ISS - June 25, 2012

1. Please mention all kinds of diagrams supported by the UML, the view associated and the modeling concepts (including relationships) that can be used in each diagram. 2p

- **the static (architectural) view** is described by means of a **class diagram**. Concepts that can be used in a class diagram are: packages (system, subsystems), classes, interfaces, association classes. The relationships are: dependencies (import, access) inheritance, implement, generalization, association, instantiation (for generic (parametrized) classes). Classes may include attributes and operations.

- **the functional view**, described by means of **use case diagrams**. The concepts used are: actors and use cases and the relationships: generalizations, associations that can be stereotyped with uses, extend. Also, extensions points may be included in use cases.

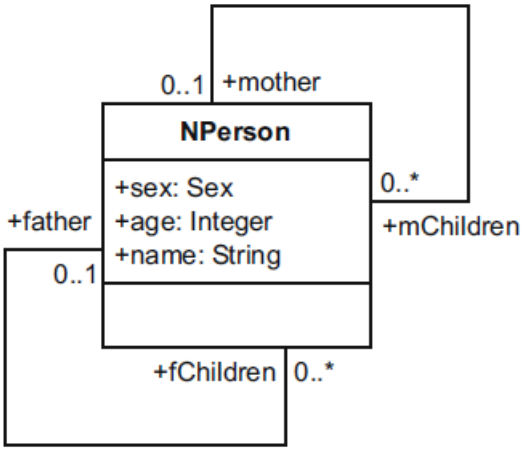
- **the behavioral view**, that can be described by means of **sequence** or **collaboration diagrams** or by means of a **state-transition diagram**s or **activity diagrams**. The concepts used in sequence and collaboration diagrams are: objects (instances), actor instances, messages that have attached an info about the message order, object constructors and destructors. In state transition diagrams there are states and pseudostates, branches, transitions, triggers (stimulus), guards, activities, swimlanes (in case of concurrent states). In activity diagrams, there are activities, branches, transitions, join and fork in case of concurrent and swimlanes.

- **the deployment view** described by using **deployment diagrams**. The concepts are nodes (of a distributed application - nodes having attached appropriate information), association describing the communication infrastructure and associated info about the communication protocol.

- **the component view** described by means of **component diagram**. The concepts used are components having appropriate ports that enable components to be connected. The components have appropriate info like: the version, the standard a.s.o.

2. Each person is characterized by name, sex and age and by its parents, that in some cases are unknown. Using UML, please represent by means of a class diagram the architecture of a person - family model, enabling to compute in an easy manner the mother, father, sisters

and brothers of a person. Using the OCL, please specify the observers computing the above mentioned info. Also, please specify an invariant constraining the sex of father at male and of mother at female and requiring that parents are at least 16 years older than their children. 3p




```
context NPerson
inv parentsSexP1:
(self.mother ->size() = 1 implies Sex:: Female = self.mother.sex) and
(self.father ->size() = 1 implies Sex:: Male = self.father.sex)

context NPerson
inv parentsAge:
self.mChildren ->reject (p | self.age - p.age >= 16)->isEmpty () and
self.fChildren ->reject (p | self.age - p.age >= 16)->isEmpty ()
```

```
context NPerson

    def bANDs:
        let bANDs: Set(NPerson) =
        let mc:Set(NPerson) = self.mother.m_children->excluding(self)
        let fc:Set(NPerson) = self.father.f_children->excluding(self)
           if self.mother->size = 1
               then if self.father->size = 1
                           then mc->union(fc)
                           else mc
                    endif
               else if self.father->size = 1
                             then fc
                             else oclEmpty(Set(NPerson))
                    endif
           endif
```

```
let brothers:Set(NPerson) = bANDs->reject(p | p.sex = Sex::female)
let sisters:Set(NPerson)  = bANDs->select(p | p.sex = Sex::female)
```

