Introduction to OOA, OOD, and UML

-Course Overview-

Week 01

College of Information Science and Engineering
Ritsumeikan University

RITSUMEIKAN

Acronyms

- OOA : Object-Oriented Analysis
- OOD: Object-Oriented Design
- UML: Unified Modeling Language

Today's class outline

- Review of basic course information
  - Contact details
  - Course objectives
  - Textbook and materials
  - Student guidelines and course policies
- Course structure and overview
  - Schedule of classes
  - Brief review of OOA, OOD, and UML
  - Topics in OOA, OOD, and UML
- About mini-tests , mid-term test, homework assignments, and self-preparation

Contact details

- **Course name (in English):**
  Introduction to OOA, OOD, and UML

- **My name:**
  Igor Goncharenko

  - E-mail: igor@fc.ritsumei.ac.jp
  - Office: Creation Core, 707 (7th floor)
  - Consultation: Daily, 13:00-18:00 (except on-line period, after May 2nd)

RITSUMEIKAN

Communication

- By e-mails (not so often)
- Web page for downloads: http://www.dh.is.ritsumei.ac.jp/UML/

You can download slides (PDFs) after classes, mini-test problems (after solving the mini-tests), mid-term test, extra materials (PDFs, diagram files and documentation)

Course objectives

- The academic plan (curriculum) of Ritsumeikan University requires this course to be taught in English
- The goal of the course is to provide you with a basic understanding of OOA and OOD, which are very powerful methodologies for modern engineering fields, including efficient software development.
- This knowledge will be extremely useful for understanding important stages of system design, systematic and analytical system development, testing and re-usability.

RITSUMEIKAN

UML : practical tool

- Learning of OOA and OOD is supported by UML (Unified Modeling Language)
- UML is *de facto* standard for reliable industrial software development
- UML (current standard version UML 2.0) is very useful for software diagram building
- UML does not require coding skills, it is a visual tool

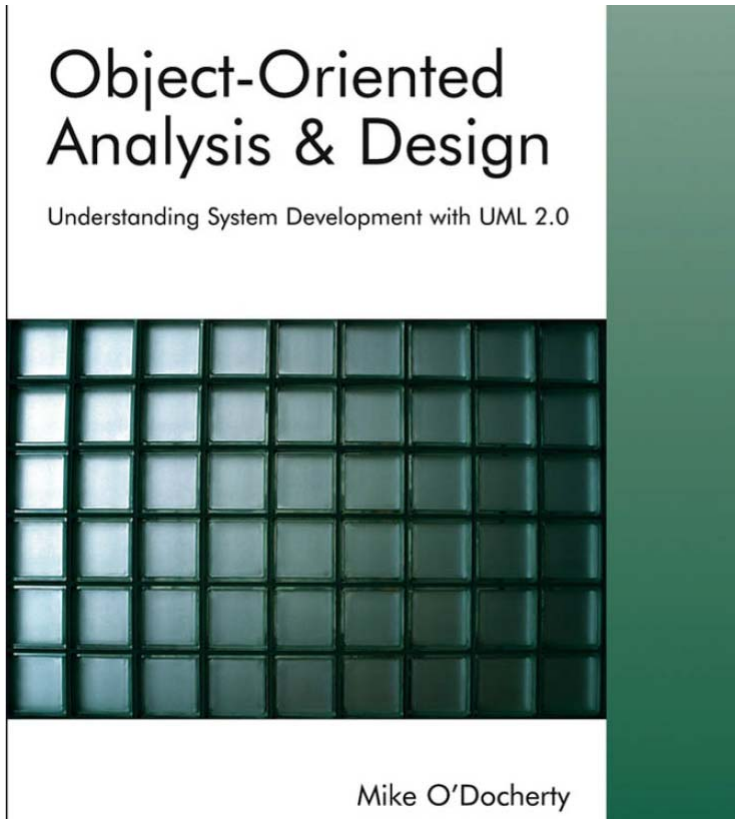
Textbooks and materials

- Most knowledge of this course you are expected to get during lecture hours.
  All materials (in PDF format) of this and all the future lectures cam be downloaded from our class Website.
- If you still do not have it, the textbook is available from the University's bookstore:

