Memo on Methodology

I. Analysis
- Use case
- First class diagram
- Relevant scenarios

II. Design
- Classes of the system
- Architecture of the system

III. Detailed design
- Behavior of the system

IV. Validation of the system
- Simulation
- Code generation
**Analysis**

- Use case of the system
- First scenarios
- Identification of main objects and classes
- Relevant scenarios of the system

**Use Case**

- **Purpose**
  - Identify services offered by the system
  - Identify the main functionalities of the system

- **Use case**
  - Relevant users of the system
  - Services offered or required by the users with regards to the target system
  - Use case focuses on the **goal** of the functions

- **How to identify use cases**
  - What functionality should be included or excluded?
  - Relations between the system under analysis and other systems
  - Users of the system?
  - Results and outputs produced by the system
  - Why do users / other systems need those functionalities offered by the target system?
Example of Use Case

- **System bound**
- **Use case**
- **Actor**

Basics of Actors

- **Actor** = user of the system with regards to the way it interacts with the system
- **User** = human person or another system such as a device, etc.
- **Identifier of the actor** = role of this actor onto the system
- **If the same actor plays several distinct roles onto the system, it is divided as many times as necessary**
How to Identify Actors?

- Who benefits from several needs listed in the requirements?
- Who will use the system?
- In which organization will the system be located?
- Who will provide the system with input data? Who will use these data? Who will use output data? Who will maintain the system?
- Who will perform technical support onto the system?
- Does the system use external resources?
- Does a user or another system play different roles onto the system?

Relations between Use Cases

- **Inclusion**
  - A use case includes the behaviour of another use case (=sub function)

- **Extension**
  - A use case (subfunction) extends the behaviour of another use case

- **Generalization/Specialization**
  - A use case is defined as a refinement of another use case
  - Generalization / specialization also applies to actors
Example of Relations among Use Cases

Another Example

Condition: client selected proxy
Extension point: selection
Profession-Driven Use Cases

- DefineContent
- Prepare
- Lecture
- Evaluate

Student
CourseManager
Teacher
Companies

Location-Driven Use Cases

- makeFrame
- SettleWheels
- Integrate

BucarestFactory
BarcelonaFactory
ParisFactory
StuttgartFactory
RomaniaFactory
PatelFactory
Exercises

- **A Client in a shop**
  - Enters in the shop
  - Cruises into the stands
  - Asks for information
  - Tries items
  - Pickups items or book them if not enough are available
  - Pays
    - May get a special rebate
    - May pay by check (amount > 10€, identity card required), credit card (amount > 15€), or by cash
  - May get a delivery for heavy items
    - Free or not

From Use Cases to ...

Functional decomposition

Object decomposition
Analysis

- Use case of the system
- First scenarios
- Identification of main objects and classes
- Relevant scenarios of the system

Purpose of the First Scenarios

- Use case diagrams -> functions and actors of the system
- First scenarios
  - Describe the interactions between the actors and the functions offered by the system
  - Non exhaustive scenarios -> only relevant scenarios
  - Scenarios are performed using UML Sequence Diagrams
UML Sequence Diagrams

- Basics of sequence diagrams
  - Gives clear visual clues to possible flows of control over time
  - Emphasizes time ordering
  - Shows object lifeline
  - Shows the focus of control

- UML 1.5
  - Notion of message (or stimulus) and of lifeline
  - Observation of time
  - Temporal constraints
  - Activation of an object

- UML 2.2
  - Suspension, interaction, duration constraints

Example of Sequence Diagrams

- Customer
- Controller
- Hardware

```
Coin(10)
Coffee (Additives(Milk=true,Sugar=1.))
FillWater ()
WaterOK ()
FillCoffee ()
CoffeeOK ()
HeatWater ()
Warm ()
CupOfCoffee (Additives(Milk=true,Sugar=1.))
```
Basic Syntax

Object1

Object2

Object3

message1

message2

Time

Basic Syntax (Cont.)