3. By means of snapshots, please represent different cases complying or breaking the above mentioned invariant. 2p

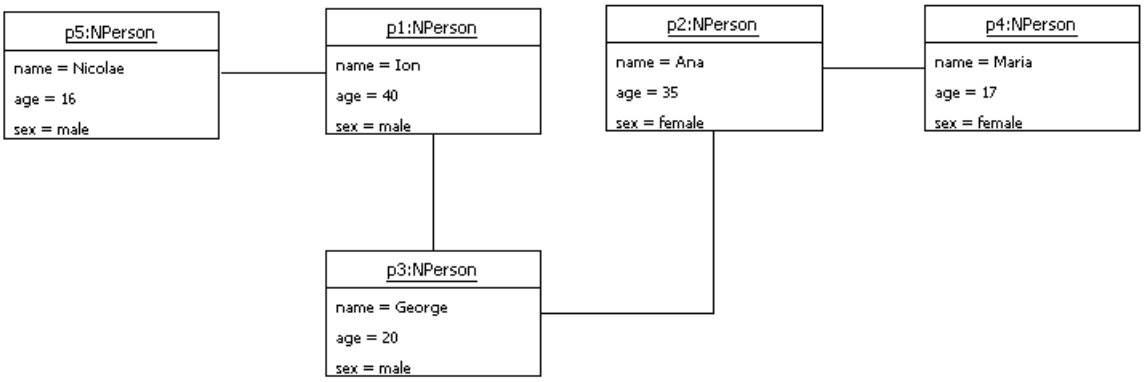


Figure 1 - complying snapshot

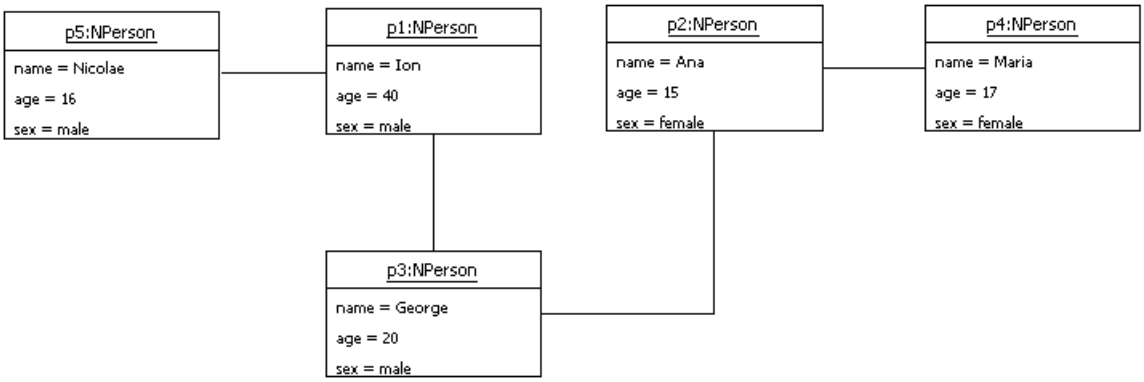


Figure 2 - breaking snapshot

4. Describe the behavior specified in Figure 1, and explain if some information is missing or not. In case of an affirmative answer, include the missing information. Name the UML concepts used in this diagram and explain the difference between the two kind of states. 2p

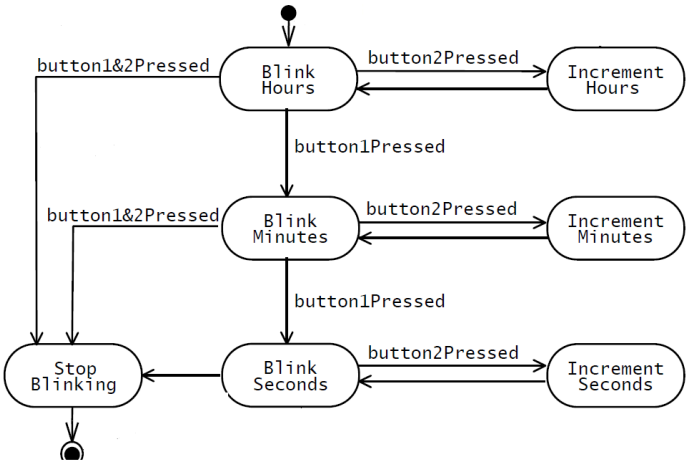


Figure 3 - An UML state-transition diagram

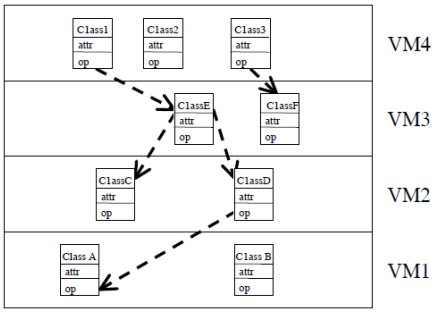
The transition from Blink Seconds to Stop Blinking can be done when pressing the button1&2Pressed. The transition from Increment Hours to Blink Hours is realized when button2Released. The modeling concepts are: states, pseudostates and transitions, and

