



Subject Matter Model

Chapter 6



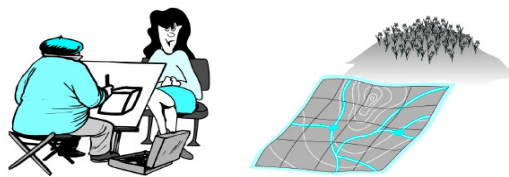
What Do We Analyze?

- We have looked at this before
- We can nearly always analyze the subject matter of the system-to-be
 - (Sometimes known as the real world or the application domain or the problem domain)
- Only sometimes can we analyze an existing solution
 - Systems analysis
 - Which used to be much more common than it can be today

First Model


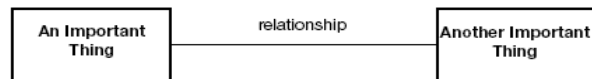
- We have already spoken about the reason for the emphasis on modeling
- A computer system is always a model
- Therefore we can always see our job as modelers
- We can't model directly in lines of code
 - The disparity between the modeled and the modeling medium is too great
- We can't jump into modeling with classes and types
 - There are simply too many questions; the disparity is still too large
- As our first modeling step we must find a way to abstract our subject matter

Models





The Graph model



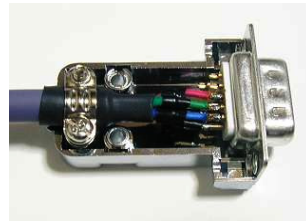
Chunks, cohesion and coupling

- Model must be:
 - Understandable
 - Reorganizable (Maintainable)
 - Divisible
- We have learned that for anything to be understandable and maintainable, we need
 - Chunks (modularity, componentization, ...)
 - High cohesion
 - Low coupling

Chunks

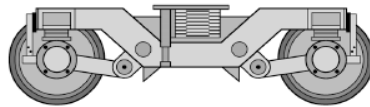


How do you imagine a cable end?