  Object-Oriented Analysis and Design: Understanding System Development with UML 2.0 (by O'Docherty, Publ. by Wiley, ISBN 0470092408)

RITSUMEIKAN

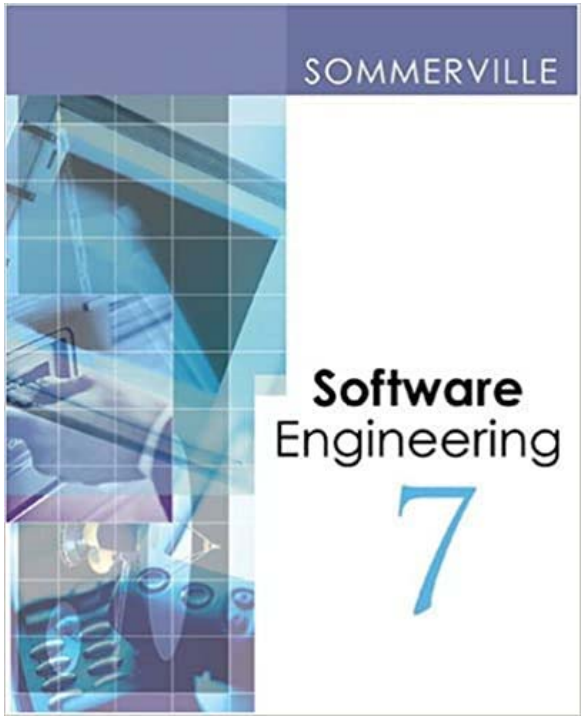
Textbook cover



- You can also buy this book through Amazon.com (it's cheaper there!)

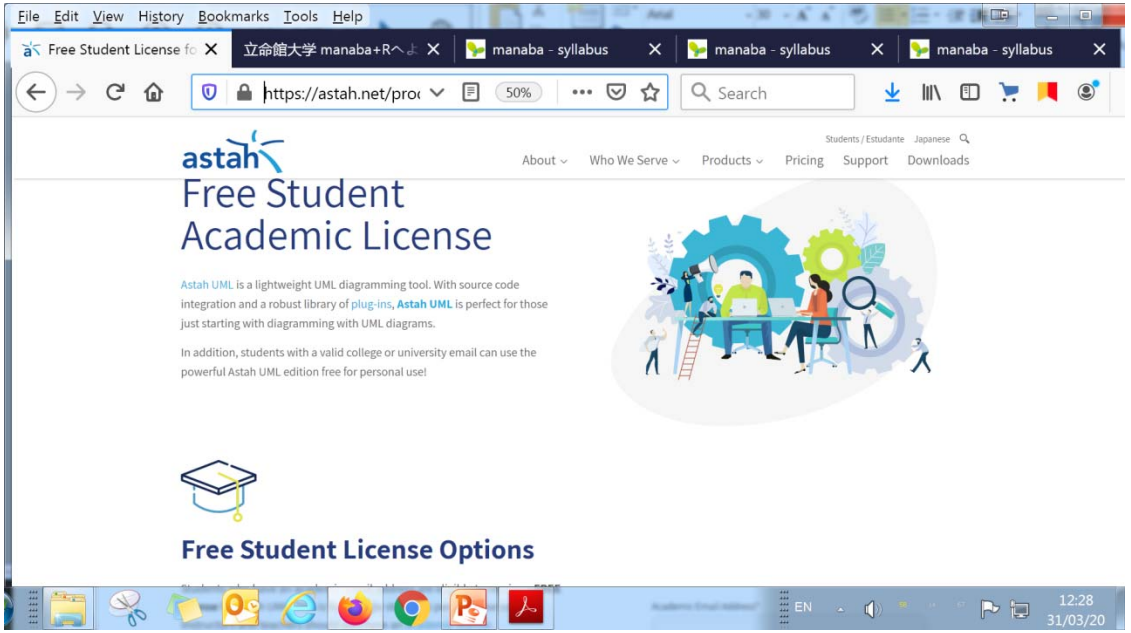
RITSUMEIKAN

Companion textbook



- Software Engineering, 7th edition or newer, by Ian Sommerville, ISBN-10: 9780321210265

