UNIFIED MODELING LANGUAGE

Generic Mechanisms

Radovan Červenka, October 1998 (version 0.04)

Context

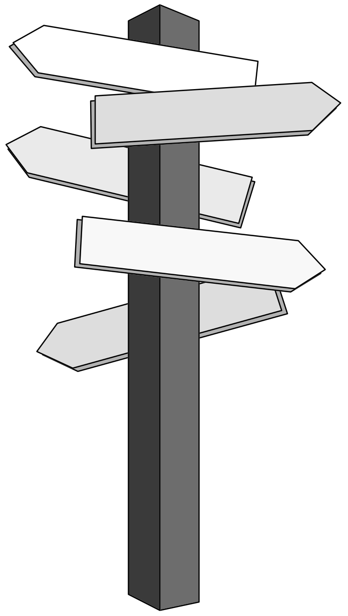
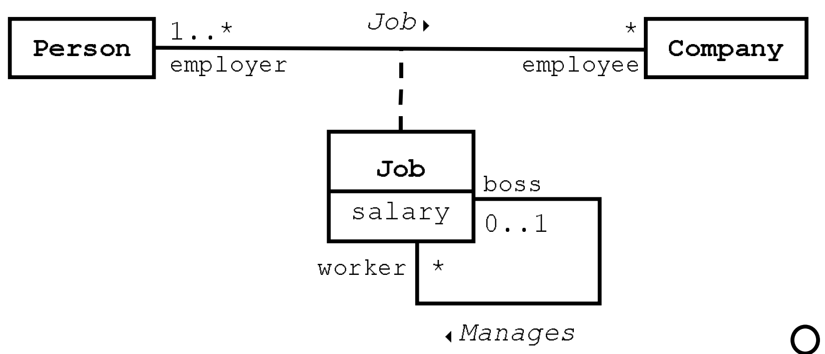
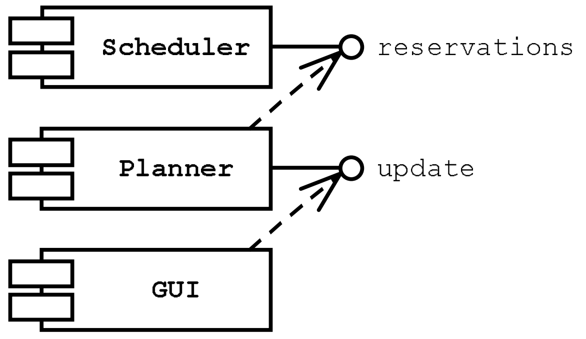
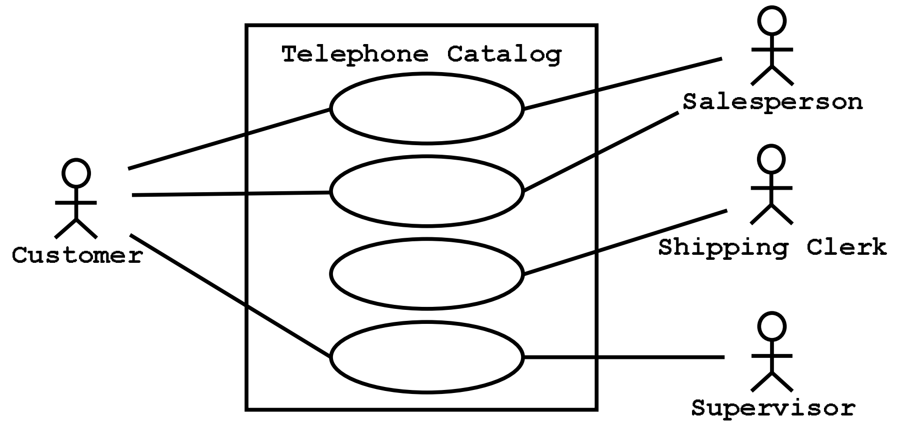


Diagram Elements

→ generic notation mechanisms used in various ways in other parts of the language







Graphs, Drawing Paths, Hyperlinks, ...

Graphs and their Contents

- UML diagrams are mainly graphs
- the information is mostly in the topology
- graphical constructs: icons, 2-d symbols, paths and strings

Drawing Paths

→ a series of line segments whose endpoints coincide

Invisible Hyperlinks and the Role of Tools

- arrangement of model information into the hyperdocument
  - ⇒dynamics notation defined for a particular tool

∗ out of scope of UML

Background Information

- suppression of some element information
- textual or tabular format of some information

∗ their format is out of scope of UML

String, Name and Label

String

→ a sequence of characters (of any character set)

Name

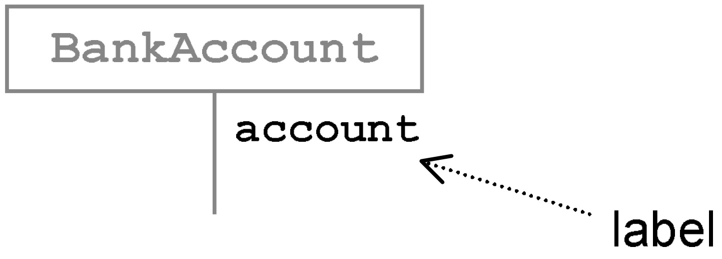
→ a string uniquely identifying a model element