Cohesion

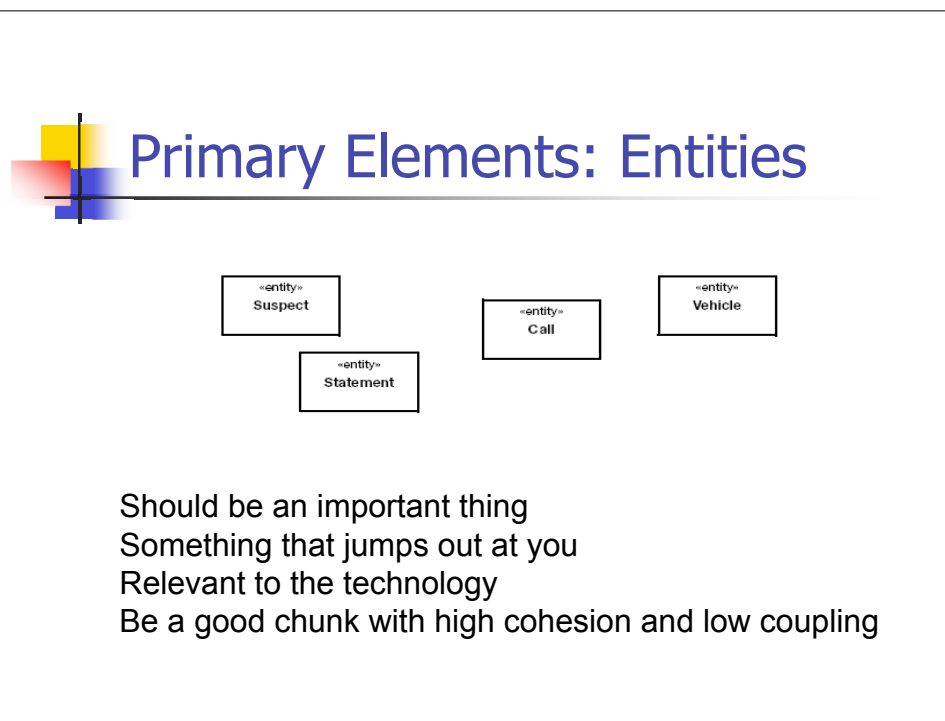
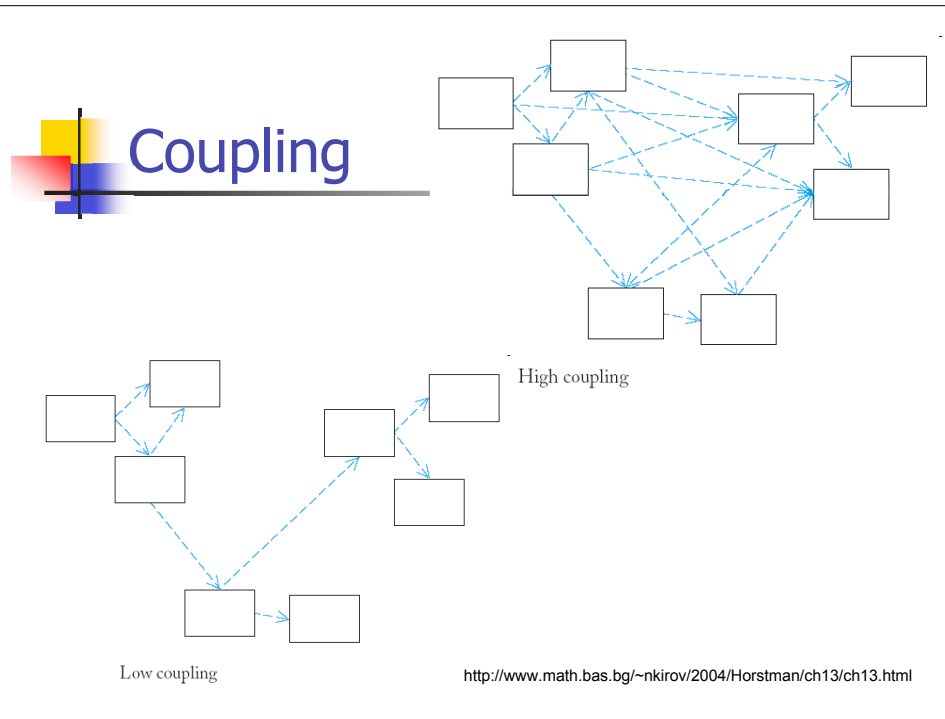
Cohesive – a bogie



Uncohesive – an axle with one wheel

A **bogie** (UK) or **wheel truck** (US) is a structure underneath a **train** to which **wheel axles** (and, hence, wheels) are attached. If they are used there are usually two for each **carriage**, wagon and **locomotive**, or alternatively, they are at the connections between the carriages or wagons. The connections of the bogies with the cars allow a small rotational movement around a vertical axis.







Finding Entities

- Nouns
- Talk the subject matter or to an expert
- Read what other experts have written
- Read the requirements document (remember, its an analysis input)
- Read proposals
- Listen to the voices in your head



... Finding Entities

Nouns can hide though

- Especially in English, where verbs can be nouned and nouns can be verbed
- Pronouns can hide important entity names
- The passive voice can hide important entity names
- ...



Assessing Entities

- Cohesion and Naming
- Relevance
- Identity
- Subject Matter Membership



Cohesion and Names

Cohesion

- One thing we begin here, that we will continue throughout the development phases, is
 - Cohesion (or focus, oneness, normalization)
- An entity box represents a set of entity instances but,
 - Each such entity instance should represent one clear and crisp thing from the subject matter
 - Not half a thing or several things or a vague piece of “arm waving”



... Cohesion and Names

- Naming is part of the assessment of cohesion,
 - And important anyway in its own right
- It's usually singular
 - To emphasize the cohesion, the oneness
- It should also be
 - Precise
 - complete, honest and accurate
 - Colorful
 - No “manage(r)”s, “handle(r)”s, “process(or)”s, etc.
 - Concise
 - But not at the expense of the first two