Free UML software



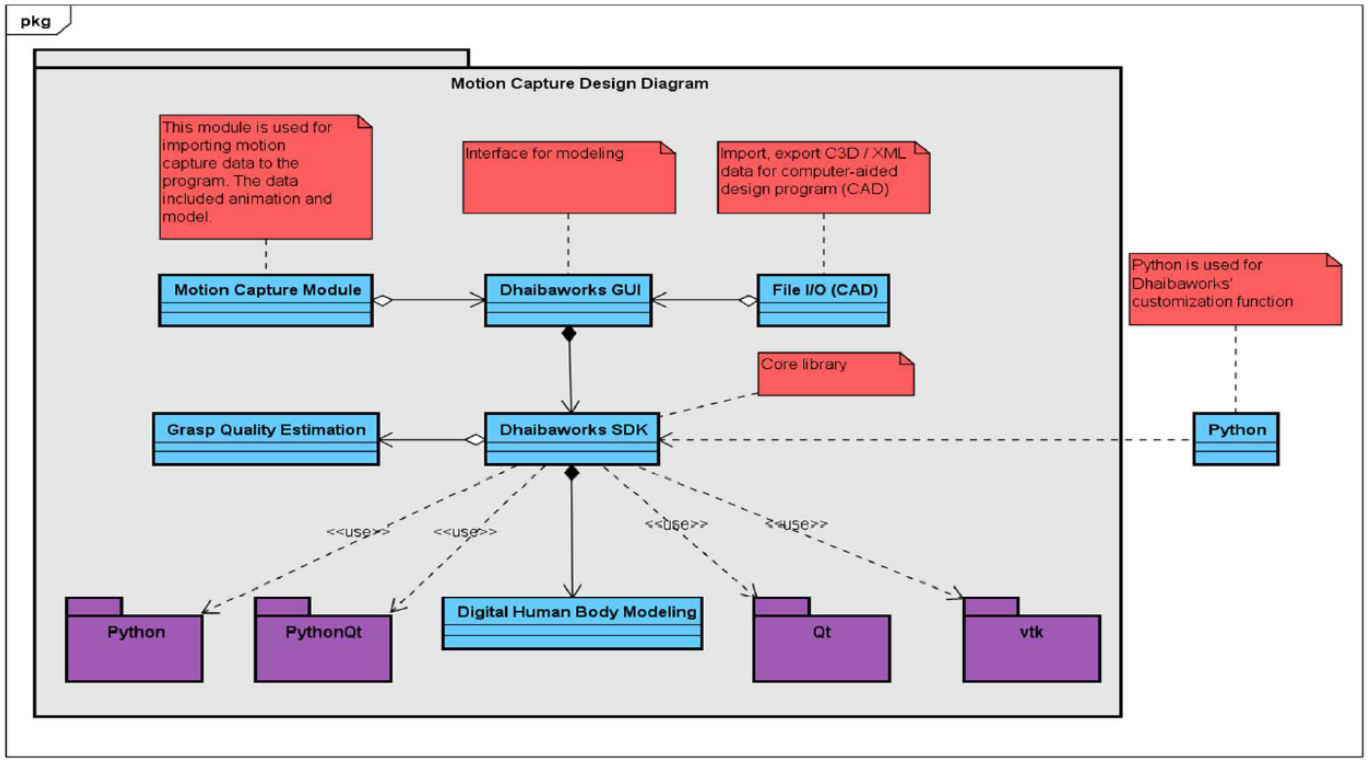
- Free download for first time from:

  https://astah.net/products/free-student-license/

- Professional academic license (not free) will be purchased by the instructor in April

