Model-Driven Software Engineering (MDSE)

Bahman Zamani, Ph.D.
bahmanzamani.com

Department of Computer Engineering
Faculty of Engineering
University of Isfahan

Slides are prepared by Dr. Abdelwahab Hamou-Lhadj and are used by permission

What is a Meta-metamodel?

- A metamodel describes information about models
- A meta-metamodel describes information about metamodels
- Metamodels that are defined using the same meta-metamodel
  - Can exchange information
  - Can be used by the same CASE tools that understand the meta-metamodel

2

What is MOF?

- MOF stands for Meta Object Facility
  - enables meta-metamodeling of UML level metamodels
- It defines a small set of concepts (such as package, class, method, attribute…) that
  - allow one to define and manipulate **models** of metadata (data about data)
  - are described using a subset UML notation

3

OMG 4-Layer Architecture

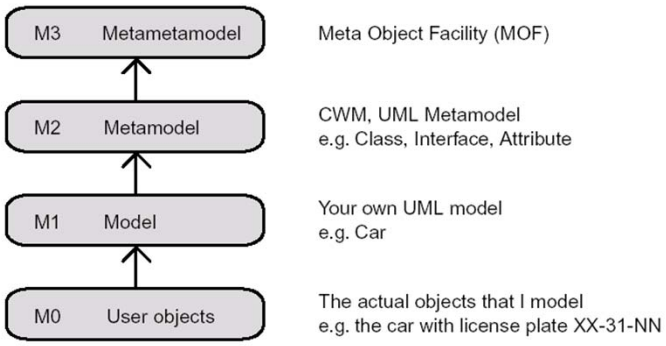


Figure 1 *OMG 4-layer architecture*

4

MOF Key Abstract Classes (Cont.)

- **ModelElement** common base Class of all M3-level Classes. Every ModelElement has a name
- **Namespace** base Class for all M3-level Classes that need to act as containers
- **GeneralizableElement** base Class for all M3-level Classes that support generalization (i.e. inheritance)
- **TypedElement** base Class for M3-level Classes such as Attribute, Parameter, and Constant whose definition requires a type specification
- **Classifier** base Class for all M3-level Classes that (notionally) define types. Examples of Classifier include Class and DataType

5

The MOF Model: Main Concrete Classes

- The key concrete classes (or meta-metaclasses) of MOF are as follows:
  - Class
  - Association
  - Exception (for defining abnormal behaviours)
  - Attribute
  - Constant
  - Constraint

6

The MOF Model: Key associations

- Contains: relates a ModelElement to the Namespace that contains it
- Generalizes: relates a GeneralizableElement to its ancestors (superclass and subclass)
- IsOfType: relates a TypedElement to the Classifier that defines its type
  - An object is an instance of a class
- DependsOn : relates a ModelElement to others that its definition depends on
  - E.g. a package depends on another package

7

The UML Metamodel

- A UML model is an instance of the UML metamodel
- The UML metamodel
  - Describes the UML model elements
  - Is defined using a subset of UML
  - Is organized in the form of packages

8

UML Metamodel (cont.)

