STI · INNSBRUCK

Modeling elements

Lecture 2

11.10.2010

Course sylabus

STI · INNSBRUCK

- **Introduction and motivation**. Definitions, principles, properties and characteristics. Typical usage scenarios. Model engineering.
- **Modeling elements. Core relationship types and their properties.**

NEXT class: Nov 15, 2010

- ER modeling. Basic modeling elements, foundational theory, examples. Differences to other modeling paradigms.
- OO modeling. Basic modeling elements, UML. Differences to other modeling paradigms.

Summary of the previous lecture

STI · INNSBRUCK

- **Introduction and motivation**.

- Conceptual modeling.
- Data and process models.
- General principles.
- Model engineering.
- Modeling languages.
- Use cases.

Today‘s lecture

STI · INNSBRUCK

- **Modeling elements**.

- General principles.
- Basic elements and their properties: entities, attributes, associations, specialization/ generalization.
- Relationships and their characteristics.
- Large example: modeling the wine domain

Basic principles

STI · INNSBRUCK

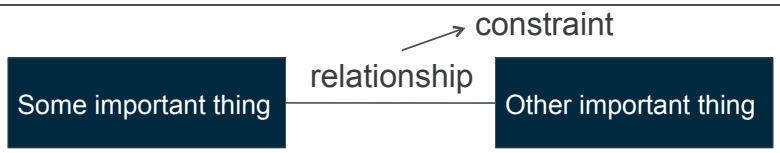
- Abstraction
  - Ignoring certain aspects in order to simplify the handling of something or to better understand other aspects.
  - The modeler decides what it is important or not and then chooses a representation that is more tractable than the original.
  - A representation of something can not be greater than that something.

More basic principles

STI · INNSBRUCK

- Models should be divisible.
- Components/modules/chunks should be highly cohesive.
- Use informative names.
- Modules should have low coupling.

Basic elements



The node is easy to find in the domain
Candidates for nodes:
- steps in flowcharts.
- functions in data flow diagrams.
- states in state machine diagrams.
- things or entities in ER models, knowledge bases.
- classes in OO modes.

Relationships/associations/relations/properties/attributes hold between instances of the entities.

Constraints/axioms/restrictions/rules further specify the nature of relationships.

Some definitions

- Cohesion:
  - In computer programming, cohesion is a measure of how **strongly-related and focused** the various responsibilities of a software module are. Modules with **high cohesion** tend to be **preferable** because high cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability whereas low cohesion is associated with undesirable traits such as being difficult to maintain, difficult to test, difficult to reuse, and even difficult to understand.
  - Cohesion vs. Coupling
  - Cohesion → loose coupling

Entities

STI · INNSBRUCK

- An entity represents a set of instances.
- An entity should be highly cohesive, precisely nameable, relevant.
- An entity should have a strong identity.

How to find entities

STI · INNSBRUCK

- Interview: talk to subject matter experts.
- Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
- Observation and reflection.

How to find entities

STI · INNSBRUCK

- Typical candidates for entitites: NOUNS.
  - But: actors of use cases do not necessarily correspond to entities.
- Other terms as well
  - Verbs: an association which starts to take on attributes and associations of its own turns into an entity: „Officer arrests suspect“.
  - Verbs: events: „Illness episode“.
  - Passive form: re-formulate in active form.
  - No pronouns.

Cohesion and identity

STI · INNSBRUCK

- An entity should represent one thing, all of that thing and nothing but that thing.
- You can prove cohesion by
  - Giving the entity a representative name.
  - Noun (+ adjective, sometimes however also captured as attribute value).
    - Blackmail victim, robbery victim.
    - Blue car, red car.
    - **Cars** is not cohesive.
- Avoid ambiguous terms.
  - Manager, handler, processor, list, information, item, data…
- Instances vs classes of things.
- Identity: entities change values, but are still the same entity
  - Child/Adult: age

Relevance

STI · INNSBRUCK

- Going out too far vs. going down too far.
- Investigate homonyms and synonyms.
  - Homonym: two words are homonyms if they are pronounced or spelled the same way but have different meanings (bank of the river and financial institution)
  - Synonym: two words that can be interchanged in a context are said to be synonymous relative to that context ("buy" and "purchase")
- Questions
  - Can medicine and drug be considered synonyms?
  - Do they have the same properties/characteristics/attributes/ relationships?
  - Do they have a critical mass of commonalities?