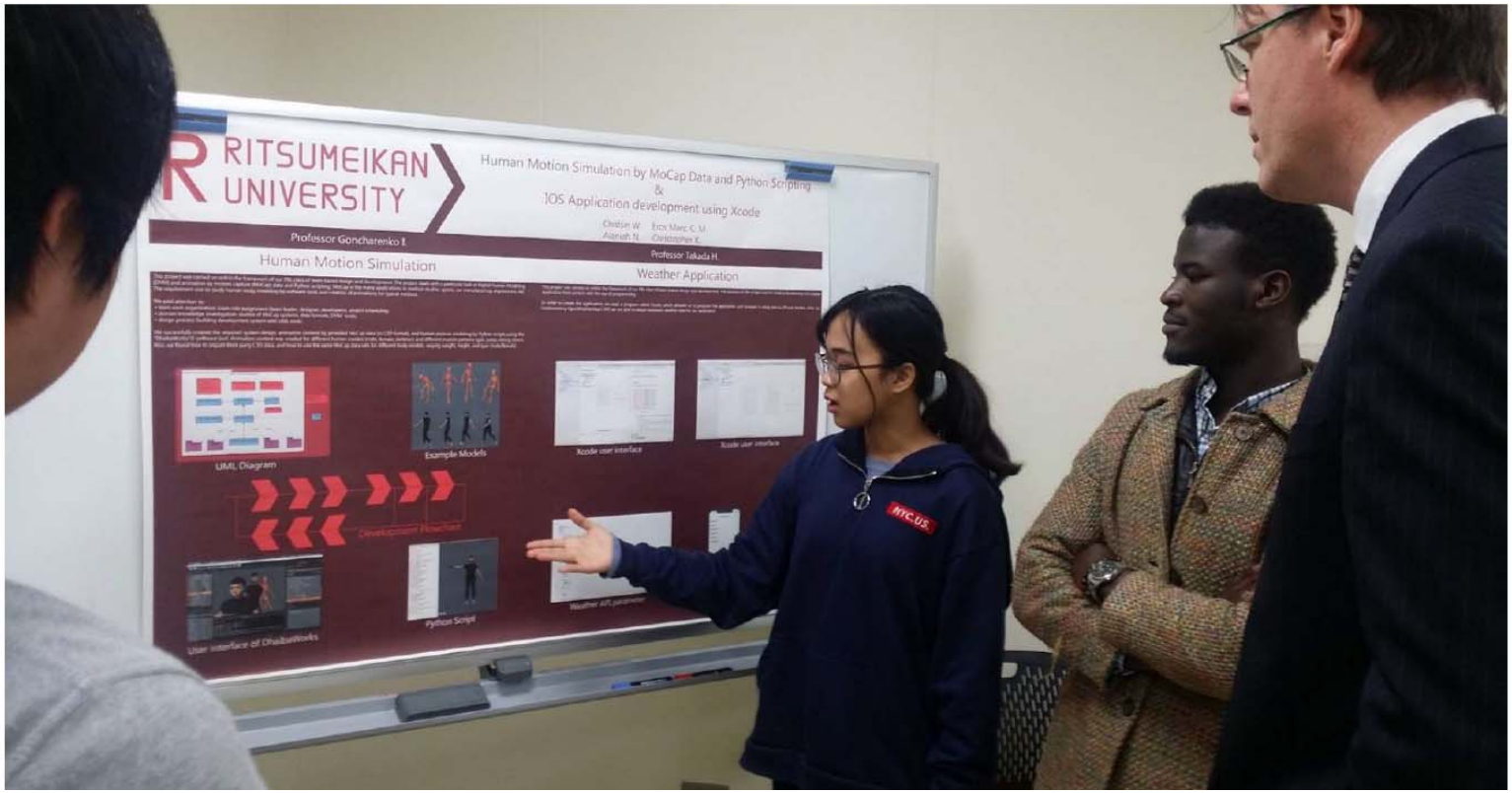
Usage of UML for PBL-2,-4

- Example: diagram of software structure reconstruction



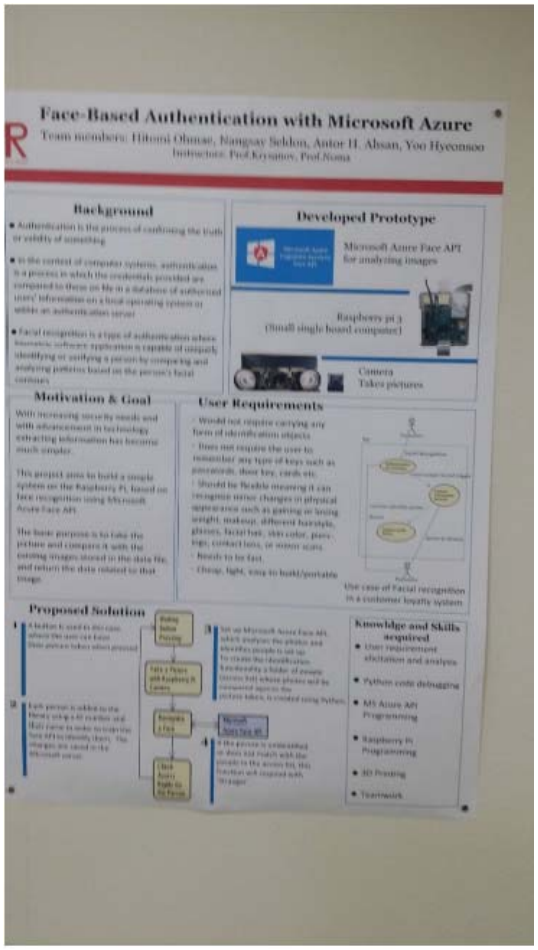