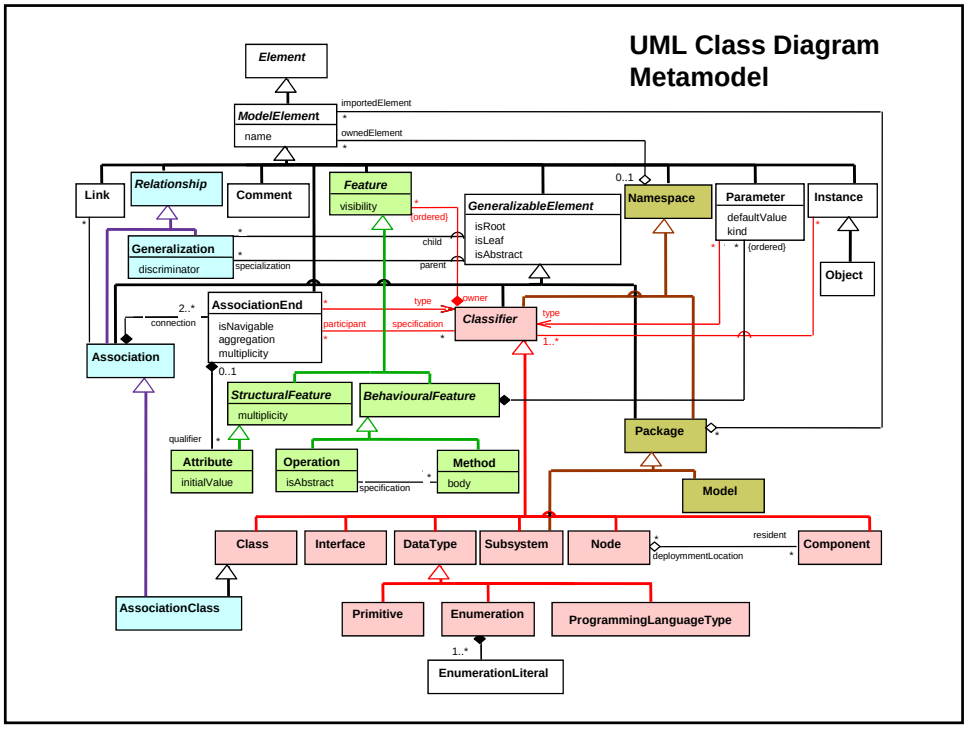
- UML metamodel is based on MOF
- Every model element in UML metamodel maps to an element in MOF
- There is a metamodel to every UML diagram:
  - class diagrams
  - use case diagrams
  - etc.
- UML extension mechanisms have also a metamodel that defines them

9

The UML Metamodel (cont.)

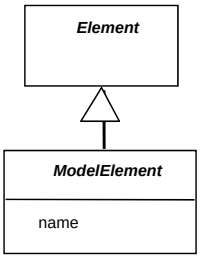
- UML metamodel is defined according to the following concepts:
  - Abstract Syntax: The metamodel of UML is described using UML class diagrams
  - Well-formedness rules: Well-formedness rules are used to express constraints on the model elements
    -E.g. a class cannot have two names
  - Semantics: describes using the natural language the semantics of the model elements

10



Model Elements

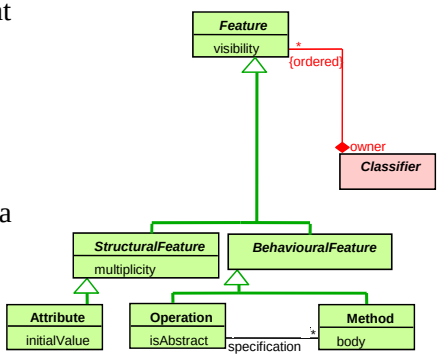
- An element is an atomic constituent of a model.
- Element is the top metaclass in the metaclass hierarchy
- A model element is a named entity in a Model
- It is the base for all modeling metaclasses in UML
  - All other modeling metaclasses are either direct or indirect subclasses of ModelElement



12

Features

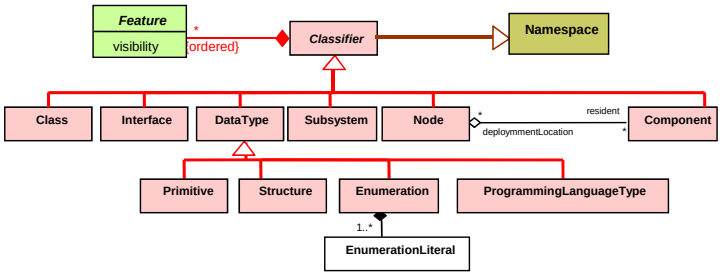
- Feature is an abstract class that declares a behavioral or structural property of
  - an instance of a Classifier
  - the Classifier itself
- A behavioral feature refers to a dynamic feature of a model element
  - E.g. operation or method
- A structural feature refers to a static feature of a model element
  - E.g. attribute