Relevance

- We said that analysis was definitive
 - However, it's the content that's definitive; not the scope
- We have to make a judgment as to whether an candidate entity is relevant
 - Necessary and sufficient test
 - Homeless attributes
 - Butler's riposte



... Relevance

- Necessary and sufficient
 - Is a candidate necessary, or have we already got it, perhaps under a synonym?
 - Is an entity sufficient, or is it trying to stand for more than one concept
 - Is it possible to name it according to the rules just given?
If its name can't be good, it probably isn't good
- Homeless attributes
 - If an entity were deemed irrelevant and removed, would there be any important attributes that would become homeless?



Butler's Riposte

- We assume that the system-to-be has correspondence with the subject matter (and the entities in it),
 - Use that as a test
- We imagine that the entity changes state in some way
 - (We can apply this test to attributes as well)
 - Would the system-to-be need to discover this change?
 - In order to keep the correspondence; to keep "in sync"?
 - Yes: then it's relevant; no: it might not be relevant



Identity

- Helps decide if something relevant is best modeled as an entity, or as an attribute
- Identity is an important property of application (“business”) software objects as well
 - So identity is an important concept to understand
- Does the thing exhibit identity? Or “merely” value?
 - Would you tend to point at it? Or would you “merely” measure it?
 - Do they get enumerated (“how many?”), or evaluated (“how much?”)
- Entity instances are identified; attributes are just measured

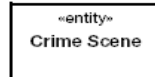


Sticking to the Subject Matter


- Only include the entities that exist in the absence of the system-to-be
- Don’t include entities that are artifacts of an anticipated system-to-be
- Use the terminology of the subject matter, not the implementation technology
- Don’t invent things
 - Such as serial numbers that don’t exist in the subject matter



Depicting Entities



- Label, stereotype, keyword
 - Tells what the box is doing on the diagram
- Name of the Entity in bold in the middle



Secondary Model Elements: Properties and Connections

- Attributes
 - The relevant, measurable properties of the relevant entities
 - Intrinsic state
 - Remember state is our modeling friend
- Characterizing relationships
 - Association, aggregation and composition
 - Extrinsic state
 - State contributions from the identity of other entities rather than values



Attributes

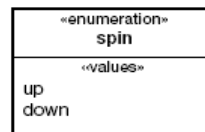
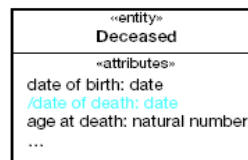
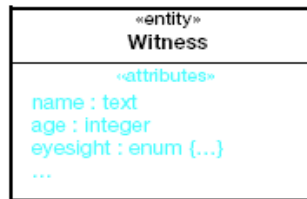
- A characteristic
- An attribute is a relevant, measurable property of an entity
- Usually the value will be scalar: a choice from a range of values (no data structure)
- If it is complicated to measure, its an entity
- Types
 - Integers
 - Real numbers
 - Enumerations
 - Boolean
 - Text (string)



Assessing Attributes

- Ensure they aren't really entities
 - The point or measure test
- Relevant
- Precisely and colorfully named
- How much how many test
- "Irreducible"
 - Although one could come up with a method where attributes could have structure of their own,
 - 99% of the time it's better and easier to deem such attributes to be entities

Depicting Attributes



Assessing What We Have

- Entities are not classes
 - We hope and expect that the subject matter entities will lead us to useful kinds of object instances
 - We must not expect them to deliver a list of classes; classes come later
- Attributes are not instance variables
 - Attributes lead us to query services of objects
 - They don't lead us to instance variables; attributes come much later