Usage of UML for PBL

- Example: usage of UML for your posters



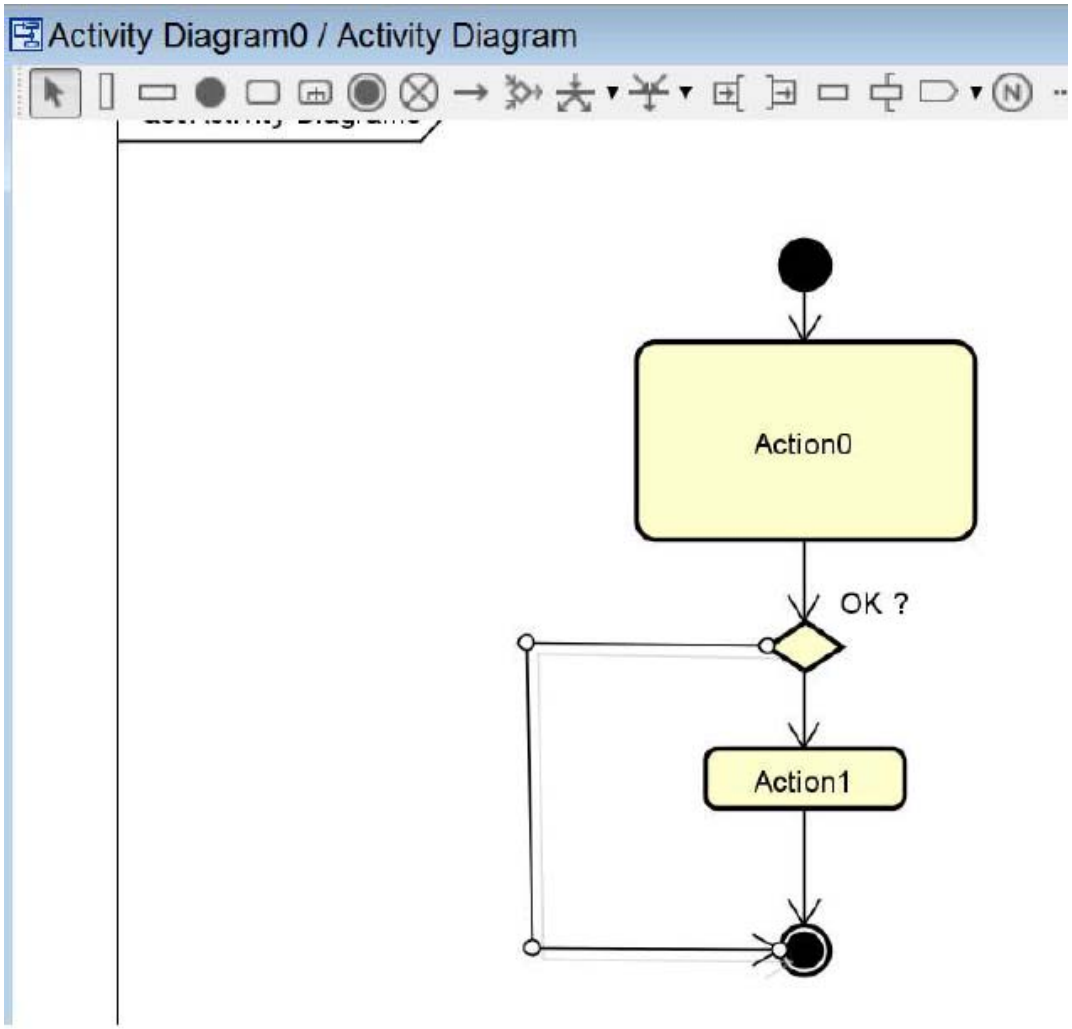
Usage of UML for PBL

- Example-2: “Use case diagram” and “activity diagram”