Characterizing entities

STI · INNSBRUCK

- Two types of principal characteristics:
  - Measurable properties: attributes.
  - Inter-entity connections: associations.
  - Use associations to model something with an identity.
  - Arrest details as attribute of the suspect vs. Arrest as an association.
    - Do we measure degrees of arrestedness or do we want to be able to distinguish between arrests?
  - Color of an image as attribute vs. entity.

Attributes

STI · INNSBRUCK

- An attribute is a measurable property of an entity.
  - Scalar values: choice from a range of possibilities.
  - An attribute is NOT a data structure. It is not complicated to measure.
- Value of attributes: integer, real numbers, enumerations, text.
- Attributes do NOT exhibit identity.
- Attributes should have precise representative names.
- Derived attributes.

How to find attributes

STI · INNSBRUCK

- Interview: talk to subject matter experts.
- Documentation: read what experts have written about the subject matter, read the requirements documentation, etc.
- Observation and reflection.

How to find attributes

STI · INNSBRUCK

- Nouns in „-ness“
  - Velocity-ness, job-ness, arrested-ness…
- „How much, how many“ test.
  - If you evaluate this, then it is probably an attribute.
  - If you enumerate these, it is probably an entity.
- Status attributes are problematic because of open-ended range or fixed, but very large possible values, or because of complex state dependencies.
  - Status attributes enable and disable the functionality of an item over time. Each status attribute allows you to enable the item for a particular use. For example, if you set the status attribute *Purchasable* to *Yes*, you can put the item on a purchase order.

Ranges of attributes

STI · INNSBRUCK

- Age abstracted as an integer.
- Latitude and longitude: real numbers.
- Names abstracted as text.

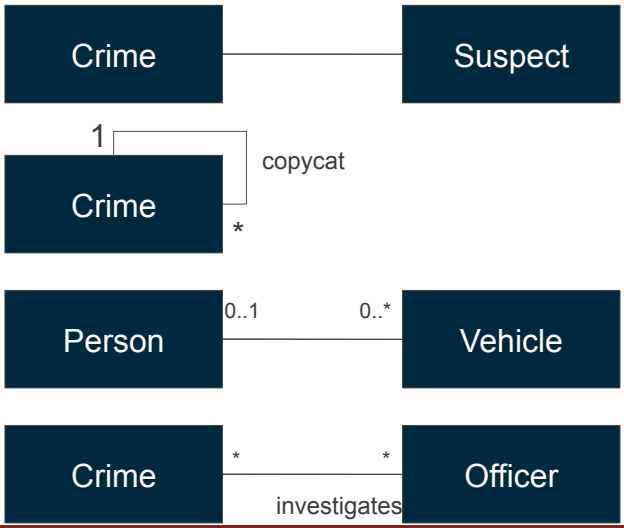
Associations

STI · INNSBRUCK

- Associations are relationships where entities instances are aware of, and characterized by, other entity instances.
- Characterizing relationships vs. use relationships.
  - The former are part of the domain model.
- Properties of associations: reflexivity, cardinality, functional, inverse-functional, discountinuous multiplicity, transitivity, symmetry etc.
- Arity
- Derived associations.
- Associations can stand for any way in which one entity can be characterized by another.
- Associations should be labelled.

Examples

STI · INNSBRUCK



How to find associations

STI · INNSBRUCK

- Interview: talk to subject matter experts.
- Documentation: read what experts have written about the subject matter, read the requirements documentation, etc.
- Observation and reflection.

- Verbs, verbal phrases and things that could have been verbs.
  - „*The buttler murdered the duchess*“

Association as roles

STI · INNSBRUCK

- *Never draw more than three associations without going back and starting to label them.*

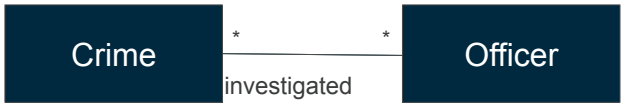
- The most common way to label associations is with role names.

