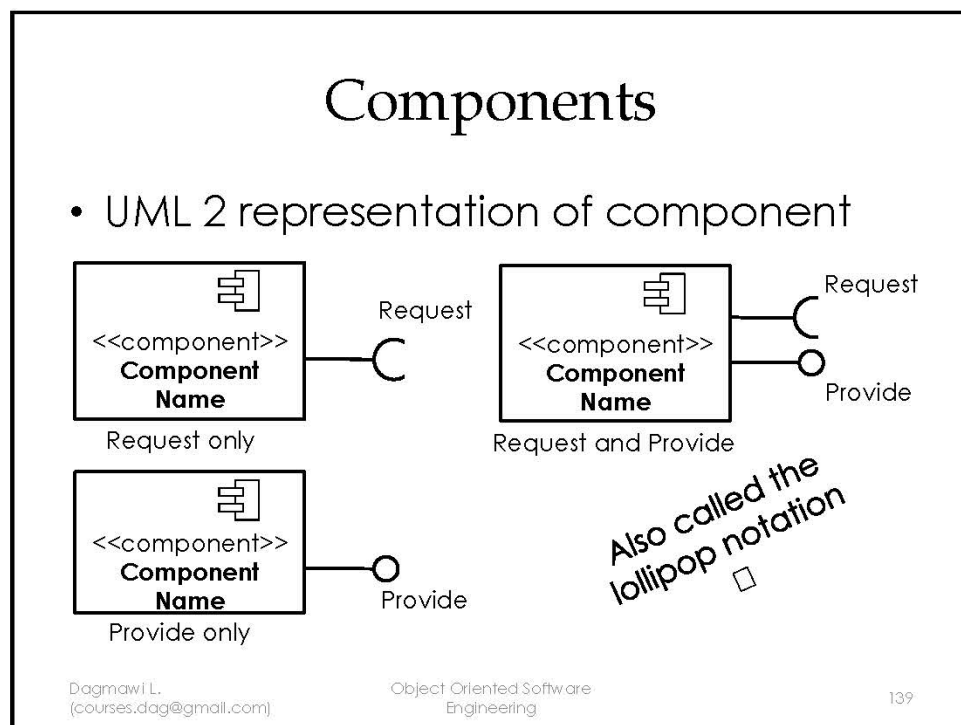
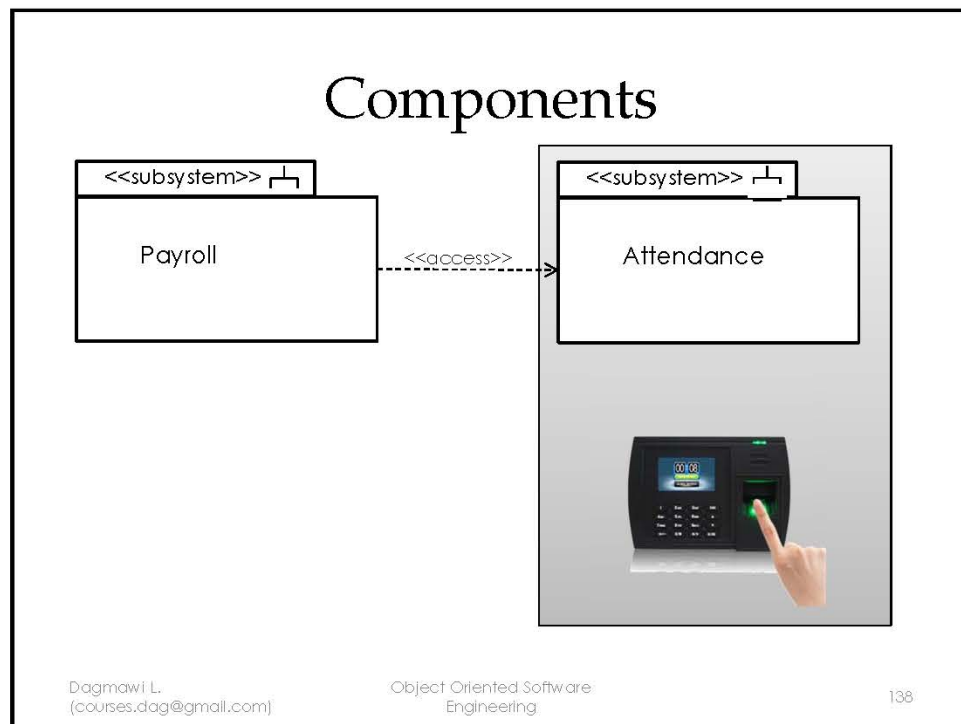
- A software component is **a software element that conforms to a component model and can be independently deployed** and composed without modification according to a composition standard