13

Classifier

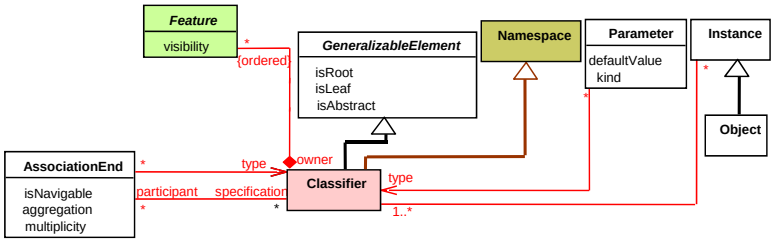
- A classifier is an element that describes behavioral and structural features
  - E.g. class, data type, interface, component
- Classifier is an abstract class that
  - declares a collection of Features, such as Attributes, Methods…
  - has a name, which is unique in the Namespace enclosing the Classifier



14

Classifier (cont.)

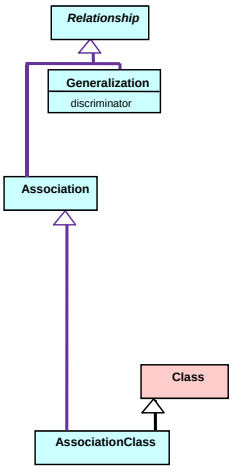
- A classifier defines a namespace and is a generalizable element
- Can have
  - association ends
  - parameters
  - instances



15

Relationships

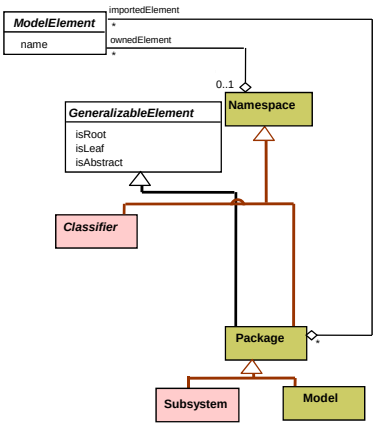
- A relationship is a connection among model elements
- UML defines several relationships such as:
  - Association
  - Generalization
- UML defines other types of relationships that are not shown in this diagram, such as:
  - Dependency
  - Flow



16

Namespace

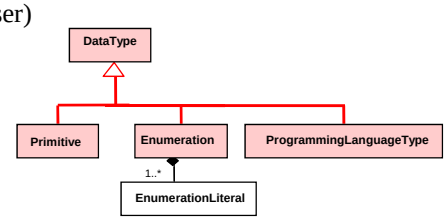
- A namespace is a part of a model that contains a set of other model elements
  - E.g. Associations and Classifiers
  - the name of an owned model element is unique within the namespace
- Namespace is an abstract metaclass and it subclasses are
  - Classifier
  - Package