Associations

- Finding associations
- Assessing associations
- Depicting associations



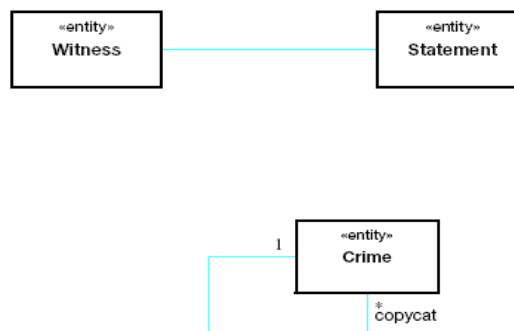
Finding Associations

- Finding associations seems to cause more difficulty than most things
- The notion of characterization is very useful
 - As well as deriving character from the values of its attributes,
 - Does an entity instance also derive character from its awareness of, its relationship with, other entity instances?

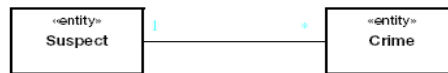
Assessing Associations

- We have just given the main extra test
 - Above and beyond relevance, good name, ...
 - Does the association model necessary, non-intrinsic characterization?
- I.e. is the model correct in saying that if an entity instance's set of associated entity instances changed,
 - It will make a difference to the entity
 - The entity will be richer, poorer, more expensive, better paid, ...

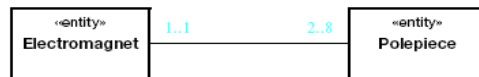
Depicting Associations



Multiplicity



More Multiplicity



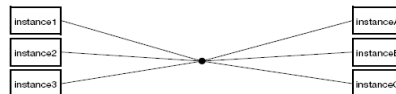
Many to many associations



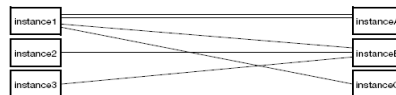
We need to be careful with many-to-many associations for at least three reasons
It's not obvious as to exactly what we mean by many-to-many
They're difficult to implement
They sometimes mask complimentary pairs of one-to-many associations

Three types

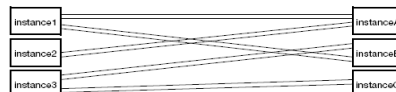
Spoke



Lattice-1

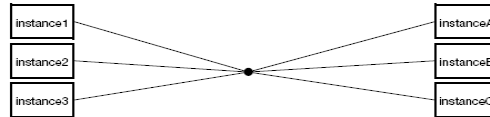


Lattice-2



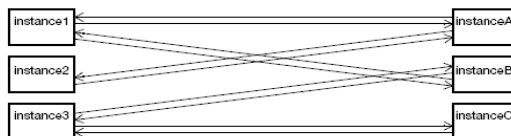
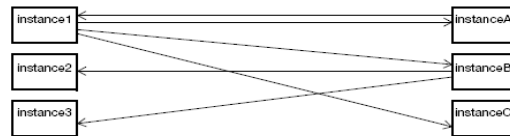
Spoke

Are you missing an entity?



Lattices

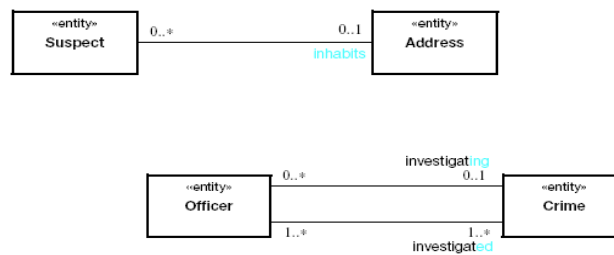
- Consider the direction of the relationships



Roles

- Older approaches named associations with a single name in the middle
 - The UML still permits this but a pair of names is now more common and more useful
 - The role names
- A role name names the role that the instance(s) at this end play in the life of the instance at the other end
 - And, of course, should be a precise, colorful name
 - Role names are some of the hardest names to get right; BUT don't spare any effort; good role names are vital