Bill Council George T. Heineman
- A Component is a physical piece of a system, such as a compiled object file, piece of source code, shared library or Enterprise Java Bean (EJB); it can also be a commercial off-the-shelf (COTS)
  - even a hardware as long as it provides the service required in the system (e.g., barcode reader, fingerprint reader for keeping the attendance)

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

137



## Component

- A component can have
  - Interfaces
    - An interface represents a declaration of a set of operations and obligations
  - Usage dependencies
    - A usage dependency is relationship which one element requires another element for its full implementation
  - Ports
    - Port represents an interaction point between a component and its environment
  - Connectors
    - Connect two components
    - Connect the external contract of a component to the internal structure

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

140

## Interface

- A component defines its behavior in terms of **provided** and **required** interfaces
- An interface
  - Is the definition of a collection of one or more operations
  - Provides only the operations but not the implementation
  - Implementation is normally provided by a class/ component
  - In complex systems, the physical implementation is provided by a group of classes rather than a single class

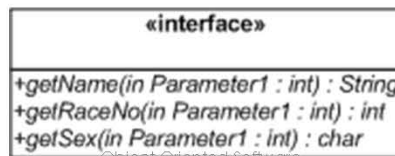
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

141

## Interface

- A component defines its behavior in terms of **provided** and **required** interfaces
- An interface
  - Is the definition of a collection of one or more operations
  - Provides only the operations but not the implementation
  - Implementation is normally provided by a class/component
  - In complex systems, the physical implementation is provided by a group of classes rather than a single class
- Can be shown using a rectangle symbol with a stereotype <<interface>> preceding the name



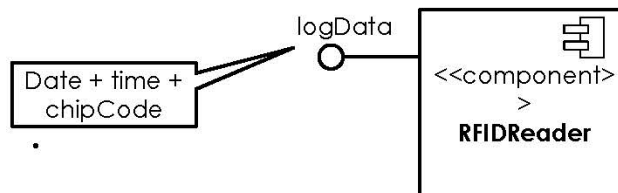
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

142

## Interface

- A provided interface
  - Characterize services that the component offers to its environment
  - Is modeled using a ball, labeled with the name, attached by a solid line to the component



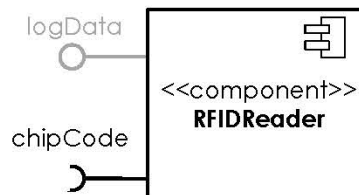
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

143

## Interface

- A required interface
  - Characterize services that the component expects from its environment
  - Is modeled using a socket, labeled with the name, attached by a solid line to the component
  - In UML 1.x were modeled using a dashed arrow



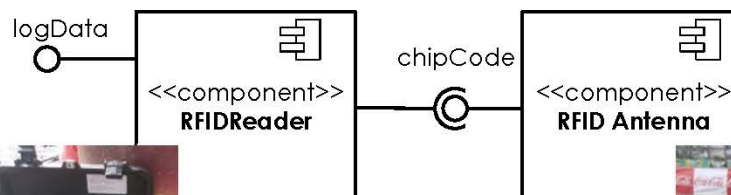
Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

144

## Interface

- When one component provided and the other request the same service, then the ball and socket can be combined

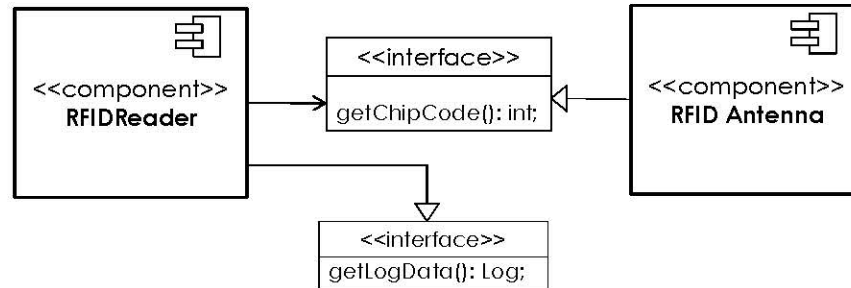


Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering



## Interface



- If the interface is represented using rectangle symbol then the dependency arrow shall be used

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

146

## Summary

- Designing internal (**low-level design**) focus on specifying the **decomposed elements** of the software
- In **structured approach** the **dependency** between **modules** is represented using **structured chart**, and the type of **coupling** shall be represented using the direction of communication and the appropriate notation (e.g. flag, data, etc.)
- The structured chart can also be used to show the **sequence** or the **logic** how a particular module is initiated

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

147

## Summary

- In OO approach **grouping of classes** could help to define the **subsystem**
- It is also possible to **map use-case/s** to subsystem so as to define the subsystem by the service it provides
- Service/s that is/are provided by a subsystem can also be gained by using a component
- Component is part of the software that can be independently installed, **COTS** can be an alternative

Dagmawi L.  
(courses.dag@gmail.com)

Object Oriented Software  
Engineering

148