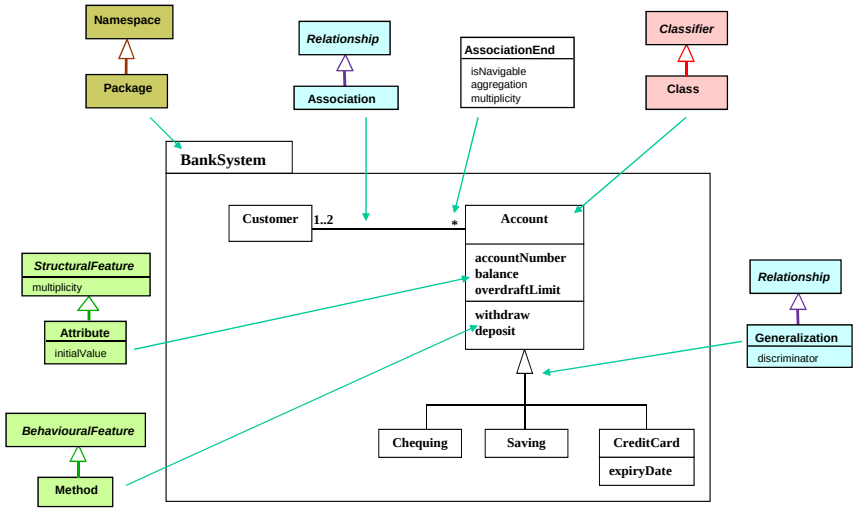
17

Data Types

- UML Data types include
  - primitive built-in types (such as integer and string)
  - definable enumeration types (such as Boolean whose literals are false and true)
- Programming languages data types
  - are specified according to the semantics of a particular programming language
  - are not portable among languages (except by agreement among the languages)
  - do not map into other UML classifiers
- Enumerations are a user-defined data types whose instances are literals (specified by the user)



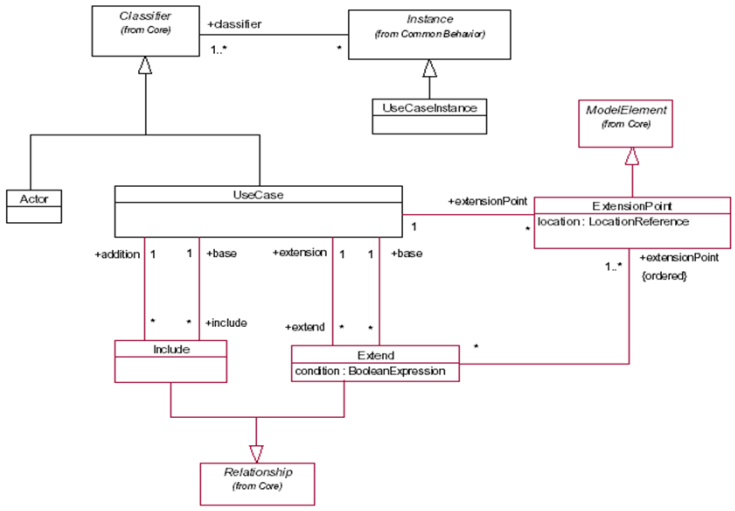
18



Mapping of UML Models to Metamodel Elements (Example)