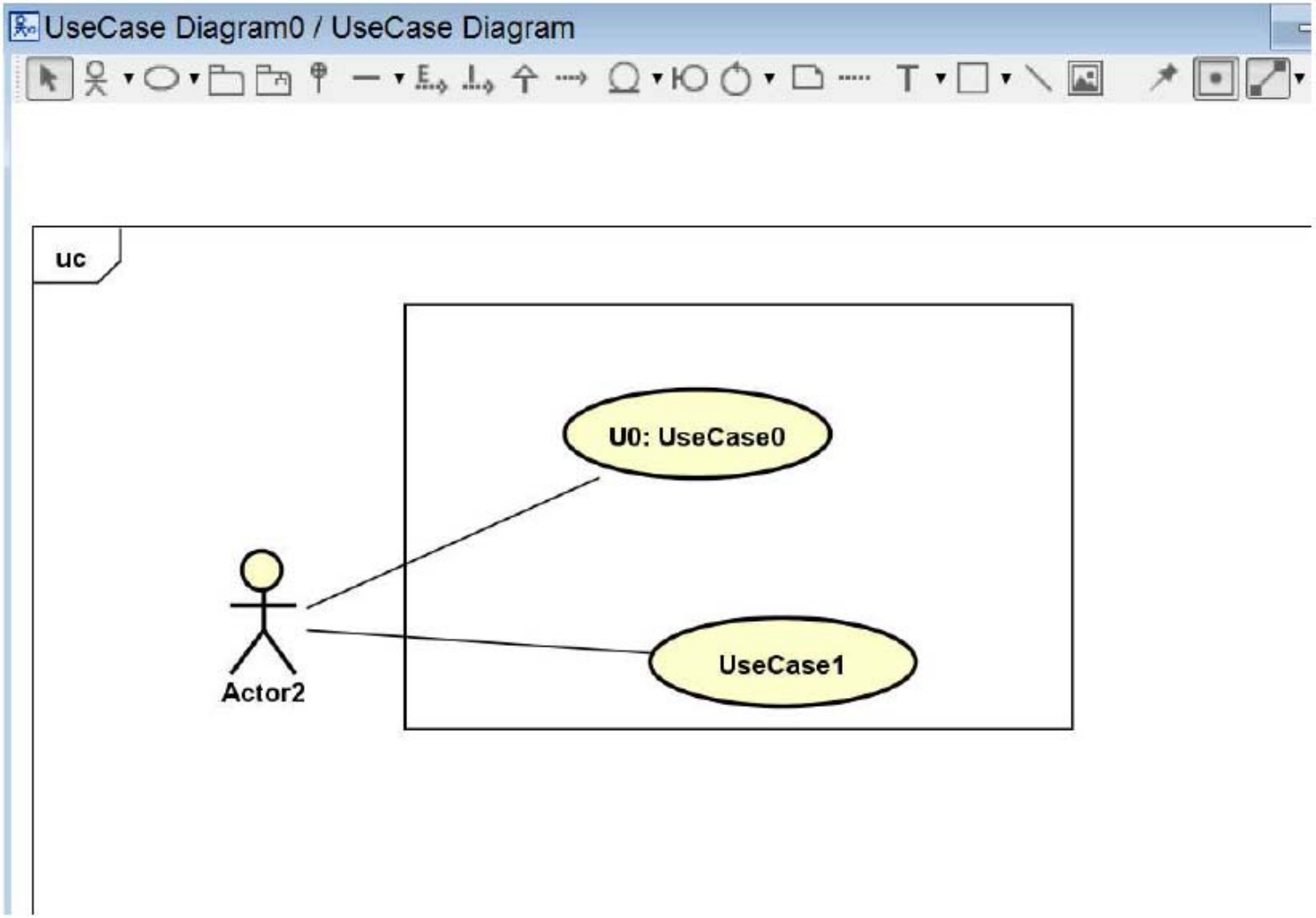
RITSUMEIKAN

Activity diagram



RITSUMEIKAN

Use case diagram



Student guidelines

- Read the relevant materials – slides, etc. – before every class
- Study the textbook before every class
- Complete all self-preparation assignments in due time
- Whenever needed, learn new English vocabulary
- Maintain good self-discipline and pay attention in class
- Do your best in daily mini-tests

