Business-Oriented Data Modelling Masterclass –
Balancing Engagement, Agility, and Complexity

A compressed two-day version developed for Booking.com
presented by
Adept Events and Clariteq Systems Consulting

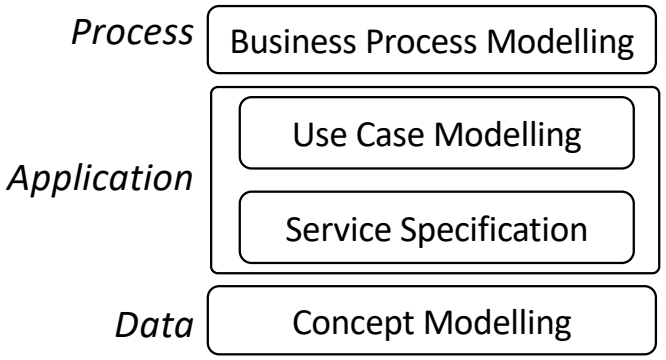
Alec Sharp
Consultant
Clariteq Systems Consulting Ltd.
West Vancouver, BC, Canada
asharp@clariteq.com
www.clariteq.com

Booking.com ///AdeptEvents CLARITEQ

Instructor / course developer background…

Alec Sharp, Clariteq Systems Consulting – *asharp@clariteq.com*

- 40+ years experience as an independent consultant:
  - Business Process Change – discover, model, analyse, and design/redesign processes
  - Application Requirements Specification
  - Data Modelling and Management My roots!
  - \+
  - Facilitation & Organisational Change
  - Project Recovery

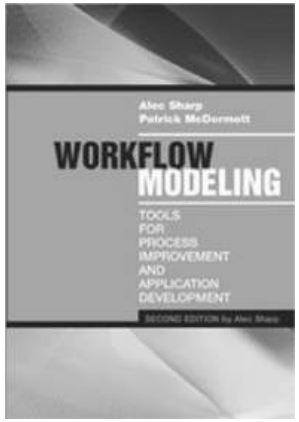


- Consulting, teaching, speaking globally (pre-pandemic)

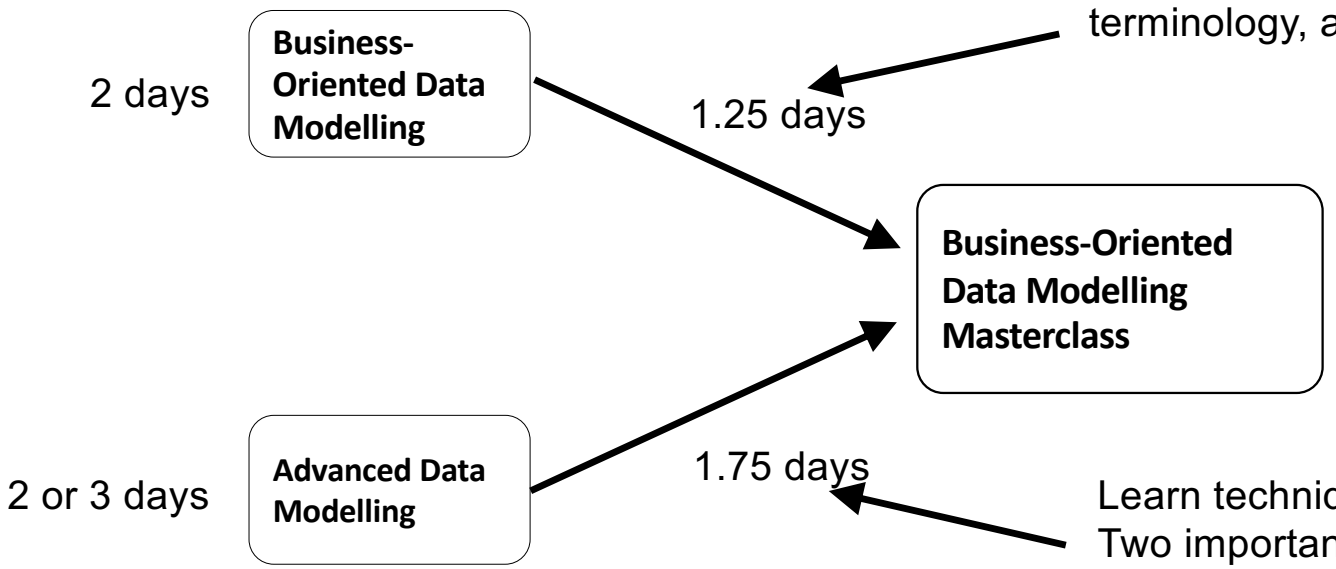
- Awarded DAMA's global Professional Achievement Award for contributions to "human-friendly" data modelling

Check out the nice reviews on Amazon - http://amzn.to/dHun1o

- Author of "Workflow Modeling"
  - best-selling book on process modelling & improvement
  - second edition – a complete re-write



Background for this course



Get everyone, even experienced modellers, to the same baseline of concepts, conventions and terminology, and our unique *business-friendly* approach.

~~3 days in-person~~

2 days in-person for this Booking.com version. *Core content is still here!*

Learn techniques for modelling more complex situations. Two important notes:

1. Our focus is on engaging businesspeople in *conceptual and logical modelling* of *operational data* - master data, transactional data, & reference data.
2. This is *not* a class on physical database design or implementation strategies (e.g., Data Mesh or Data Lake) and is *not* a class on Data Governance, Data Management, or Data Strategy – it's about *modelling*.

Overview and logistics

➡	*Fundamental and Advanced Topics*

1. Introduction and level-set
   - Issues and Principles
   - Essentials of Concept Modelling
   - Transition from Conceptual to Logical
2. Interesting structures
   - Types vs. Instances
   - Recursive Relationships
   - Subtyping / Generalisation
   - Meeting New Requirements
3. Rules on relationships and associations
   - Multi-way Associatives & Complex Rules
   - Advanced Normal Forms (4NF & 5NF)
4. Presentation techniques for data modellers
   - A (painful) Learning Experience
   - Core Techniques for Presenting
   - A Real-life Example

Schedule (CET)

- 09:00 - start
- 09:00 - 10:30 *class*
- 10:30 - 10:40 break
- 10:40 - 12:30 *class*
- 12:30 - 13:30 lunch
- 13:30 - 15:00 *class*
- 15:00 - 15:10 break
- 15:10 - 17:00 *class*
- 17:00 end

Finally…***you***:

- Name (how should I address you?)
- Brief description of your work
- Is there a topic you are especially interested in?
- *Please try to keep your introduction to one minute or less*

What is a Concept Model / Business Object Model / Domain Model…?

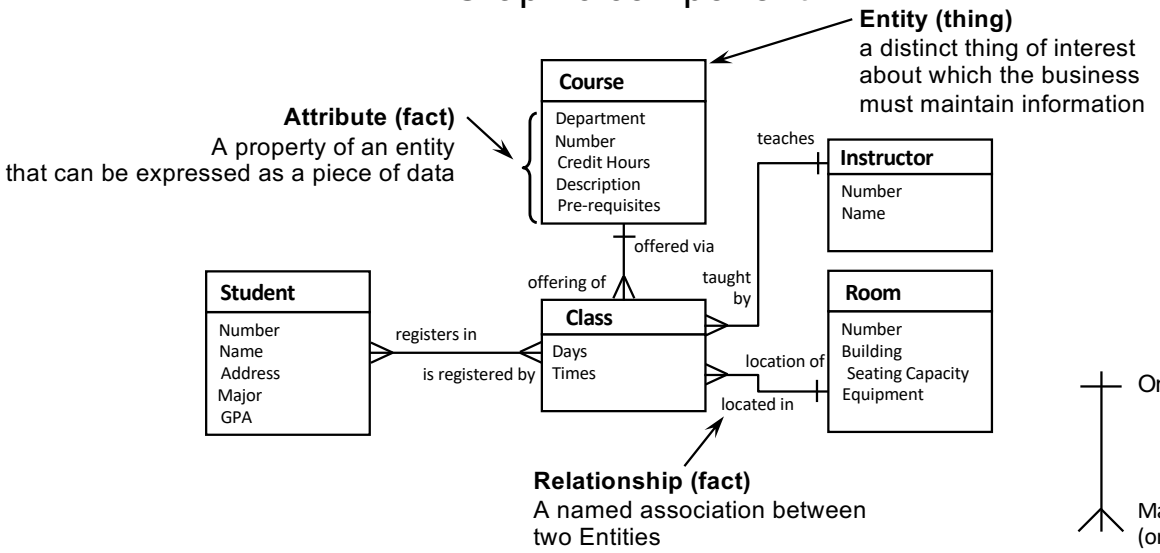
- A *description of a business* in terms of
  - **things** it needs to maintain records of – *Entities*
  - **facts about those things** – *Relationships & Attributes*
  - **policies & rules** *governing those things and facts*
- Models a view of the **real world**, not a technical design (therefore, stable and flexible)
- Can be comprehended by mere mortals (at least initially)
- Graham Witt – "A narrative supported by a graphic"

"Things" first, data later!

Narrative component

Student definition:
A Student is any person who has been admitted to the University, has accepted, and has enrolled in a course within a designated time. Faculty and staff members may also be Students

Graphic component

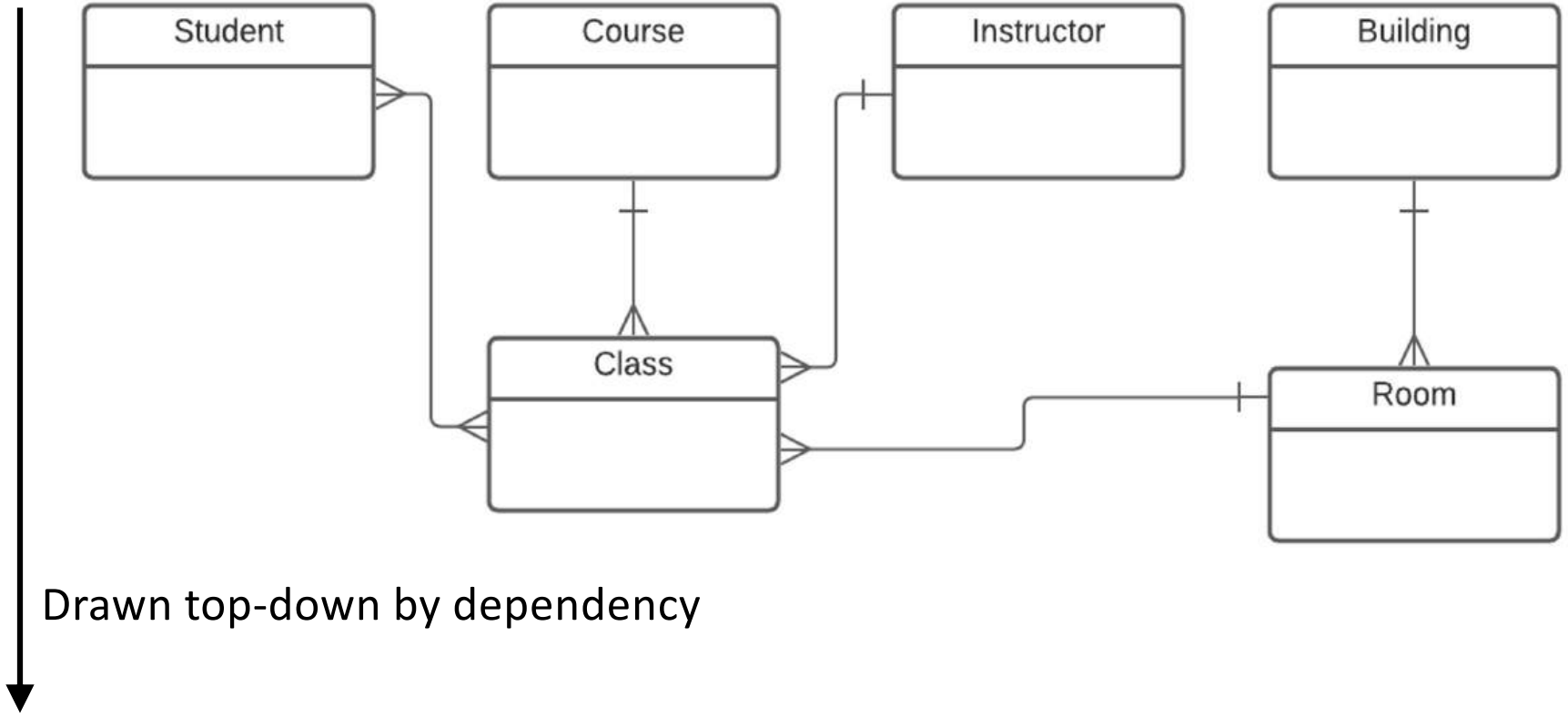


Plus "Assertions" (policies & rules)
- Each Course is offered through one or more Classes
  Each Class is an offering of a single, specific Course
- Each Instructor teaches one or more Classes
- Each Class is taught by one Instructor
  (which may or may not be true…)

Many rules can't be shown on the diagram…
- A Student can not register in two Classes of the same Course in the same Academic Term

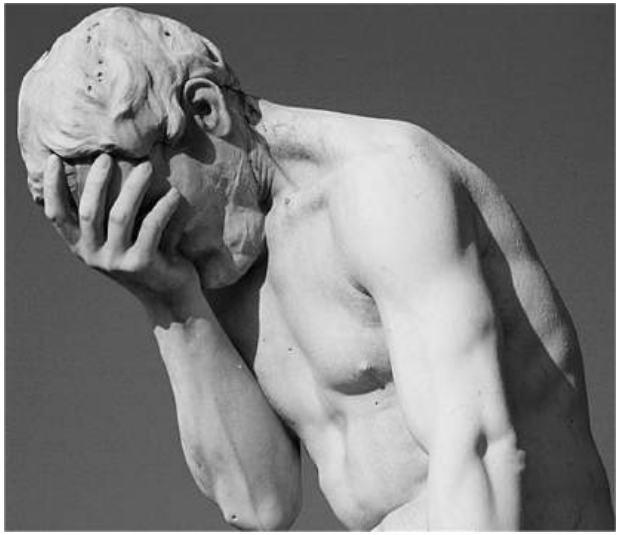
A better looking version of the model on the previous slide

Independent Entities at the top



Drawn top-down by dependency

Data Modelling – out of favour for a while, but things are getting better!



"We don't need data modelling because..."

- "We're going Client-Server!" (~1986)
- Agile ("We'll ~~refactor~~ rehacktor as necessary!")
- Packaged software / COTS
  ("The vendor has seen it all and has this figured out!")
- Big Data ("It's schema-less!") and IoT
- Data Science/Analytics
  ("The algos will discover all the connections!")
- Data Lake, Data Mesh, Data Lakehouse, … ("Fill it and they will come!")
- …and many other Silver Bullets that will *Save The Day!*
  (Chat GPT, Gen AI, LLM, … anyone?)

And then, starting ~ 5 years ago:

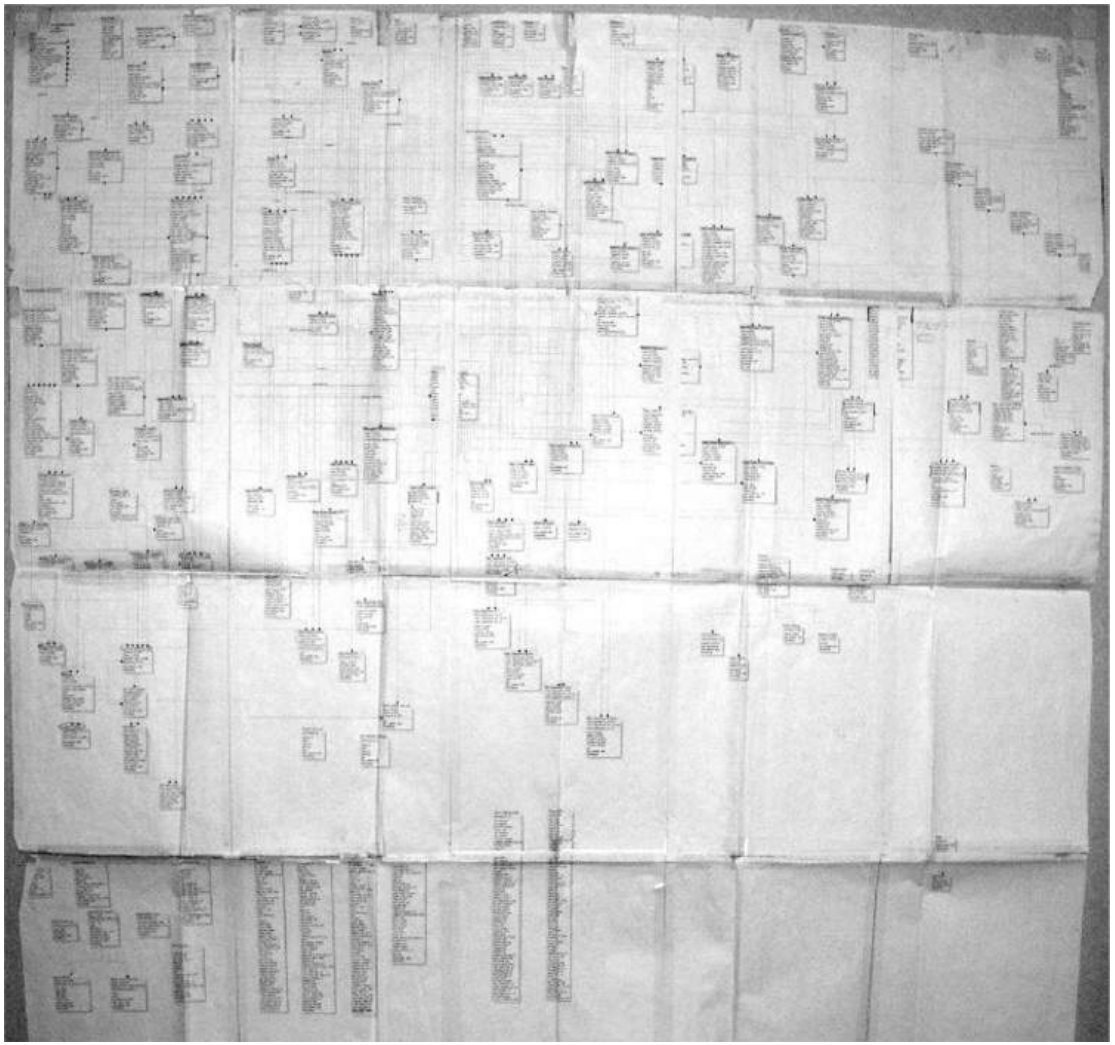
- "Could you build a 'Data Modelling for Data Scientists' class?"
- At a public workshop …
  "We aren't building a Data Lake, we're building a Data *Swamp!*"

Why Data Modelling fell out of favour

In general,
"data people" can make "data" far too difficult

1 – Confusion between data modelling and database design…

*"Help –
everyone hates our data model!"*

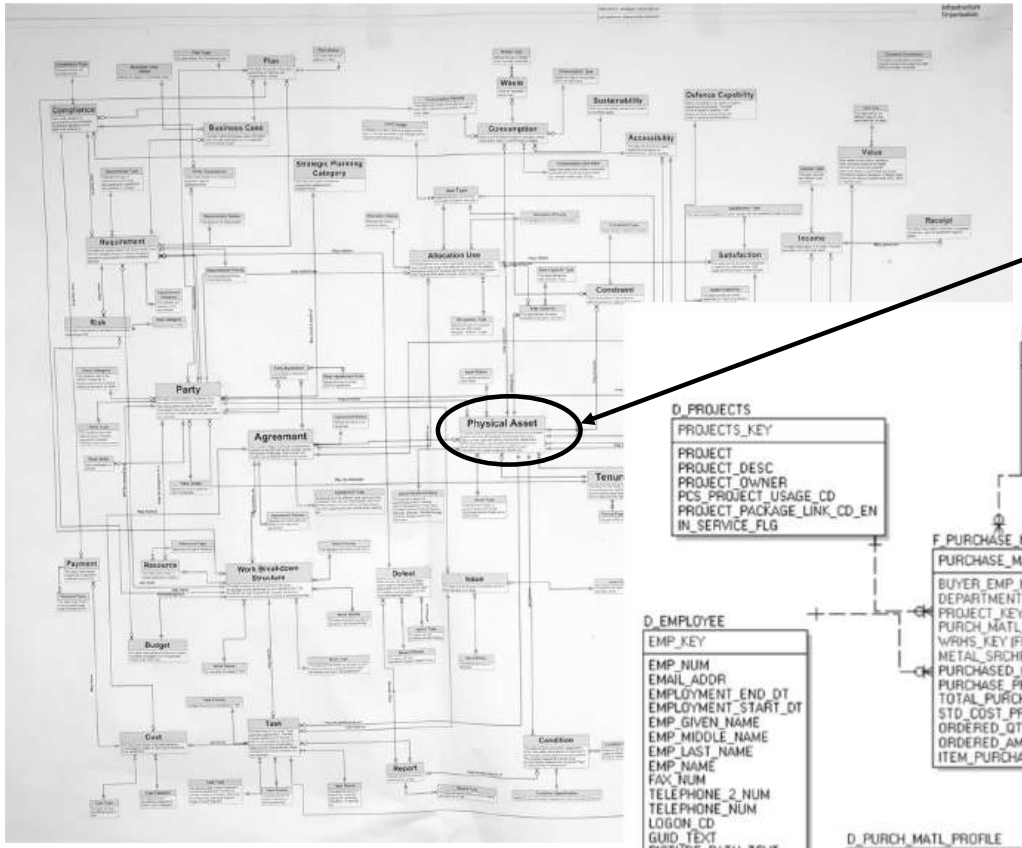


"Data people" make "data" far too difficult

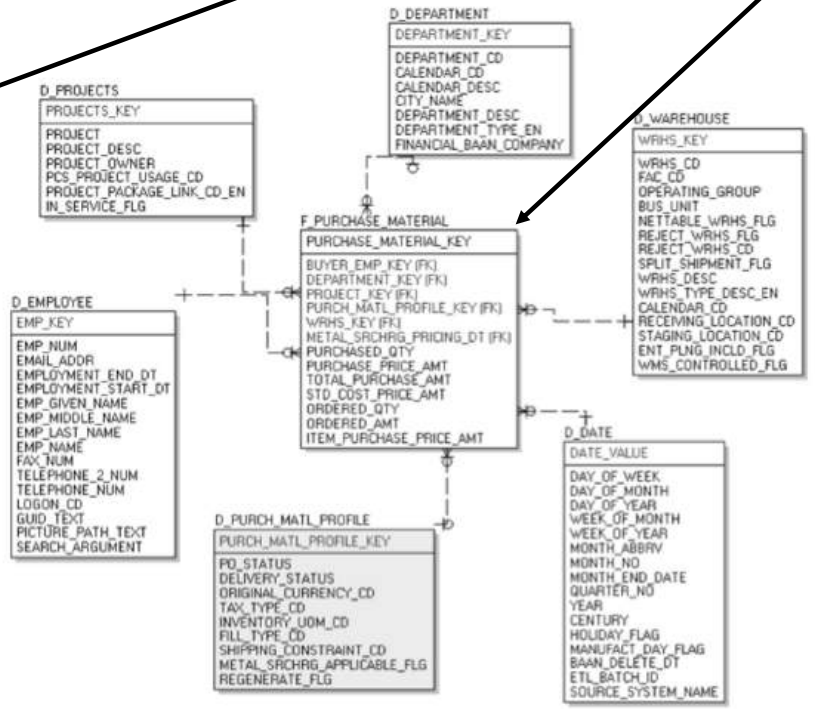
2 – Terrible diagramming…
A common error – *"the most important entity should go in the centre of the diagram."*

An excellent model *structurally*, but very difficult to follow – *no sense of direction.*

Concept Models / ER Models should be drawn top-down by dependency.



"Fact" in the middle - fine for Dimensional, terrible for E/R



Why Data Modelling fell out of favour

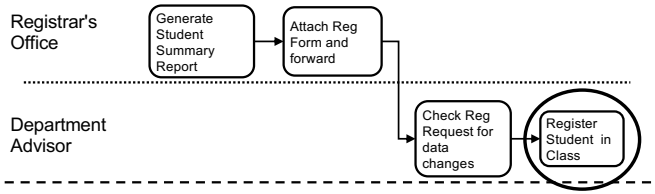
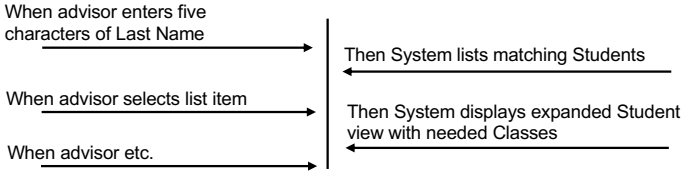
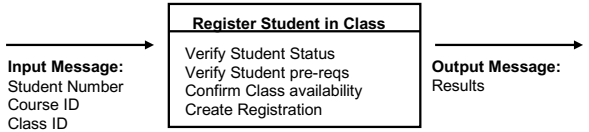
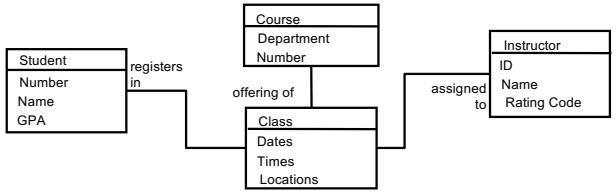
3 – No clarity on different types of models for different purposes

1 Contextual (Scope – Planner's View)	2 Conceptual (Overview – Owner's View)	3 Logical (Detail – Designer's View)
✓ *Context model* ✓ Agreement on "big picture," context, and some vocabulary ✓ A block diagram of "subject areas," higher level than individual entities ✓ Shows the scope or "footprint" ✓ Optional – not useful on smaller projects	✓ *Concept Model* ✓ Agreements on basic concepts, vocabulary, and rules	✓ *Logical Data Model* ✓ Complete detail for physical design
	Some important differences	
	✓ Main ("recognisable") entities only - a singular noun used daily ✓ Main attributes only, many are non-atomic ✓ M:M relationships ✓ Doesn't show keys ✓ Not normalised ✓ A "one-pager"	✓ All granular entities – many too detailed to come up daily ✓ All attributes included, all are atomic ✓ All M:M resolved ✓ Shows primary & foreign keys ✓ Fully normalised ✓ Five times as many entities

My most plagiarised slide!
More details later.

Why Data Modelling fell out of favour

The Clariteq Framework for Business Analysis

	Framework Layer	Technique sample	What it covers
Goals	*Business Objectives*	The university is initiating the "Strategic Enrollment" program to raise Student graduation rates in part by ensuring Classes are available for Student registration when needed.	✓ ***Project Charter*** – documents the rationale, objectives, scope, and success measures for the project
Process	*Business Process*		✓ ***Process Model*** - shows "what" in a Scope Model, then "who & how" in a Workflow Model – the steps done by the actors in the process
Application	*Presentation Services (user interface)*		✓ ***Use Case*** – models how an actor interacts with a system to obtain (trigger) a service, typically to complete a step in a process
	Business Services (rules & logic)		✓ ***Service Specification*** - describes a service – a package of rules and logic – that is triggered to complete or respond to a business event
Data	*Data Mgmt. Services (databases)*		✓ ***Concept Model*** - depicts the things and the facts about things the organisation needs to record; the things (the entities) are what processes and solutions act on.

This is not a sequence!

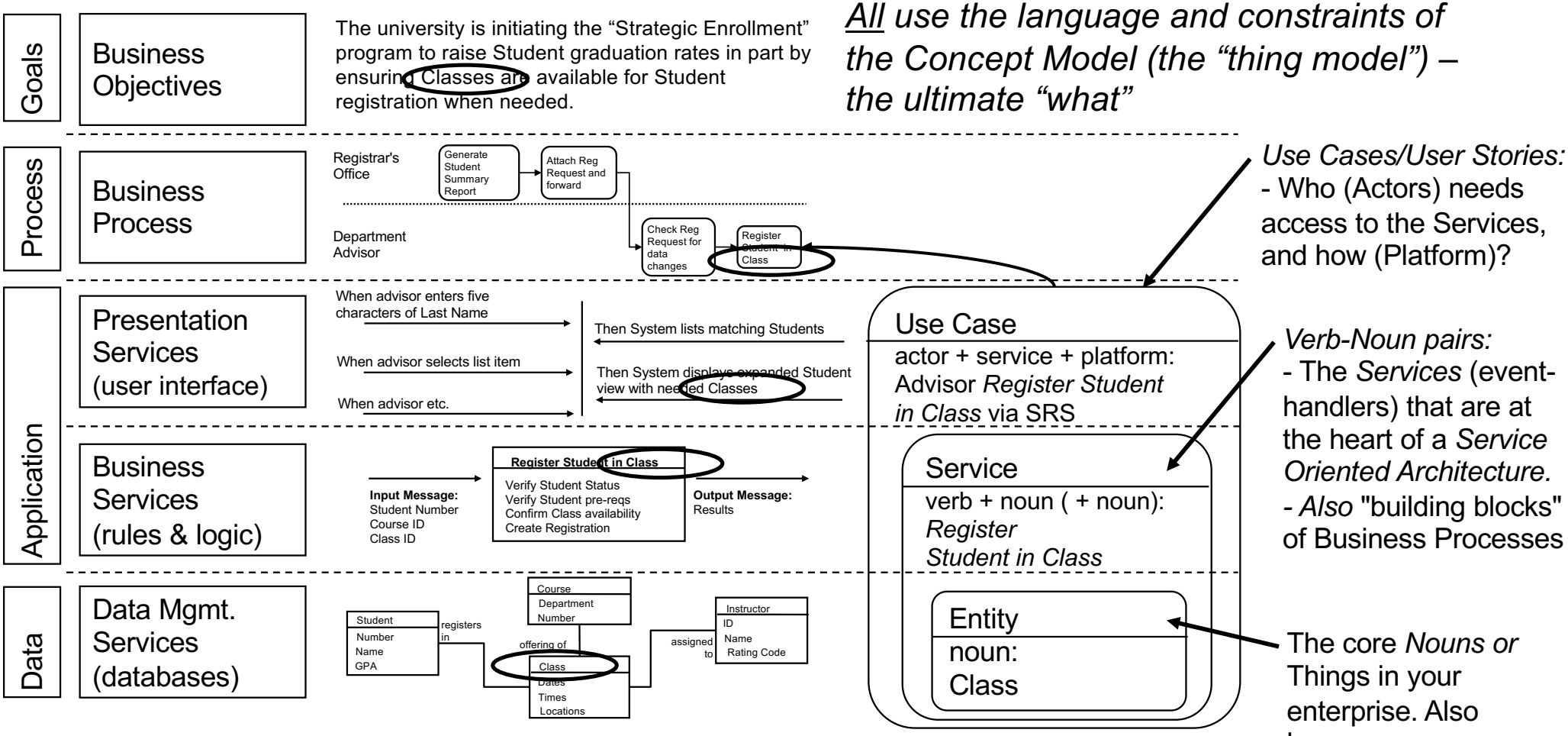
Business Process: gives great *context* for *Business Analysis*

Use Cases and Services: where we capture *Functional Requirements*

Concept Model / Data Model: a great *platform* for *Business Analysis*

4 – No clarity on how it relates to other models & BA techniques

Key point! Everything relies on the Concept Model



<u>All</u> use the language and constraints of the Concept Model (the "thing model") – the ultimate "what"

Use Cases/User Stories:
- Who (Actors) needs access to the Services, and how (Platform)?

Verb-Noun pairs:
- The *Services* (event-handlers) that are at the heart of a *Service Oriented Architecture.*
- *Also* "building blocks" of Business Processes

The core *Nouns or* Things in your enterprise. Also known as *Business Objects.*

Bonus – great starting point to discover your Events/Services and Use Cases/User Stories

Case study – Concept Model, Services, Use Cases, Business Processes

Client –

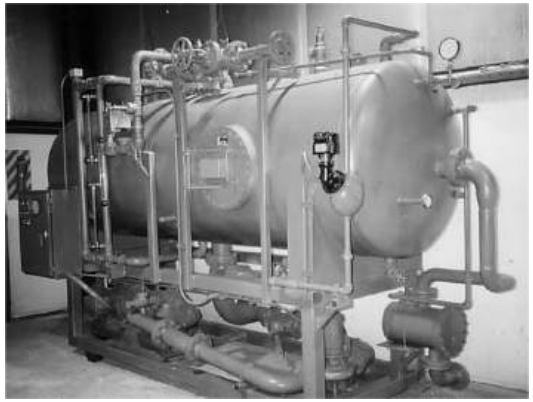
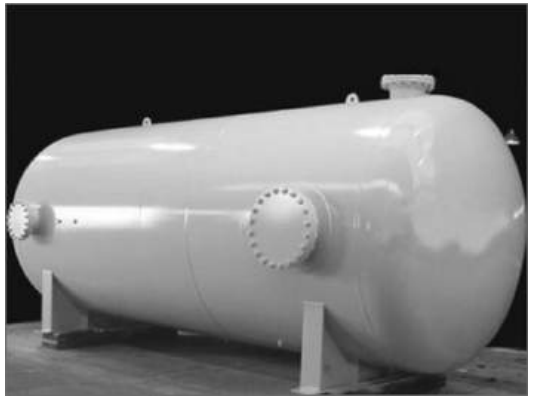
- Regulatory agency ensuring the safe design, installation, and use of technical equipment
- Natural gas systems, electrical systems, boilers and pressure vessels, elevating devices, & many more

Goal –

- Shift from an inspection-based model (~800 inspectors!) to client-managed safety programs
- Clients will apply for a *Client Safety Management Program Authorisation (CSMP Authorisation)*
  - must show effective processes and accurate record-keeping
- Clients will pay a fee for managing *their own safety programs!* Still beneficial!