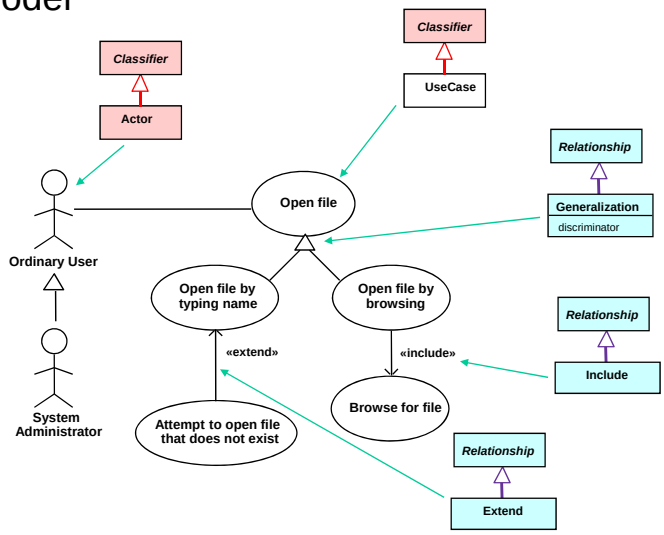
19

Use Case Diagrams Metamodel



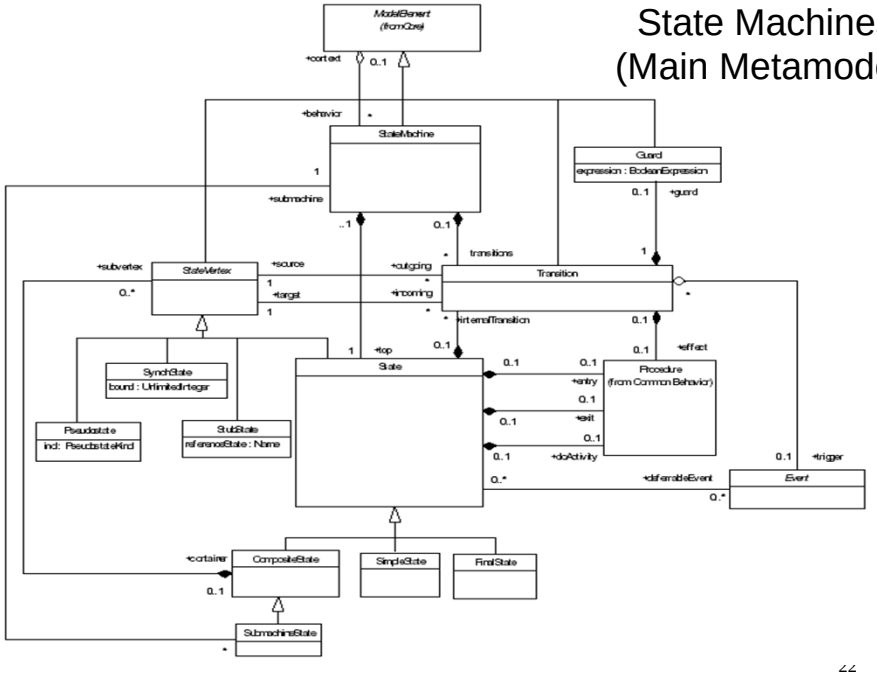
20

Mapping Use Cases Model to Metamodel



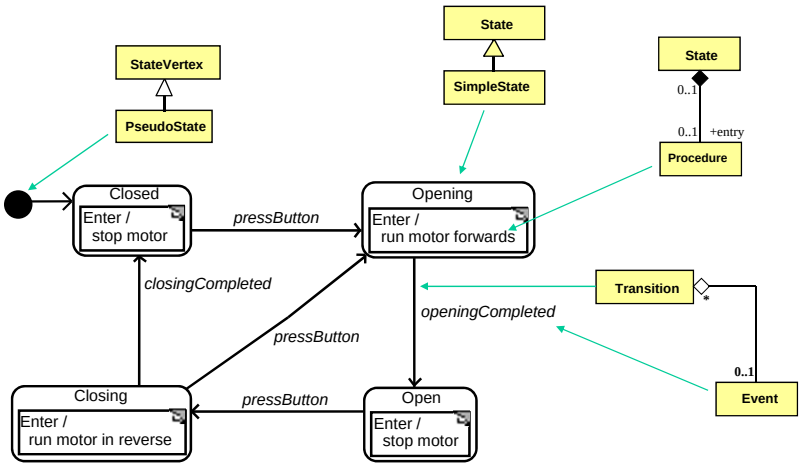
21

State Machines (Main Metamodel)



22

Mapping State Machines to Metamodel



23

UML Extension Mechanisms

- Although UML is very well-defined, there are situations in which it needs to be customized to specific problem domains
- UML extension mechanisms are used to extend UML by:
  - adding new model elements,
  - creating new properties,
  - and specifying new semantics
- There are three extension mechanisms:
  - stereotypes, tagged values, constraints and notes

24

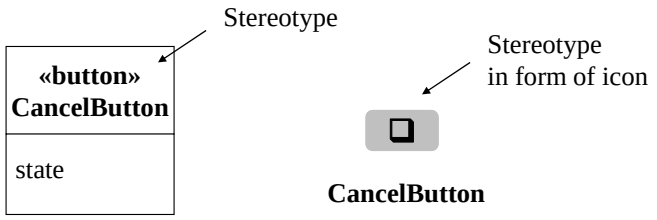
Stereotypes

- Stereotypes are used to extend UML to create new model elements that can be used in specific domains
- E.g. when modeling an elevator control system, we may need to represent some classes, states etc. as
  - «hardware»
  - «software»
- Stereotypes should always be applied in a consistent way

25

Stereotypes (cont.)