- may be linked together by delimiters into *pathname*

`BankAccount`, `controller`, `long_underscored_name`,
`MathPack::Matrices::BandedMatrix.dimension`

Label

→ a string that is attached to a graphics symbol



Keyword and Expression

Keyword

→ reserved name

- usually used to distinguish element types which don’t have their own graphical representation

«*keyword*»

Expression

→ linguistic formulas that yield values when evaluated at run-time

- language-dependent ⇒ in UML treated as string

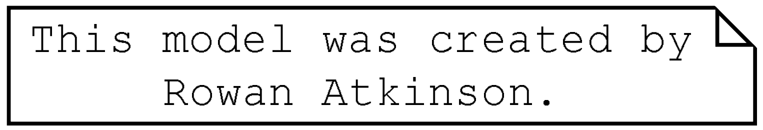
```
BankAcount
BankAccount * (*) (Person*, int)
array [1..20] of range(-1.0 .. 1.0) of Real
[i > j and self.size > i]
```

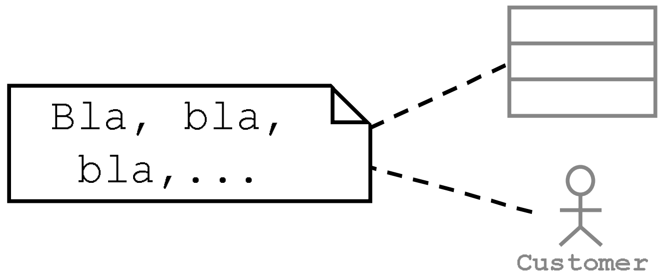
Note and Type-Instance Correspondence

Note

→ textual information attached to some semantic element

- of various kind, e.g. constraint, comment, method body, tagged value





Type-Instance Correspondence

- dual form of modeling elements: type and instance
  - Class-Object, Association-Link, Parameter-Value, Operation-Call, etc.
- notation: the same geometrical symbol and name strings of instance elements are underlined

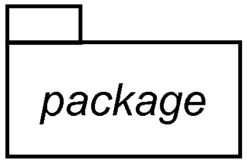
Model Management: Packages

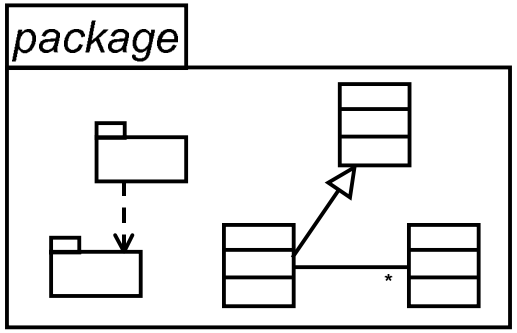
Package

→ grouping of model elements; it is itself a model element

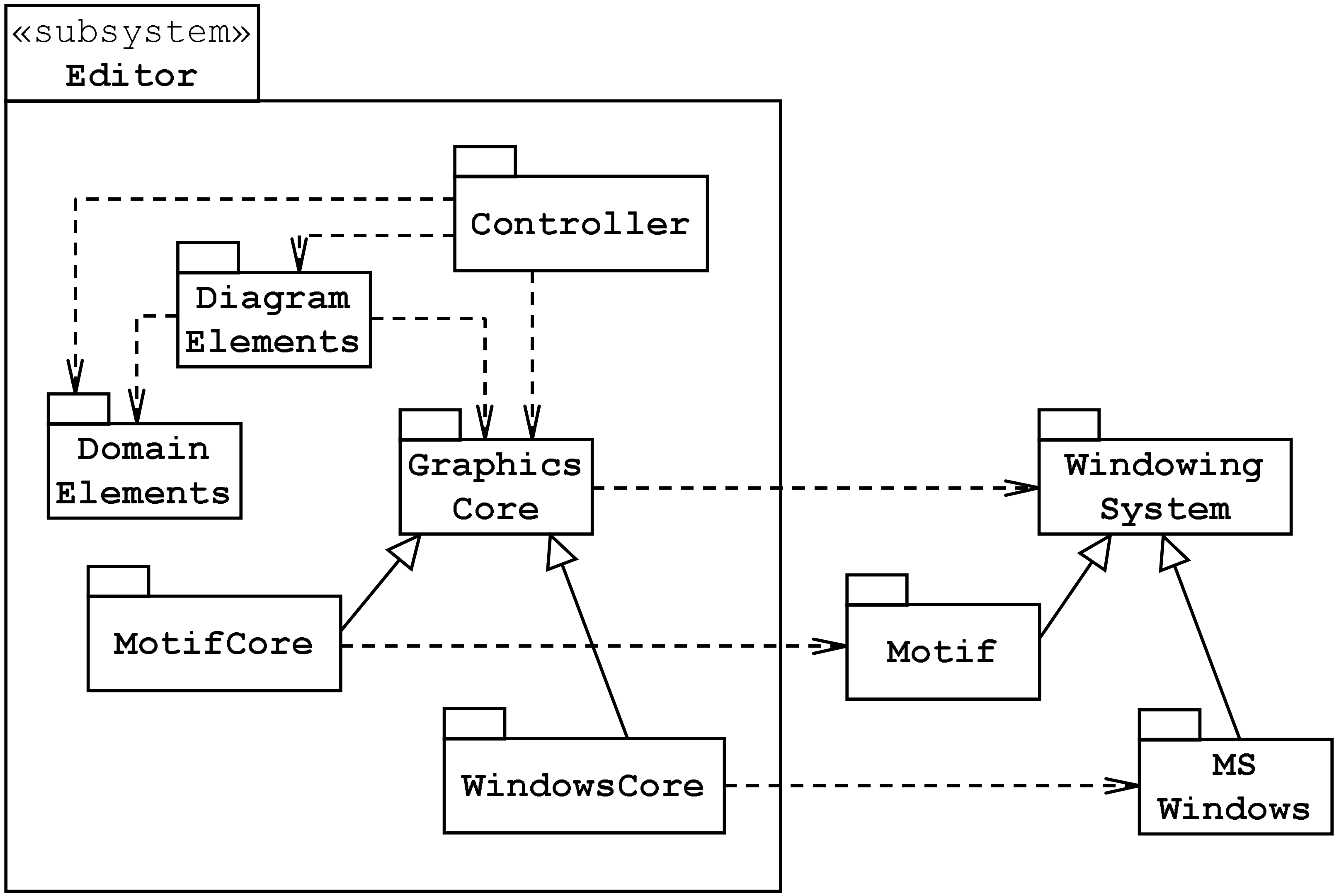
- owns or references other model elements
  - ⇒packages themselves may be nested within other packages
- used in
  - Use Case View ⇒ functional decomposition
  - Static Structure View ⇒ logical high-level architecture
  - Component View ⇒ modular decomposition
  - Deployment View ⇒ physical HW decomposition