Case study – Concept Model, Services, Use Cases

- Business Development chooses Pilot Program –
  boilers and pressure vessels in Oil & Gas fields

- Current systems won't support CSMP, time-consuming and expensive to change them –
  IT and Finance suggest 18 – 24 months of work
- BD is unimpressed by IT and Finance objections ("You're being mindlessly obstructionist!")
  and proposes work-around procedure. *Guess which tool they intend to use?*
- I'm hired to identify end-to-end implications –
  "Design a process and determine IT requirements that will allow this procedure to work."
- *Concept Modelling was a critical tool in understanding the underlying policies,*
  and developing the process & requirements

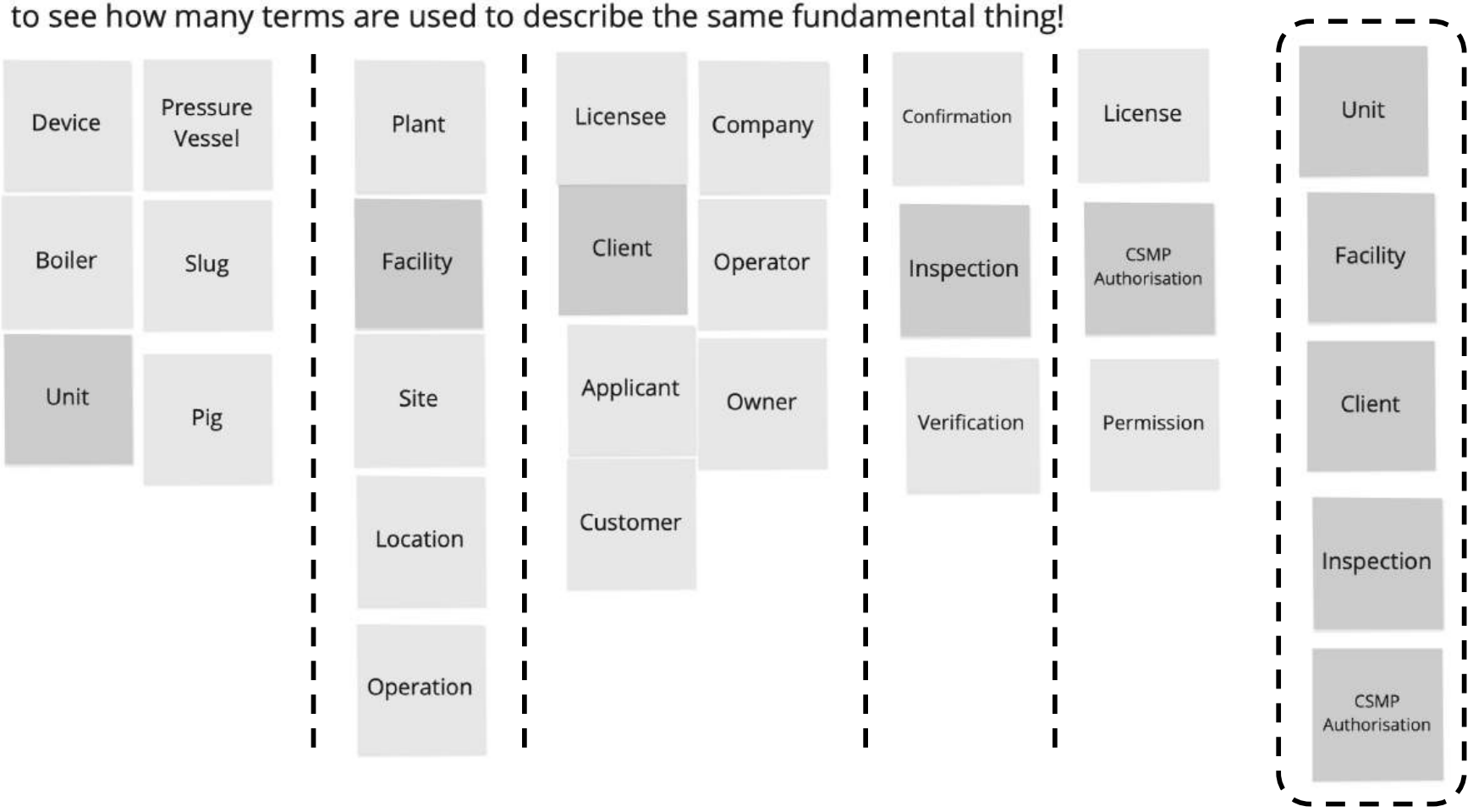
Always start with terminology (the "things")

From one-on-one interviews with 8-10 key stakeholders we gathered ~200 terms related to CSMP (Client Safety Management Program) – "anything that went by a name."
Here are 24 that met the criteria to be a "thing"– the candidate *Entities.*

Identify synonyms and select one term.
How do these relate to one another?
What do you need to know about each?

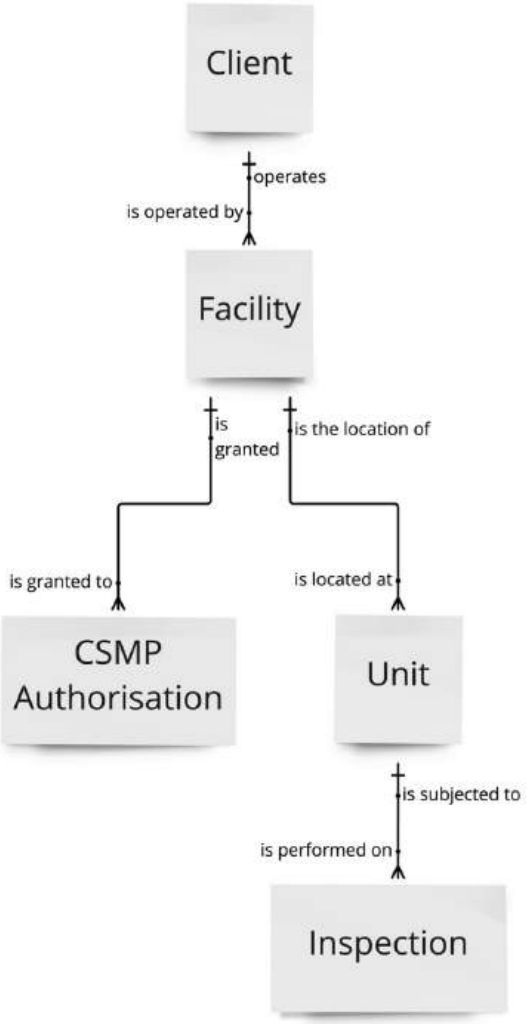
Review of a Miro example – Terminology Analysis

Terminology analysis (continued):
Let's arrange these terms into columns of synonyms. It's always a surprise for the business to see how many terms are used to describe the same fundamental thing!



Concept Model Version 1; not perfect, but a good start

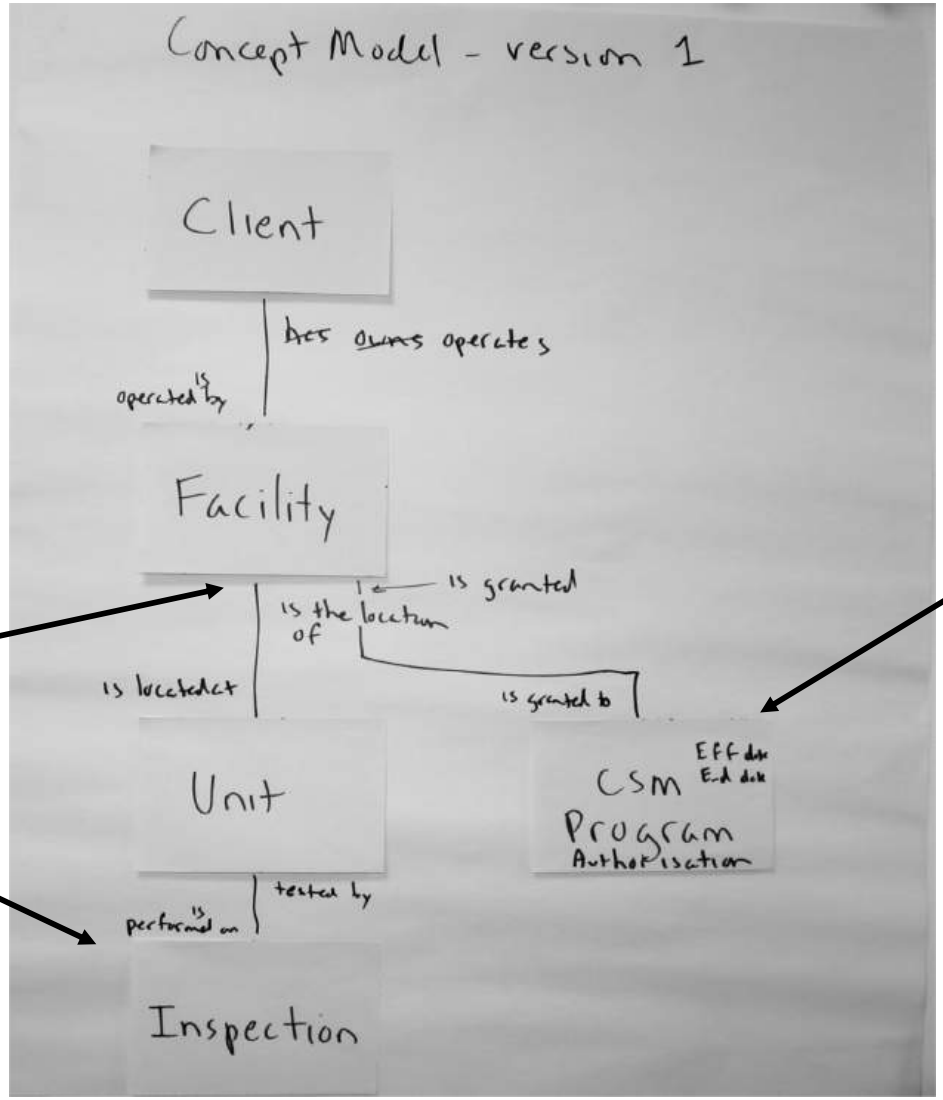
1. We arranged the entities / business objects by dependency
2. Then we drew relationship lines
3. Then we added a relationship name in each direction
4. Only then did we state (in words) the cardinality (1:1, 1:M, M:M) and then update the diagram with hash marks (⫞) and crowsfeet (⅄)



Definition -
A CSMP Authorisation is a permission (or license) to operate a self-managed safety program (a Client Safety Management Program) at a specific Facility, for a specified time period, usually 1, 2, or 5 years.

The CSMP Authorisation is "all or nothing" - it covers ALL the Units at a Facility.

Just boxes and lines, but raises important questions



Are Units permanently part of one Facility?

What do we Inspect?

What do we issue the Authorisation to?

Concept Model Version 1; state Assertions and challenge them

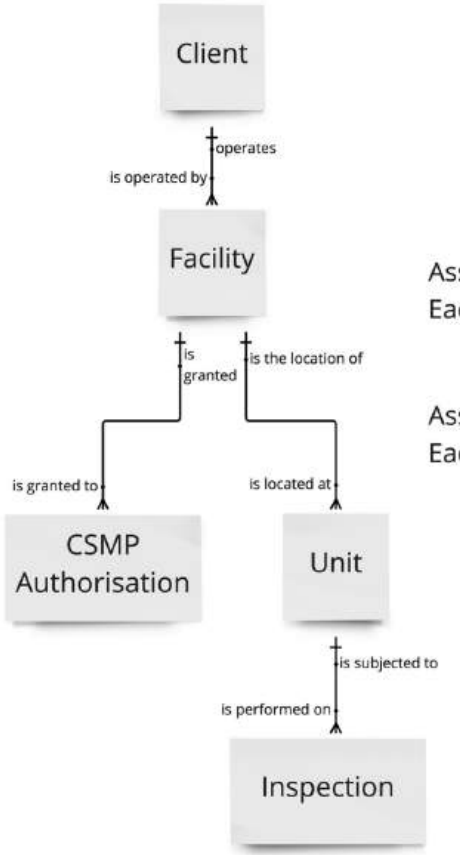
Now, state the relationships ***emphatically*** as Assertions. ***Each*** Client operates ***one or more*** Facilities! Then, ***challenge*** them!
Again, don't worry yet about ***optionality*** – whether the relationship ***must be*** or ***may be*** be present.
We only care now about the ***maximum*** – each ObjectA is related to a ***maximum*** of ***one*** or ***one or more (or many)*** ObjectB.

Assertion:
Each Client operates ___

Assertion:
Each Facility is operated by ___

Assertion:
Each Facility is granted ___

Assertion:
Each CSMP Authorisation is granted to ___



Assertion:
Each Facility is the location of ___

Assertion:
Each Unit is located at ___

Assertion:
Each Unit is subjected to ___

Assertion:
Each Inspection is performed on ___

Concept Model Version 1; revised Assertions from challenges

Now, state the relationships ***emphatically*** as Assertions. ***Each*** Client operates ***one or more*** Facilities! Then, ***challenge*** them!
Again, don't worry yet about ***optionality*** – whether the relationship ***must be*** or ***may be*** be present.
We only care now about the ***maximum*** – each ObjectA is related to a ***maximum*** of ***one*** or ***one or more (or many)*** ObjectB.

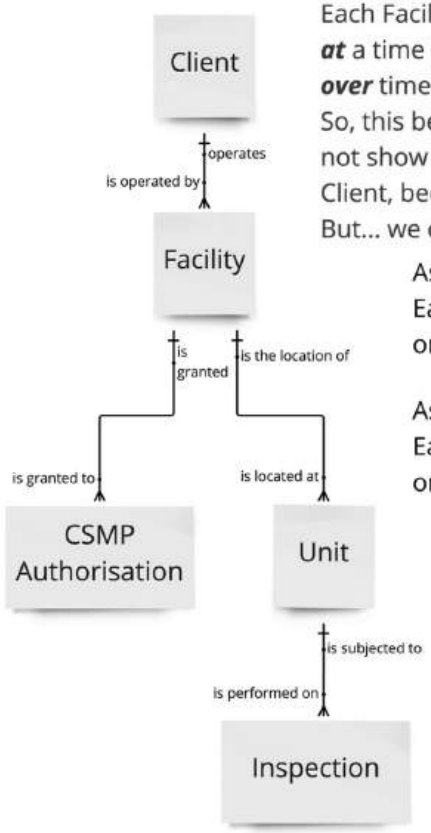
Assertion:
Each Client operates
one or more Facilities

Assertion:
Each Facility is operated by
one Client

Assertion:
Each Facility is granted
one or more CSMP Authorisations

One CSMP Authorisation ***at*** a time,
but one or more ***over*** time

Assertion:
Each CSMP Authorisation is granted to
one Facility



Each Facility is operated by one or more Clients
at a time (Joint Ventures) and
over time (changes in Ownership or Lease.)
So, this becomes a M:M relationship, and we should not show a Facility as being dependent on a single Client, because a Facility is an independent thing.
But... we don't always get our way!

Assertion:
Each Facility is the location of
one or more Units

YES, but one or more Facilities ***over*** time, because Units can move between Facilities. So, this becomes a M:M relationship, and we cannot show a Unit as being dependent on a single Facility, because a Unit is an independent thing

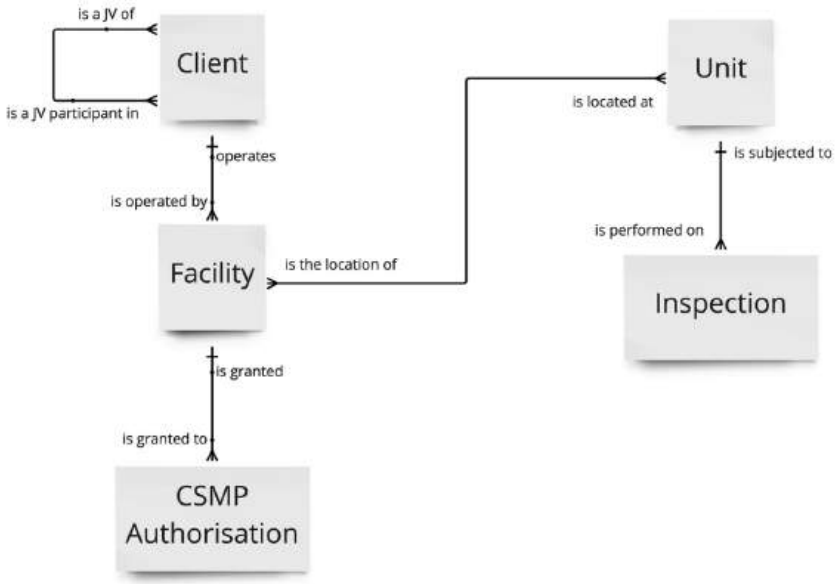
Assertion:
Each Unit is located at
one Facility

Assertion:
Each Unit is subjected to
one or more Inspections

Assertion:
Each Inspection is performed on
one Unit

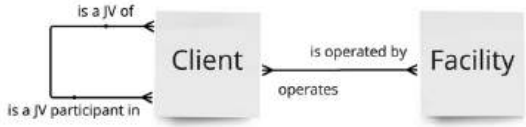
Concept Model Version 2; revised from challenging Assertions

Now we will re-draw the initial Concept Model based on changes that came from challenging the Assertions in Ver. 1.

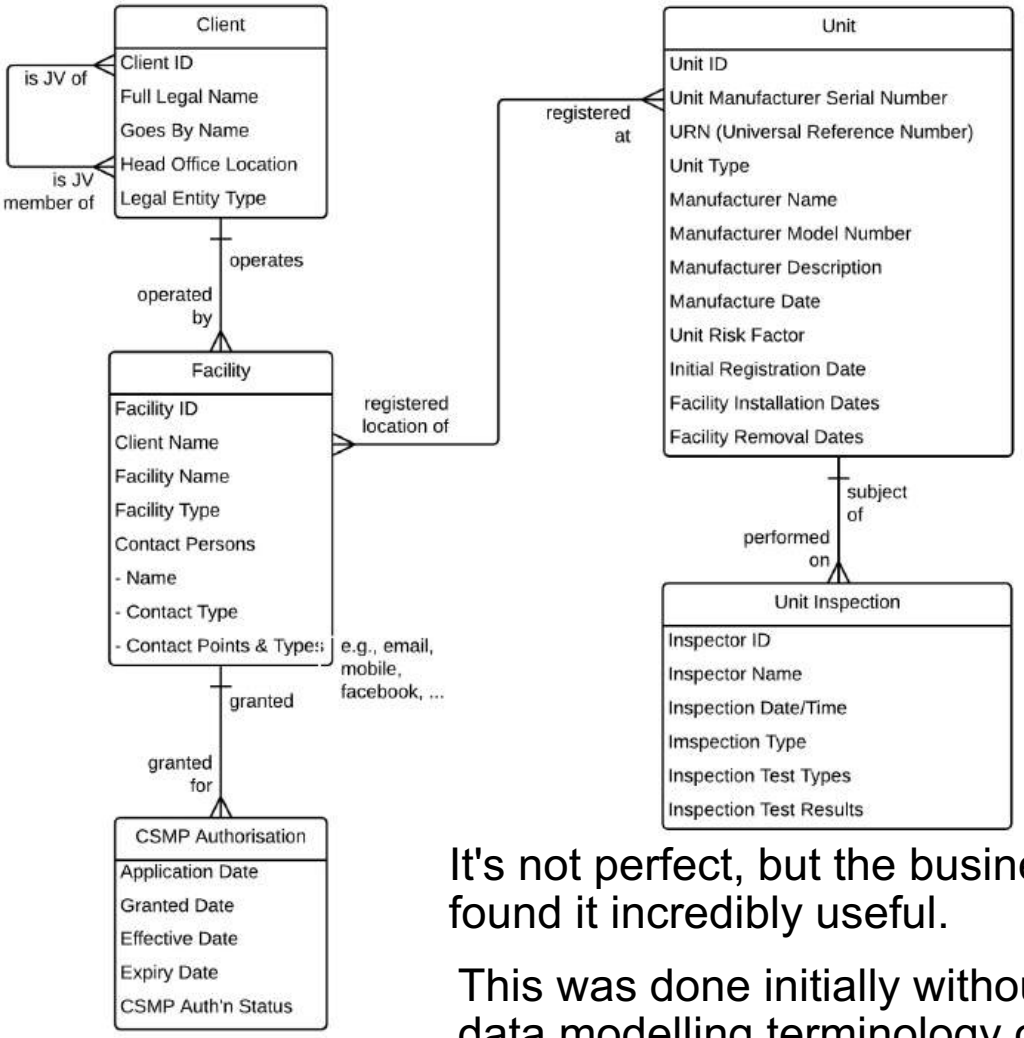


Note:
You don't always get what you *want* or what you think is the *right* thing in Concept Modelling. In this case the client (the Regulator) said they always wanted a Facility to be operated by ONE AND ONLY ONE Client.
If a Facility was operated by multiple Clients, they would require the Clients to form a new Joint Venture Client. This was to ensure that if there were legal difficulties, there was only ONE Client to go after.
Or, as they put it, "one throat to choke."

Later in the project, they realised they needed a history of the Clients that had operated a Facility, so the Client-Facility relationship became Many-to-Many, and Facility was modelled (correctly) as an independent Entity, as shown here:



"What do you need to know about the things in the Concept Model?"



Sketching this out was *fast, and* raised many questions that had not occurred to the client…

- Is there one CSMP per Client, per Facility, or some other basis?
- Do Units frequently relocate, or even turn up at another Client?
- What is inspected – the Facility or the Unit?
- Does the CSMP cover all or some Units at a Facility?
- …and MANY more…

It's not perfect, but the businesspeople found it incredibly useful.

This was done initially without any data modelling terminology or symbols!

Model took ~90 minutes

Summary – what an analyst can do with a Concept Model?

First, clarify language. (A platform)

Second, establish policies and rules.

And then, identify events and services, e.g.,
A **Unit** is…

- Registered (requiring the service "Register Unit")
- Loaded (requiring the service "Load Unit")
- Idled (requiring the service "Idle Unit")
- Reactivated (requiring…)
- Repaired
- Inspected
- Relocated
- Retired
- …

We did the same for Client, Facility, CSM Program, …

These are the *essential* capabilities.
In Business Analysis "essential"
means **what** with no reference to
who or **how**

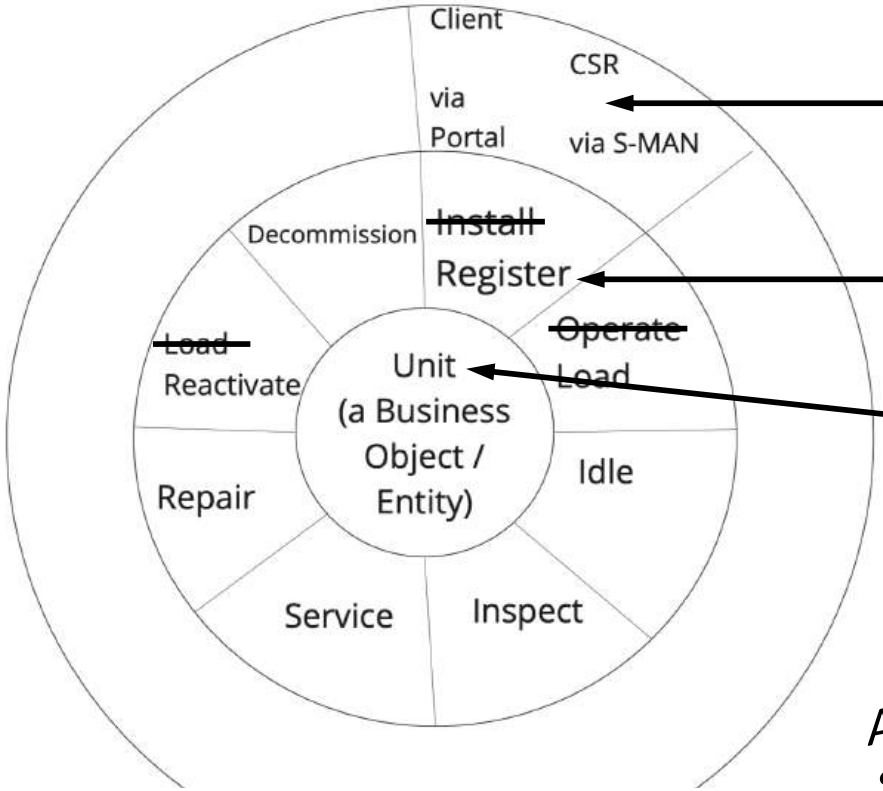
Something I *always* do when
evaluating/selecting COTS S/W

Identify Services (Events) then Use Cases / User Stories

Finally, we'll identify the Services (verb - noun pairs) we need, and the Use Cases / User Stories by which the Services will be accessed

What events happen to a Unit - what are the needed services? (Verb - Noun)

- ...
- ...
- ...
-



Who needs access to each Service, and How?

Use Case	*Use Case or User Story - add Who and How*
Service Specification (Events)	*Service (or Event) - add a Verb to the Noun*
Concept Model	*Entity or simply a "thing" - a core Noun*

A Concept Model is a great starting point for discovering your Services and Use Cases (User Stories)

Supports *Service-Oriented Business Analysis*

Clarify scope of the new process and identify participants

Trigger:
Client submits request to enter into a CSMP

Grant Client Safety Management Program

Accept CSM Application	Audit CSM Program	Verify CSM Equipment	Determine & Collect Equipment Fees	Determine & Collect Consultation Fees	Issue CSM Authorisation

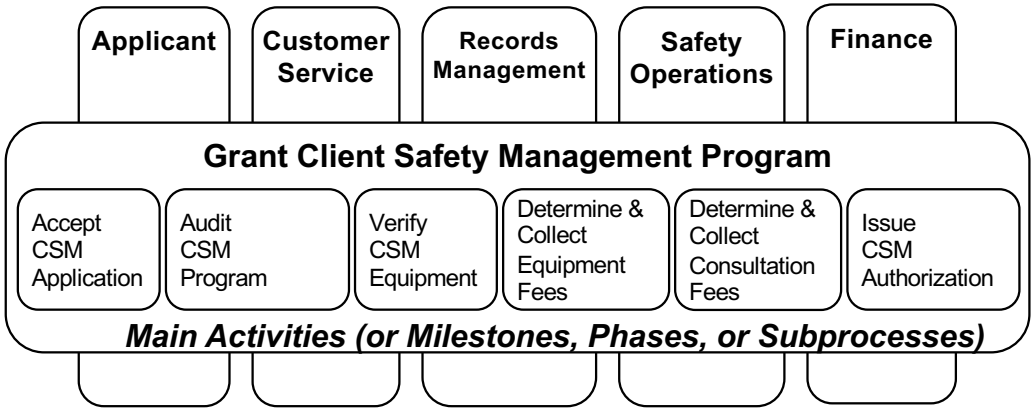
Main Activities (or Milestones, Phases, or Subprocesses)

Client Result:
Approval granted for a self-managed safety program.

Agency Result:
Revenue collected.
New participant in CSMP; confirmation that regulations are satisfied

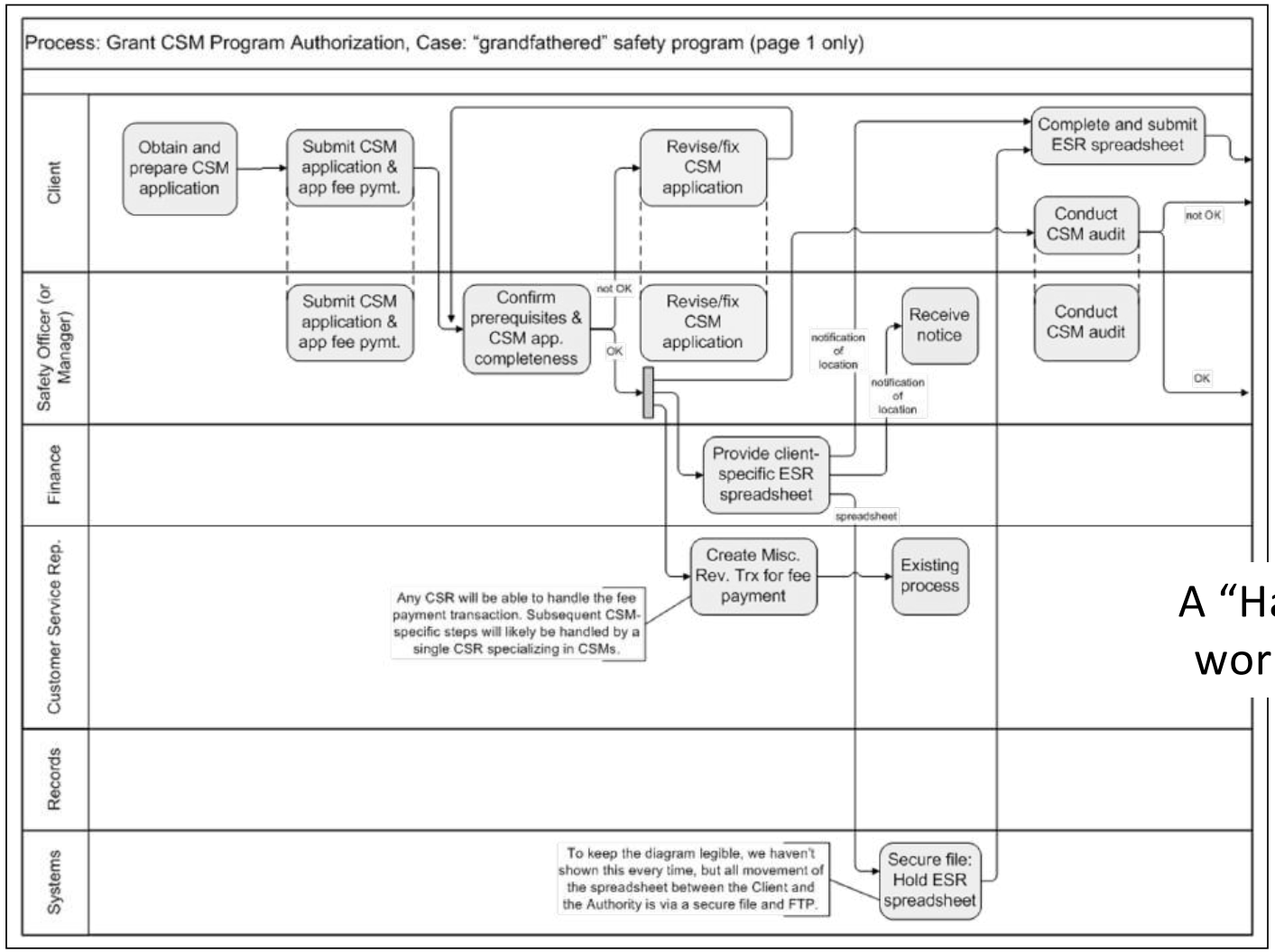
Cases:
- New
- Grandfathered
- Ownership Change

Process Scope Model – pure "what"…



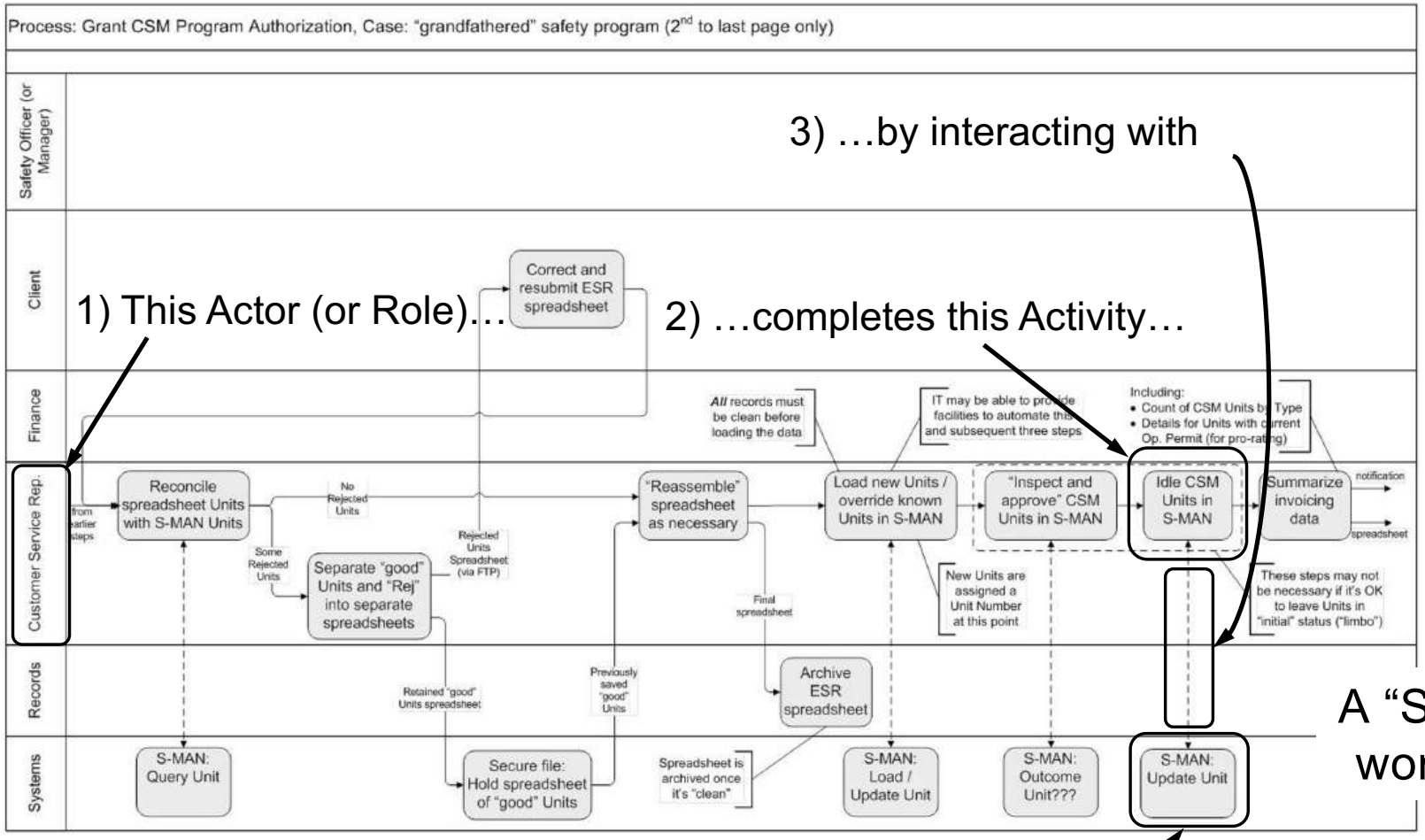
Process Summary Chart – simplified "what," plus "who"

The initial, business-friendly workflow model



A "Handoff Level" workflow model

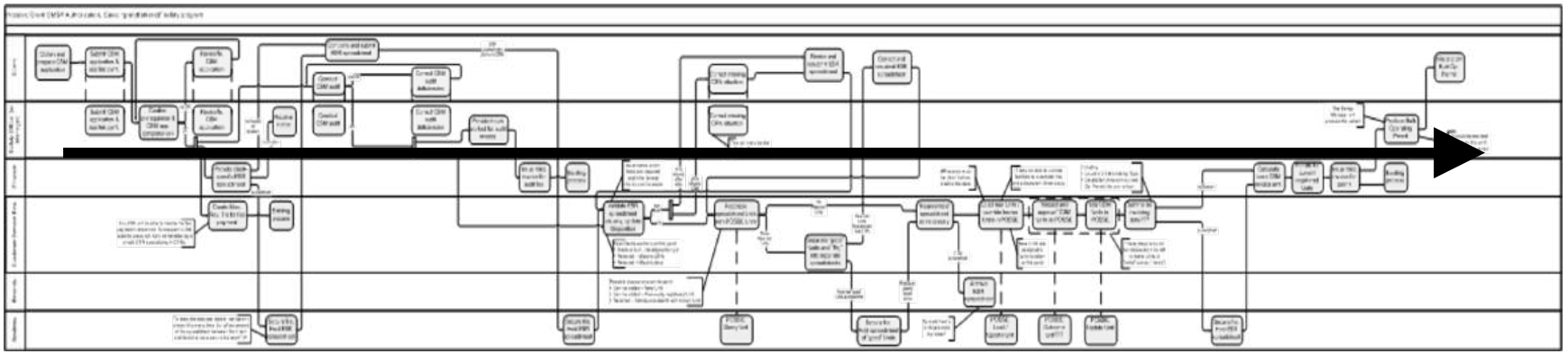
Then detail showing where use cases & services fit



A "Service Level" workflow model

Mission accomplished! Conclusions:

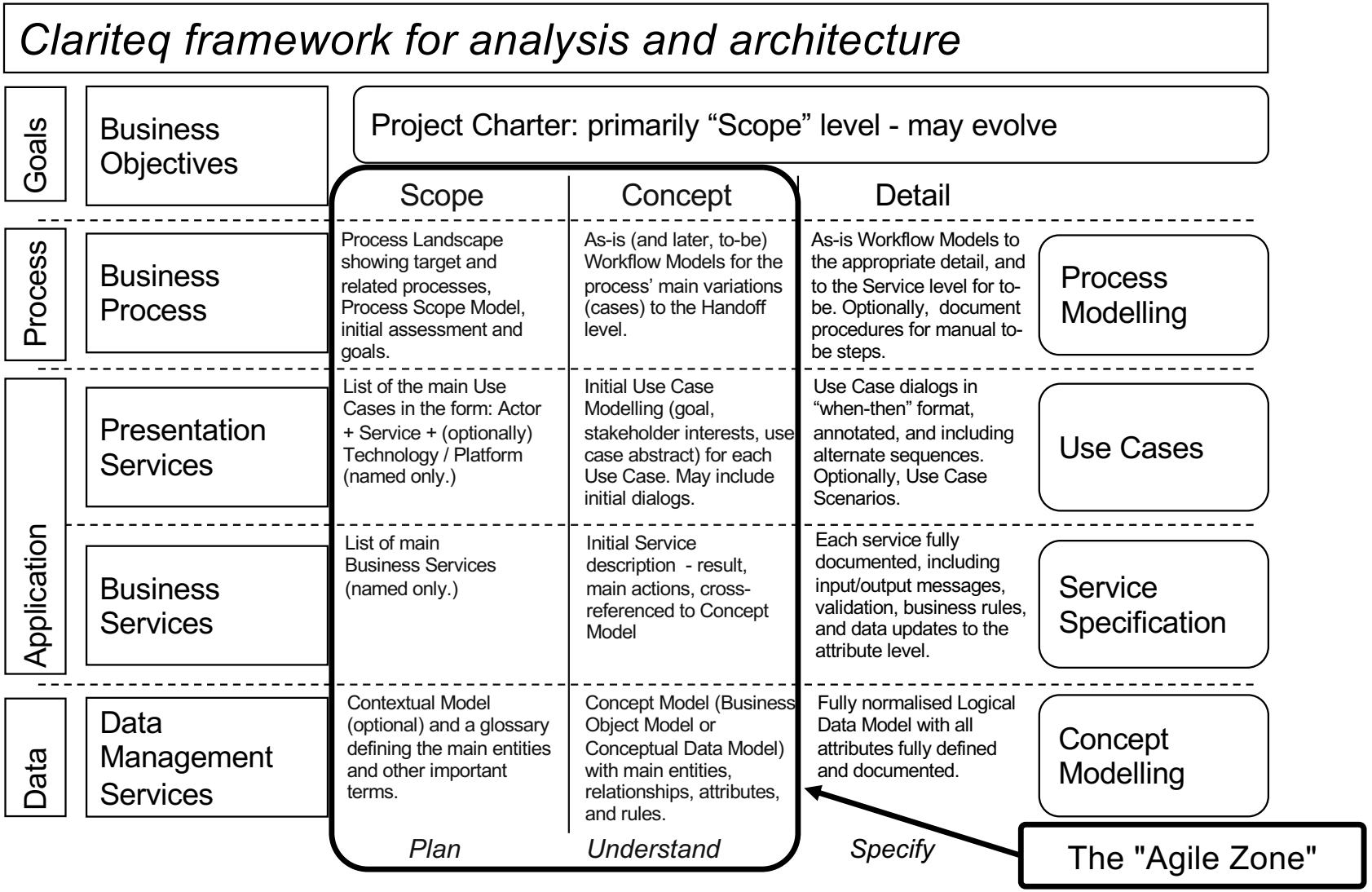
- "Plan A" rejected – agreement that Unit data *must* get into S-MAN
- “Plan B” (change the app) looks good, but the vendor estimates are *HIGH*
- “Plan B Minus” (existing functionality plus CSR work) is *worth the cost*



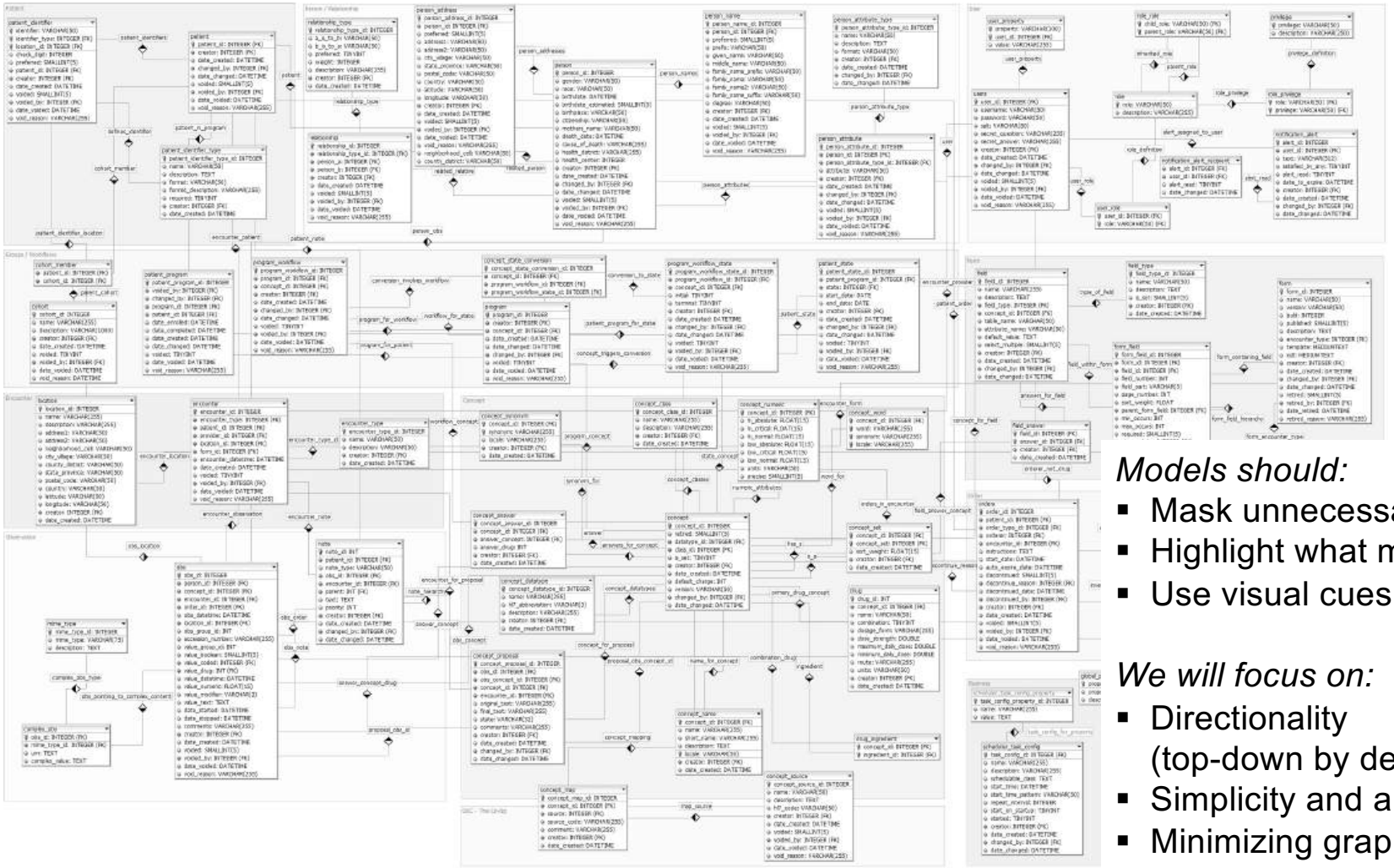
1. If requirements, issues, assumptions, etc. are in lists, people will argue endlessly; if they are in an *integrated* and *understandable* set of models, it’s much harder to dismiss the reality of the situation
2. Process Models, Use Cases, Service Specs, & *Concept Models*: ***essential!***

Concept Modelling supports Agile

Clariteq framework for analysis and architecture



Concept Modelling principles



Models should:

- Mask unnecessary detail
- Highlight what matters
- Use visual cues consistently

We will focus on:

- Directionality
  (top-down by dependency)
- Simplicity and abstraction
- Minimizing graphic "widgets"

The basics: ERA – *Entities*

A distinct thing about which the enterprise must maintain facts in order to operate.

Criteria –

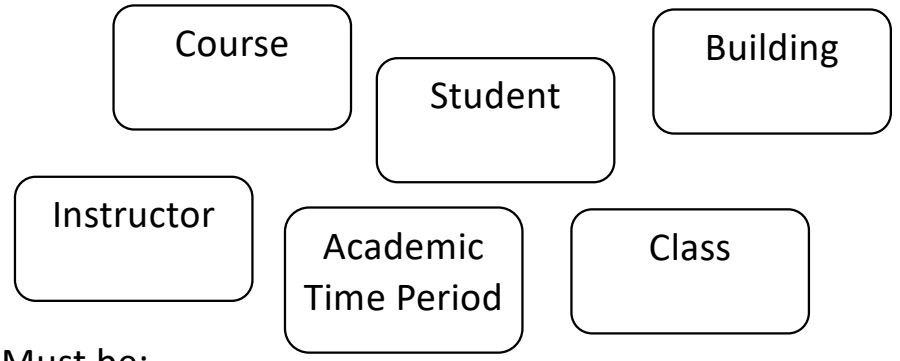
- *singular noun* – we can talk about *one of them* ("Employee," not "Staff")
- *multiple* instances
- must *need to* and be *able to* keep track of *each* instance
- has *facts* (attributes & relationships) that must be recorded
- makes sense in a *"verb-noun"* pair
- *NOT* an *artifact* like a spreadsheet or report

Fundamental to business analysis.
Entities are the things

- processes act on
- applications manipulate
- databases record
- BI & reporting tools provide info about

Two basic types:

- independent – can stand alone
- dependent – must have one or more parents

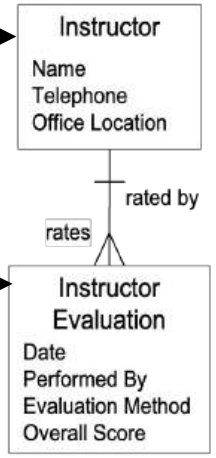


Must be:

- named: business-oriented noun / noun phrase
- defined: "What is one of these things?" or "What do you mean by _____?"

Independent
- "strong"
- no relationships "on top" (no parents)

Dependent
- "weak"
- one or more relationships "on top" (to parent(s))



Naming and definition – the essence of Concept Modelling

Organisations need a *common language* more than ever…

- Data integration (data lake, data mesh, data fabric, data virtualisation, data warehouse, operational data store, …)
- Mergers/acquisitions/partnerships/…
- Business analysis – most requirements can't be stated without using a term from the Concept Model
- Performance measures, e.g., KPIs

Note – it often works best if you don't start by talking about *Concept Modelling* or *Data Modelling…*

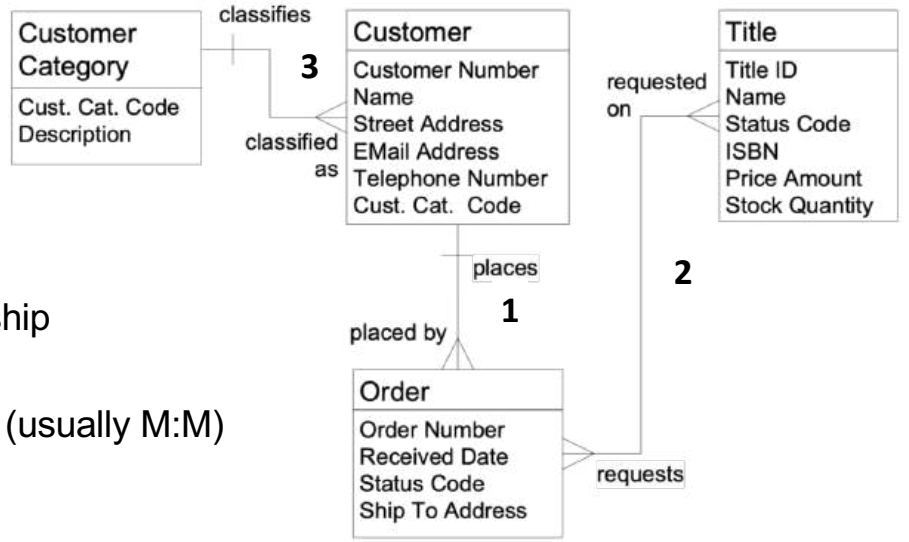
"Now! That should clear up a few things around here!"

The basics – ERA – *Relationships*

An association between Entities that the business must keep track of

Named in both directions

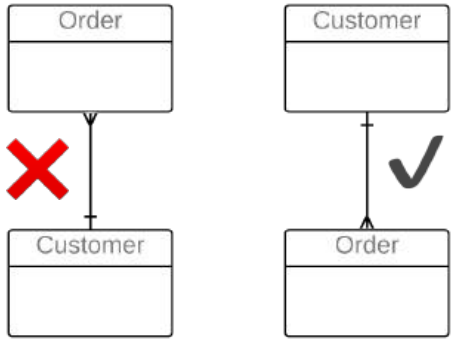
- verb-based phrase
- the line tells us they *are* related,
  the name tells us *how*



Different types of relationships

1. parent-child or characterising – "bottom to top" relationship
   from an entity to a dependent entity (1:M)
2. associating – "side to side" relationship
   between entities that are not dependent on one another (usually M:M)
3. classifying – "side to side" relationship
   from reference data to the classified entity
   (seldom shown in the Concept Model)

Dependency is shown top down – No Dead Crows

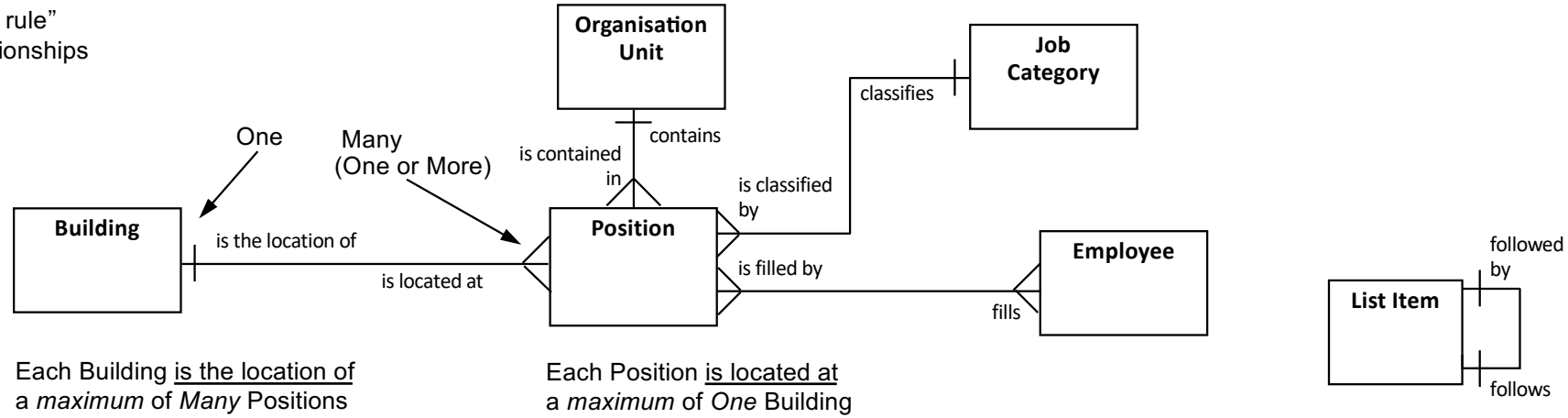


Relationships have rules

- cardinality – 1:1 (almost certainly wrong,) 1:M, M:M
- optionality – relationship *may be* present or *must be* present
  (not shown until later, in the logical model)

Relationship cardinality (maximum cardinality)

A kind of “business rule”
that applies to relationships



Each Building <u>is the location of</u>
a *maximum* of *Many* Positions
(or, ... *One or More* Positions)

Each Position <u>is located at</u>
a *maximum* of *One* Building

One to One (1:1) relationships in a conceptual or logical model are almost invariably an error except in recursive relationships.

To determine cardinality, *first name the relationships properly, and only then:*

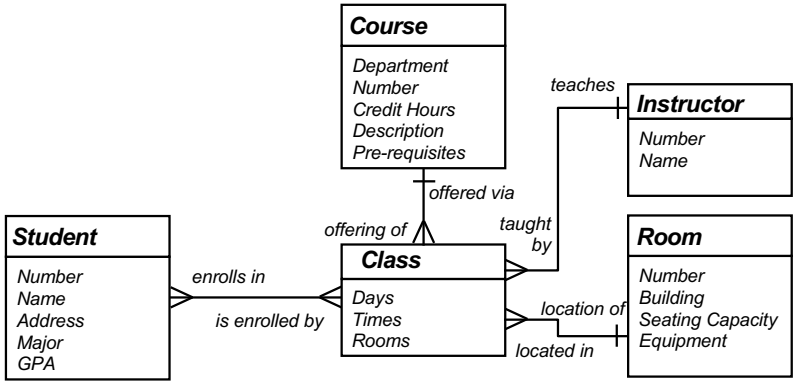
- for each entity, ask
  “Can one of these be related to a *maximum* of *One* of the other or a *maximum* of *Many* of the other?”

- record the answer (One or Many) at the “other” end;
  "One or More" works better for businesspersons than "Many"

- possibilities – 1:1 (error), 1:M (common), M:M (more work, eventually)

Relationships – state as assertions

1. You *must* state the relationship name as an assertion, in both directions (for clarity and confirmation)
2. Be clear on whether cardinality is “one” or “one or more” (don't worry about “may” and “must” at first)
3. *Emphatically* begin the assertion with the word “Each”
4. Try it on this model…



Note –
A Class is a scheduled offering of a Course during an Academic Time Period, e.g. a Semester or an Academic Year.
During an Academic Time Period there may be one or more Classes for a Course. Each Class is held on specific Days (e.g. Monday & Wednesday,) at specific Times (e.g. 10:30-11:30,) in specific Rooms (e.g. AQ3100 & CC7232.)

Each Instructor teaches one or more Classes (Sounds good…)

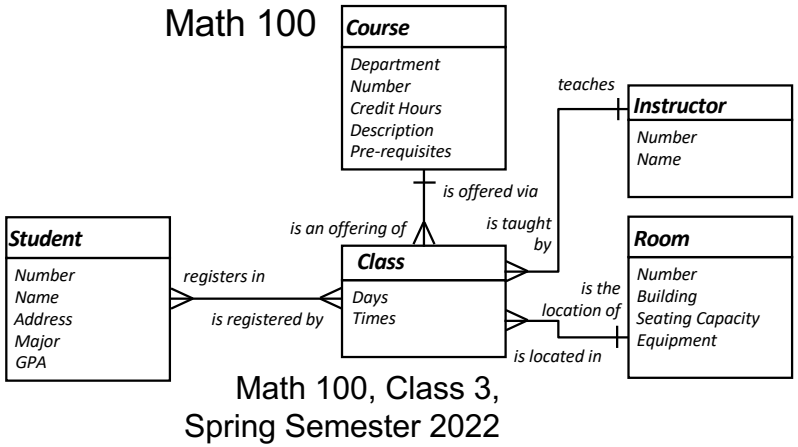
Each Class is taught by one Instructor…

1. Student-Class
2. Course-Class
3. Instructor-Class
4. Room-Class

Which ones might be *incorrect?*

Discussion – state as assertions, identify incorrect ones

In some universities, Students in the same Class could be earning credit for *different* Courses – it could be a M:M relationship.



1. Student-Class
   Each Student *registers in* one or more Classes
   Each Class *is registered by* one or more Students ✓
2. Course-Class
   Each Course *is offered via* one or more Classes
   Each Class *is an offering of* one Course ? – depends on Policy
3. Instructor-Class
   Each Instructor *teaches* one or more Classes
   Each Class *is taught by* ~~one~~ One or More Instructors
4. Room-Class
   Each Room *is the location of* one or more Classes
   Each Class *is located in* ~~one~~ One or More Rooms

Each Class is taught by One or More Instructors. On what basis?

- team teaching
- backup
- replacement
- specialist
- guest lecturer
- lab assistant
- teaching assistant
- …

We are discovering reference data to describe an Instructor's Role.

All of this has an impact on the Business Process! It's easier to resolve these rules before working on the Process.

The basics: ERA – *Attributes*

A fact about an entity recorded as a piece of data.
If facts are needed about a relationship,
we will later (in the Logical Data Model) create an entity that represents the relationship and records its facts

Like Entities, attributes are named and defined

Not every possible fact – just the ones we need

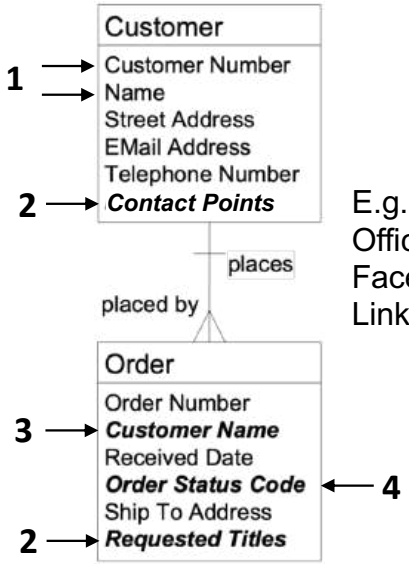
Have properties that we address during the transition from Concept Model to Logical Data Model

1. base or fundamental attribute
2. single-valued vs. multivalued –
   one attribute can have multiple values,
   at a time or *over* time
3. fundamental vs. redundant –
   the same value is recorded multiple times
   in different entities
4. "user-entered" vs. constrained –
   attribute can only come from a limited set,
   as in a drop-down list

Traditionally alphanumeric data; now includes richer types e.g., retinal scan image or voice audio clip

Eventually, an entity will contain only base / fundamental / *essential* attributes:

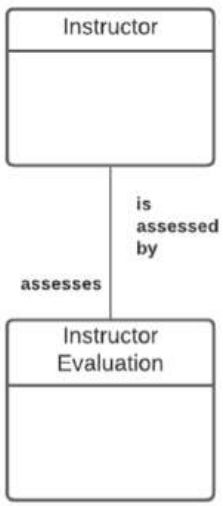
- an *essential fact* about that thing (entity)
- *not* multi-valued
- *not* redundant
  (a redundant attribute is an attribute that is really an essential fact about a *different* entity, so its value is recorded multiple times, redundantly)
- and *not* derived or calculated from other attributes; otherwise, clearly flagged "derived"



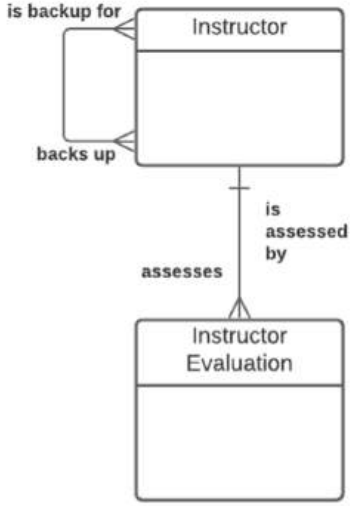
E.g., Mobile Number, Office Telephone Number. Facebook ID, LinkedIn ID, …

For reference – the Information Engineering symbol set

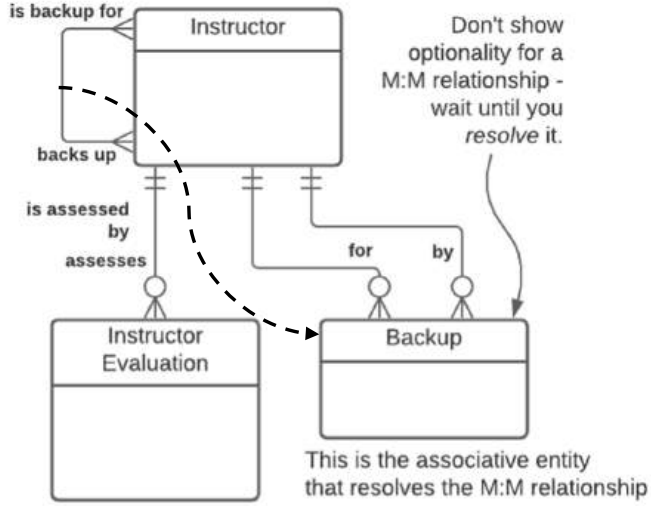
- This symbol set was refined and developed by Clive Finkelstein.
- Known in some tools as the "Martin IE" symbol set.
- Strengths are:
  - symbols are not "overloaded" – they explicitly convey only *one* idea.
  - can show as much or as little as needed in terms of rules.



The two entities are related - that's all this shows

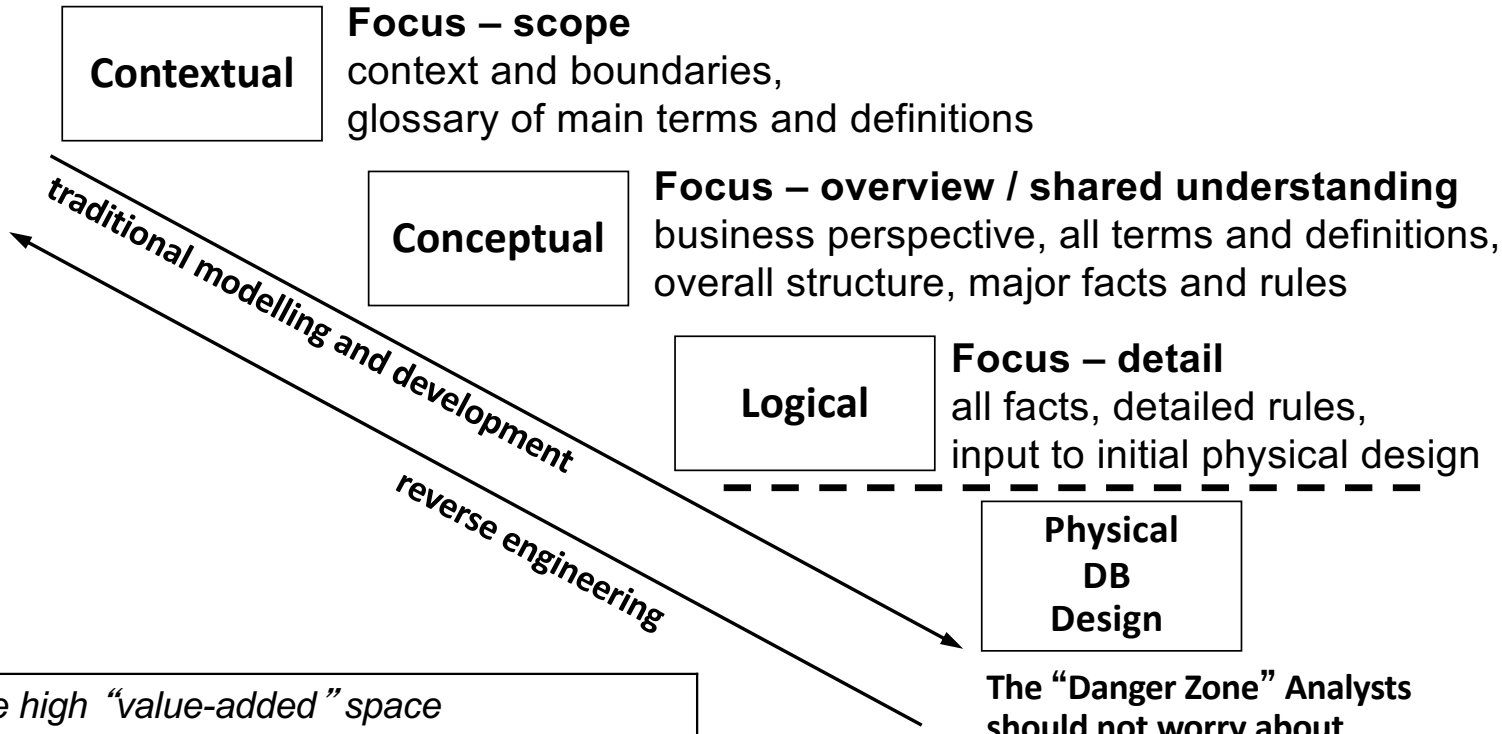


There is a 1:M relationship from the parent entity (business object) to the child entity (business object.) Optionality is not shown.



There is a 1:M relationship from parent to child, *optional* for the parent and *mandatory* for the child. (The parent *may* have a child, the child *must* have a parent.) This is by far the most common relationship in a logical model.

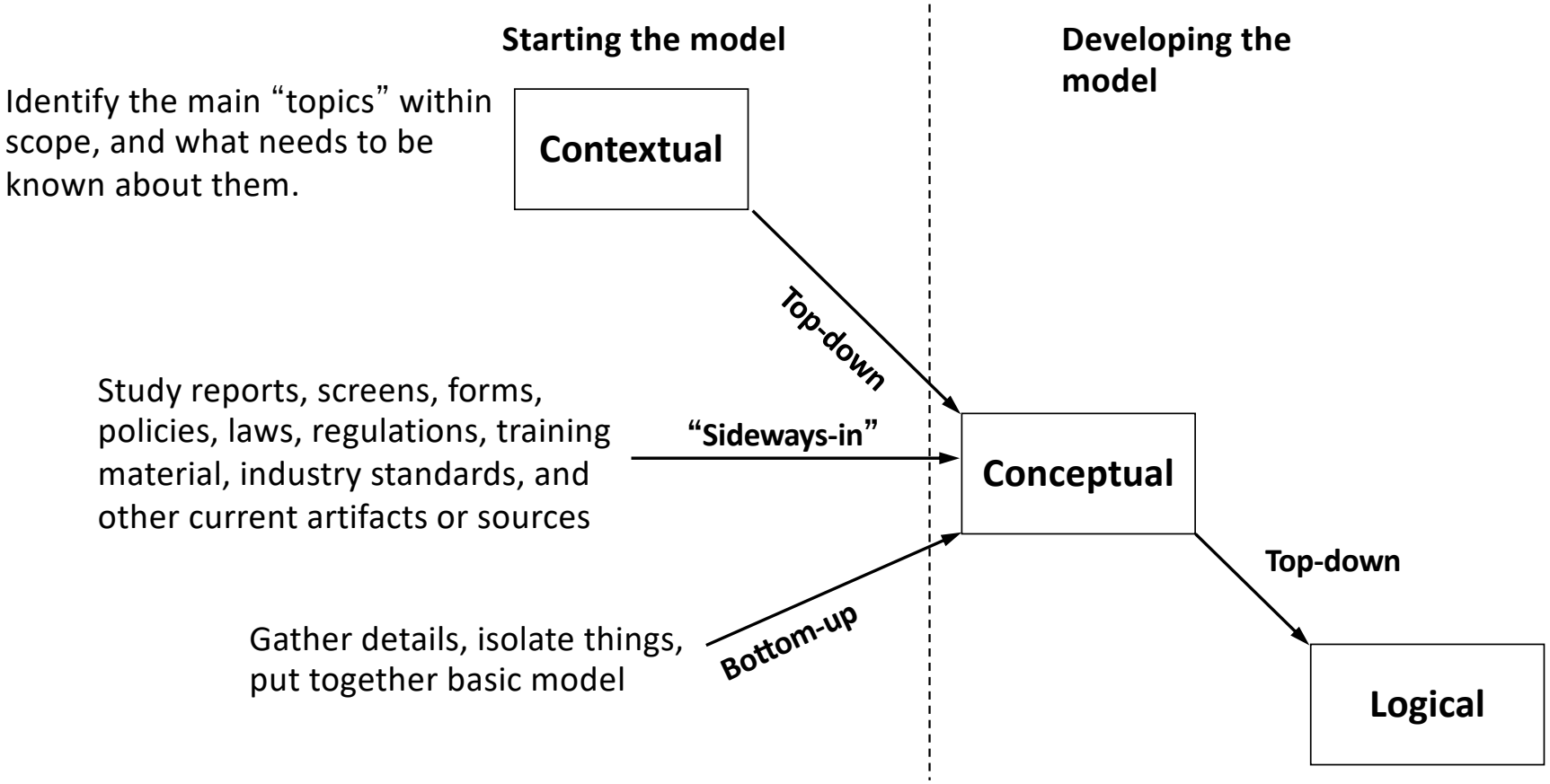
A natural progression



!	*Get into the high "value-added" space*

- Contextual – helpful for large models
- Conceptual – a great way to add value
  - Improve communication among all players
  - Highlight disconnects – terms, rules, scope, …

Different ways to get started



Some advice on starting the concept model

<u>Don't</u> begin with a lecture on data modelling (but I have a painful story that had a happy ending)

If you can, don't even mention "data modelling"

We use "terminology analysis" – starting with the <u>nouns</u> – at the outset of every project. This was demonstrated earlier in the Client Safety Management example.

Starting a data model bottom-up

1) Interview business representatives about their area:
   mandate and activities, goals and objectives, issues and opportunities,
   needs and wants, likes and dislikes, etc.…

   Nod sympathetically but ignore it all (almost!)

   Instead, capture "terms" – anything that goes by a name.
2) Later, write each term on a large Post-it
3) In a facilitated session, participants sort terms into categories:
   - Things (entities, but don't use the term… yet)
   - Facts about things (add new "thing" if it's not there already)
   - "Other stuff"

As needed, introduce criteria to be a"thing" (an entity)

"Other stuff" includes:

- Metrics
- Organisations, departments, jobs, roles, …
- Processes, functions, activities, tasks, …
- Systems, tools, equipment, mechanisms, …
- Reports, forms, screens, queries, …
- Other – too vague, only one instance,
  a "fact of life," not a thing we track, etc.

Exercise 1: Starting a conceptual data model

The assignment:
The following describes project tracking at Amalgamated Automaton. Read it over and be prepared to discuss the things about which the business needs to record information, and the important facts about them. The instructor will lead the development of an initial data model.

Amalgamated Automaton, Inc. has a growing Information Systems department. Until recent years, the department was concerned almost entirely with selecting, installing and maintaining purchased software packages. Recently, however, the focus has shifted towards the in-house development of application software.

One of the problems confronting the IS department is that they have no base of historical data to aid in trend analysis or estimating development effort, nor any effective means of charging back development costs. The proposed solution is to develop a simple Project Tracking System, which will work in conjunction with the existing Personnel and General Ledger Systems.

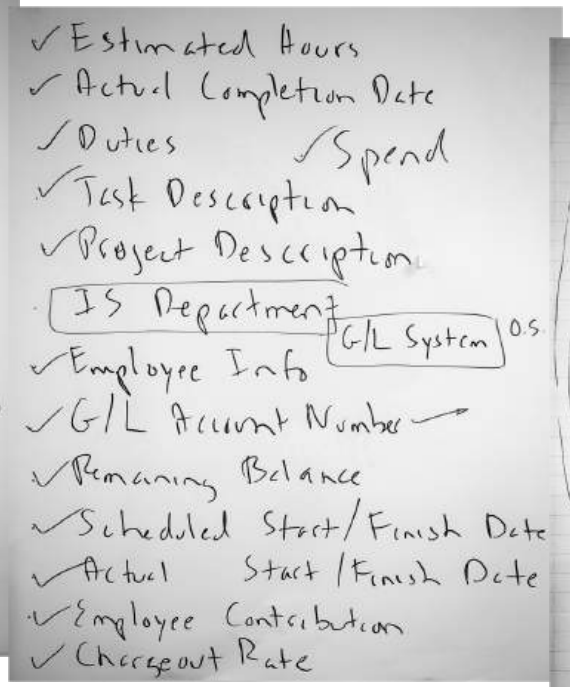
When a development project is initiated, a project name and a short description are recorded, among other things. Soon, before any further work is done on the project, a new account is created on the G/L System, identified by a G/L account number. Project costs will be charged to this account, and the project budget is recorded as the initial account balance in dollars.