Classroom policies

- **Do not sleep in class.** If you must sleep, leave the classroom
- **No cell phone use during the class**
- **Do not talk or even whisper. If you make noise then you will be brought out of the classroom**
  - When you make noise in class, you disturb the teacher
  - When you whisper in class, you disturb other students
  - When you talk in class, you waste your time

Classroom policy (on-line)

- Do copy assignment solutions from each other
- The solution must be your own work !

Grading policy

- **Students who miss more than 5 lectures will automatically receive an 'F' mark and no credits**
- **Points:**
  - Daily mini tests (which include self-preparation assignments) → up to 30
  - One mid-semester test → up to 30
  - Final exam → up to 40

- Grading: less than 60 points → F,
  60-69 → C,
  70-79 → B,
  80-89 → A, over 89 → A+

Point policy during on-line period

- Points will be given based on assignment completeness (usually, one week will be given to complete one assignment)
- Assignments will be uploaded to Manaba+R in DOC/DOCX format (and text format, when possible)
- Point policy during on-line period might be slightly changed, when you get full access to Internet via your personal computers

Academic integrity

- Cheating will result in an automatic ‘F’ for the semester and turning the case over to the College administration. There will be no second chances
- When you feel you need help with your studies, contact the teacher but never try just to copy test answers from your friend or someone else
- Your tests must be your own, original work

RITSUMEIKAN

Schedule of classes

1. Course overview (today)
2. Objects
3. Classes
4. Inheritance
5. Type systems
6. Intermediate overview and evaluation
7. Development methodologies – I
8. Development methodologies - II
9. Requirement analysis
10. Problem analysis
11. System design
12. Subsystem design
13. Specifications and re-usability
14. Testing
15. Course overview
16. Final exam

RITSUMEIKAN

About mid-term test

- Mid-term test is very helpful for your final exam!
- Once you successfully solve the mid-term test, you will be almost 90% prepared for the final exam
- You can find the mid-term tests and all mini-tests on our UML class Website during the semester

What is the aim of this course ?

- The aim of this course is to give you a basic understanding of the processes and techniques used in *object-oriented* (OO) software development. The course clarifies the following questions:
  - How to create programs using object-oriented techniques?
  - What is needed prior program coding ?
  - How to conduct OO analysis and design?
  - What techniques are needed to maintain software long life and re-usability ?
  - How to use UML for OO design ?

Features of the course

- Even skills in programming are not required, some topics are illustrated by simple pieces of codes of object-oriented languages (Java, C++, C#); code constructions will be explained
- When possible, diagrams will be given in UML style
- Code commenting examples will be given in *Doxygen* style

- The course knowledge is useful for your future Engineering PBL classes and real project development !

RITSUMEIKAN

Further lectures

- The following slides will introduce each lecture topic and give you some good reasons for wanting to learn it

RITSUMEIKAN

Objects

- The concept of object is considered
  - Mainly, from the point of view of object-oriented software;
  - Depicting of objects by UML will be considered
- Encapsulation of objects is done by hiding of their private attributes (fields)
- Association and aggregation are two types of connections of objects. They combine objects in graph- or tree- structures. The links can be navigable.
- Objects can collaborate, e.g., by message exchange

RITSUMEIKAN

Classes

- The concept of class is considered
  - Class is fundamental category in programming;
- For classes, hierarchy is important, which is defined by inheritance, or generalization
- A lot of information can be stored in classes, e.g., name, fields, methods, operations
- Important feature of a class is its code re-usability, which results in faster and simpler development, easier software maintenance

RITSUMEIKAN

Inheritance

- Inheritance allows us to design new classes by using a *parent* class and adding new elements to the parent class
- Class hierarchy design will be considered, together with adding implementation to a class hierarchy
- Classes can be abstract and concrete, with re-definition of methods
- Example: stack implementation