∗ high-level static structure diagrams containing only packages are called *Package Diagrams*





Example of the Package Diagram

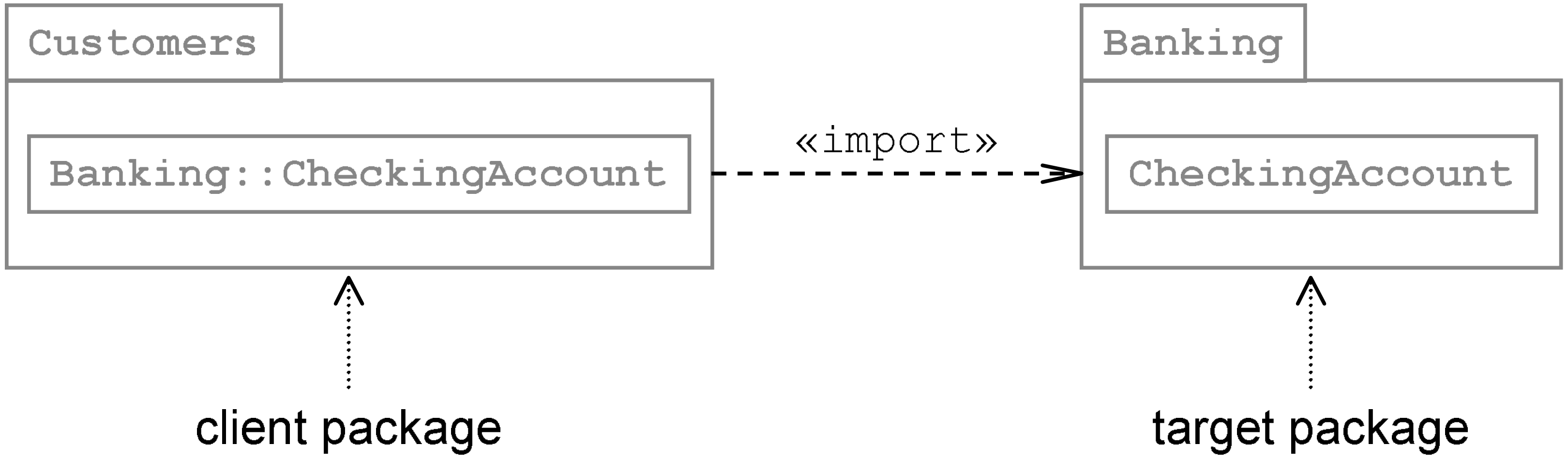


Importing a Package

→ the contents of the *target package* may be referenced by the *client package* or packages recursively embedded within it

- notation: dependency relationship with stereotype `«import»`
- full element identification:

package name`::` … `::`*package name*`::`*element name*



Summary

- Graphs and their Contents
- Drawing Paths
- Invisible Hyperlinks and the Role of Tools
- Background Info
- String
- Name
- Label
- Keyword
- Expression
- Note
- Type-Instance Correspondence
- Package
- Importing a Package