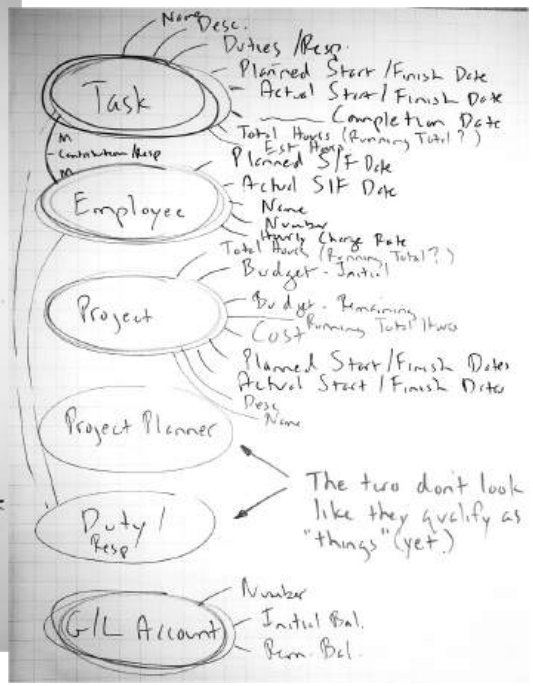
Project planners break a project down into many tasks, perhaps hundreds. A typical project task might be “Test Order Entry Module”. Some of the facts which are required about tasks include a brief task description, estimated work hours, and the scheduled start and finish dates.

Eventually, individual employees are assigned responsibility for the tasks. Some tasks will be the responsibility of many employees, and an employee might be assigned to many tasks. As each employee is assigned to a project task, their planned start and finish dates, their contribution to the task (not a “kind of work,” but their specific duties on the task – e.g., “Develop test scripts”), and the estimated number of hours they are to spend on the task are recorded. Employee information such as the employee name and number are available from the existing Personnel System, although it will have to be modified to record the employee's hourly charge out rate.

When an IS employee begins work on a new task, their actual start date is recorded. A running total of the number of hours that they have worked on each started task is updated regularly. At the same time, the remaining balance in the project account is updated. When an employee completes a task assignment, the actual completion date is recorded.

Workshop example





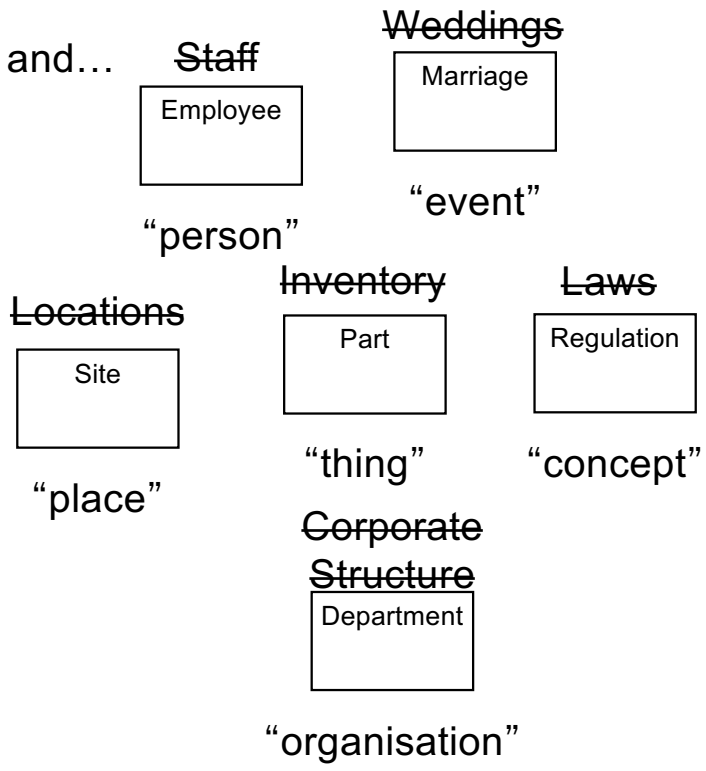
Introduce "thing criteria" as necessary:

- *singular noun* – can talk about *one of them* (*Worker* not *Staff*, *Item* not *Inventory*)
- *multiple* instances
- must *need to* and be *able to* track *each* instance (uniquely identify each)
- has *facts* that must be recorded
- makes sense in a *"verb-noun"* pair
- *NOT an artifact* like a spreadsheet or report (not a Call Log or Worker Directory or…)

Entities – more specific criteria

An *entity* is a distinct thing the business *needs* to know about,
often described as a *person, place, thing, event, concept,* or *organisation* and…

- is named with a *singular noun* that implies a single instance
  - not a plural or collective noun, list, set, collection, report, etc.
  - we can discuss "one of them," e.g. "Weather" is not a good name
- has *multiple occurrences* (or instances)
  - *need* to and *can* keep track of (differentiate) each occurrence
- has *facts* that must be recorded, e.g.
  - *Student* attributes: Number, Name, Birth Date, Major, GPA, …
  - *Student* relationships: "majors in" *Subject*, "enrolls in" *Section*
- is acted on by *processes,* so they make sense in a "verb-noun" pair
- refers to the *essence*, not the implementation ("What, not who or how") –
  the most common error is to identify artifacts (forms, reports, spreadsheets, …) as entities!



Let's look at some common errors…

Identifying Entities – three common errors

1. Treating an "artifact" (a spreadsheet, report, web page, form, etc.) as an Entity – an Entity is a fundamental thing – *"what"* – with no reference to *"who or how."*
   Artifacts typically contain attributes from *multiple* Entities
   e.g., *"Admission Request Form"* or *"Orders Summary Spreadsheet"*
   or *"Daily Call Log"* or *"Class Roster" or "Materials List Fax"* or…

2. The "types vs. instances" problem – failing to clarify if the Entity deals with *types* of things (or *categories* or *kinds* or *classes* of things)
   vs. specific *instances* of things
   e.g., "*Test"* is this a *type of Test*, or a *specific instance of a Test?*

3. Identifying an Entity that exists in the real world, but whose *instances* can't be uniquely identified
   e.g., *"Transit System Passenger"*

Types vs. Instances – "What do you mean by a Bus?"

A category of Bus – a "meta-Type?"
(transit, articulated, intercity, minibus, …)
A Make and Model of Bus – a Type?
An individual Vehicle? – an Instance?

Model	Length	Width	Introduced
Xcelsior[18]	35 feet (11 m) 40 feet (12 m) 60 feet (18 m)	102 inches (2.6 m)	2008
MiDi	30 feet (9.1 m) 35 feet (11 m)	96 inches (2.4 m)	2013

"What do you mean by a Bus?"

254 British Properties

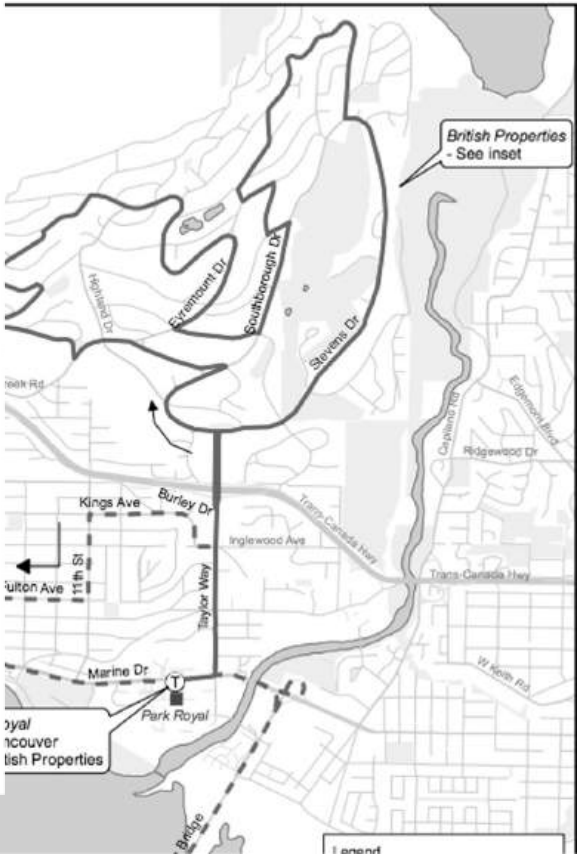
Inbound From Glenmore and Bonnymuir via Bonnymuir, Stevens, Taylor Way to Park Royal terminus (extends to Downtown Vancouver during Monday-Friday peak hours).

Outbound From Park Royal (from Downtown Vancouver during Monday-Friday peak hours) via Marine Drive, Park Royal South, Taylor Way, Southborough, Eyremount, Cross Creek, Chartwell, Crestwell, Eyremount, Fairmile, Southborough, King Georges Way, Robin Hood, Kenwood, St. Andrews, Bonnymuir to Glenmore terminus.

Park Royal to British Properties and return to Park Royal

MONDAY TO FRIDAY

Connecting Buses Leave Downtown Vancouver	Leave Park Royal	Leave Eyremount at Highland	Leave Bonnymuir at Glenmore	Leave Eyremount at Highland	Leave Marine at 14th	Arrive Park Royal	Arrive Downtown Vancouver Connecting Buses
6.35	6.53R		7.03	7.15	7.31	7.34	7.54
6.45	7.23R		7.33	7.45	8.01	8.04	8.24
7.47	8.07R		8.17	8.28	8.44*	8.47	9.16
8.20	8.40	8.53	9.06		-	9.15P*	9.41
9.22	9.47P	10.00	10.13		-	10.22P*	10.43



A Bus Route?
A Bus Route Scheduled Departure
An instance of a Bus Route Scheduled Departure?

<u>Never</u> be afraid to ask "What do you mean by…?"

Discussion – good Entity or not?

Which of the following might ***not*** be valid entities?
And if not, *why* not?

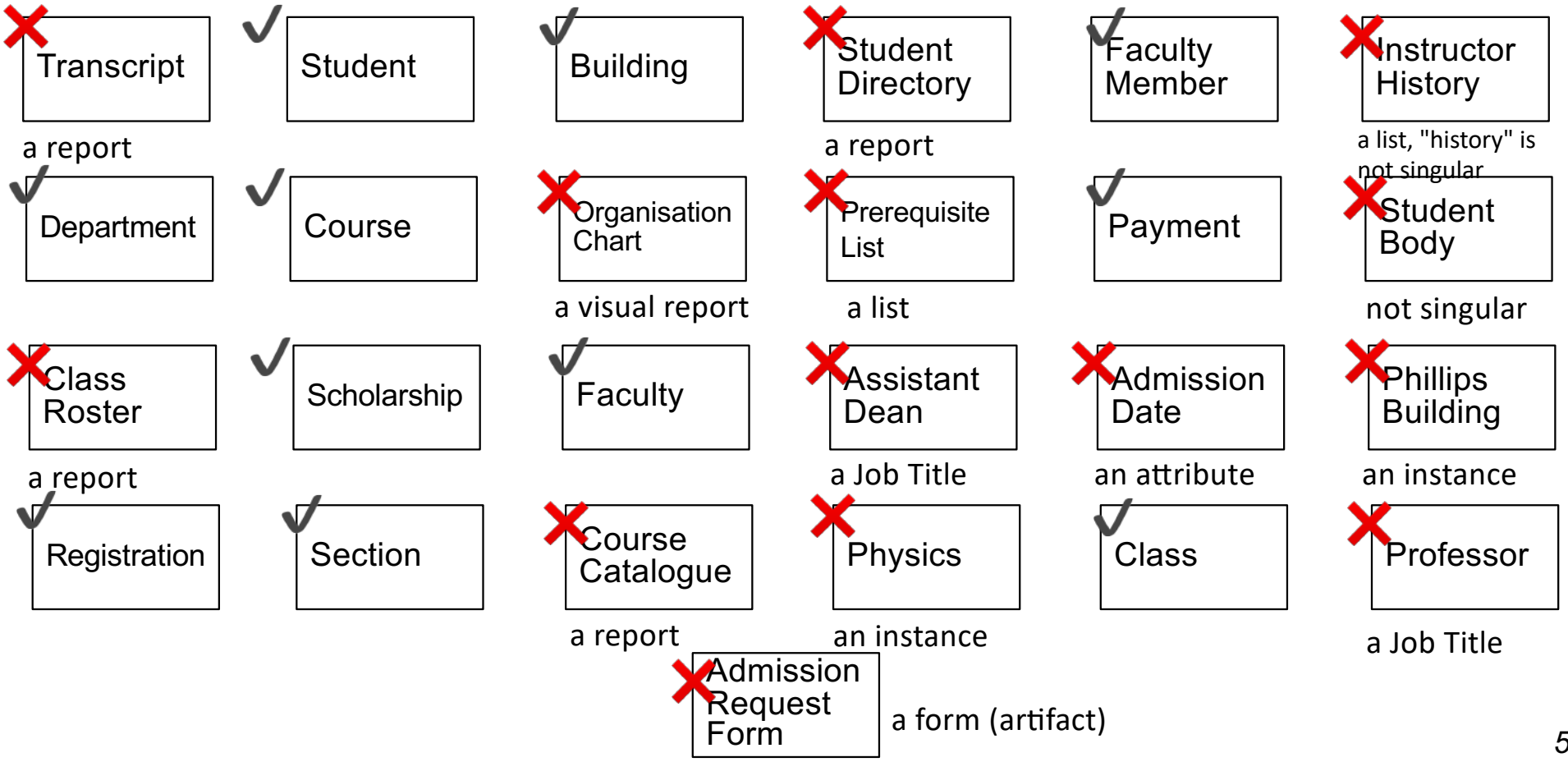
Transcript	Student	Building	Student Directory	Faculty Member	Instructor History
Department	Course	Organisation Chart	Prerequisite List	Payment	Student Body
Class Roster	Scholarship	Faculty	Assistant Dean	Admission Date	Phillips Building
Registration	Section	Course Catalogue	Physics	Class	Professor

Admission Request Form

Thinking space…

Discussion – good Entity or not?

Which of the following might ***not*** be valid entities?
And if not, *why* not?



Entity definition – bad example then a good format

~~**Customer**
We have a variety of Customers that operate in multiple geographies, and these must be tracked in order to consolidate purchasing statistics and enable our rating process to identify our best Customers.~~

Not a good definition

- Interesting background and miscellaneous points
- *Doesn't* answer the question "What is *one* of these things?"

Customer
1. A Customer is a person or organisation that is a past, present, or potential user of our products or services.
2. Current examples include Solectron (contract manufacturer,) Cisco Systems (OEM,) Arrow Electronics (distributor,) Best Buy (retailer,) M&P PCs (assembler,) and individual consumers.
3. Excludes the company itself when we use our own products or services but includes cases where the Customer doesn't have to pay (e.g., a charity.)

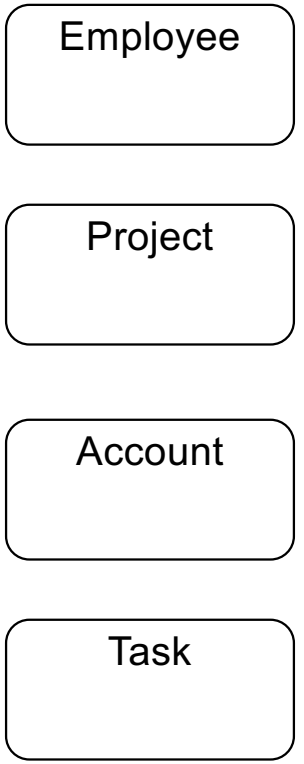
Entity definition format:

1. A description of which real-world things will be included in scope.
   This might be developed from a list of standard "thing types" – person, organisation, request, transfer, item, location, activity, etc.
   Be sure to identify any specific inclusions ("This includes…" or "This is...")
2. Illustrate with examples:
   - 5 – 10 sample instances
   - diagrams or scenarios
   - illustrations such as reports or forms
3. Interesting points – anomalies, synonyms, common points of confusion, etc.
   May include specific exclusions ("This excludes…" or "This is not...")

Discussion – starting an Entity definition

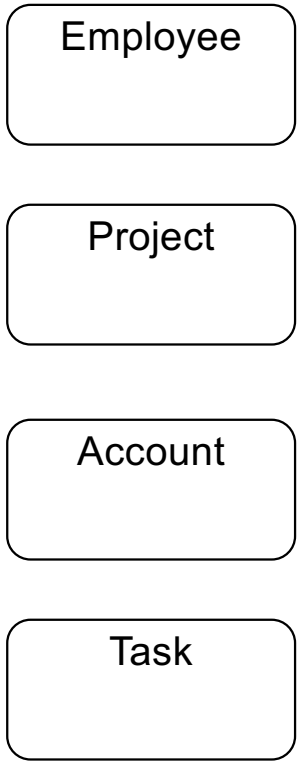
*"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Employee" means?*

-
-
-
-
-
-
- *...*



Brainstorming space

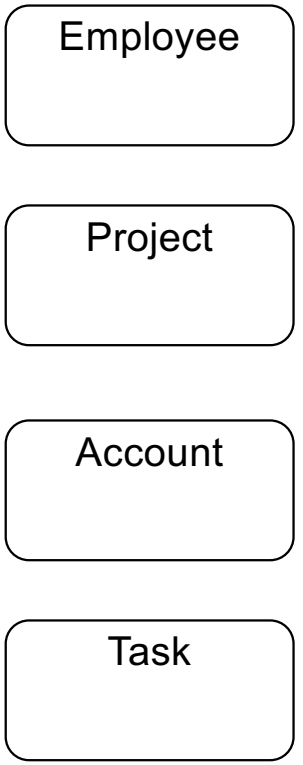
*"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Employee" means?*



Starting an Entity definition

*"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Employee" means?*

F/T vs. P/T?	– *Both*
Only IS Department?	– *No*
Include management, or only individual contributors?	– *Yes, everyone*
Still in recruitment (an applicant)?	– *No*
Onboarded? on probation? active? retirees?	– *Yes, all*
Include contractors, student interns, vendor staff, etc.?	– *Yes, all*
Volunteers?	– *Yes*
A type of worker (DBA or tester) or a specific person?	– *No, only a specific person*
A robotic, automated, or AI agent?	– *No, only a real person*



Defining the Entity "~~Employee~~" – "Worker"

Definition format:

1. A description of which real-world things are within in scope, and any specific inclusions ("This *includes*…" or "This *is*...")

2. Illustrate with examples – 5 to 10 sample instances or types

3. Interesting points – anomalies, synonyms, common points of confusion, etc.
   May include specific exclusions
   ("This *excludes*…" or "This *is not*...")

Worker (renamed from Employee):

A *Worker* is a person, whether or not directly employed by *the company,* but with some sort of employment contract or arrangement, who has been or may be assigned to a Project.

Worker includes:

- Full or Part-time Employees who have been onboarded, including Probation, Active, Seconded, Suspended, Retired…
- Contractors
- Consultants
- Student Interns
- Vendor Staff Persons
- Company Owners and Managers

Key points:

- "Worker" was chosen as the entity name because it is more generalised than "Employee."
- A Worker may not necessarily be billable on a Project, e.g., a non-chargeable Subject Matter Expert or Volunteer
- Worker excludes:
  - Job Roles, e.g., DBA or Technical Writer
  - Robotic, Automated, or AI Agents (this might change)

Another example – starting an entity definition for Task

*"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Task" means?*

-
-
-
-
-

Worker

Project

Account

Task

Another example – starting an entity definition for Task

"Can anyone think of examples that might surprise someone else – that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Task" means?

Key points that typically arise:

- A *type* of Task or a *specific* Task? (the types vs. instances problem)
- Part of a *specific* Project or used across *multiple* Projects?
- Produces a specific *deliverable* or *state*?
- Time-bounded or ongoing?
- Performed by *one* Worker or *one or more* Workers?
- …

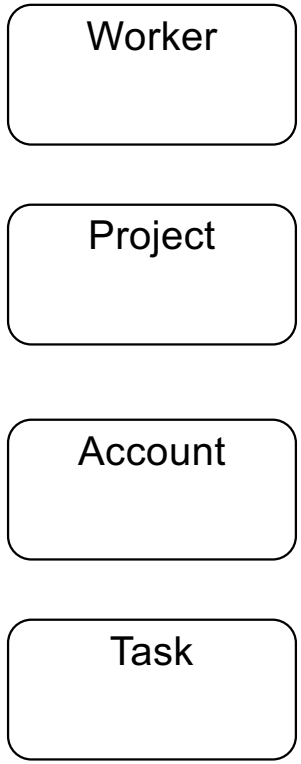
A ***Task*** is a specific, time-bounded, unit of work, within a single Project, intended to be performed by one or more Workers, that produces an intended deliverable or achieves a specific state.

Examples:

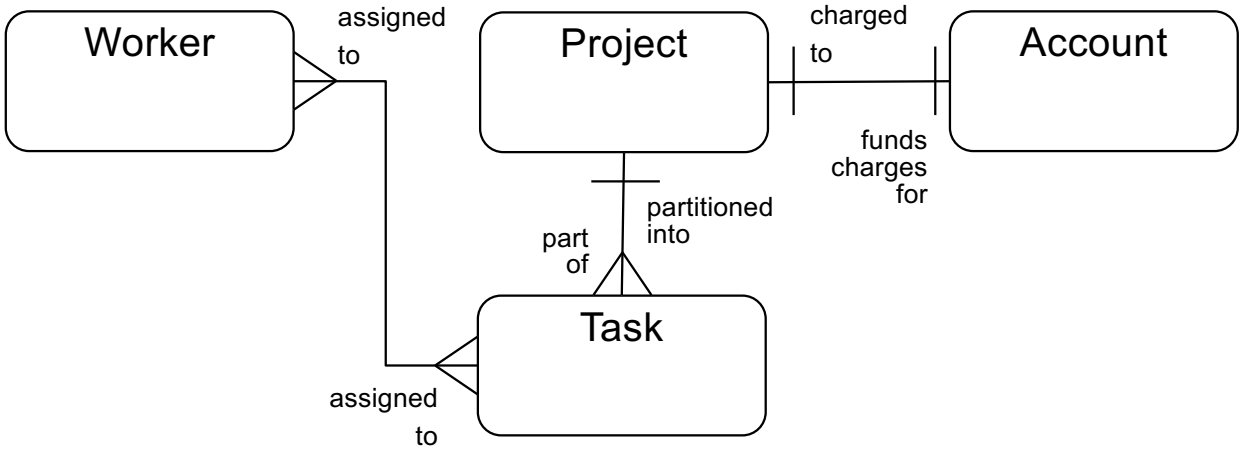
- Code *Place Order* service
- Test *Place Order* service

Excludes:

- types of Tasks
- ongoing (non time-bounded) activities such as management or administration



Now we have definitions – it's "safe" to draw the ER model



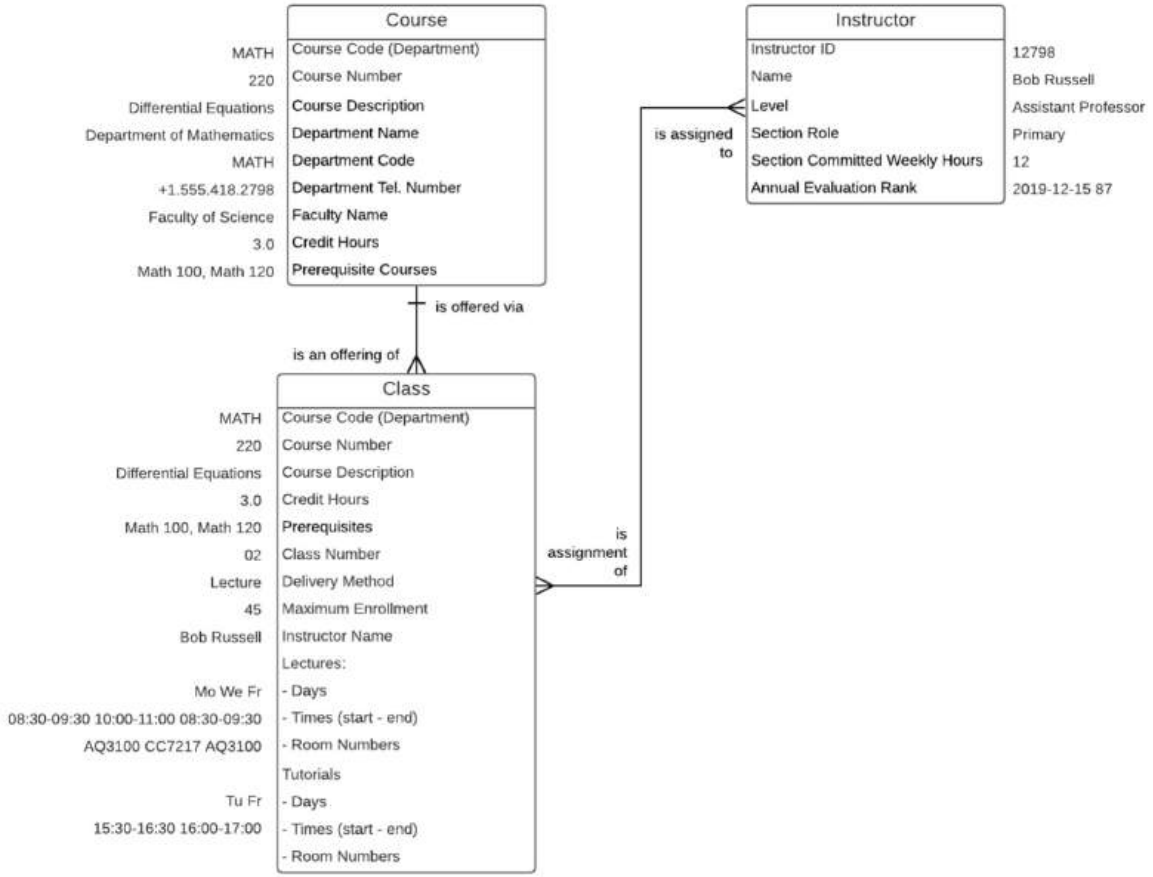
First arrange entities top-down by dependency.

Then add relationships with a verb-based phrase.

Then add cardinality (1:1, 1:M, M:M.)

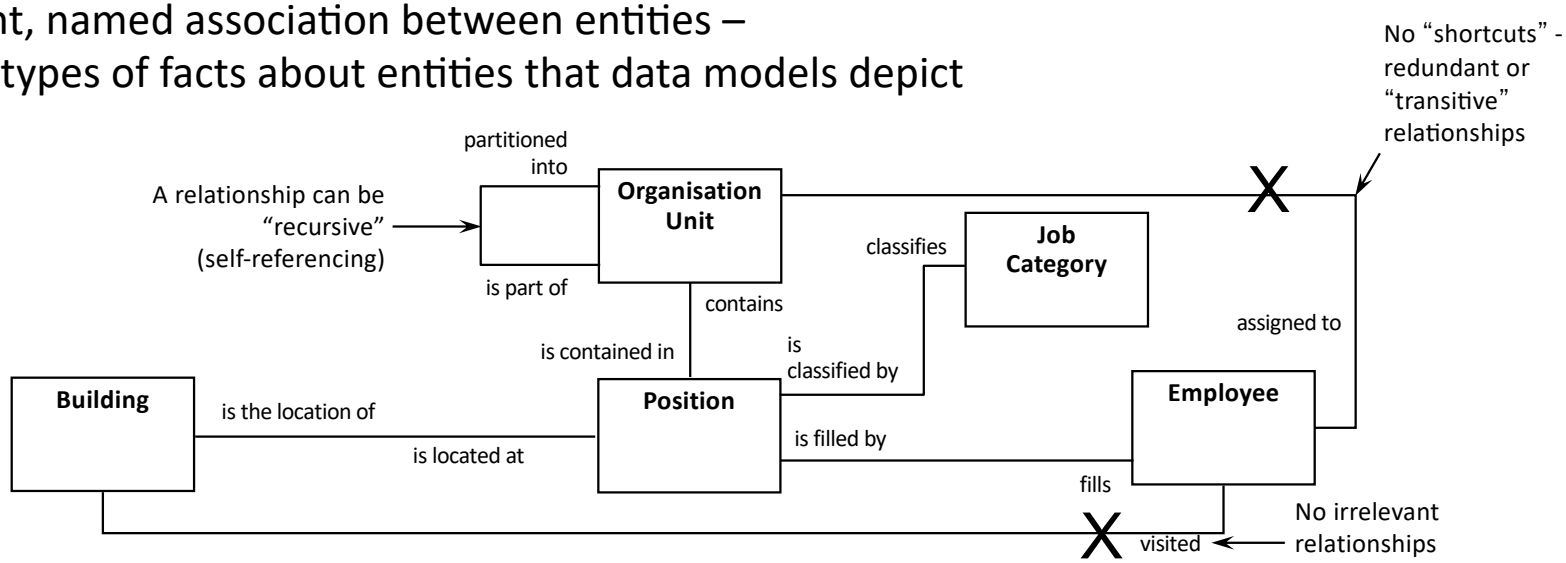
"Demonstrate the Data"

In addition to Entity definitions, it can be helpful to show *sample data values* on an E-R Diagram.



Relationships – a few more points

A significant, named association between entities –
one of the types of facts about entities that data models depict



Guidelines

- named with a descriptive, verb-based phrase – not "has" or "is related to"
  (the line tells us they *are* related; the name tells us *how*)

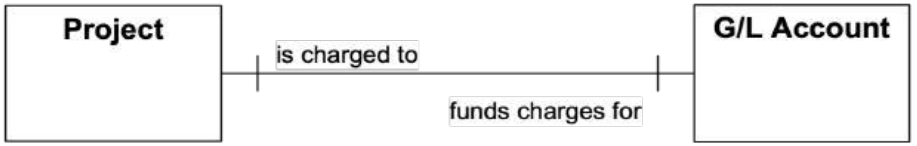
- named in both directions – try to use the same root word at both ends
  (e.g., "classifies" and "is classified by")

- the complete name reads like a sentence (noun verb noun) –
  "Position is classified by Job Category"

1:1 relationships – almost always an error!

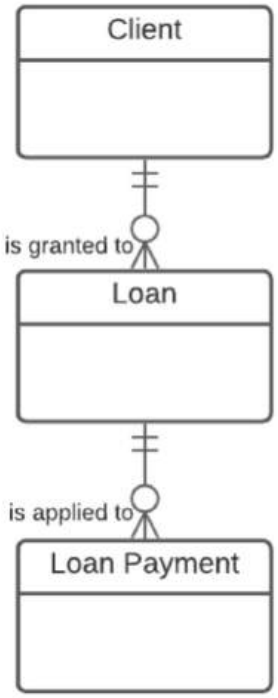
- Note – a 1:1 relationship might be necessary in the Physical Database Design
  e.g., "Fixed Asset" records financial data about a "Network Component" but they are in two separate systems (the G/L System and the Configuration Management System) connected by a 1:1 relationship

- ✘ Incorrect analysis
  e.g., Project costs are probably prorated across *many* Accounts



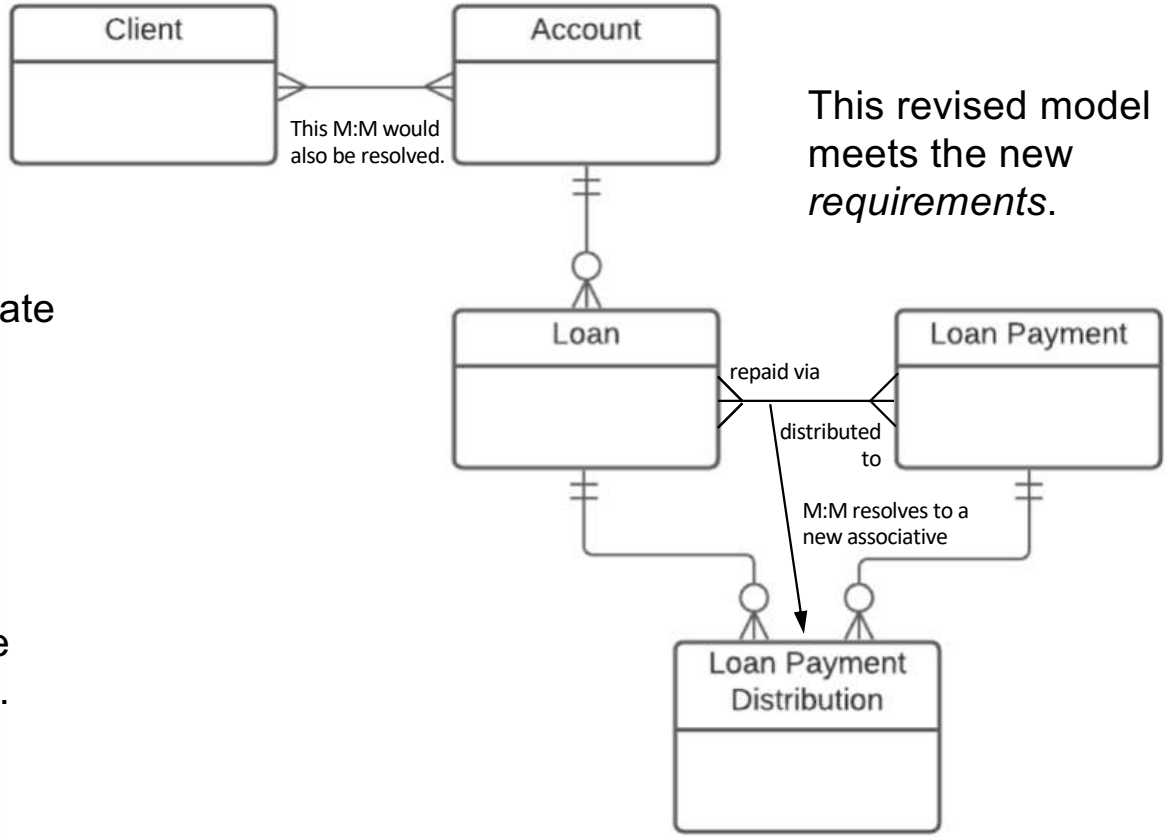
- ✘ Failing to account for changes over time
  e.g., an Employee may hold only *one* Credit Card at a time, but *many over time,* and we virtually always want history.
  The most common written constraint in Concept Modelling is
  "one <u>at</u> a time but many <u>over</u> time."

Future-proofing – "Challenge the Ones"



A Loan is granted to
one and only one Customer –
really?
No – multiple Clients can participate
in a Loan via a shared Account.
(A new *requirement*.)