- Ways of representing a stereotype:
  - Place the name of the stereotype above the name of an existing UML element (if any)
    - The name of the stereotype needs to be between «» (e.g. «node»)
    - Don't use double '<' or '>' symbols, there are special characters called open and close guillemets
  - Create new icons



26

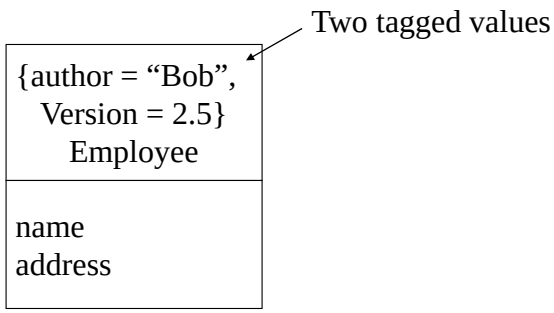
Tagged Values

- Tagged values
  - Define additional properties for any kind of model elements
  - Can be defined for existing model elements and for stereotypes
  - Are shown as a tag-value pair where the tag represent the property and the value represent the value of the property
- Tagged values can be useful for adding properties about
  - code generation
  - version control
  - configuration management
  - authorship
  - etc.

27

Tagged Values (cont.)

- A tagged value is shown as a string that is enclosed by brackets {} and which consists of:
  - the tag, a separator (the symbol =), and a value



28

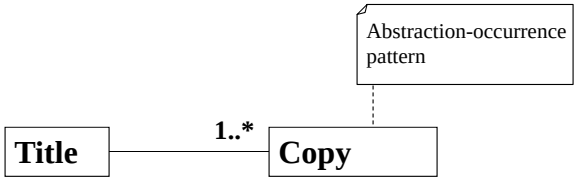
Constraints

- Constraints are used to extend the semantics of UML by adding new rules, or modifying existing ones.
- Constraints can also be used to specify conditions that must be held true at all times for the elements of a model.
- Constraints can be represented using the natural language or OCL

29

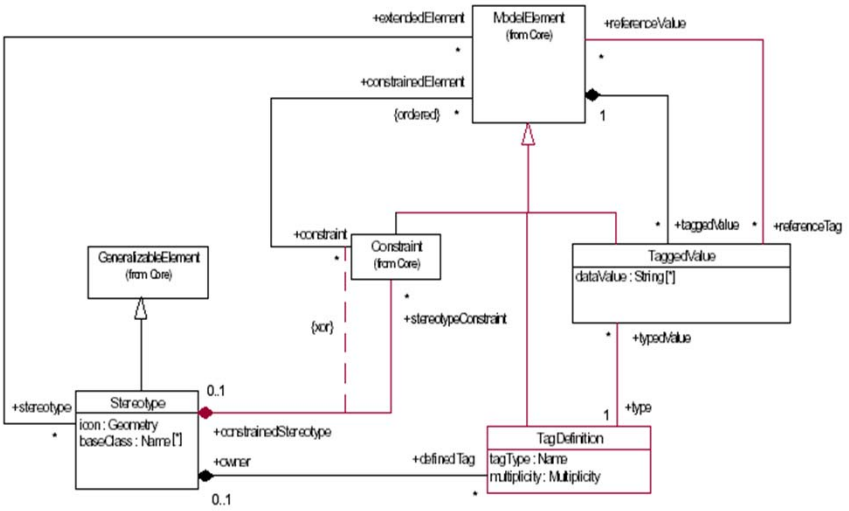
Comments

- Comments are used to help clarify the models that are being created
  - e.g. comments may be used for explaining the rationale behind some design decisions
- A comment is shown as a text string within a note icon.
- A note icon can also contain an OCL expression