Object1

Object2

Object1

Object2

message1

message2

= 

message1

message2

(C) Ludovic Apvrille

UML for Embedded Systems - Fall 2012
Various Types of Message (UML 2.0)

- Synchronous message
- Return from synchronous call
- Asynchronous message
- Instantiation of an object
- Lost message
- Found message

Object Creation and Termination

```
Object1

CreateObject2

msg1 0
msg2 0
deleteObject2 0

Object2
```
Co-regions

- Inside coregions, events are not ordered

```
msg1 0
msg2 0
msg3 0
```

Interactions

- Reference to another SD
- Alternative scenarios
- Loop structures

```
x = 0;
x = x + 1;
```

Reference to another SD
Alternative scenarios
Loop structures
Temporal Operators

- Absolute time specification
- Relative time specification

Absolute time specification:
- Object1
- Object2
- msg1
- msg2
- msg3
- msg4

{10..20}

Relative time specification:

{10..15}

Management of Timers

- Setting a timer
- Resetting a Timer
- Timer expiration

Setting a timer:
- Object1
- Object2
- msg1
- msg2
- msg3
- msg4

{10}
Methodology

- Identify which objects play a role in the interaction
- Place most important objects / entities in the center
- Specify links among objects
- Starting with messages that initiate interactions
  - Attach each subsequent messages to the appropriate link
- Set the sequence number
- If more formality is needed, add pre and post conditions to each message

Common Errors

- Arrows going up in time
- Verbs as Objects
- Notation different from UML
  - Object name
  - Object shape
  - Lifeline
- Fuzzy diagrams
  - Keep them simple!
More Practically…

- Draw interactions between actors and the system

Analysis

- Use case of the system
- First scenarios
- Identification of main objects and classes
- Relevant scenarios of the system
Identifying Objects and Deducing Classes

- Suppose to know exactly what is an object and what is a class
  - Course on objects & classes provided in annex
- “Identifying objects”: the most difficult task!
  - This process cannot be automated
- Classification: gathering of similar instances
  - All should have the same type of data, even if their value is different
  - All should behave the same way

Example

- Objects
  - The car bought by Peter at Antibes the 1st of September
  - The car lent by Peter to John last summer
- -> Two instances of the class « Car »
  - The class « car » characterizes all cars
Correctly Defining Classes

- **Parnass principle**
  - The definition of a class should provide the environment with all necessary information to correctly use its instances, and nothing else
  - The implementation of an operation should be performed according to all information necessary to the execution of its task, and nothing else

- **Choosing identifiers**
  - Start with an uppercase letter (ex: Dice), singular form
  - Use the right level of abstraction
    - Municipality instead of City
  - Do avoid ambiguous names (ex: Room is ambiguous)

Identifying Objects and Classes

- **Difficult task**
  - Not only the objects obtained from a snapshot of a system
  - Experience is a key factor!

- **Using names in text**
  1. Identify all the most important names of the textual description of the system
  2. Then, gather all the names that are equivalent or that belong to the same concept
     - The application domain must be taken into account
  3. Delete all names belonging to the external environment of the system
  4. Inside each group
     - Select names that represent objects
     - Consider others as attributes, or remove them
Exercise: Identification of Objects/Classes

- Use the method provided on previous slide to the following textual description

A grocery desires to automate its stock list. It actually has several computer terminals at cashier’s desks level. These terminals make it possible to know the number of items bought by clients for each product. The return service is also equipped with a similar terminal for returned products. Another similar terminal is located at the unloading zone to deal with products delivered by various suppliers. At last, two other terminals are located at “butchery” and “fruits and vegetables” stands to record the rebates and loss due to deterioration of products.

Idea: For the identification of objects / classes, divide your sheet into four spaces:

1. Names
2. Groups
3. Entities external to the system
4. Final list of objects with attributes and operations
Analysis

- Use case of the system
- First scenarios
- Identification of main objects and classes
- Relevant scenarios of the system

Refinement of Previously Performed Scenarios

- Identified objects and classes should be integrated into previously performed sequence diagrams
- The whole system interactions should not be described
- Relevant scenarios
  - Nominal case
  - Secondary cases
    - For example, buy an item with a loan
  - Error cases
    - The ATM crashes, etc.
How to Refine First scenarios

Retake the interactions between the actors and the system, and add interactions internal to the system

Make Relevant Scenarios for:

- The example of a client in a shop
- The example of the grocery

(C) Ludovic Apvrille  UML for Embedded Systems - Fall 2012