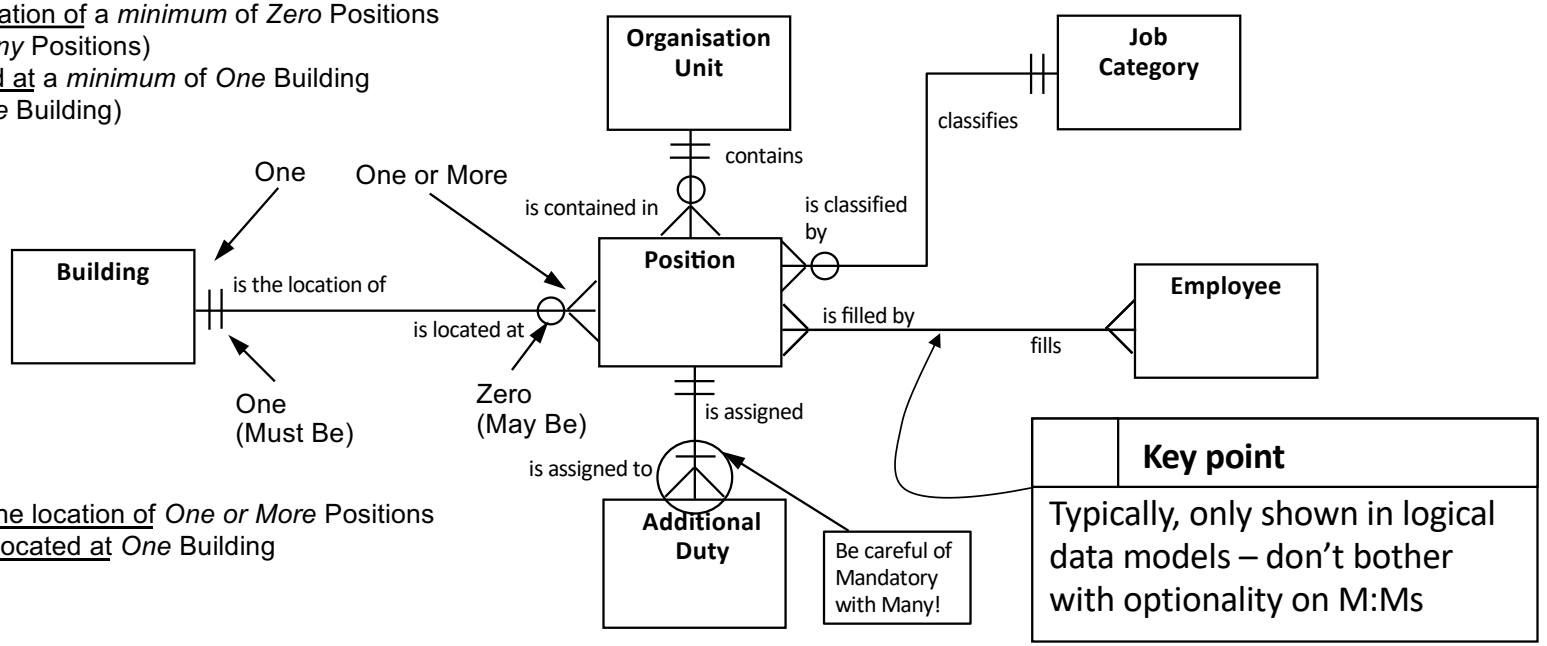
A Loan Payment applies to
one and only one Loan –
really?
No – one Loan Payment could be
distributed across multiple Loans.
(A new *requirement*.)



This revised model
meets the new
requirements.

Relationship optionality (logical models only)

Each Building is the location of a *minimum* of *Zero* Positions
(and a *maximum* of *Many* Positions)
Each Position is located at a *minimum* of *One* Building
(and a *maximum* of *One* Building)



or,
Each Building *May Be* the location of *One or More* Positions
Each Position *Must Be* located at *One* Building

To determine optionality (a.k.a. minimum cardinality)

- for each entity ask "Can one of these be related to a *minimum* of *Zero* or a *minimum* of *One* of the other entity?"
- record the answer – 0 or 1 – at the "other" end
  "zero" means an optional relationship (*May Be*) and
  "one" means a mandatory relationship (*Must Be*)
- easier form: "Each one of these *May Be* be or *Must Be* related to the other?"

Don't forget the four Ds of Data Modelling

1 Definition

- "What *is* one of these things?"
- List common and unusual instances
- "Are there any known anomalies?"
- "What are the potential differences of opinion?"

2 Dependency

- "What type of entity is this?"
- "What other entity does it depend on?"
- Essentially
  - is it a free-standing thing?,
  - is it a type of thing?,
  - is it repeating detail about some other thing?

Please let us know the key point (or points) that mattered most to you in this first section.

3 Detail

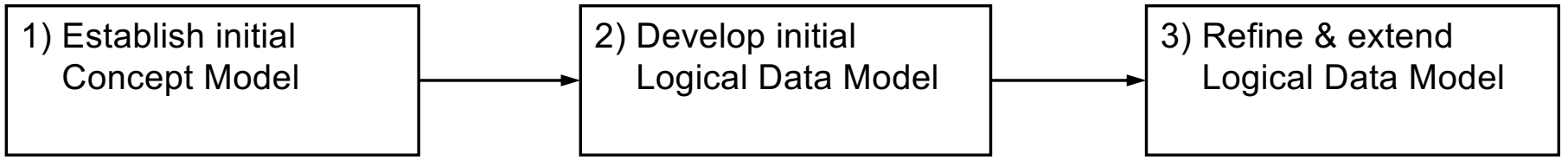
- Don't dive into detail – keep it in its place!
- GEFN!* HPDL!**

**Good enough for now!*
***Hard part, do later!*

4 Demonstration

- Assertions / narrative rules
- Sample data values or instances
- Scenarios or use cases
- Props (e.g., report layouts or common documents)

Phase 2 of three phases in data modelling



- Focus is on developing a core set of entities:
  - named
  - defined
  - minimally attributed
  - bound by basic rules and relationships
  - placed on an ERD
- Might start bottom-up: brainstorm details then synthesising "up"
- Might start top-down: build a contextual model, then flesh out required details analysing "down"
- Experiment w. alternatives
- Refine the contextual model, if you had one.

- Focus shifts to attribute rigor and structure when going to the logical level
- First check attributes for:
  - completeness
  - necessity
  - name and definition
  - placement
- Resolve attributes that are:
  - multi-valued
  - redundant
  - constrained
- Continue experimenting with alternate structures
- Refine conceptual model

- Focus is on refinement, and validation via new requirements using…
- …an event-based approach: fast and easy…
- …or full business analysis:
  - process workflow model
  - use cases (external)
  - service specs (internal)
  - profiling existing data
  - informational needs
- Resolve attributes that are semantically overloaded, non-atomic, or derived
- Document attribute properties and validation
- Specify identifiers
- Refine conceptual model

From conceptual to initial logical

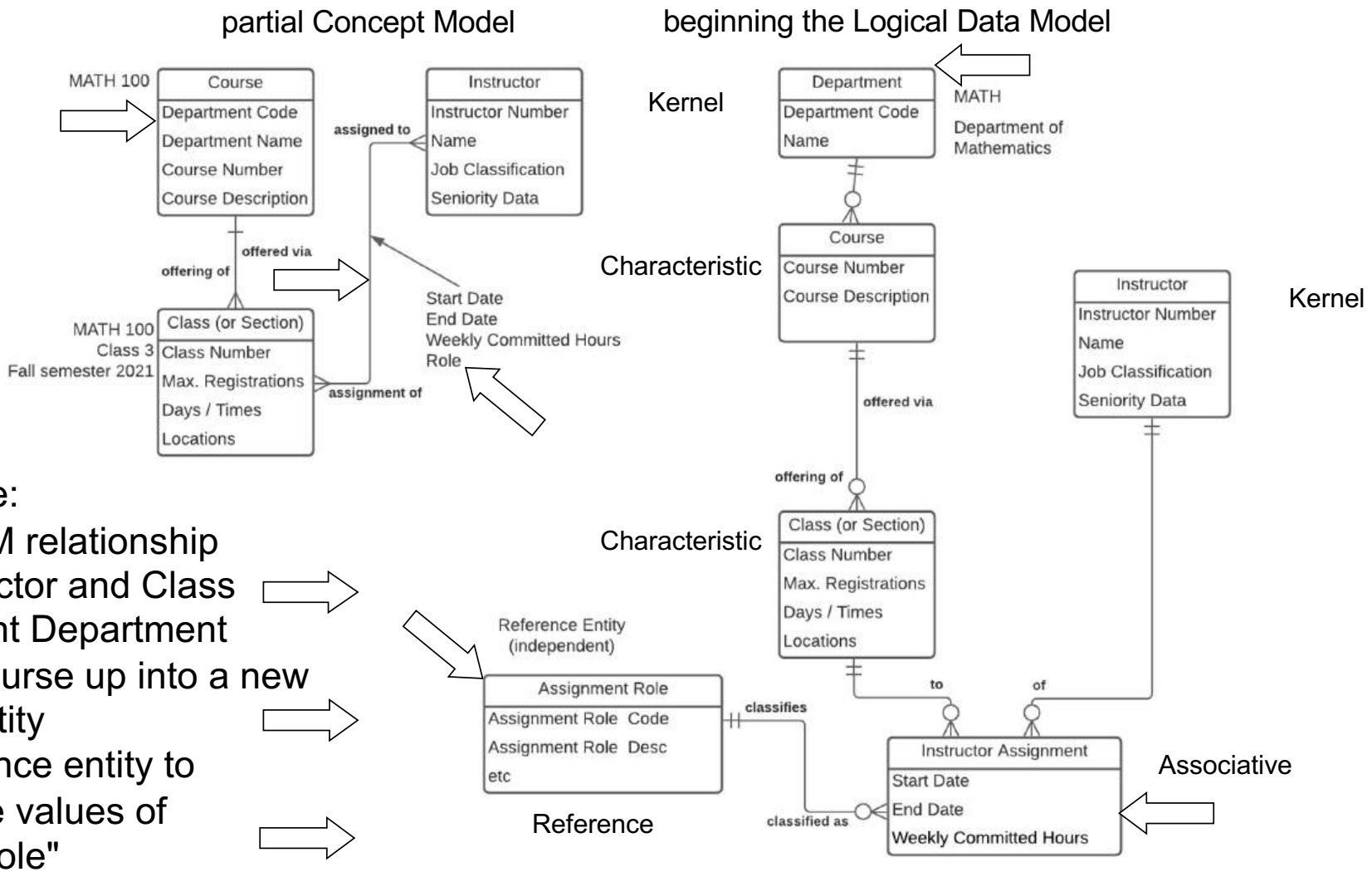
The progression from conceptual to logical is largely based on identifying and dealing with three attribute characteristics

- Multi-valued - the attribute can have multiple different values for one instance of the entity, either "at a time" or "over time"
  E.g., "Employee Name" if aliases or previous names are tracked
  - move it **down** to the "many" end of a 1:M relationship into a characteristic entity
  - if it's a fact about a M:M relationship between entities, move it down to the "many" end of a 1:M relationship into an associative entity
  - this puts the data structure into 1st Normal Form – 1NF

- Redundant - the same attribute value is recorded multiple times, in different entity instances, possibly inconsistently
  E.g., "Company Name" in a "Department" entity
  - move it **up** to the "one" end of a M:1 relationship to one of the parent (or higher) entities (2nd Normal Form – 2NF)
  - You might have to create a new parent entity where none existed before

- Constrained - a descriptive attribute needs to be restricted to a set of standard (or "allowable") values to improve integrity and reporting
  E.g., "Employee Type"
  - move it **out** to the "one" end of a M:1 relationship to a reference or other related entity (3rd Normal Form - 3NF)

A simple Concept Model to Logical Data Model example

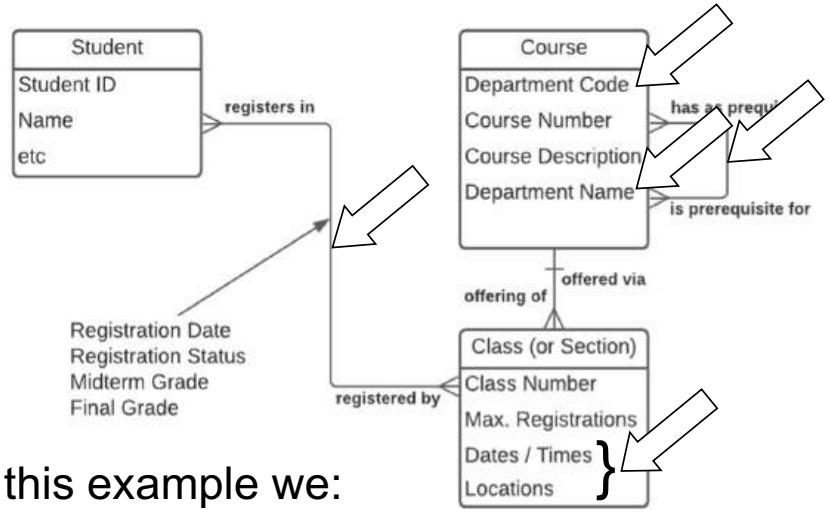


In this example we:

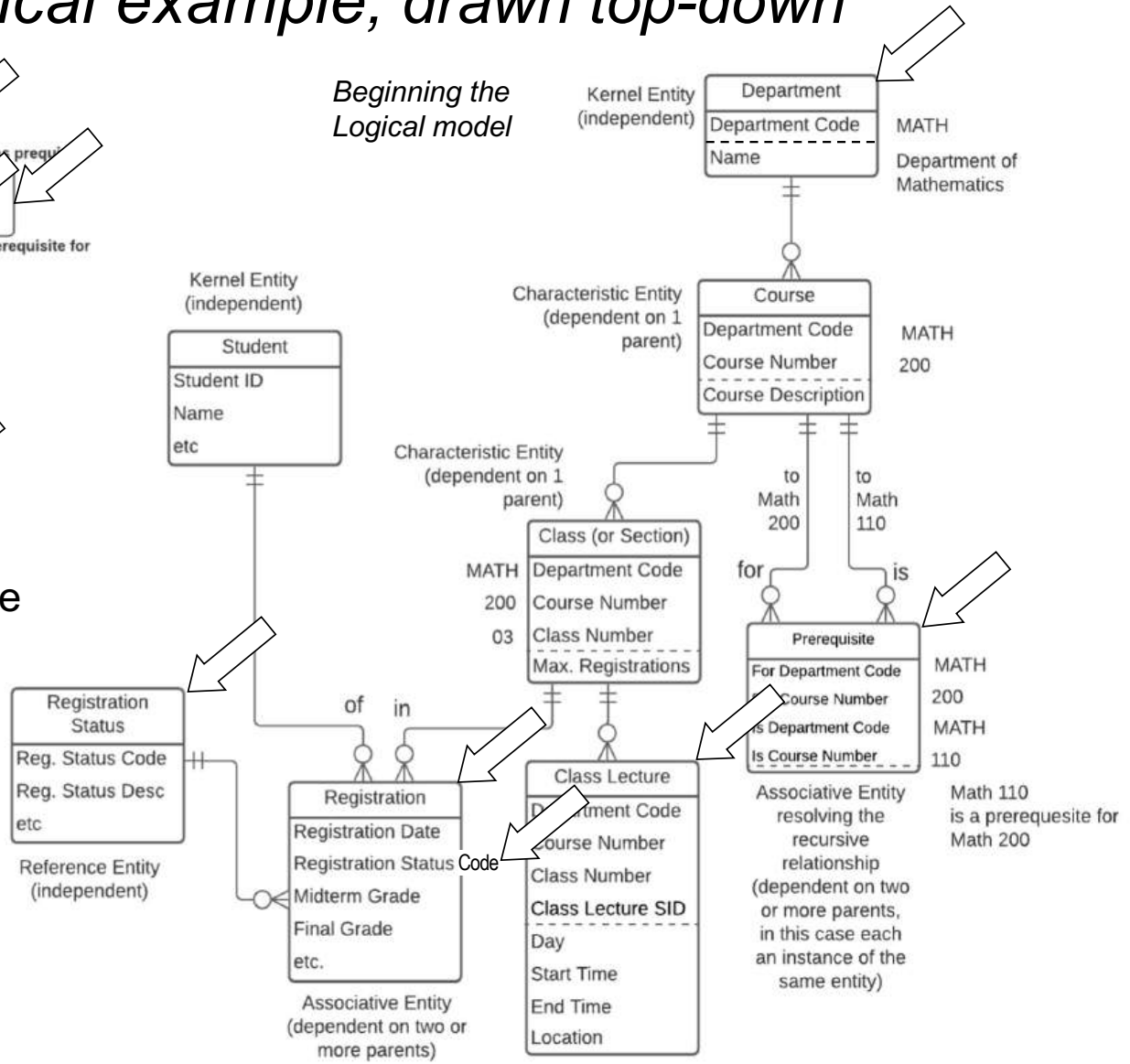
- resolve the M:M relationship between Instructor and Class
- move redundant Department attributes in Course up into a new Department entity
- create a reference entity to standardise the values of "Assignment Role"

Another Concept to Logical example, drawn top-down

Conceptual



Beginning the Logical model



In this example we:

- move multi-valued Class attributes into their own entity – Class Lecture
- resolve the M:M relationship between Student and Class
- resolve the recursive Course to Course M:M relationship
- move redundant Department attributes in Course up into a new Department entity
- move Registration Status into a reference entity

World's shortest course on normalisation

Unnormalised (UNF or 0NF)

- Contains multivalued attributes (a "repeating group")

First Normal Form (1NF)

- Repeating attributes moved *down* to a dependent Characteristic or Associative entity (create a new dependent entity if necessary.) This makes data "reportable."

Second Normal Form (2NF)

- Only applies to dependent entities
- No attribute in a child entity is really a fact about a parent (or grandparent or…)
- That is, no Characteristic or Associative entity redundantly contains facts from its parent(s) – if it does, move the fact(s) *up* (create a new parent entity if necessary)

Third Normal Form (3NF)

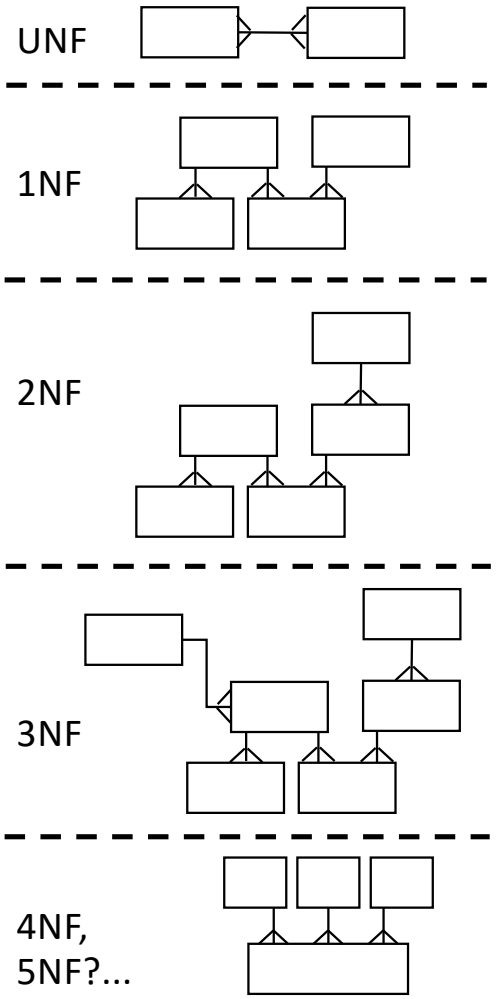
- If any entity redundantly contains facts from a related (non-parent) entity, move the fact(s) *out* to the other entity (create a new entity if necessary)

BCNF (Boyce-Codd NF – "3.5NF")

- Not an issue if you keep your wits about you

Fourth and Fifth Normal Form (4NF, 5NF)

- "Large" (3-way or more) associatives need to be broken down into more granular entities



For reference – Contextual, Conceptual, & Logical models

1 Contextual (Scope – Planner's View)

Agree context or "big picture" – the scope in terms of topics or subjects that are in or out, plus core terms and definitions

- May be a simple block diagram of topics/subjects, or primarily textual (a list)
- Optional – not necessary on smaller projects

My most plagiarised slide ever!

2 Conceptual (Overview – Owner's View)

Agreement on basic concepts and rules

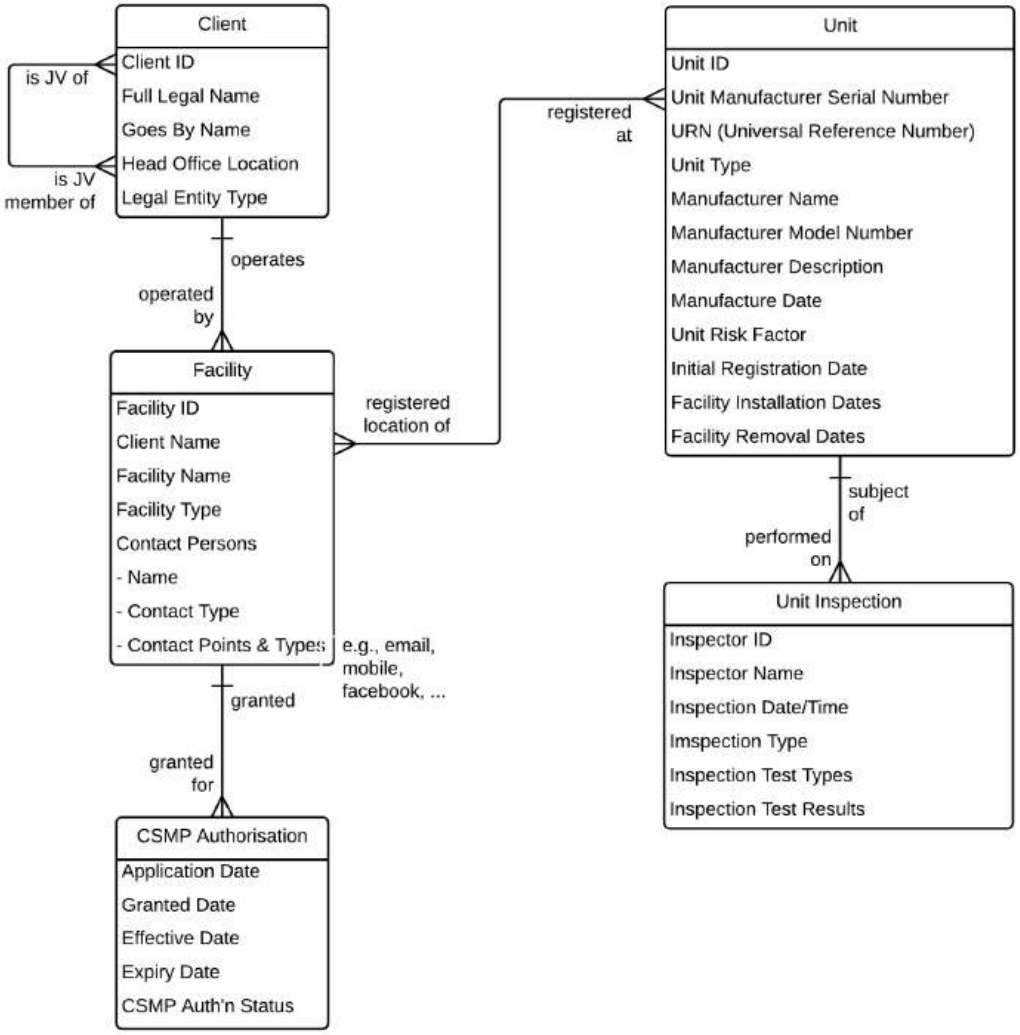
- Ensures everyone is using the same vocabulary and concepts before diving into detail
- Overview: main entities, attributes, relationships, rules
- Lots of M:M relationships
- Relationships show cardinality
- No keys
- Few or no reference entities
- Unnormalised – most M:M relationships unresolved, many attributes will be multi-valued, redundant, and non-atomic
- Verified directly by clients plus other techniques: Use Cases…
- A "one-pager"
- 20% of the modelling effort

3 Logical (Detail – Designer's View)

Full detail for physical design

- Provides all detail for initial physical database design and requirements specification
- Detailed: ~ 5 times as many entities as the conceptual model
- M:M relationships resolved
- Relationship optionality added
- Primary, foreign, alternate keys
- Lots of reference entities
- Fully normalised – no multi-valued, redundant, or non-atomic attributes. All attributes defined and "propertised"
- Verified by other means: sample data, report mockups, scenarios, …
- May be partitioned
- 80% of the modelling effort

Self-study exercise 3 – from conceptual to logical



This is unnormalised – it contains multi-valued (repeating,) redundant, and constrained attributes.

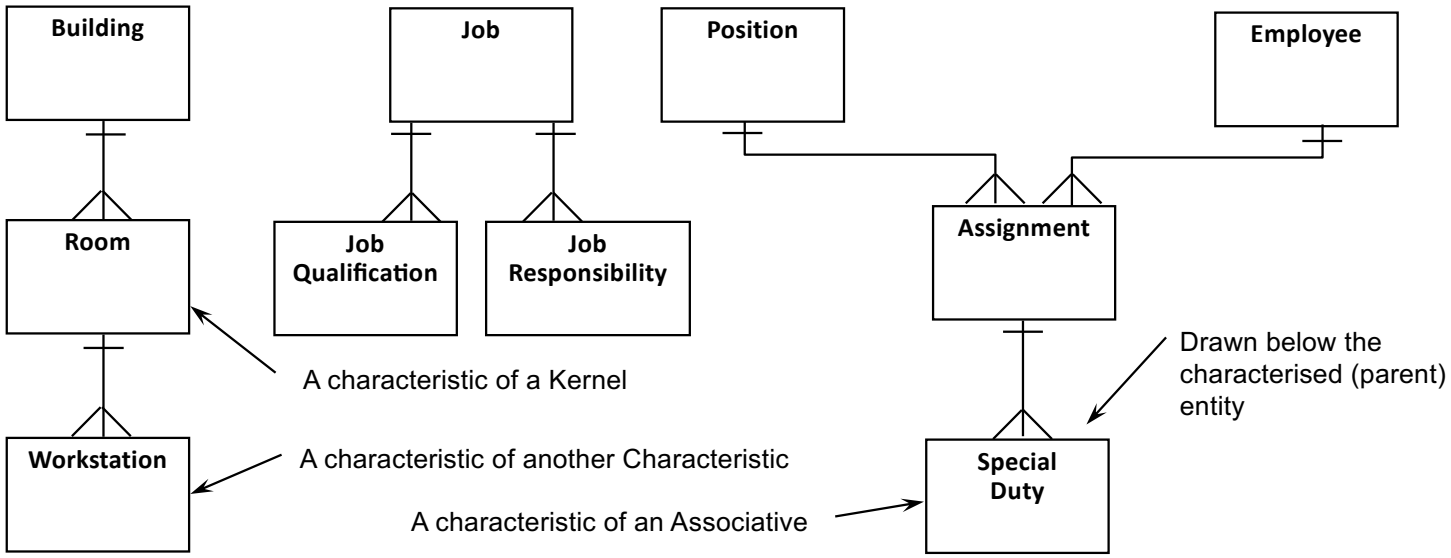
First, identify the attributes that are "correct" – they are base attributes of the entity they are in. ✓
Then, normalise it to 3NF (Third Normal Form) by identifying and dealing with attributes that are:

- Multivalued, and need to be moved *down* to a dependent entity. (1NF)
- Redundant and need to be moved *up* to a parent (or higher) entity. (2NF)
- Redundant or constrained and need to be moved *out* (sideways) to a related but non-parent entity, or to a reference entity. (3NF)

Entity types – kernels

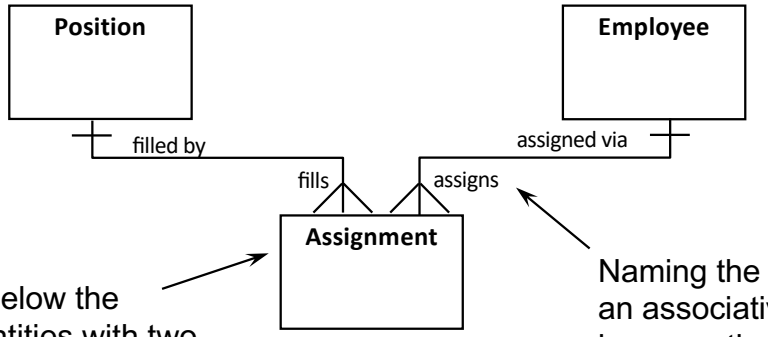
- The “central” objects in the model
  - “what it's all about”
  - everything else either further describes, associates, or classifies the kernel entities
- *Independent* – its existence is not dependent on another entity
  - “Does it make sense for one of these to exist on its own?”
  - is not a child of another entity
- Drawn at the top of the diagram, or subject area within a diagram

Entity types – characteristics



- Records repeating, multi-valued facts about a parent entity that have been "cast out" from the parent entity
- It "characterises" the parent entity (any type of entity)
- *Dependent* on one parent entity, and is drawn below that parent
- Drawn below the parent entity

Entity types – associatives

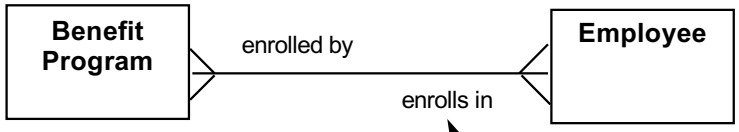


Drawn between and below the associated (parent) entities with two or more crowsfeet on the top edge

Naming the relationships from an associative is often difficult, because the associative entity itself is a relationship

- Relates ("associates") two or more other entities –
  records facts about the association (M:M relationship) between those other entities
  - sometimes so important it is discovered directly (Order, Contract, …) and is shown on the Conceptual Model – the remainder are added on the Logical Model
  - other times it evolves from "resolving" M:M relationships
- Can associate any combination of different entity types
- *Dependent* on two or more parent entities
- Drawn between and below the parent entities

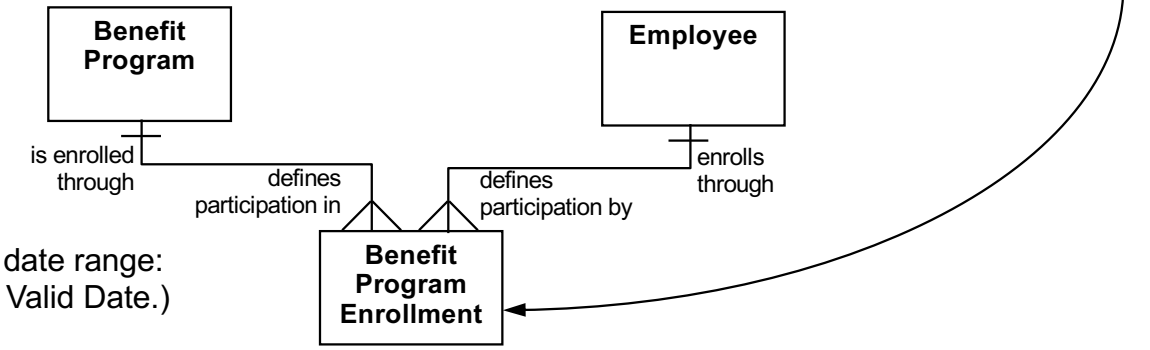
Associatives – notes



Where will we record facts about the relationship, such as Enrollment Date?

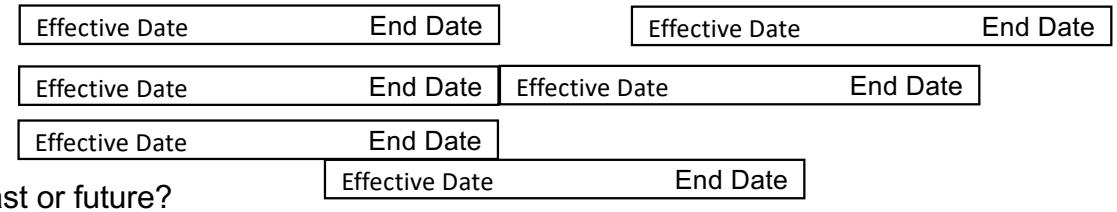
becomes...

The M:M relationship name is often the basis for the associative entity name



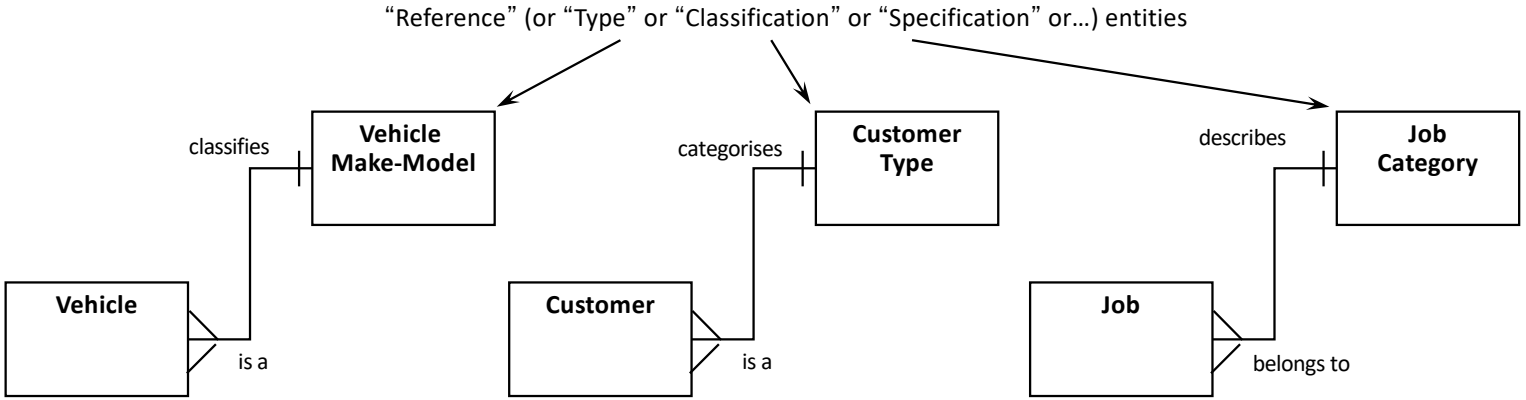
Associative entities often contain a date range: Effective Date to End Date (or Last Valid Date.)
Always check for the following:

- Gaps are allowed?
- Must be contiguous?
- Overlaps are allowed?
- Effective Dates can begin in the past or future?



- Is the date range *until* or *through* the End Date? (tot en met)
- Must an End Date be specified, and if so, what format is used – “null” or “HighDate – 99991231?”
- Must the date range fit within a parent's date range?
- Do global time zones need to be handled?

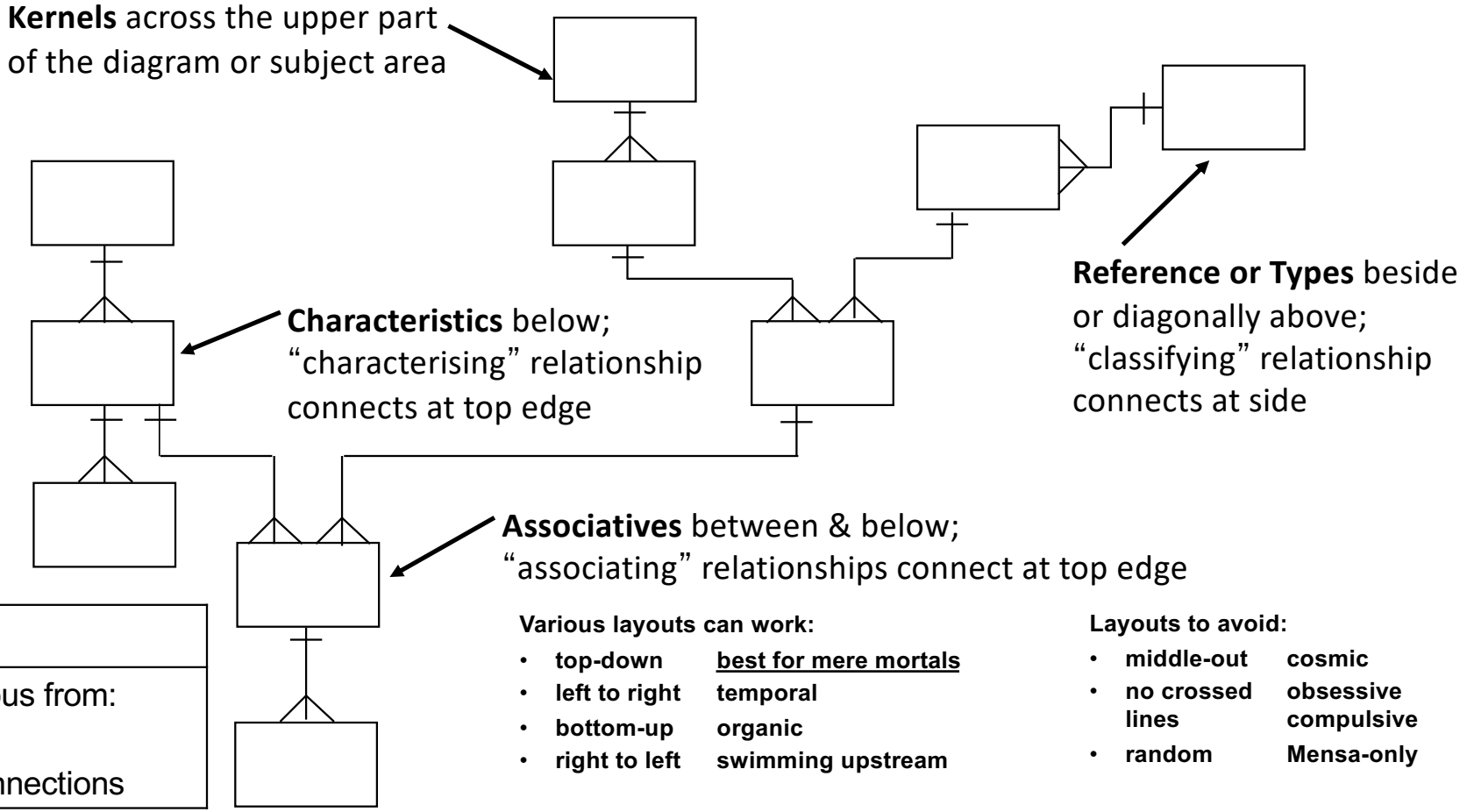
Entity types – reference or type



- An entity that classifies or categorises other entities and/or allows the recording of standardised values for a descriptive attribute
- *Independent*
- Purpose may be served by an attribute in the Concept Model
  (i.e., a Customer Type attribute in the Customer entity)
- Only critical Reference entities are shown on a Concept Model
  (i.e., when a Reference entity ties together different parts of the model)
- Drawn beside or diagonally up from the classified entity

Graphic guidelines – the "no dead crows" principle

Draw the same kinds of things the same way every time!