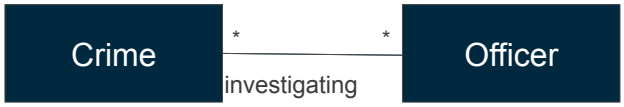
- Nouns, adjectives.

- Verbs: indication of time‘s passing.
  - **Short**-term, one-to-one associations should be named with present participles. (investigating)
  - **Longer**-term, one-to-many associations should be named with past participles, or with the simple present third-person singular. (investigates, investigated)

Examples

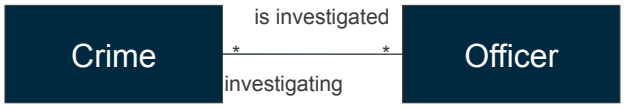
STI · INNSBRUCK





Direction

STI · INNSBRUCK



Aggregation

STI · INNSBRUCK

- Aggregations are relationships where entities instances are
  - aware of, and
  - characterized by,

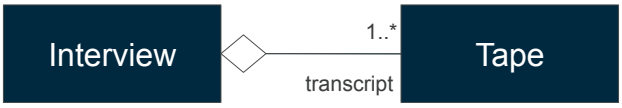
  other entity instances.

- Difference between aggregation and association:
  - An association describes a relationship between **instances of one or more classes**.
  - An aggregation describes **a part-whole** relationship.
  - Existence-dependence

Existence-dependence

STI · INNSBRUCK

- Provides a reliable test to distinguish between an aggregation and an association.
- Component is mandatory for the existence of the composit.

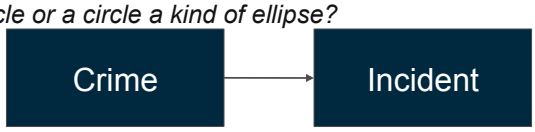


- 1..* vs. existence-dependence.

Classifications

STI · INNSBRUCK

- Classification is a relationship between **entities and sets of instances**.
- It is done to communicate understanding and to reduce redundancy.
- *„Tools“: Generalization/specialization, typing, is-a-kind-of, is-a, inheritance.*
- Some kind of things is a kind of another kind of thing and yet they couldn‘t both be said to be instances of the same kind of thing.
- *Is the ellipse a kind of circle or a circle a kind of ellipse?*
- *Is a warrant an arrest?*



How to find classifications

STI · INNSBRUCK

- Top-down, bottom-up, middle-out.
- Are all instances of entity A also instances of entity B?
- Are all A‘s also B‘s?
- Roles.
- Difference between classifications, associations, and aggregations.

Approaches

STI · INNSBRUCK

- Bottom up
  - Collection of most fine-grained entities first and then their „more general“ counterparts
  - Example: Dalmatian, Retriever, Setter; then Hunting Dog; then Dog; then Animal etc.
- Top down
  - Collection of most general entities first
  - Example: Animal; then carnivor and mammal, etc.
- Middle out
  - Combination of both

STI · INNSBRUCK

STI · INNSBRUCK

BASIC RELATIONSHIPS AND THEIR PROPERTIES

Relationships overview

STI · INNSBRUCK

- Spatial relationships
- Classification relationships
- Meronymy relationships
- Temporal relationships
- Case relationships

Definitions

STI · INNSBRUCK

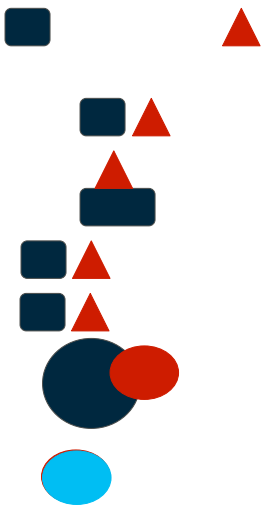
- Reflexivity.
  - Every element related to itself (<= is reflexive, < not)
- Antisymmetry.
  - R(a,b) and R(b,a), then a = b
- Asymmetry
  - For all a and b in X, if a is related to b, then b is not related to a.
- Transitivity.
  - A is related to an element b, and b is in turn related to an element c, then a is also related to c

Spatial relationships

STI · INNSBRUCK

- left-of, right-of.
- above, below.
- in-front-of, behind.
- inside.
- outside.
- between.