RITSUMEIKAN

Type systems

- The meaning of type and type system will be considered:
  - Dynamic type systems
  - Static type systems

- A polymorphic variable refers to a different types at different times, and a polymorphic message has different methods associated with it

- Type casting (implicit and explicit) and template usage allow us to convert types

RITSUMEIKAN

Development methodologies - I

- Classical phases of development methodologies will be considered:
  - Requirements
  - Analysis
  - Specifications
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance and re-use

RITSUMEIKAN

Development methodologies -II

- Waterfall development methodology is a classic methodology of software development
- Object-oriented methodologies:
  - Role of UML and UML tool in development
- More flexible methodologies
  - Spiral
  - Incremental
  - Iterative
  - Hybrid

RITSUMEIKAN

Requirement analysis

- Requirement (functional and non-functional) is the important phase, which should be done prior any coding
- As the first recommended step, actor list and project glossary are created for requirement analysis
- Next, modeling the business context and system functionality (using high-level business use cases) is completed
- Optionally, system requirements can be modeled with the use case models, and UML communication diagrams and activity diagrams can be used for modeling

RITSUMEIKAN

Problem analysis

- Overview of the analysis process will be given
- Two analysis methodologies will be considered:
  - Static analysis
  - Dynamic analysis
- Improvement of analysis can be done using communication diagrams and state machine diagrams

RITSUMEIKAN

System design

- Main steps of system design and design priorities will be considered
- It is important to decompose a system into physical and logical components
- Example of complicated software/hardware system design with UML diagrams will be investigated
- Introduction to concept of UML packaging will be given

RITSUMEIKAN

Subsystem design

- The role of subsystem design is to decide exactly what objects we are going to implement and what interfaces they should have
- Design of user interfaces will be discussed
- To avoid writing fresh code (“from scratch”), looking for existing patterns, libraries, frameworks is recommended
- UML examples of subsystem will be given

RITSUMEIKAN

Specifications and re-usability

- There are formal and informal specifications. Very often customer's specifications are very informal
- Important examples of Doxygen style commenting will be presented
- Doxygen software tool use example will be given
- Techniques of patterns' re-use will be considered

RITSUMEIKAN

Testing

- Testing as one of the final stage of software development can be categorized as:
  - Black-box testing
  - White box testing
  - Modular testing
  - Integral testing
- Testing terminology and testing strategies will be considered
- Example of test table will be given

RITSUMEIKAN

UML diagrams

- You will learn several diagram types (minimum, 5)
- Below, they are ordered by development stage

No.	Phase	Primary Diagram type	Purpose	Importance	Other diagrams for the phase
1	Requirements	Use Case	Actor (Customer, user, ext. system)- system interaction; who uses	Moderate	Communication, Activity, State machine
2	Analysis	Communication	Collaboration between objects, messages, object relationship	Moderate	Class (sketch only), Object (sketch), activity (rough solutions), sequence, state machine
3	Design	Class	Detailed software design	Very High	
	Design	Activity	Algorithm description, thread organization, calculation activity	High	
	Design	Deployment	Software part distribution among computers, machines	Moderate	

RITSUMEIKAN

Summary

- This course covers important stages which should be done prior actual coding, and namely, OOA and OOD
- OOA and OOD is supported by UML tool
- This course is useful for practical project development in our PBL labs
- This course can be accompanied by learning of:
  - Programming in Java
  - Programming in C++ or C#

RITSUMEIKAN

Homework

- Download these slides and read this lecture materials again
- Download Astah UML (free student edition from astah.net), if you have computer. You have to register via your Ritsumeikan student e-mail to get the key-code for Astah UML activation.
- Download self-preparation assignment and complete it within one week
- Check access to our class Webpage http://www.dh.is.ritsumei.ac.jp/uml/

RITSUMEIKAN

Next class

- Objects
  - concept
  - encapsulation
  - association and aggregation
  - first use of UML tool

RITSUMEIKAN

About tests

- Beginning from class “Week03”, there will be our first mini-test (included into assignment for Week-03)
- A test will typically consist of 1-3 problems, which you must solve individually.
- The problems will be from the previous lecture topics

- There will be also a mid-semester test (Week 06)