!	*Key point*
Entity type is obvious from: ▪ Placement ▪ Relationship connections	

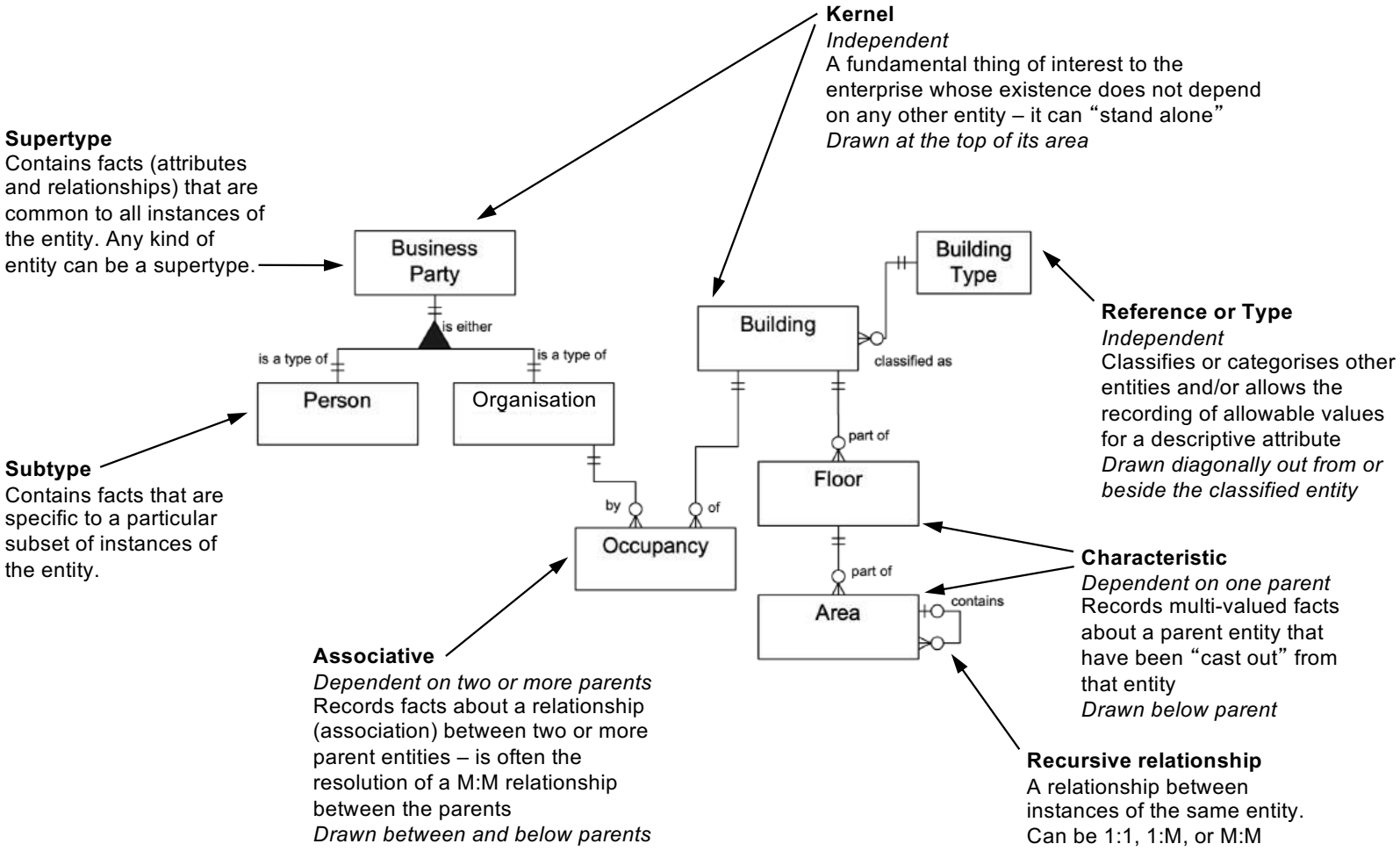
Various layouts can work:

- **top-down** — **best for mere mortals**
- **left to right** — **temporal**
- **bottom-up** — **organic**
- **right to left** — **swimming upstream**

Layouts to avoid:

- **middle-out** — **cosmic**
- **no crossed lines** — **obsessive compulsive**
- **random** — **Mensa-only**

Summary – entity types and conventions



Kernel
Independent
A fundamental thing of interest to the enterprise whose existence does not depend on any other entity – it can "stand alone"
Drawn at the top of its area

Supertype
Contains facts (attributes and relationships) that are common to all instances of the entity. Any kind of entity can be a supertype.

Subtype
Contains facts that are specific to a particular subset of instances of the entity.

Reference or Type
Independent
Classifies or categorises other entities and/or allows the recording of allowable values for a descriptive attribute
Drawn diagonally out from or beside the classified entity

Characteristic
Dependent on one parent
Records multi-valued facts about a parent entity that have been "cast out" from that entity
Drawn below parent

Associative
Dependent on two or more parents
Records facts about a relationship (association) between two or more parent entities – is often the resolution of a M:M relationship between the parents
Drawn between and below parents

Recursive relationship
A relationship between instances of the same entity. Can be 1:1, 1:M, or M:M

Interesting structures

➡	*Outline*

1. Conceptual & Logical models, level-set
2. Interesting structures
3. Rules on relationships and associations
4. Presentation techniques for data modellers

★	*Topics*

- Types vs. Instances
- Recursion and generalisation, in general
- Recognising lists, trees, and networks, and modelling them with recursive relationships
- Generalisation (subtyping) - when to use it, and when not to
- Meeting new requirements and modelling difficult rules

!	*Themes*

- Communication!
- Consistency
- Contextual models
- Complexity

Exercise: libraries and bookstores

Your local library and your local bookstore share some obvious similarities:

- Libraries loan books to cardholders (what the library calls a customer) and bookstores sell books to customers. Customers get to keep their purchases, but cardholders have to return whatever was loaned to them within a stated time period.
- Bookstores and libraries both keep track of all transactions (“purchase” or “loan”), but:
  - the library always records the cardholder for the transaction
  - the bookstore only records the customer for the transaction if they belong to their “frequent buyer” program.

Some miscellaneous points:

- Purchases and loans can both cover multiple items.
- Both of them use the term “book” somewhat loosely; they also deal with different Format Types - audiotapes, videotapes, CDs, CD-ROMs, and so on.
- They both call a "book" a "Title," and when a Title is available on a specific Format Type (Softcover, Hardcover, CD, DVD, Audiobook, e-Book, etc.) they call that a "Release."
- Both organisations care about the “Book's” title and author
- Both organisations deal solely with “Books” (whatever you decide to call it) – they do not carry other types of products, or at least Books are all we care about.

What are the most important differences between the two models?

Build a simple data model for the library, and one for the bookstore. Make a guess at a few important attributes for each of the entities in your model.

Exercise work area

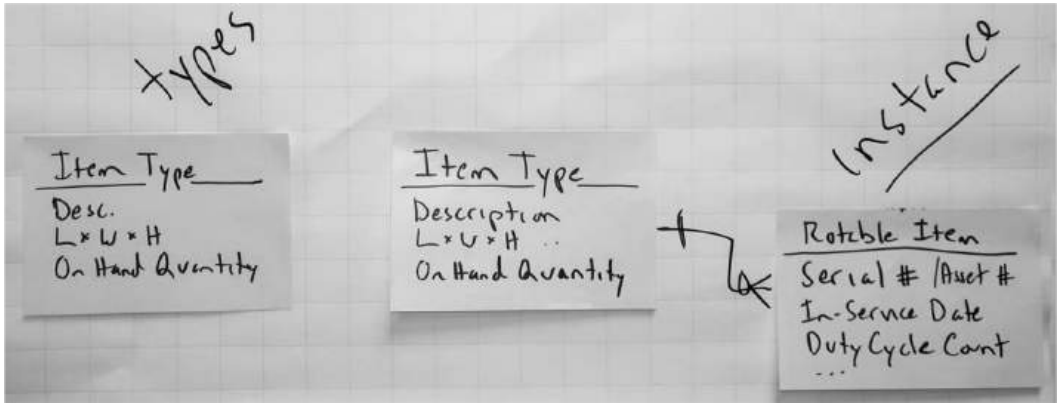
Solution: differences and notes

Differences – library vs. bookstore

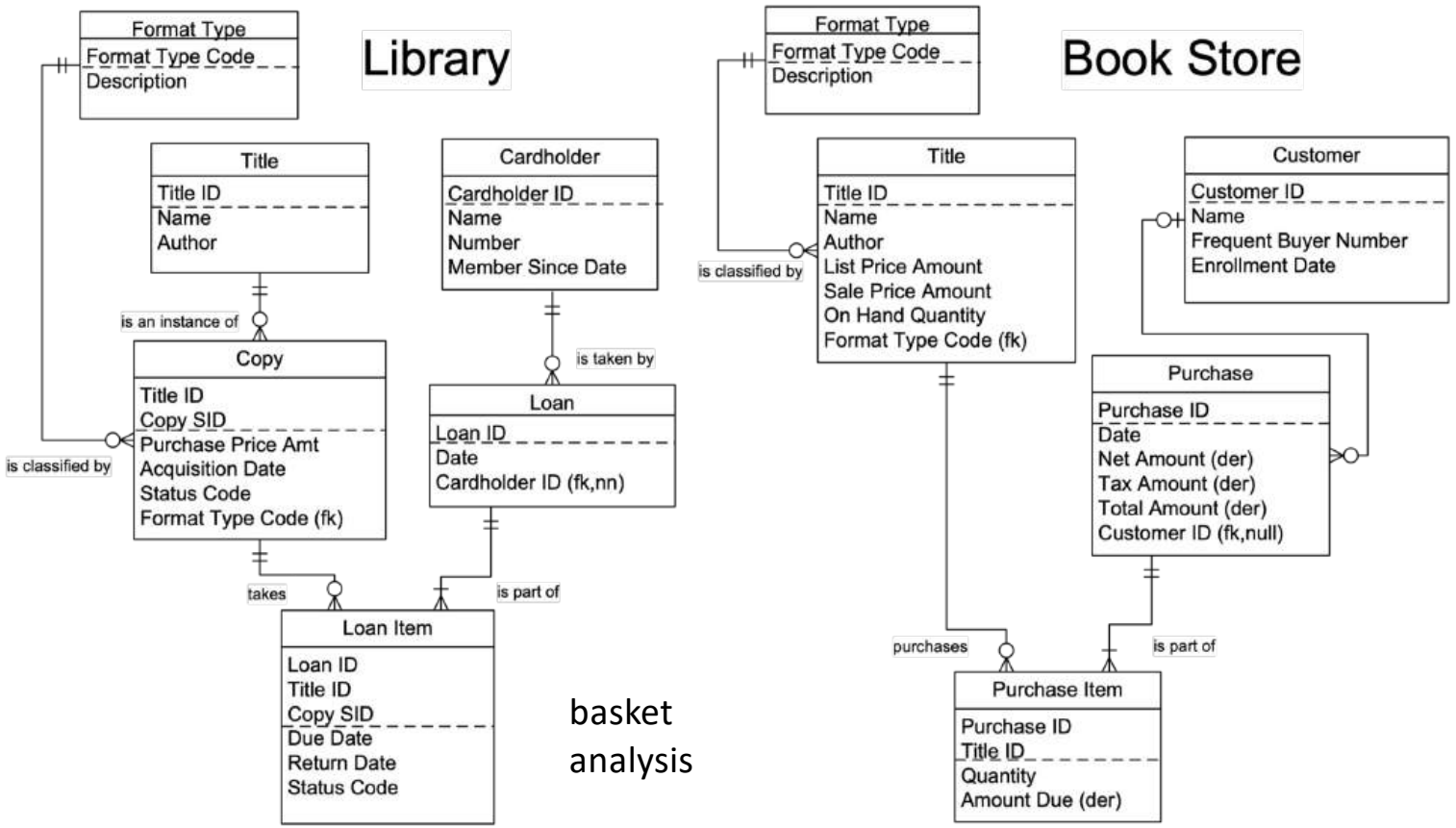
- loan vs. purchase: loan – we want it back; purchase – one-time
- one book is loaned many times in a library, but sold once in a bookstore
- library: cardholder (mandatory); bookstore: customer (optional)
- library – loan has a return date; bookstore – we hope there is NO return
- bookstore – has a price (but the library may sell books)
- *types vs. instances – bookstore: type (Title); library: each instance (Copy)*

Diane McKellar, civic government – "Get it wrong and live in pain forever!"

An Inventory Management system was selected that only tracked TYPES of Items. During implementation they discovered they also needed to track INSTANCES of certain Items – Items that could be rebuilt and "rotated" in and out of service.
The solution was a complex and expensive "shadow system" built in Excel.

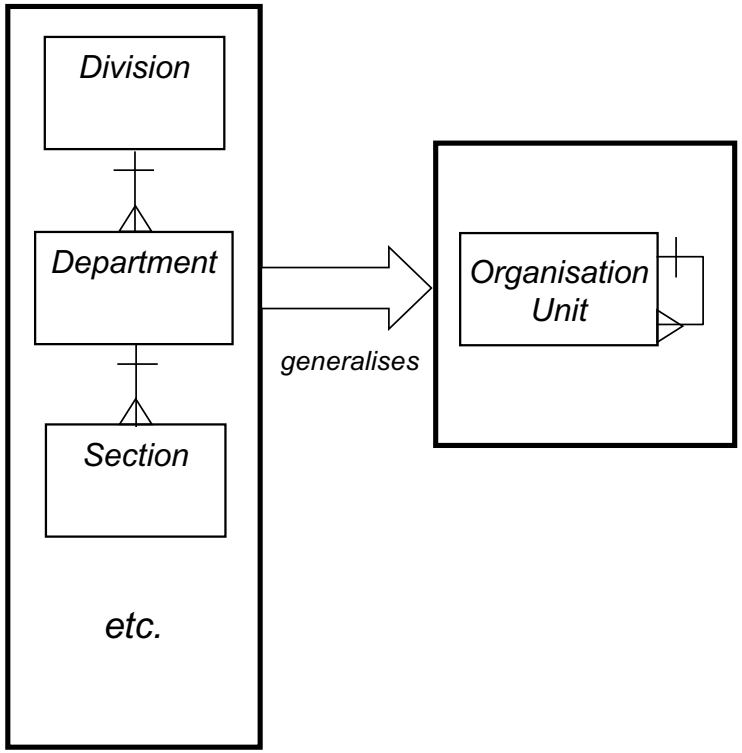


Solution: libraries and bookstores



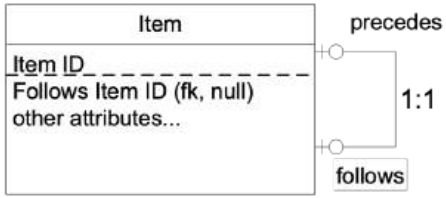
Recursion

- ✓When one entity occurrence can be related to another occurrence of the same entity type
- ✓Also know as a "self-referencing" relationship
- ✓Three variations:
  - 1:1
  - 1:M
  - M:M
- ✓Often involves "generalising"

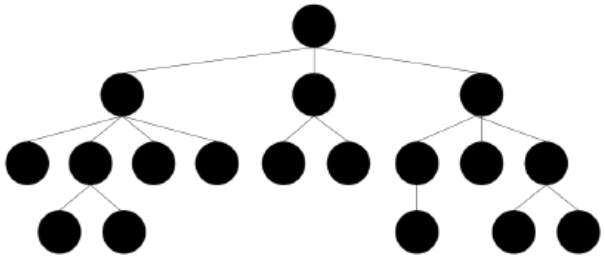


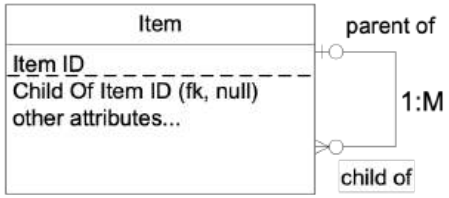
Recursion - recognising the data structure

Items arranged in a *linked list:*

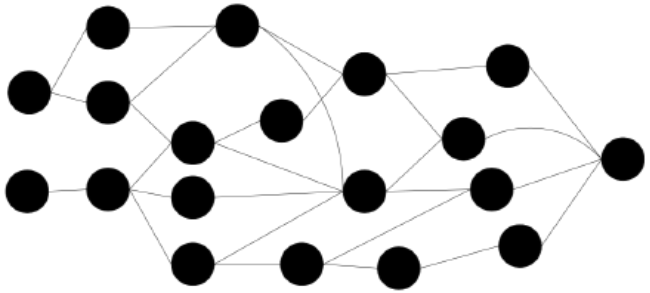


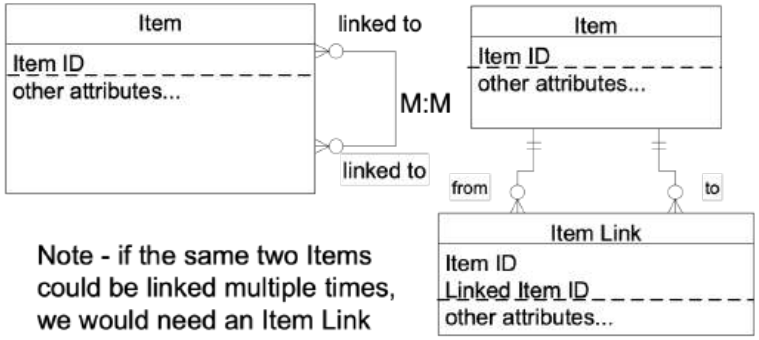
Items arranged in a tree:





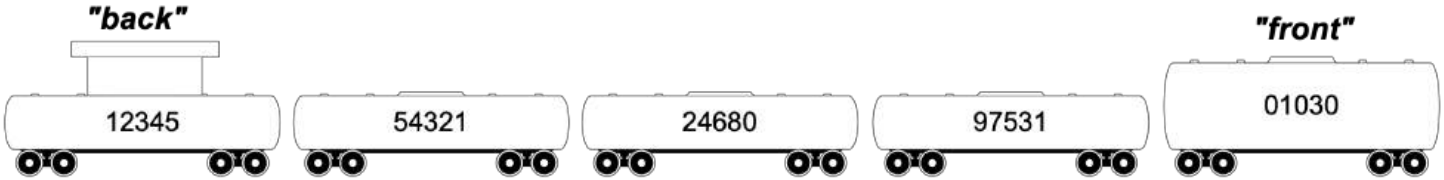
Items arranged in a network:



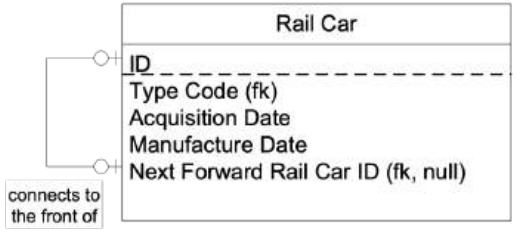


Note - if the same two Items could be linked multiple times, we would need an Item Link SID in the primary key

Recursive relationships – 1:1 example



- ✓ The train above is an example of a "linked list"
- ✓ A linked list can be handled with a recursive 1:1 relationship
- ✓ All the items in the list must be generalised into the same type of entity(or a supertype with multiple subtypes - "Rail Car" would subtype into "Freight", "Locomotive", "Passenger", etc.)
- ✓ The foreign key can either "point ahead" or "point back" – depends which end you add new instances from
- ✓ As always, the recursive relationship is "fully optional"
  - the first car doesn't have a car in front of it
  - the last car doesn't have a car behind it.

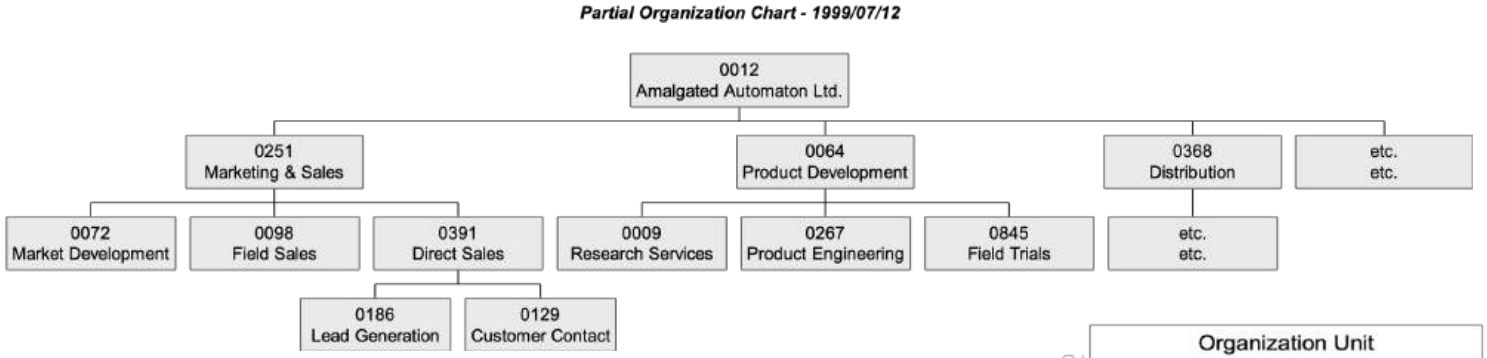


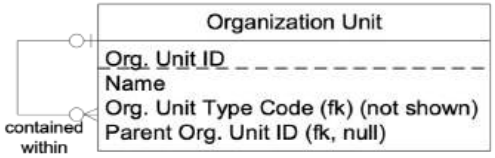
"Rail Car" Sample Instance Table

Rail Car ID	Next Forward Rail Car ID
12345	54321
54321	24680
24680	97531
97531	01030
01030	null

Can you think of an example where you would use this?

Recursive relationships – 1:M example





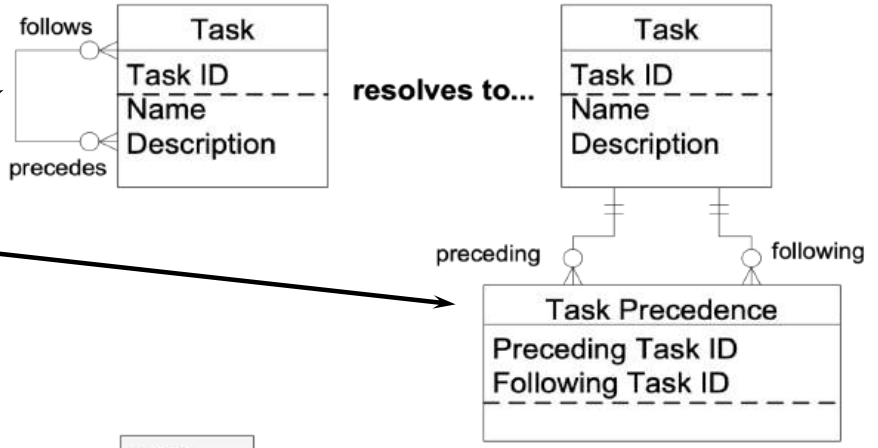
- ✓ The organisational structure is a "hierarchy"
- ✓ A hierarchy can usually be handled with a recursive 1:M relationship
- ✓ Again, this requires all the items to be generalised into the same type of entity
- ✓ The foreign key must be at the "Many" end, so the child "points to" the parent
- ✓ As with all recursive relationships, this is "fully optional"

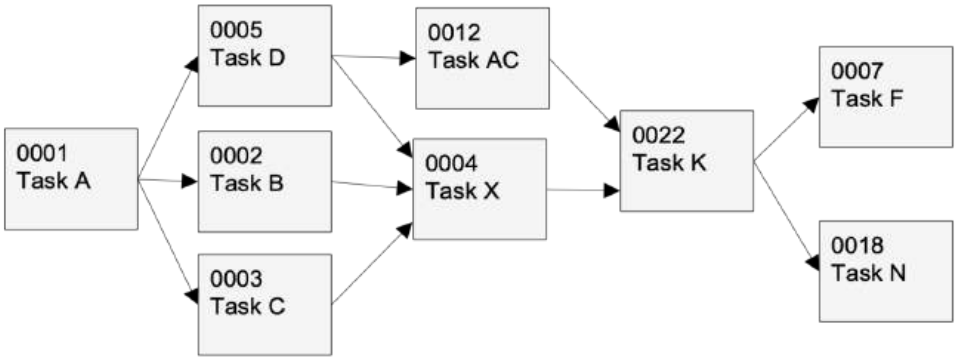
"Org Unit" Sample Instance Table	
Org. Unit ID	Parent Org. Unit ID
0012	null
0251	0012
0064	0012
0368	0012
0072	0251
0098	0251
0391	0251
0186	0391
0129	0391
0009	0064
0267	0064
0845	0064
etc.	etc.

Can you think of an example where you would use this?

Recursive relationships – M:M example

- ✓ This project plan is a "network"
- ✓ A network is handled with a recursive M:M relationship
- ✓ This resolves to an associative entity linking two different instances of the same entity



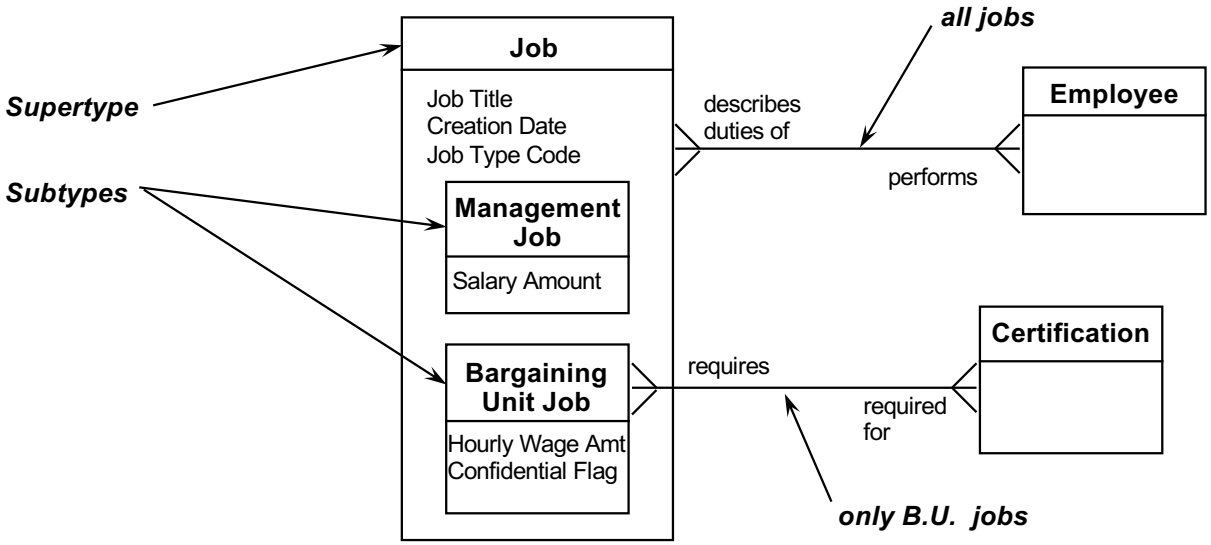


"Task Precedence"
Sample Instance Table

Preceding Task ID	Following Task ID
0001	0005
0001	0002
0001	0003
0005	0012
0005	0004
0002	0004
0003	0004
0012	0022
0004	0022
0022	0007
0022	0018

Can you think of an example where you would use this?

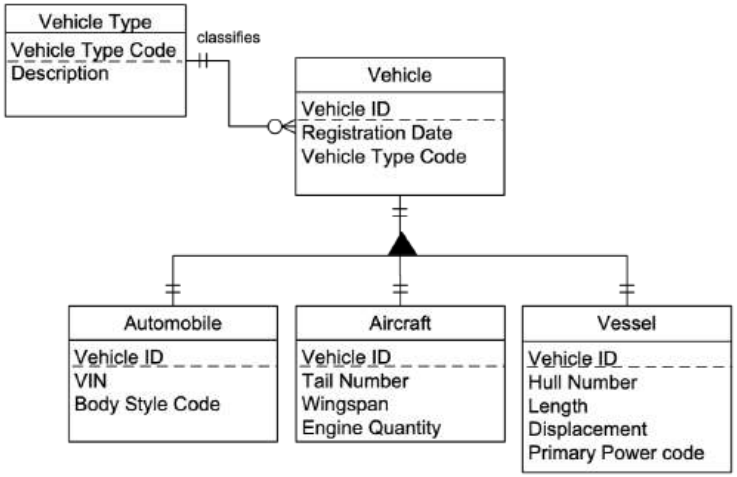
Supertypes and subtypes



- Breaks an entity down into two *or more* 'subtypes', or generalises two or more into a single 'supertype'
  - common relationships and attributes go into *supertype*
  - unique relationships and attributes go into *subtype*
- Subtypes are mutually exclusive and mandatory – there is exactly one subtype instance for each supertype
- a.k.a., generalisation-specification, or gen-spec

Generalisation vs. subtyping

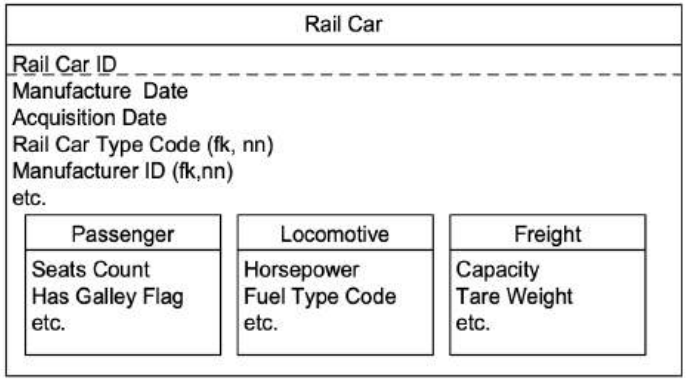
- ✓ "*Generalisation - Specialisation"* is typical O-O terminology; *"Supertype - Subtype"* is typical E-R terminology. Gen-spec.
- ✓ Generalise whenever two or more entities, each with their own *distinct* attributes and relationships, also *share* other attributes and relationships
- ✓ Automobile, Aircraft, and Vessel have common attributes that could be generalised into Vehicle…
- ✓ …or, Vehicle could be sub-typed into Automobile, Aircraft, and Vessel, with the same outcome
- ✓ Note that it's common for a subtyped entity to also be classified by a *type* or *reference* entity. In this example, Vehicle Type Code is the "subtype discriminator."