1. Please characterize the Closed Architecture and Open Architecture Styles 1.5p

Closed Architecture (Opaque Layering)

- Each virtual machine can only call operations from the layer below

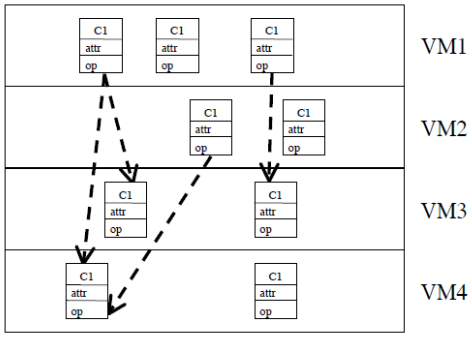
Design goals:
Maintainability, flexibility.



Open Architecture (Transparent Layering)

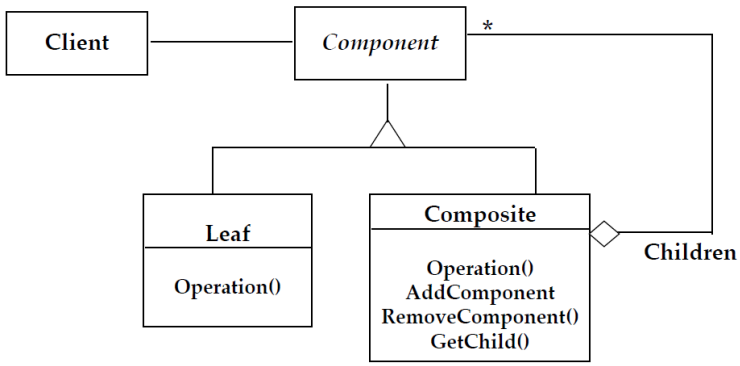
- Each virtual machine can call operations from any layer below

Design goal:
Runtime efficiency



2. Please describe and exemplify the Composite Pattern 2p

- Models tree structures that represent part-whole hierarchies with arbitrary depth and width
- The Composite Pattern lets a client treat individual objects and compositions of these objects uniformly



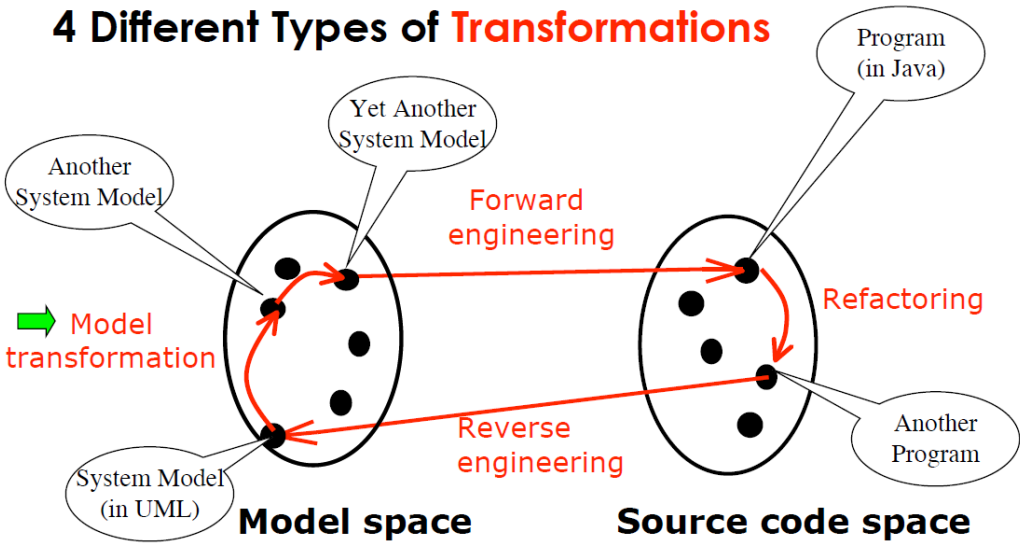
3. Is it possible to completely test any nontrivial module or system? Please justify your statements. 2p

Observations

- It is impossible to completely test any nontrivial module or system
  - Practical limitations: Complete testing is prohibitive in time and cost
  - Theoretical limitations: e.g. Halting problem
- "Testing can only show the presence of bugs, not their absence" (Dijkstra).
- Testing is not for free

=> Define your goals and priorities

4. Please shortly describe the types of transformations that can be realized between models, code, models to code and vice-versa and explain their usefulness. 2p



5. By means of a class diagram, please represent the work categories in a software project and their relationships 1.5p