Roles





Think Instance

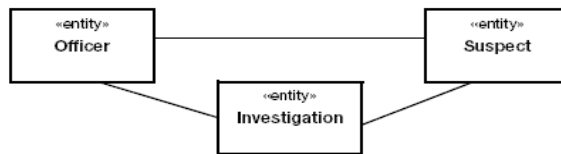
- Many of the questions we've wrestled with,
 - Mutuality
 - Role
 - Multiplicity
- Cannot be answered by reasoning about boxes in a diagram, where the boxes represent whole sets of instances
- You must picture and reason about the instances that the boxes represent



Derived Associations

- It's traditional to assume that if a is associated with b ,
 - And b is associated with c ,
 - Then a is implicitly (transitively) associated with c
- This is probably OK but be careful,
 - It's one of those areas where the real world, databases, and object technology do not match
- Associations will be implemented via links in private instance variables, so,
 - If a can message b , and perhaps b can message c ,
 - It won't mean that a can message c

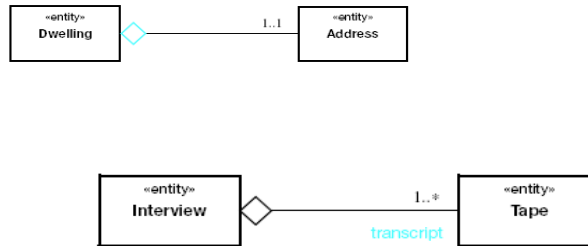
Derived Association



Aggregation

- Aggregation is the composition, whole/part, has-a relationship
- Contributes to state
- Is another characterizing relationship
- Useful clue about the linkages between instances
- Does the part ‘belong’ to the aggregate or does it exist on its own

Depicting Aggregation



Case Study- Alice

- Candidate entity names
 - Narrative analysis
- Finding the attributes
- Modelling the Relationships

Case Study: ALICE

- The system would, at a minimum:
 - assist with placing and recording bookings, helping ensure optimal table utilization, balancing the avoidance of double bookings with the need to take one or two "floating" bookings;
 - take table orders, optionally splitting order printouts to departments (starter, main, desert, etc.; accept event information, such as "table away", in order that, for example, tables still awaiting their main courses can be intuitively depicted;
 - produce bills;
 - assist with menu production and usage history;
 - store recipes for some dishes;
- The system might also:
 - maintain stock level information,
 - especially for the more expensive items.



<http://www.bestprices.com/cgi-bin/vlink/075992743921BT.html>

Candidate Entities

- **Bill**
- **Booking (Reservation)**
- **Chef**
- **Customer**
- **Dish**
- **Ingredient**
- **Inventory**
- **Meal Order**
- **Menu**
- **Menu Section**
- **Party**
- **Recipe**
- **Repertoire**
- **Restaurant**
- **Serving**
- **Shift**
- **Sitting**
- **Sommelier (Wine Steward)**
- **Stock Item**
- **Table**
- **Waiter**

Bill
amount requested: money, amount tendered: money, when paid: time and date

Booking
date: date, start time: time, requested duration: duration, booker name: text, covers: integer

Chef
name: text, contact number: phone number

Customer
name: text, address: text

Dish
number: text, name: text, description: text, price: money

Ingredient
name: text, quantity used: real, units: enum {gram, ml, cup, each}

Meal Order
time taken: time

Menu
date of use: date

Menu Section
course: enum {none, starter, main, desert}

Party
covers: integer

Recipe
name: text, extra preparation time: integer, instructions: text

Restaurant
numbered dishes: boolean, normal opening time: time, normal closing time: time, maximum people: integer

Serving
quantity: integer, preference: text

Shift
start time: time, end time: time

Sitting
name: text, typical meal duration: integer, normal opening time: time, normal closing time: time

Sommelier
name: text, contact number: phone number

Stock Item
total quantity: real, units: enum {gram, ml, cup, each}

Table
ID: text, nominal covers: integer, ζ smoking: boolean?

Waiter
name: text, contact number: phone number