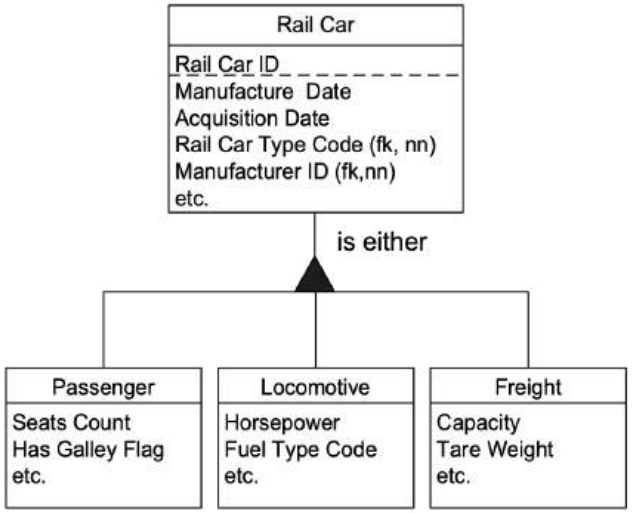
Subtyping - alternative formats

Two different diagramming approaches are widely used -

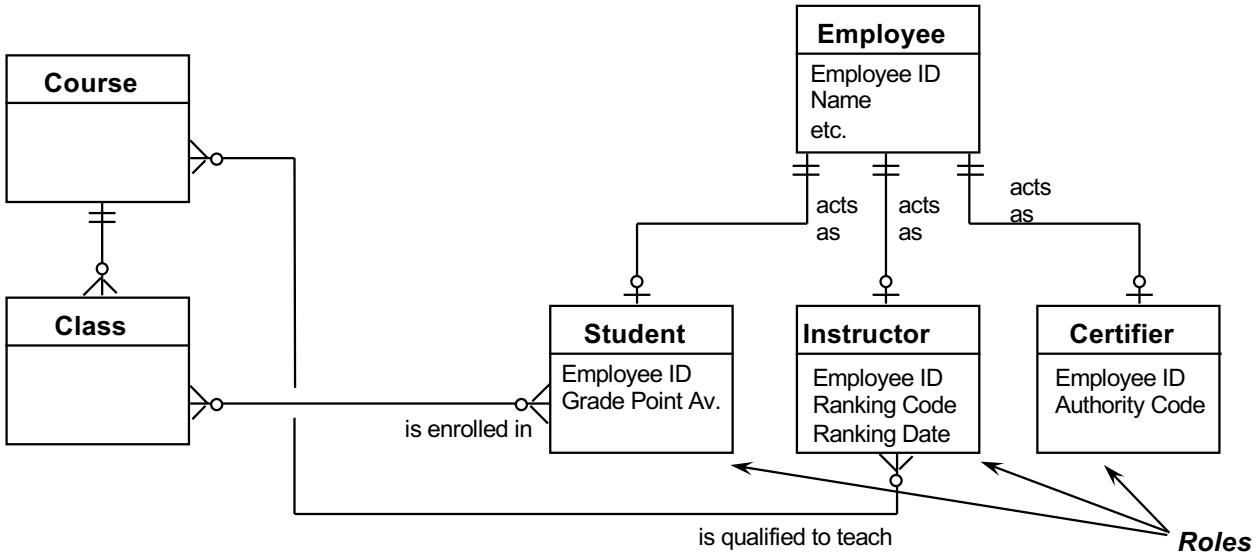
- "Box in box"
  (e.g., Oracle modelling tools)



- Generalisation hierarchy
  (e.g., most ER- or UML-based modelling tools)



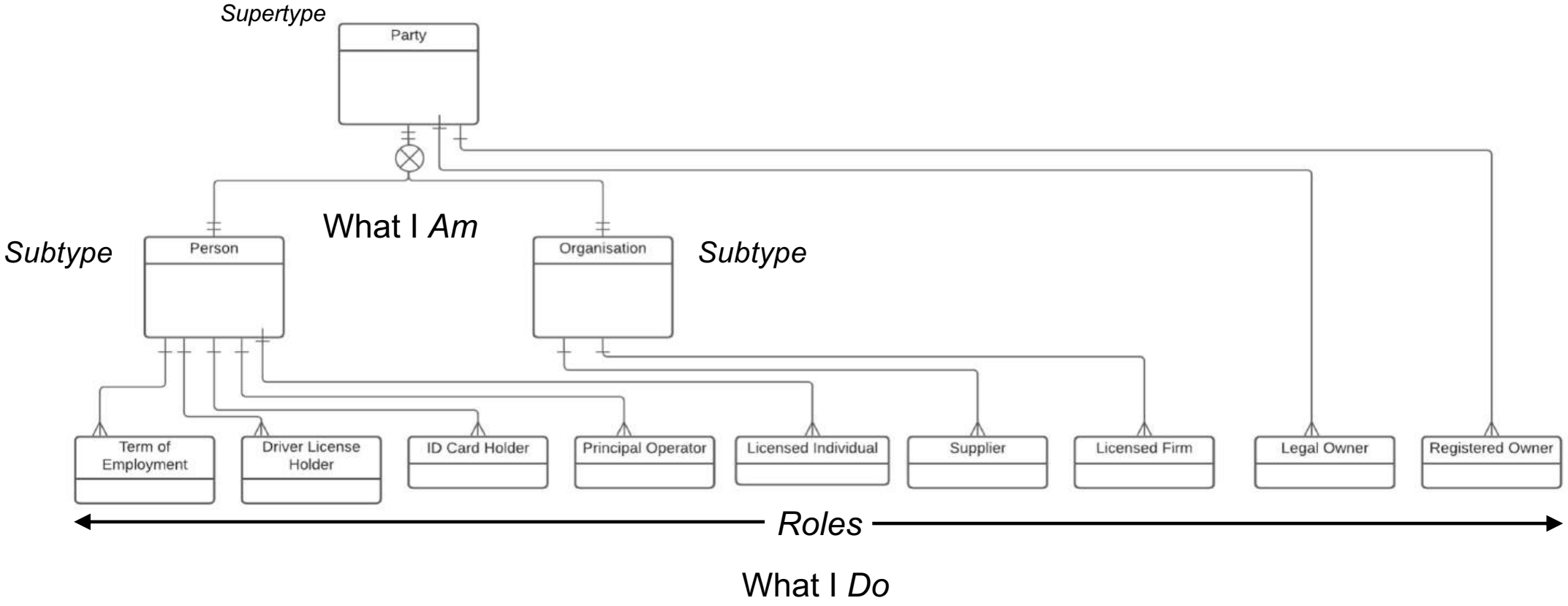
Don't confuse "roles" and "subtypes"



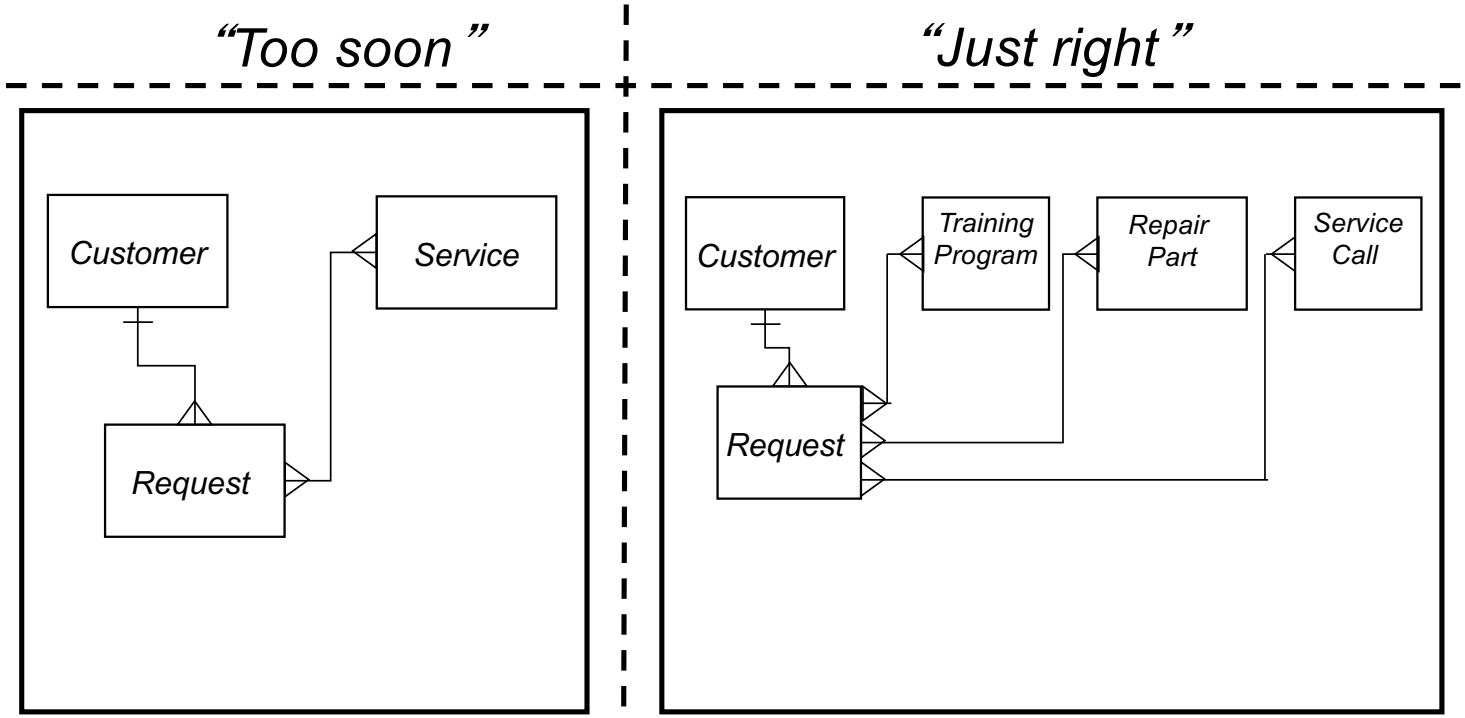
- ✓ A "role" structure is used when there are two or more different "roles" that an entity type can take on.
- ✓ *Similar* to subtyping
  - unique attributes and relationships go in the role entity
- ✓ *Different* from subtyping
  - the roles are *not* mutually exclusive
  - the parent does not necessarily have to take on any role

An example with supertype, subtypes, and roles

Note that some roles are valid for one *subtype* (Person or Organisation.)
If a role is valid for both, we connect it to the *supertype* (Party.)



Don't generalise too soon

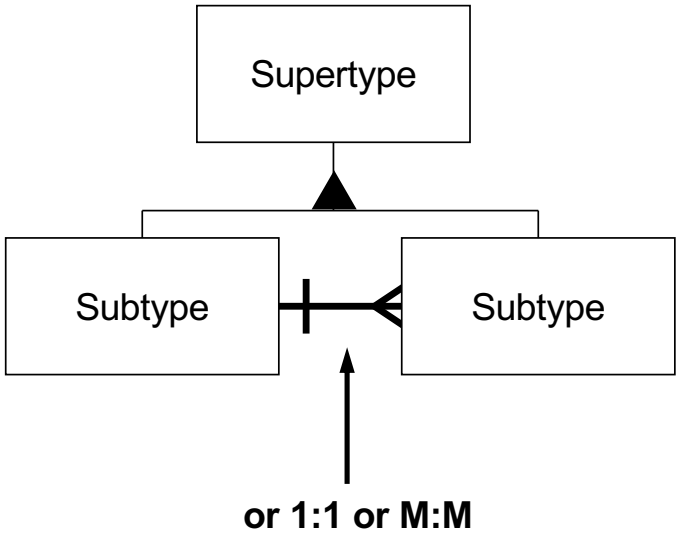


- ✓ *Confuses the client*
- ✓ *Reduces the chance of discovering "specifics"*
- ✓ *Specifics first, generalisation later*

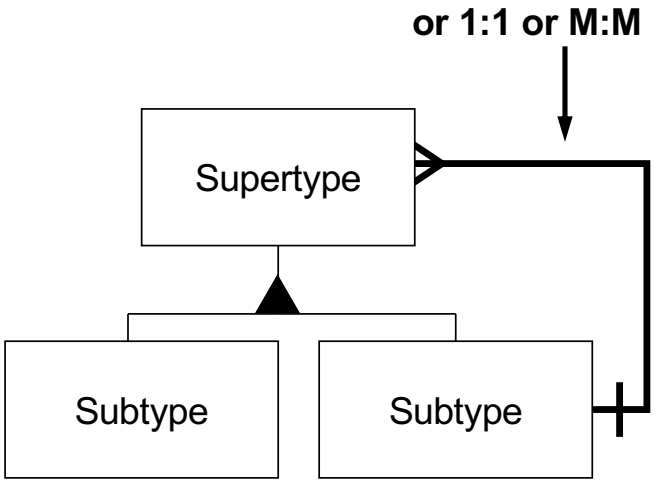
Combining recursion and generalisation

- ✓ *Business rules can often be handled by a recursive relationship involving supertypes and subtypes:*

Between subtypes



Between super and subtype



Meeting a new requirement…

Confirm and extend the model:

- ✓ *discover new requirements, using a variety of techniques*
  e.g., look for multi-valued attributes

Philosophy

- ✓ *don't dive in – start simple, add detail in layers*
- ✓ *start out in "natural language"*

See example
on the next page from our
"Good entity?" exercise

1) State the new requirement as an *assertion*

- ✓ Start out using the *client's* language
- ✓ *Then*, ensure that the assertion uses terms from the data model (entity names, relationship names, etc.) This "leads" you to the solution.
- ✓ ***Confirm it!***

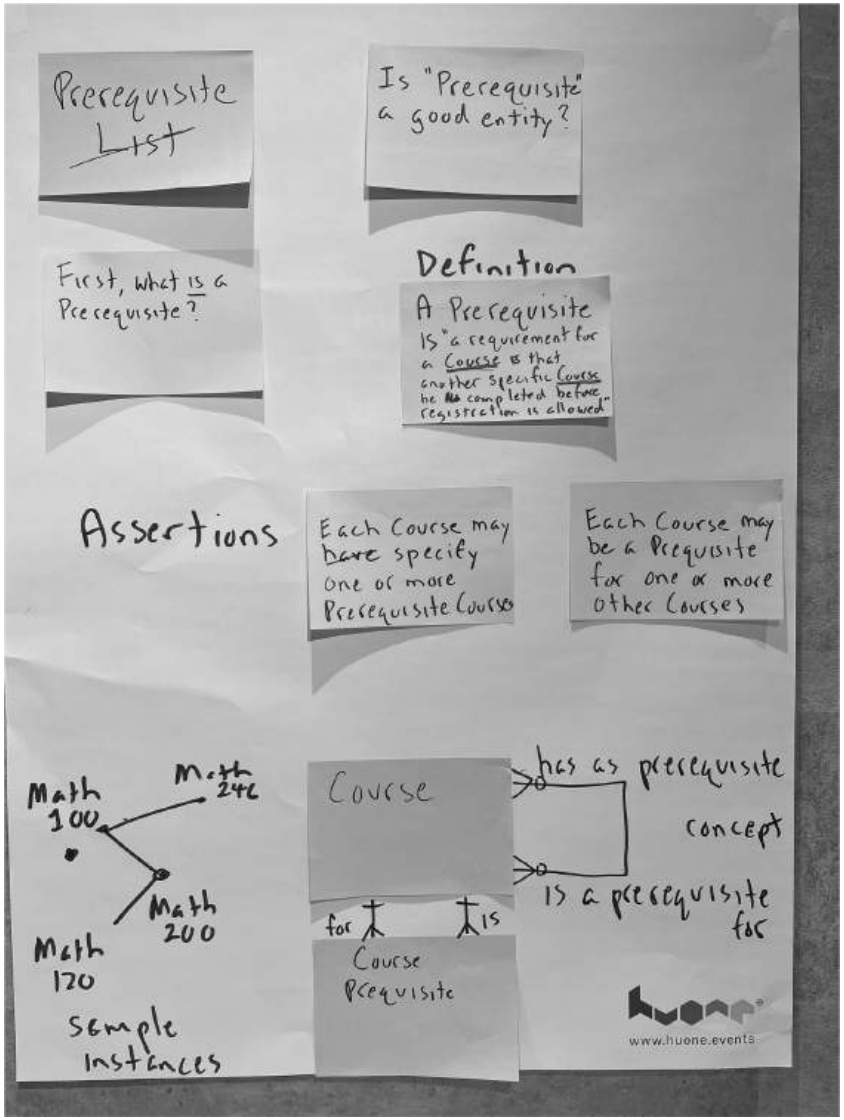
2) Develop a conceptual solution

- ✓ Look for the simplest option first: no change needed, a new reference attribute, a multi-valued attribute(s), M:M relationship, new entity
- ✓ Explore rules, like "what is the basis for multi-valued?"
- ✓ ***Confirm it!***

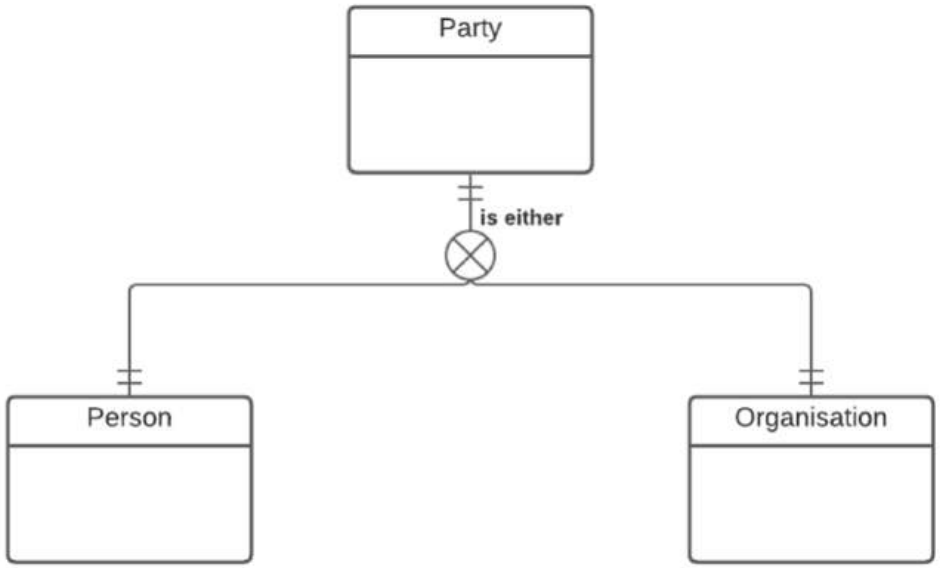
3) Develop a logical solution

- ✓ Fully normalised, fully attributed
- ✓ Follow an "orderly script" –
  don't get ahead of yourself or the client
- ✓ ***Confirm it!,*** possibly using other easy-to-follow formats such as screen or report mock-ups.

Example – meeting a new requirement



If time permits, an exercise with subtyping, recursion, & rules



Key point:
State the constraint or fact you are trying to model in plain language before drawing the model.

Extend the model so that it can record these additional facts:

1 – Organisational structure
An Organisation must be one of the recognised types, such as Corporation, Partnership, or Society.

An Organisation may be made up of multiple Organisation Units (an internal subdivision,) each of which might break down further into lower-level Organisation Units, and so on.

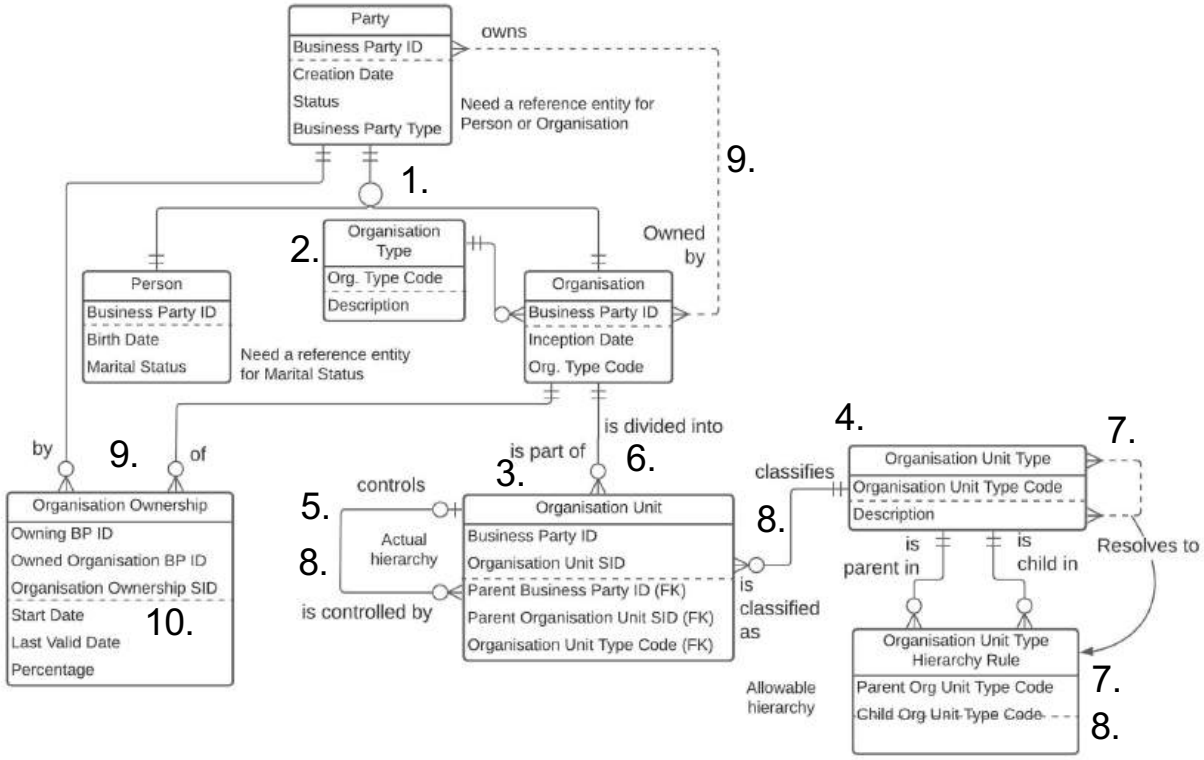
Each Organisation Unit has only one parent Organisation Unit. Some Organisation Units have no parent Organisation Unit, because they depend directly on an Organisation. That is, they are the highest level of Organisation Unit.

2 – Rules on Organisational structure
Each Organisation Unit is of a specific type, such as division, department, area, team, section, etc. Only certain relationships between types are valid. E.g.,
- a division can contain a department, but a department cannot contain a division.
- a team can be contained within an area or a division
Note – only certain types of Organisation Units can be immediately subsidiary to an Organisation, but we won't model that constraint at this time.

3 – Organisation ownership
An Organisation may have multiple owners, each of which could be another Organisation or a Person.

Exercise work area

Solution



Assertions:

1. Each Party is either a Person or an Organisation
2. Each Organisation must be classified as one Organisation Type and
   Each Organisation Type may classify one or more Organisations
3. Each Organisation may divide into one or more internal Organisation Units and
   Each Organisation Unit must be part of exactly one Organisation
4. Each Organisation Unit must be classified as one Organisation Unit Type and
   Each Organisation Unit Type may classify one or more Organisation Units
5. Each Organisation Unit may control one or more other Organisation Units and
   Each Organisation Unit may be controlled by one other Organisation Unit
6. The controlled and the controlling Organisation Units must be part of the same Organisation
7. Each Organisation Unit Type may control one or more other OU Types and Each Organisation Unit Type may be controlled by one or more other OU Types
8. The Organisation Unit Type of the controlled Organisation Unit and the Organisation Unit Type of the controlling Organisation Unit must be a pair found in Organisation Unit Hierarchy Rule
9. Each Organisation may be owned by one or more Persons or Organisations (which is to say one or more Parties) and
   Each Party may own one or more Organisations
10. A Party may own an Organisation multiple times over a period of time

Exercise: stock exchange trading

Please build a data model from the following facts:

Companies issue shares in various stocks. For instance, Algonquin Industries has issued common stock, preferred A stock, and preferred H stock.

Each stock may be listed on multiple stock exchanges. For instance, Algonquin's common stock is listed on the Vancouver and Toronto exchanges, but its preferred stocks are only listed in Vancouver.

When a customer wishes to buy or sell shares of a particular stock, they place an order on one of the exchanges. The order says, in effect, that "I am offering to buy (or sell) X quantity of stock Y for price Z for the next W days" If it is a sell order, the customer must also (by law) indicate if they are short selling (Short Sale Flag is set)

The Automated Trading System matches up buy and sell orders if the prices are within certain parameters. A complex algorithm determines the actual price of the sale. Note that an order may not be satisfied all at once (i.e., with one sale). For instance, an order to sell 10000 shares may be matched with a buy order for 5000 shares, another for 3000 at a later time, and it may expire before the remaining 2000 shares sell.

Build an initial E-R data model illustrating all the relevant entities, and their relationships.

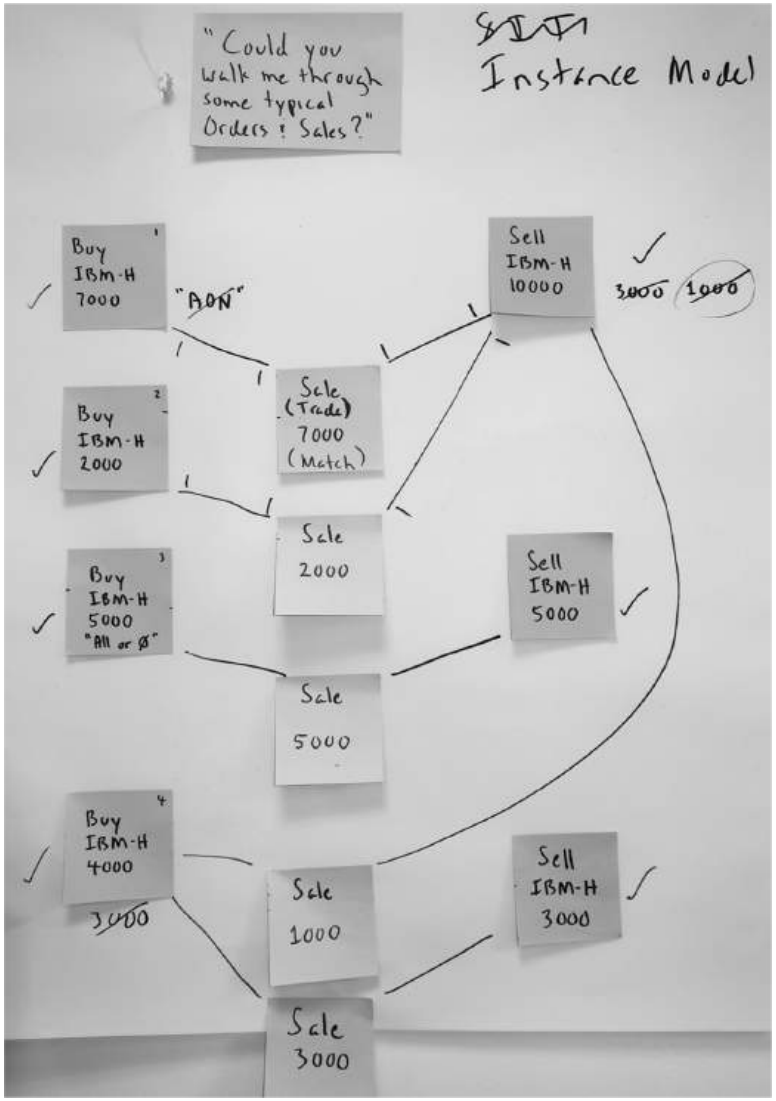
1. Identify the main entities
2. Agree simple definitions
3. State assertions that describe the scenario
4. Arrange entities by dependency
5. Add and name relationships
   - name the relationship first
   - only then add cardinality (ones and manys)

Assertions and clarifications

- Each Company may issue one or more Stocks and each Stock must be issued by one Company
- Each Stock must be classified as one Stock Type and each Stock Type may classify one or more Stocks
- Each Stock may be listed on one or more Exchanges and each Exchange may list one or more Stocks
  (The Company chooses which Exchanges to list on)
- Each Customer may place one or more Buy or Sell Trade Orders &
  Each Buy or Sell Trade Order must be placed by one Customer
- Each Trade Order is either a Buy Trade Order or a Sell Trade Order
- Each Buy Trade Order may be filled by one or more Sell Trade Orders and each Sell Trade Order may be filled by one or more Buy Trade Orders
- The Buy and Sell Trade Orders for a Trade must be placed by different Customers
- Each Trade must be a match of one Buy Trade Order and one Sell Trade Order

- A Listing is an authorisation to buy or sell a specific Stock on a specific Exchange
- A Buy Trade Order is an offer to buy …
  A Sell Trade Order is an offer to sell…
  - a stated quantity
  - of a specific Stock
  - on a specific Exchange
  - at a specified price
  - during a specified time period
- The matching of a Buy Trade Order and a Sell Trade Order is referred to as:
  - a Sale
  - a Match
  - a Fill
  - ***a Trade***
  - a Buy/Sell Transaction…

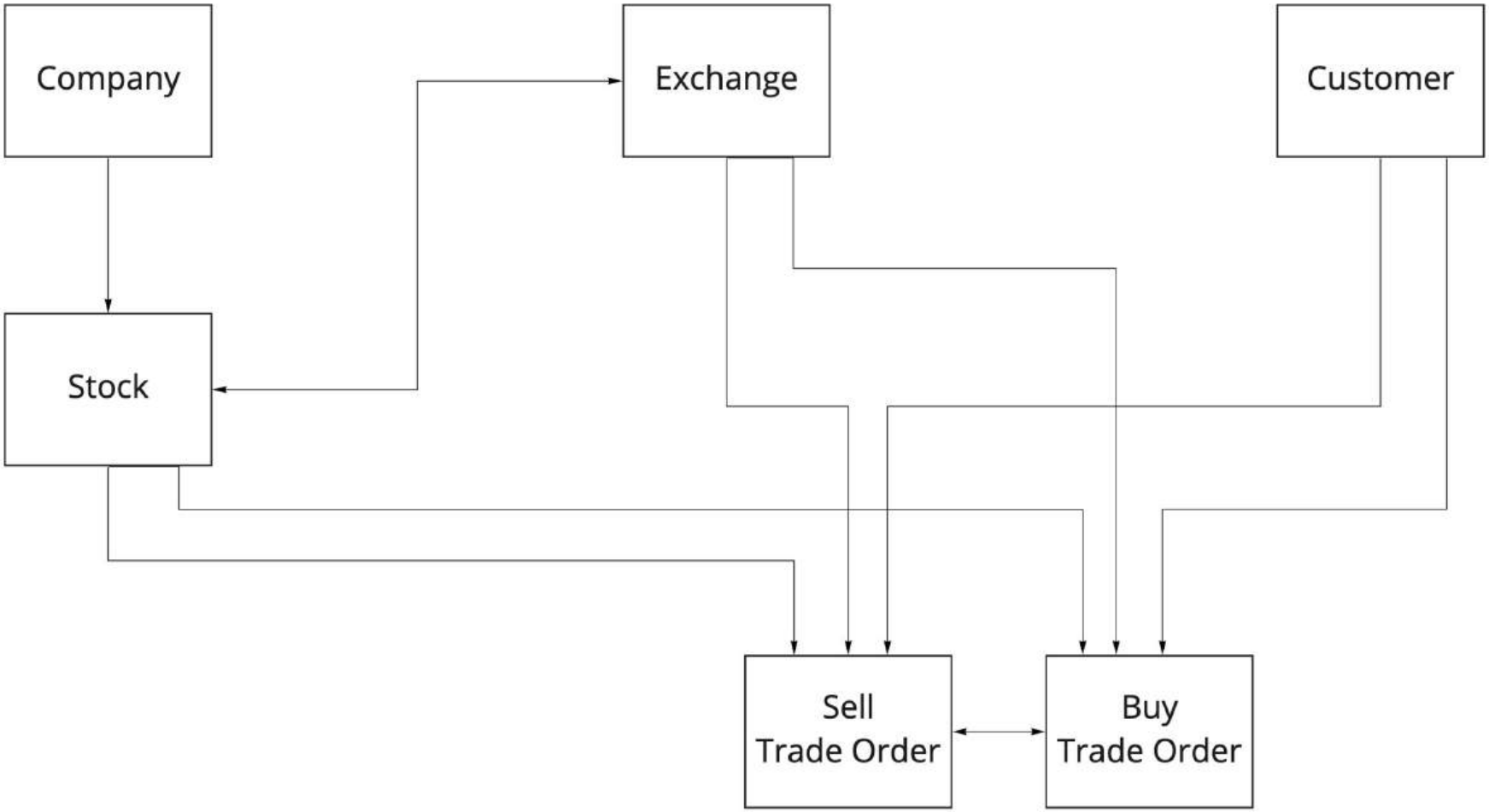
Sample Instance Model



Possible entities for the Stock Exchange model

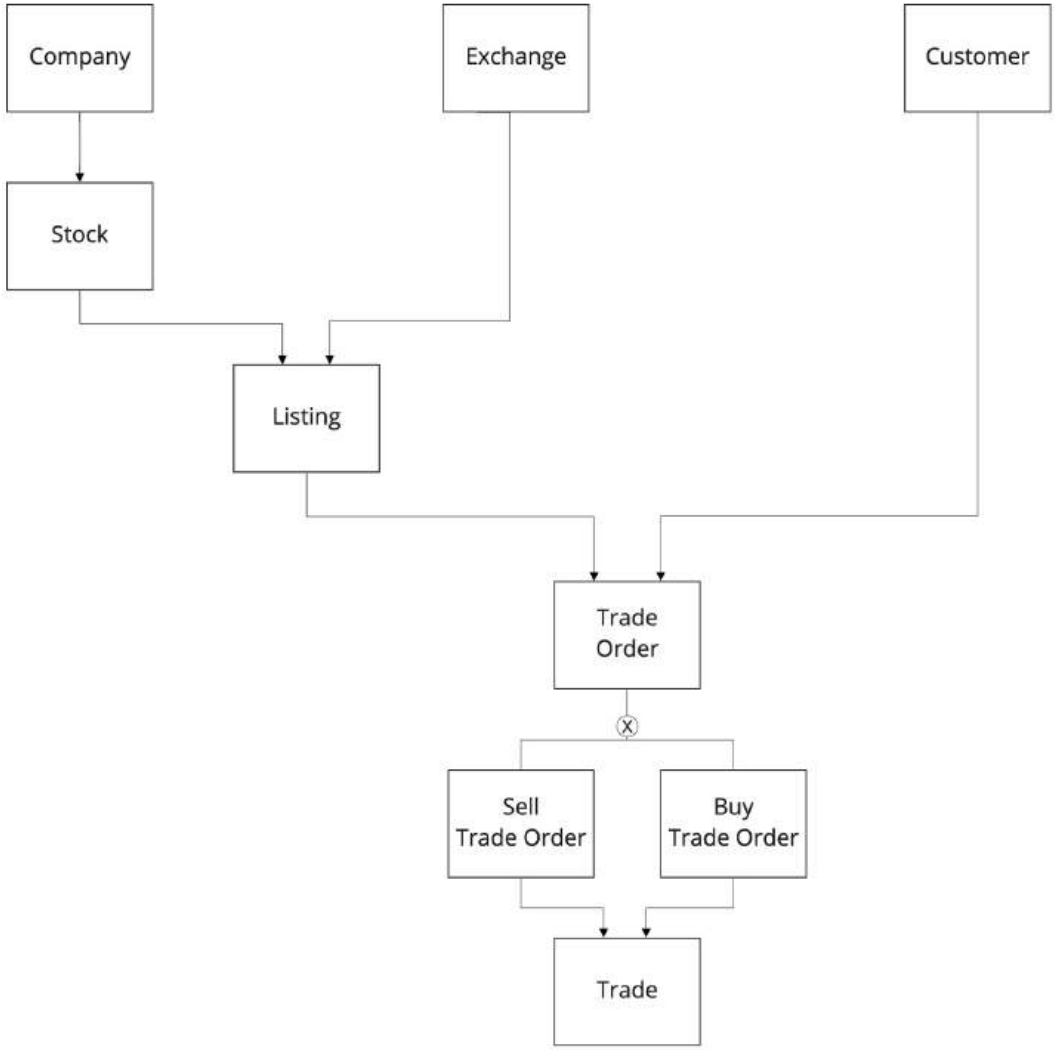
Initial Concept Model for Stock Exchange Trading

Instead of the "hash mark" (One) and "crowsfoot" (Many) this uses the symbol Miro offers – an arrowhead

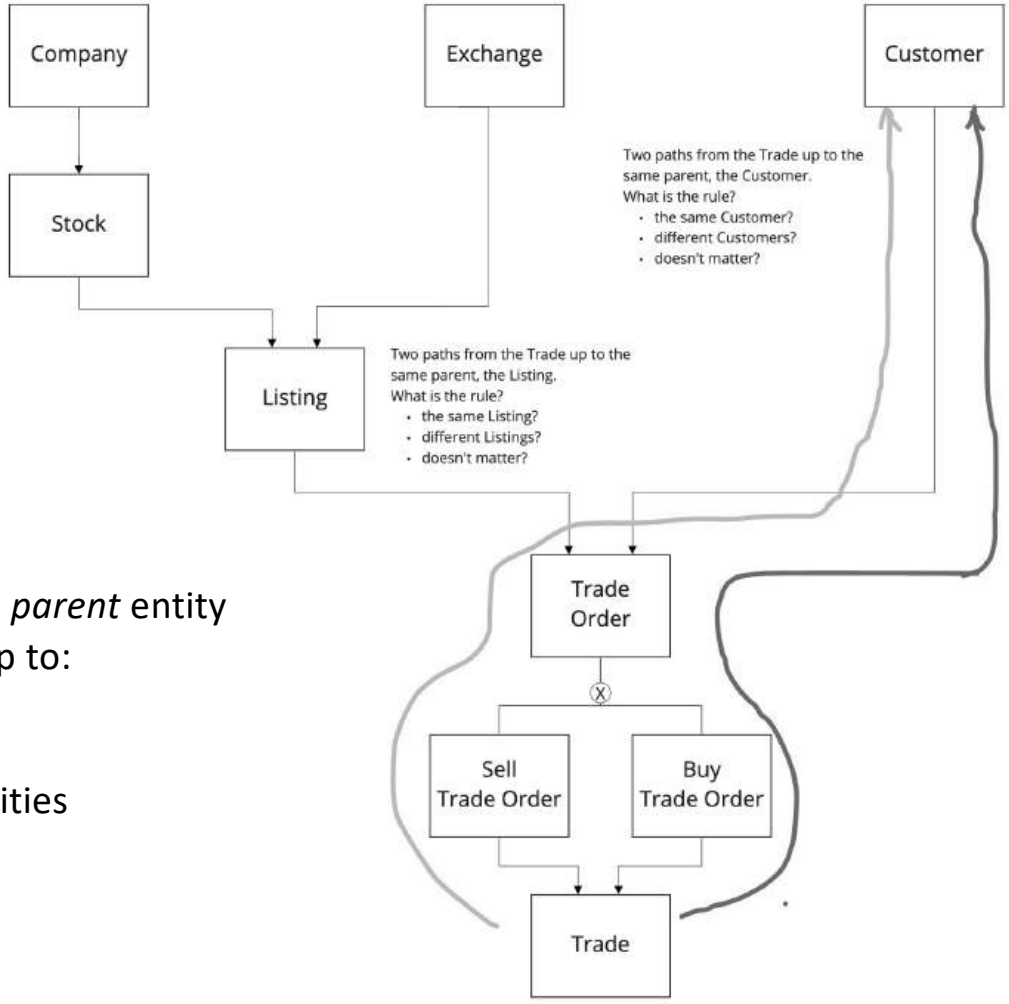


Second iteration Concept Model for Stock Exchange Trading

Key point!
Resolving the M:M between *Stock* and *Exchange* (creating *Listing*) and
generalising *Sell Order* and *Buy Order* into *Trade Order* makes the model *easier* to understand