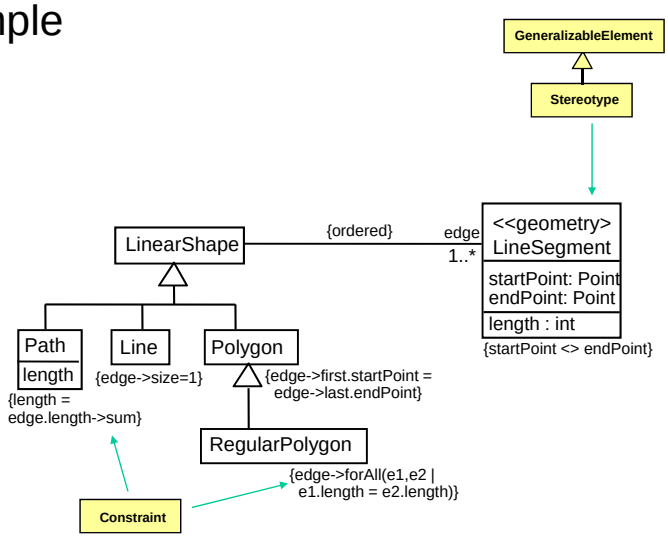
30

Extension Mechanisms Metamodel



31

Example



32

UML Profiles

- UML Profiles provide an extension mechanism for building UML models for particular domains
  - e.g. real-time systems, web development, etc…
- A profile consists of a package that contains one or more related extension mechanisms (such as stereotypes, tagged values and constraints)
  - that are applied to UML model elements
- Profiles do not extend the UML metamodel. They are also called *the UML light-weight extension mechanism*

33

UML Profiles (cont.)

- A UML profile is a specification that does one or more of the following:
  - Identifies a subset of the UML metamodel (which may be the entire UML metamodel)
  - Specifies stereotypes and/or tagged values
  - Specifies well-formedness rules beyond those that already exist
  - Specifies semantics expressed in natural language
- We will cover UML profiles in the next class

34

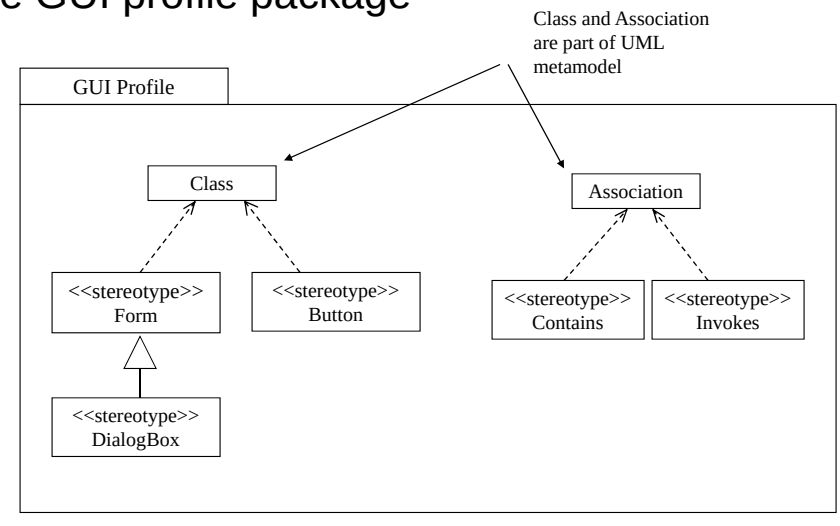
Example of a profile

inspired by the research report of Cabot et al. (2003)

- We would like to create a UML profile for representing basic GUI components.
- We suppose that our GUI contains the following components:
  - Forms (which can also be dialog boxes)
  - Buttons
- Constraints: (in practice, we need to be more precise)
  - A form can invoke a dialog box
  - A form as well as a dialog box can contain buttons

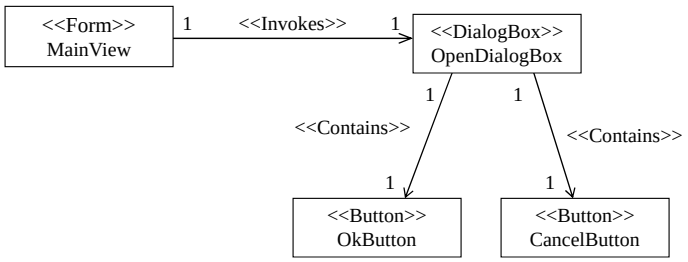
35

The GUI profile package



36

Instance Diagram of the GUI Profile



37