- far.
- near.
- touching.
- beside.
- disjoint
- intersect
- coincident



Spatial Characteristics

STI · INNSBRUCK

- Reflexivity (coincident).
- Irreflexivity (all others).
- Symmetry (near, far, beside).
- Asymmetry (left-of, right-of, above, behind, inside).
- Transitivity (left-of, right-of, above, behind).
- Mutual exclusivity (examples: below, coincident, inside; touching, far; intersect, inside).

Classification relationships

STI · INNSBRUCK

- Capture the intuitive semantics of the is-a relationship.
- Functional inclusion
  - A chair is-a piece of furniture.
  - A hammer is a tool.
- State inclusion
  - Polio is a disease.
  - Hate is an emotion.
- Activity inclusion
  - Tennis is a sport.
  - Murder is a crime.
- Action inclusion
  - Lecturing is a form of talking.
  - Frying is a form of cooking.
- Perceptual inclusion
  - A cat is a mammal.
  - An apple is a fruit.

Classification Characteristics

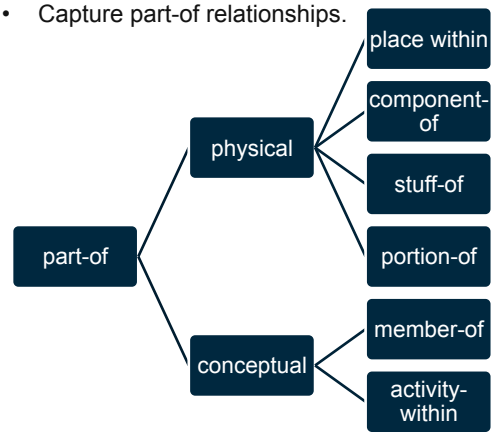
STI · INNSBRUCK

- Essential properties of an entity are inherited by its subentities.
- All instances of a subentity are instances of the superentity.
- Reflexivity.
  - Every element related to itself (<= is reflexive, < not)
- Antisymmetry.
  - R(a,b) and R(b,a), then a = b
- Transitivity.
  - A is related to an element b, and b is in turn related to an element c, then a is also related to c

Meronymy relationships

STI · INNSBRUCK

- „the semantic relation that holds between a part and the whole “
- Capture part-of relationships.



The Alps are part-of Europe

A chapter is part of a book.

A chair is partly wood.

A yard is part of a mile.

A tree is part of a forest.

Paying is part of shopping.

Meronymy Characteristics

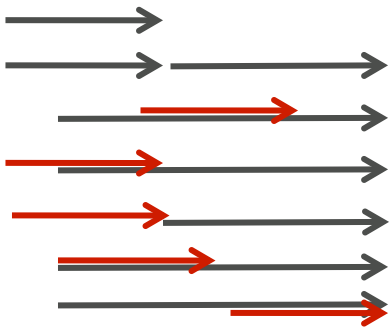
STI · INNSBRUCK

- Irreflexivity.
- Asymmetry.
- Transitivity.

Temporal relationships

STI · INNSBRUCK

- X equals Y.
- X before Y, Y after X.
- X during Y, Y contains X.
- X overlaps Y.
- X meets Y.
- X starts Y.
- X finishes Y.



Temporal Characteristics

STI · INNSBRUCK

- Irreflexivity.
- Asymmetry.
  - For all a and b in X, if a is related to b, then b is not related to a.
- Transitivity (before, during, starts, finishes).

Dependency relationships

STI · INNSBRUCK

- depends-on-causally.
- depends-on-existentially.

Characteristics

- Transitivity.

Case relationships

STI · INNSBRUCK

- They differ from other families of relationships in that they do not depend solely on the nature of the meaning of the entities they relate.

