# UML Extension Mechanisms

By Jasmine Farhad

j.farhad@cs.ucl.ac.uk



## Objectives

- Briefly describe the UML
- Introduce the UML Extension Mechanisms:
  - Specifications
  - Common Divisions
  - Adornments
  - Extensibility Mechanism(we will focus on this)
- Guidelines for use



## What is UML?

- The Unified Modelling Language
- General purpose, tool supported, standardized modelling language for writing software blueprints
- Used for specifying, visualizing, constructing and documenting the artifacts of a system intensive process
- Promotes a use case driven, architecture-centric, iterative and incremental process that is object oriented and component based
- Applicable to different types of systems, domains, methods and processes



## Is the UML enough?

- Well-defined language BUT.....
- Impossible to express ALL possible nuances of ALL models across ALL domains across ALL time, SO.....
- Need to extend it to customize/tailor it to communicate YOUR intent
- Must be done in a CONTROLLED way
- Solution: the UML Extension Mechanisms



## UML Extension Mechanisms

- There are four common mechanisms that can be used throughout the language:
- Specifications
- Common Divisions
- Adornments
- Extensibility Mechanism



## Specifications

- Class implies a full set of attributes, operations, full signatures and behaviours
- Specifications can be included in the class, or specified separately

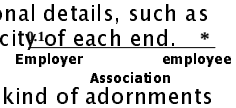
## Common Divisions

- Abstraction vs. manifestation
  - Class vs. object
  - Most UML building blocks have this kind of class/object distinction; e.g., use case, use case instance etc.
- Interface vs. implementation
  - An interface declares a contract, and an implementation represents one concrete realization of that contract

7

## Adornments

- Textual/graphical items added to the basic notation of an element
- Used for visualizing details from the element's specification
- Example: The basic notation of association is a line, but this could be adorned with additional details, such as the role and multiplicity of each end.

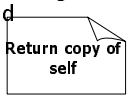



- The most important kind of adornments are *notes*

8

## Notes

- Graphical symbols used for adding constraints or comments to an element/group of elements
- Attach information to a model such as observations, requirements and explanations
- No semantic impact, ie its content does not change the meaning of the model to which it is attached
- Example:




9

## Extensibility Mechanisms

- Allows you to extend the language by adding new building blocks, creating new properties and specifying new semantics. Includes:
- Stereotypes
- Tagged values
- Constraints

10

## Stereotypes

- Extend the vocabulary of the UML by creating new model elements derived from existing ones but that have *specific* properties suitable for your domain/problem
- Each stereotype defines a set of properties that are received by elements of that stereotype

11

## Stereotypes(cont'd)

- Example: If you are modelling a network you might want to have symbols for routers and hubs.
- Can make use of stereotyped nodes so that they appear as primitive building blocks
- Another example: In Java, you sometimes have to model classes such as exceptions
- Only want them to be thrown and caught
- Can make them first class citizens in your model, ie treating them like basic building blocks, by marking them with a suitable stereotype

12

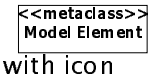
## Stereotypes(cont'd)

- Graphically, a stereotype is rendered as a name enclosed by guillemots and placed above the name of another element (eg, <<name>>)
- Alternatively, you can render the stereotyped element by using a new icon associated with that stereotype

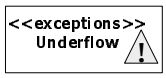
13

## Stereotypes(cont'd)

- Named stereotype




- Named stereotype with icon




- Stereotyped element as icon


14

## Tagged values

- Properties for specifying key-value pairs of model elements, where keywords are attributes
- Extend the properties of a UML building block, allowing you to create new information in that elements specification
- Can be defined for existing elements of the UML
- You can also define tags that apply to individual stereotypes . Then, everything with that stereotype will have that tagged value

15

## Tagged values(cont'd)

- Not the same as a class attribute
- Can think of a tagged value as a metadata, since its value applies to the element itself and not its instances
- Example: In the release team of a project that is responsible for assembling, testing and deploying releases, you might want to keep track of the version number and test results for each major subsystem
- Can use tagged values to add this information to your models

16

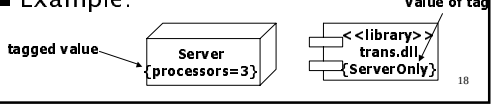
## Tagged values(cont'd)

- Another example: One of the most common uses of a tagged value is to specify properties that are relevant to code generation or configuration management.
- So you can use tagged values to specify the programming language to which you map a particular class
- Similarly, you can use them to specify the author and version of a component

17

## Tagged values(cont'd)

- Graphically, a tagged value is rendered as a string enclosed by brackets and placed below the name of another element
- The string consists of a name (the tag), a separator (the symbol =), and a value (of the tag)
- Example:




18

## Constraints

- Properties for specifying semantics or conditions that must be maintained as true for model elements
- Extend the semantics of a UML building block, allowing you to add new rules, or modify existing ones

19

## Constraints(cont'd)

- Example: If you are modelling hard real-time systems, you might want to adorn your models with information about time budgets and dead-lines
- Can use constraints to capture these timing requirements

20

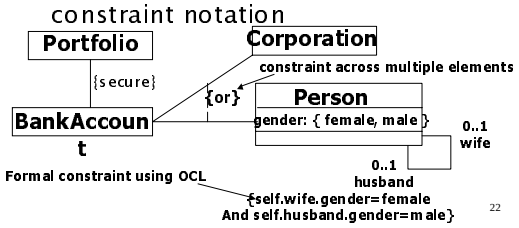
## Constraints(cont'd)

- Graphically, a constraint is rendered as a string enclosed by brackets and placed near the associated element(s) or connected to that element(s) by dependency relationships
- This notation is also used as an adornment to the basic notation of an element to visualize parts of an element's specification that have no graphical cue
- For example, some properties of associations(order and changeability) are rendered using constraint notation

21

## Constraints

- For example, some properties of associations(order and changeability) are rendered using constraint notation




22

## Guidelines for use

- Keep in mind that an extension deviates from the standard form of the UML and may therefore lead to interoperability problems
- So when you find yourself in the situation where you have to colour outside the lines you should do so in controlled ways
- Carefully weigh benefits and costs before using the extension mechanisms

23

## Summary

- Modelling is all about communication.
- UML already gives you all the tools you need to visualize, specify, construct and document the artifacts of a software-intensive system
- However, sometimes you might have to bend/extend the UML in order to tailor it to the specific needs of your domain and your development culture
- That's where the UML extension mechanisms come in!

24

## Summary(cont'd)

- Four common mechanisms:
- Specifications
- Common divisions
- Adornments
  - Notes
- Extensibility mechanisms
  - Stereotypes
  - Tagged values
  - Constraints

25

## References


- *The Unified Modelling Language User Guide* by Rumbaugh, Jacobson and Booch [Addison-Wesley]
- *The Unified Modelling Language Reference Manual* by Rumbaugh, Jacobson and Booch [Addison-Wesley]
- *Using UML, Software Engineering with Objects and Components* by Perdita Stevens and Rob Pooley


26