- Agent-Action. Dog-bark; artist-paint.
- Agent-uses-Instrument. Skier-uses-skis.
- Agent-Object. Bakes-uses-flour.
- Action-Recipient. Laydown-bed.
- Action-Instrument. Paint-brush.

Characteristics

STI · INNSBRUCK

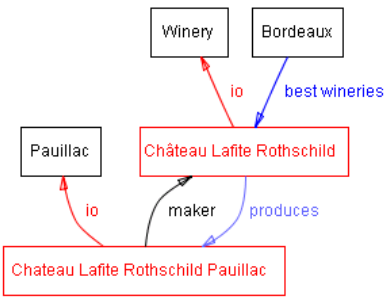
- Irreflexivity.
- Asymmetry.
  - For all a and b in X, if a is related to b, then b is not related to a.

Summary

STI · INNSBRUCK

- General principles.
- Basic elements and their properties: entities, attributes, associations, specialization/ generalization.
- Relationships and their characteristics.

STI · INNSBRUCK



Based on: Ontology Engineering for the Semantic Web and Beyond, Natalya F. Noy
Stanford University
noy@smi.stanford.edu

LARGE EXAMPLE: WINE DOMAIN

Enumerate Important Terms

- What are the terms we need to talk about?
- What are the properties of these terms?
- What do we want to say about the terms?

Enumerating Terms

wine, grape, winery, location,
wine color, wine body, wine flavor, sugar content
white wine, red wine, Bordeaux wine
food, seafood, fish, meat, vegetables, cheese

Define Classes and the Class Hierarchy

- A class is a concept in the domain
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a collection of elements with similar properties
- Instances of classes
  - a glass of California wine you'll have for lunch

Class Inheritance

- Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)
- A class hierarchy is usually an IS-A hierarchy:

an instance of a subclass is an instance of a superclass

- If you think of a class as a set of elements, a subclass is a subset

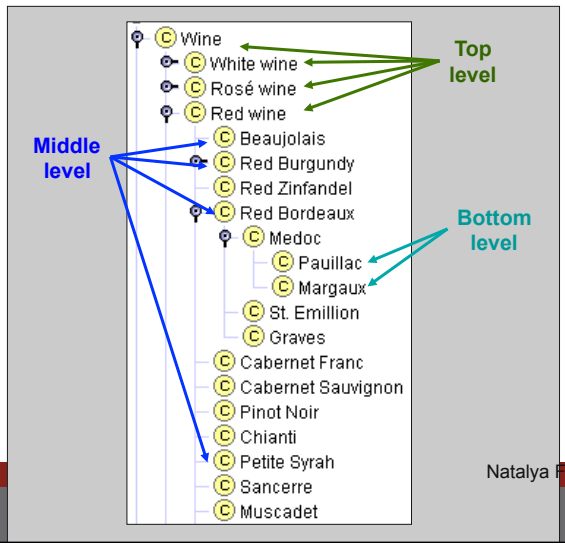
Class Inheritance - Example

STI · INNSBRUCK

- Apple is a subclass of Fruit
  Every apple is a fruit
- Red wines is a subclass of Wine
  Every red wine is a wine
- Chianti wine is a subclass of Red wine
  Every Chianti wine is a red wine

Levels in the Hierarchy

STI · INNSBRUCK



Modes of Development

STI · INNSBRUCK

- top-down – define the most general concepts first and then specialize them
- bottom-up – define the most specific concepts and then organize them in more general classes
- combination – define the more salient concepts first and then generalize and specialize them

Documentation

STI · INNSBRUCK

- Classes (and slots) usually have documentation
  - Describing the class in natural language
  - Listing domain assumptions relevant to the class definition
  - Listing synonyms
- Documenting classes and slots is as important as documenting computer code!

Define Properties of Classes

- Properties of a class definition describe attributes of instances of the class and relations to other instances

  Each wine will have color, sugar content, producer, etc.

Properties (Slots)

- Types of properties
  - "intrinsic" properties: flavor and color of wine
  - "extrinsic" properties: name and price of wine
  - parts: ingredients in a dish
  - relations to other objects: producer of wine (winery)
- Simple and complex properties
  - simple properties (attributes): contain primitive values (strings, numbers)
  - complex properties: contain (or point to) other objects (e.g., a winery instance)

Properties for the Class Wine

Template Slots

Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

(in Protégé-2000)

Property and Class Inheritance

- A subclass inherits all the properties from the superclass

  If a wine has a name and flavor, a red wine also has a name and flavor

- If a class has multiple superclasses, it inherits properties from all of them

  Port is both a dessert wine and a red wine. It inherits "sugar content: high" from the former and "color:red" from the latter

Property Constraints

- Property constraints (facets) describe or limit the set of possible values for a slot
  - *The name of a wine is a string*
  - *The wine producer is an instance of Winery*
  - *A winery has exactly one location*

Facets (values) for properties at the Wine Class

Template Slots

Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Common Facets

- Property cardinality – the number of values a property has
- property value type – the type of values a property has
- Minimum and maximum value – a range of values for a numeric property
- Default value – the value a property has unless explicitly specified otherwise

Common Facets: property cardinality

- Cardinality
  - Cardinality N means that the property must have N values
- Minimum cardinality
  - Minimum cardinality 1 means that the property must have a value (required)
  - Minimum cardinality 0 means that the property value is optional
- Maximum cardinality
  - Maximum cardinality 1 means that the property can have at most one value (single-valued property)
  - Maximum cardinality greater than 1 means that the property can have more than one value (multiple-valued property)

Common Facets: Value Type

- String: a string of characters ("Château Lafite")
- Number: an integer or a float (15, 4.5)
- Boolean: a true/false flag
- Enumerated type: a list of allowed values (high, medium, low)
- Complex type: an instance of another class
  - Specify the class to which the instances belong

  The Wine class is the value type for the property "produces" at the Winery class

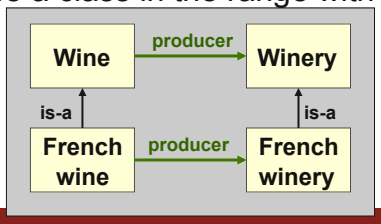
Domain and Range of property

- Domain of a property – the class (or classes) that have the property
  - More precisely: class (or classes) instances of which can have the slot
- Range of a property – the class (or classes) to which property values belong

Facets and Class Inheritance

STI · INNSBRUCK

- A subclass inherits all the properties from the superclass
- A subclass can override the facets to "narrow" the list of allowed values
  - Make the cardinality range smaller
  - Replace a class in the range with a subclass



Create Instances

STI · INNSBRUCK

- Create an instance of a class
  - The class becomes a direct type of the instance
  - Any superclass of the direct type is a type of the instance
- Assign property values for the instance frame
  - Slot values should conform to the facet constraints
  - Knowledge-acquisition tools often check that

Creating an Instance: Example

ASSIGNMENTS

Assignment 3

STI · INNSBRUCK

Describe the domain related to business trip management (cf. Assignment 1) using 10-20 entities, attributes and relationships.

Also create some instances (examples) for your entities/ attributes/relationships.

Tool: You can use simple visualization like Powerpoint or Protégé.

Protégé Stanford University
http://protege.stanford.edu/overview/protege-owl.html

Due: next class (Nov 15, 2010)

Assignment 4

STI · INNSBRUCK

a) In the context of financial accounting what might Customer, Investment, Cruise Ship and Tanker have in common?
b) In the context of a mathematics package, what might Circle and Ellipse have in common?
c) Summarize the meaning of aggregation, classification, association and provide three examples for each.

Due: next class (Nov 15, 2010)

Further reading

STI · INNSBRUCK

- Object-Oriented Analysis and Design (John Deacon, Addison-Wesley).
  - Chapters 5-8.
  - http://www.johndeacon.net/OOAandD/index.asp#TutorLecturerResources
- IDEF5 Method Report: http://www.idef.com/pdf/Idef5.pdf

STI · INNSBRUCK

Thank You!

Questions?