An important constraint to check for



When there are *two* paths up to the *same parent* entity always check if the two paths must lead up to:

- the *same* parent entity
- *different* parent entities
- either the *same* or *different* parent entities (*it doesn't matter*)

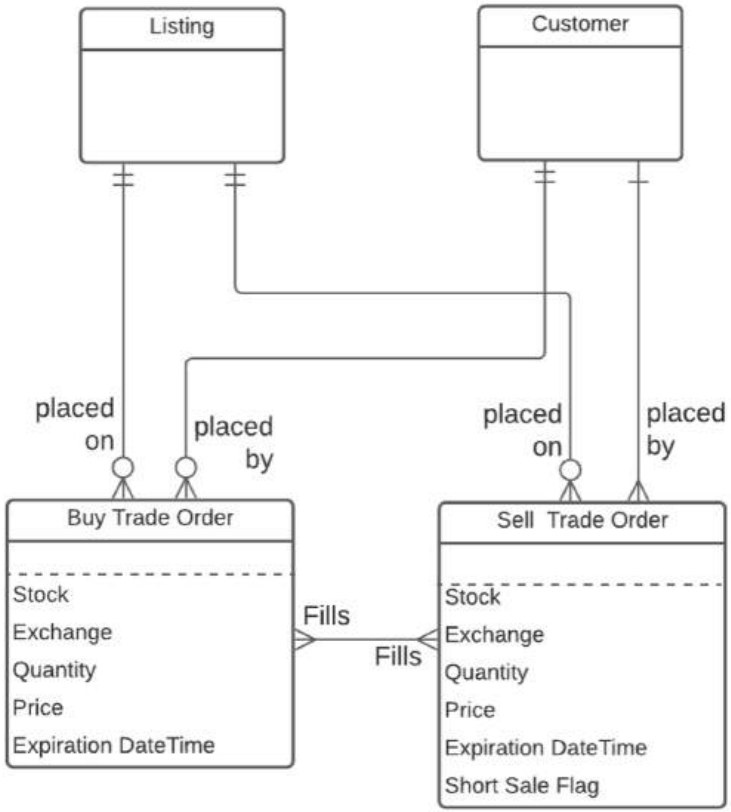
We must check the paths from

- Trade to Customer and
- Trade to Listing

Don't generalise too soon! Specifics first, generalisation later

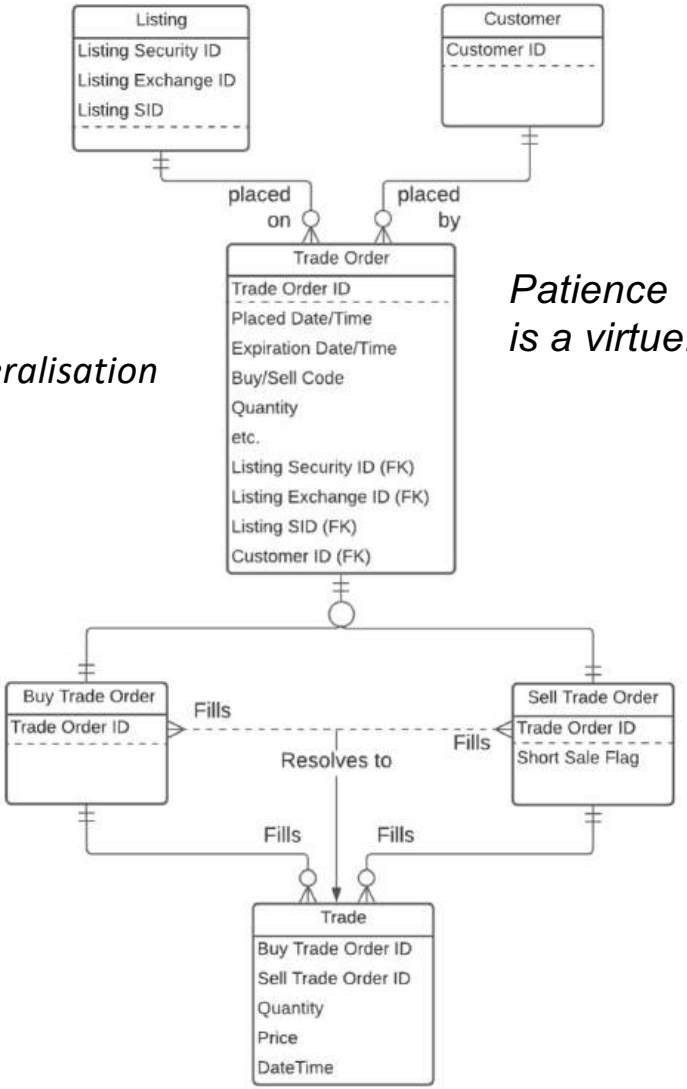
Client:
"Wow! Buy and Sell
Trade Orders
look very similar!"

Me:
"Omigosh!
That's amazing!"



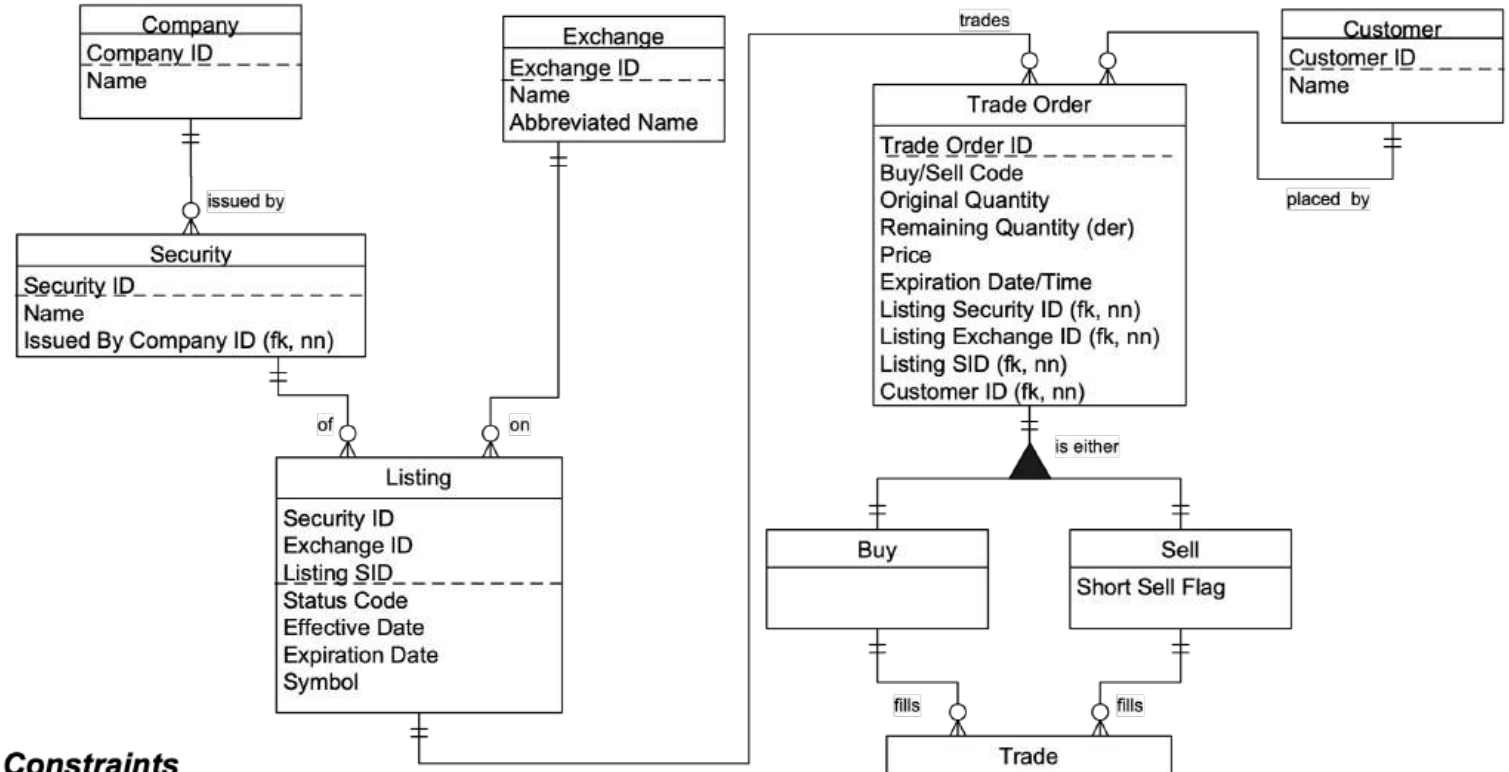
Specifics

Generalisation



Patience
is a virtue!

Complete solution: stock exchange trading



Constraints

- Buy Trade Order and Sell Trade Order involved in a Trade must reference the same Listing. Using fully inherited keys for Trade Order and Trade would make this easier to enforce.

Exercise: a flawed model including time dependent data

Jim needs to track List Price and Store Price over time for each Title the store carries. It is very important for him to be able to list the history of changes to either price, and the date, so that pricing information can be compared to sales figures.

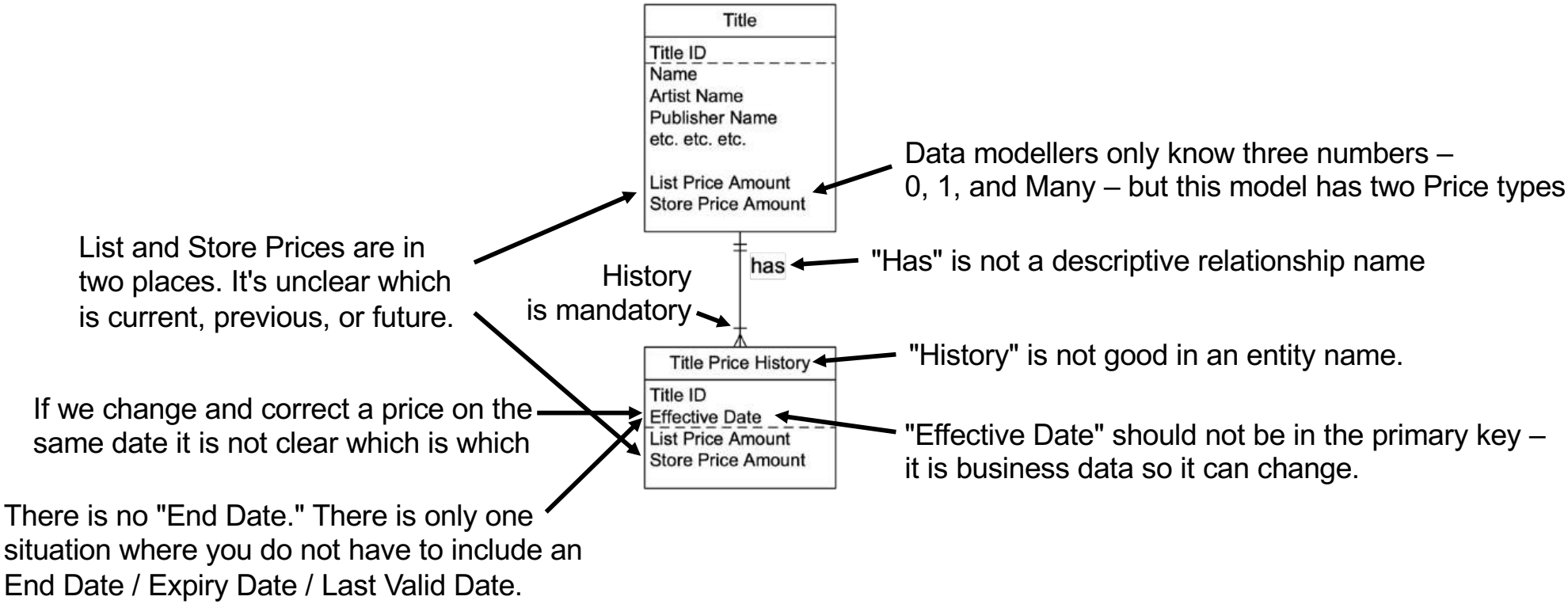
Either or both prices may change at any time. Often, a price change is known before it is due to take effect.

The model to the right has several flaws – try to list at least five.

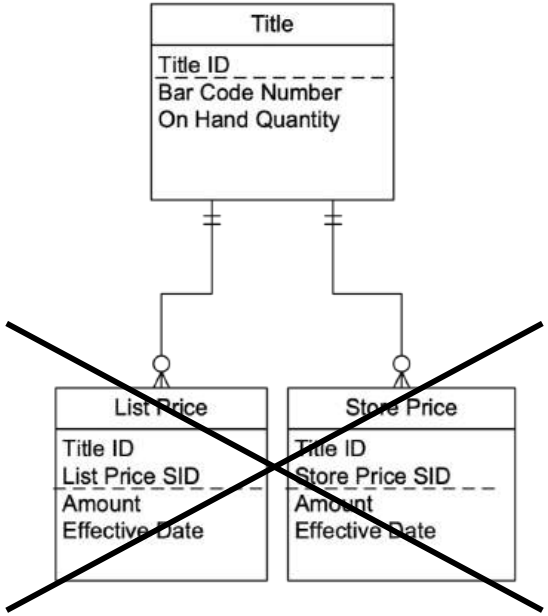
Then, construct a model (or a few alternatives) that will support Jim's needs.

A spot for you to think about that "Title Price" model

Solution – what was wrong with the flawed model?

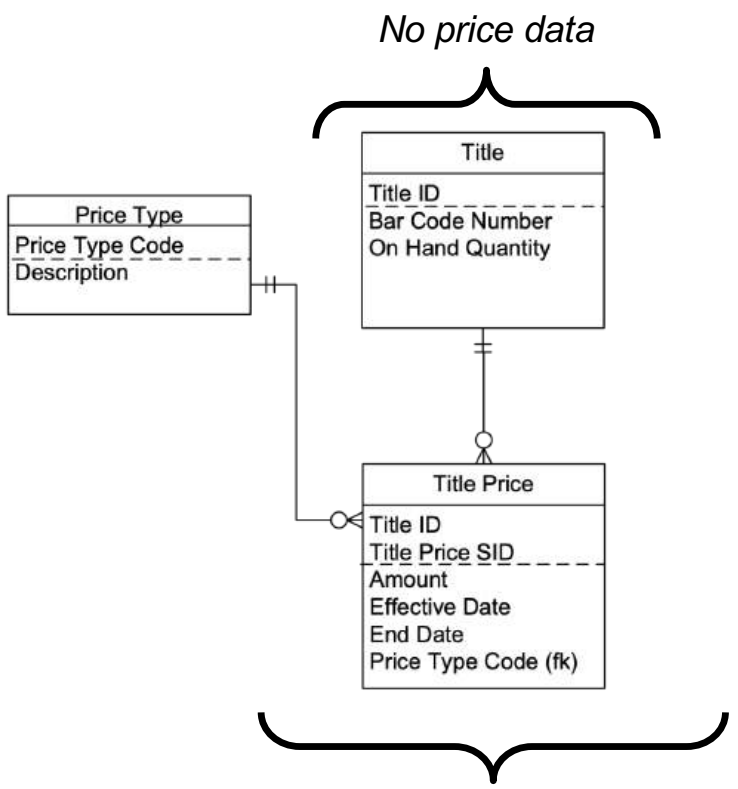


Best solution: time dependent data



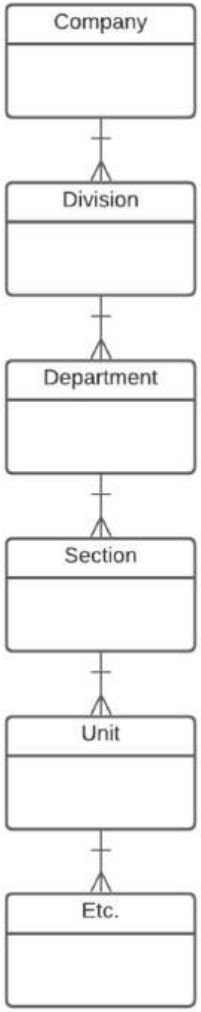
Two types of Price –
and *two* is not a number
Data Modellers recognise!

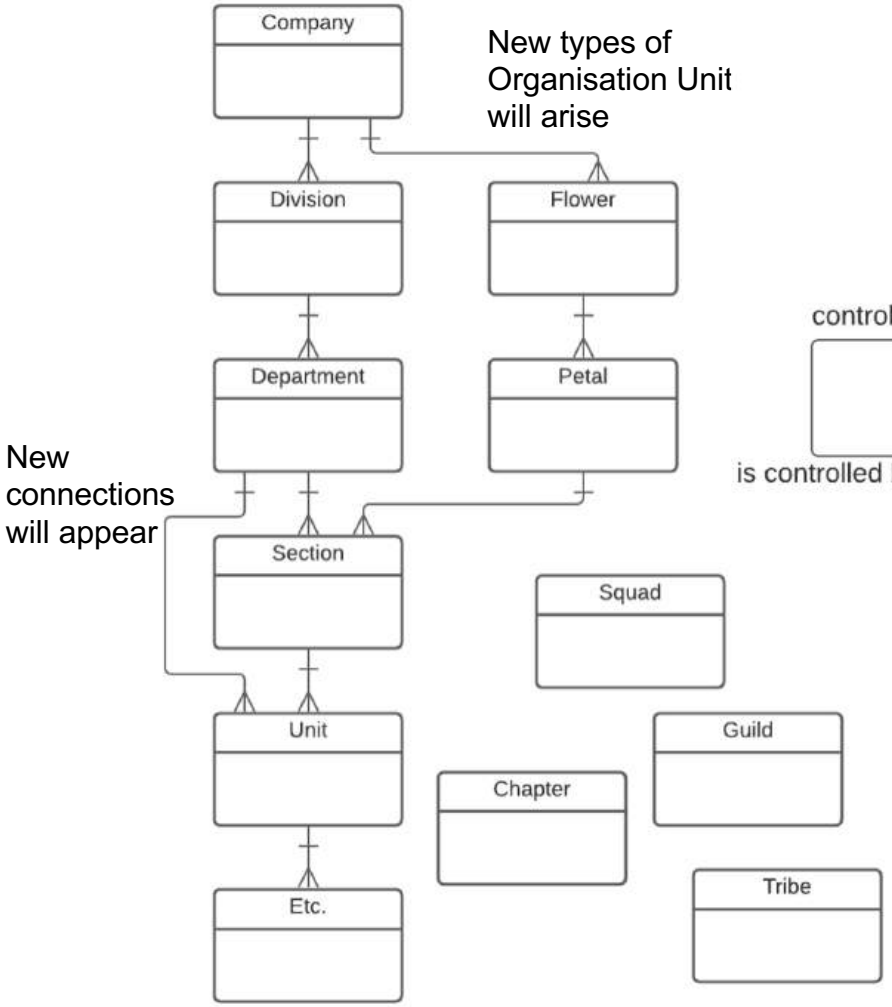
The most general solution



Future-proofing – "Avoid fixed hierarchies"

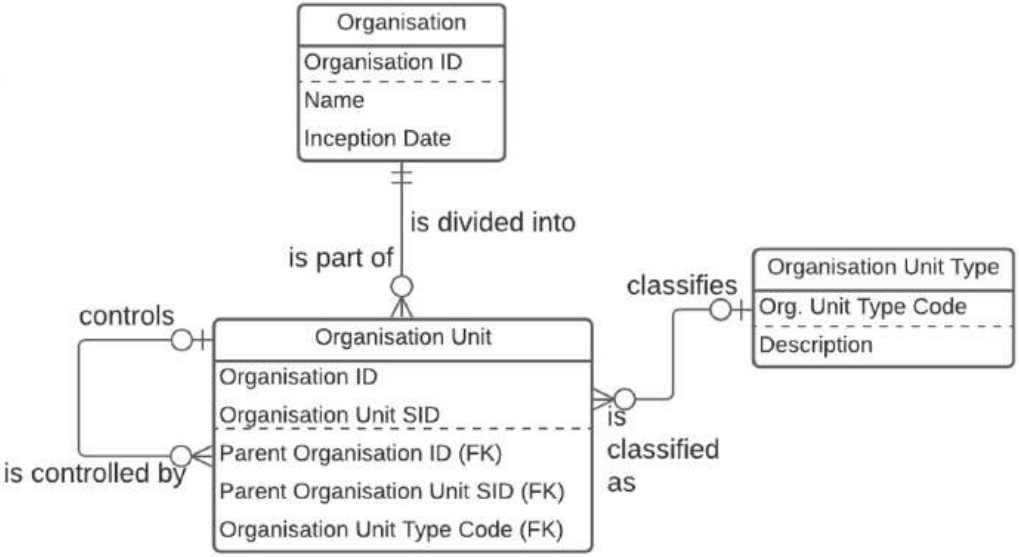
If we implement this model,
what will go wrong?





New types of
Organisation Unit
will arise

New
connections
will appear



This revised model is *far* more flexible.

This is an example of the
"generalise, recur, and classify" pattern.

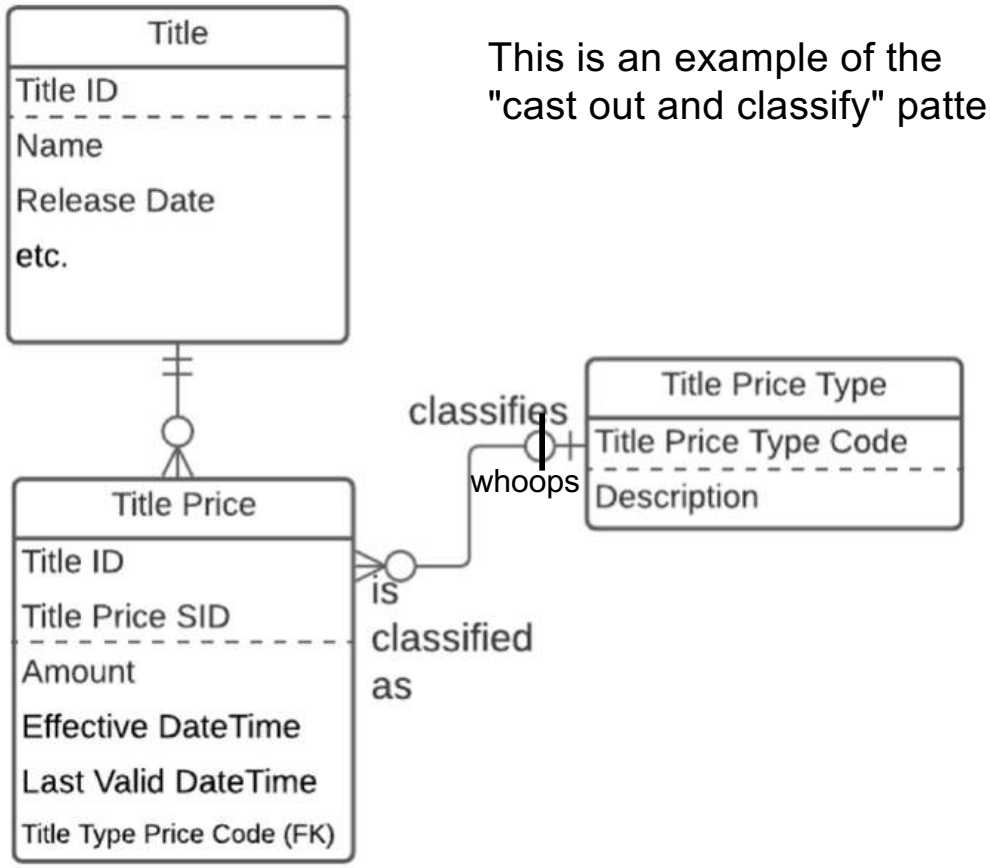
Future-proofing – "Avoid a fixed number of repeating attributes"

This model shows two types of Prices – List and Store.
Tomorrow a third will arise…
and a fourth and a fifth…

Data modellers only know three numbers –
0, 1, and Many.
We don't recognise *2*.

This revised model offers greater flexibility and supports richer queries.

This is an example of the "cast out and classify" pattern.



This slide added by mistake

Rules on relationships and associations

➡	*Outline*

1. Conceptual & Logical models, level-set
2. Interesting structures
3. Rules on relationships and associations
4. Presentation techniques for data modellers

★	*Topics*

- V-A-K – using all the tools to build a model
- Multi-way associations and complex rules
- Advanced normal forms – resolving circular relationships and cyclic dependencies

!	*Themes*

- *Say* the requirement to be sure you understand it
- *Draw* schematics and examples

Four key points about complex associations

1. You can't tell whether a model is correct or not simply by inspecting it – you must have business involvement

   This gives rise to the other three points…

2. You must draw the model in a top-down fashion (or other systematic approach) so you can actually *see* dependencies

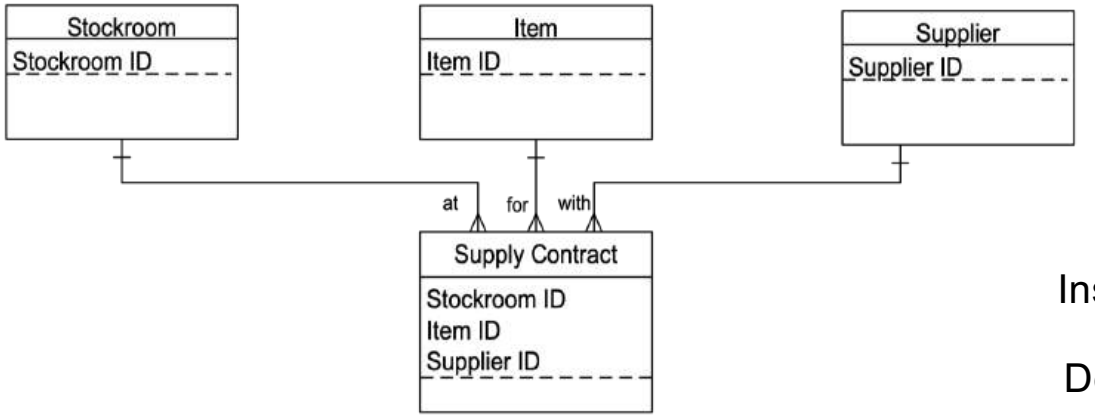
3. You must state your assumptions or understanding in narrative form as *assertions*, using terms (entity names, relationship names, and attribute names) from the data model

4. You must *illuminate* the data model by using sample data, schematic diagrams, scenarios, or some other understandable form

A quick exercise…

1. The company decides which items will be carried at which stockrooms.
2. The company qualifies suppliers to provide specific items.
   (A supplier can be qualified to provide multiple items, and an item may be provided by multiple suppliers)
3. The company enters into a contract with qualified suppliers for each item they will provide to a specific stockroom.

Will this model satisfy the business constraints?
If not, identify specific problems and develop a better model



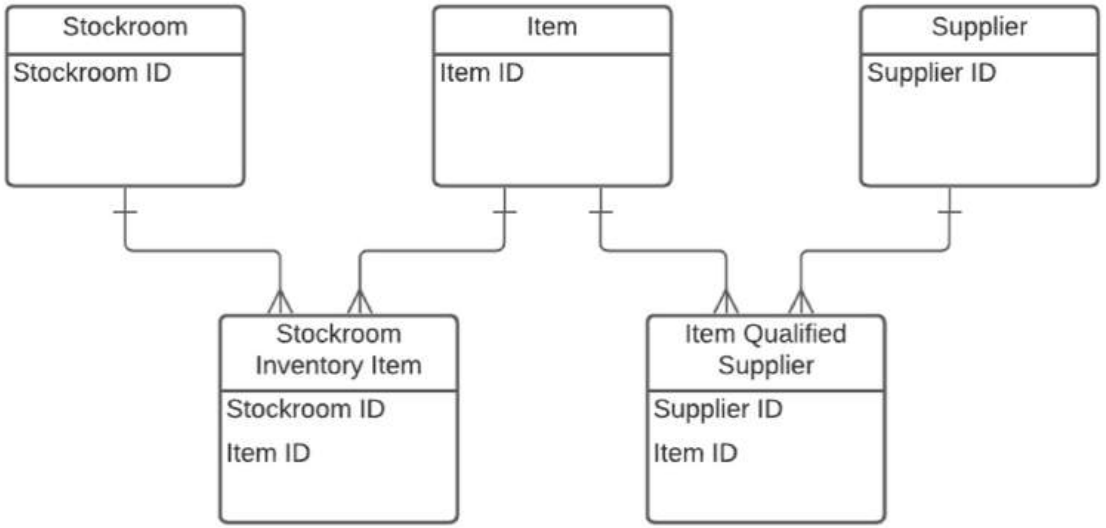
Can't record independent Supplier-Item relationship without including Stockroom – ~~"Stockroom #9999999" – a dummy Stockroom~~

Can't record independent Stockroom-Item relationship without including Supplier – ~~"Supplier #9999999" – a dummy Supplier~~

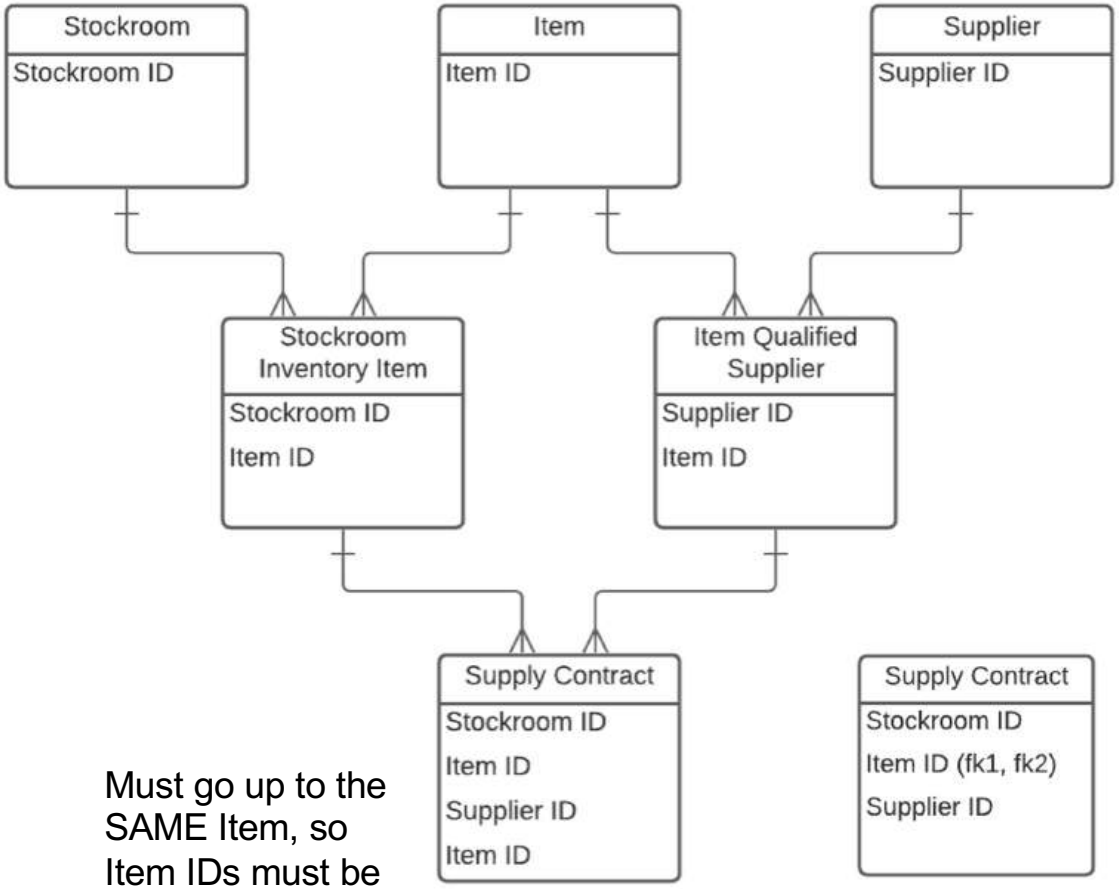
Insert anomalies

Delete anomalies

First, record independent relationships



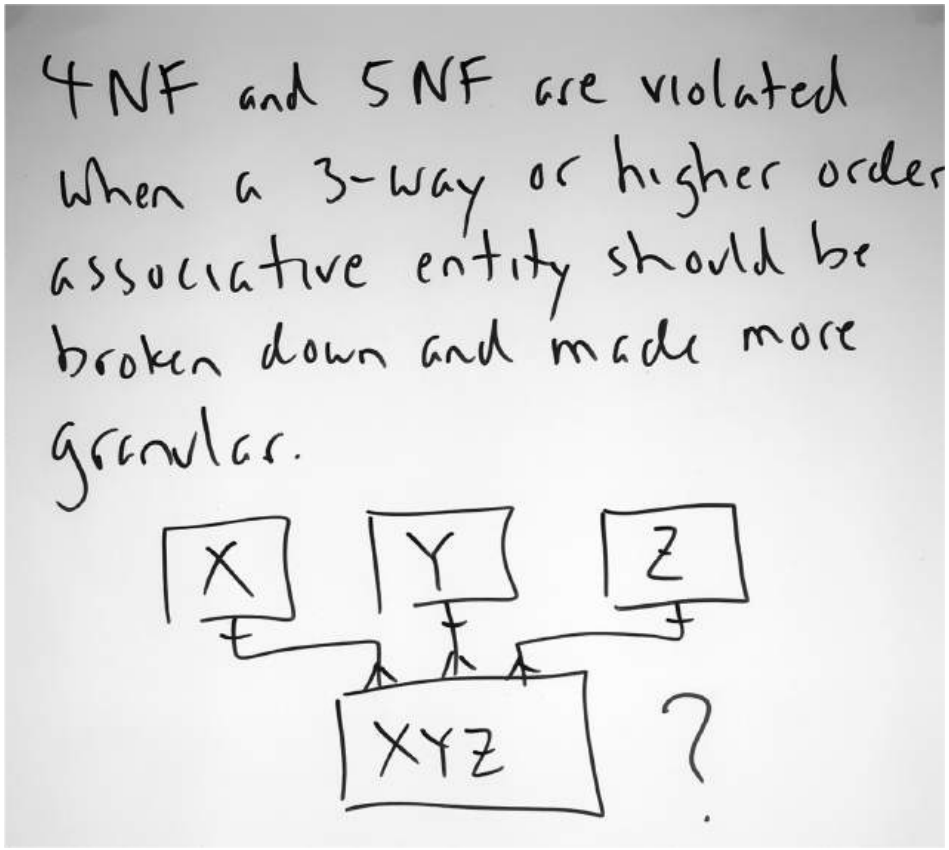
Associate the associatives

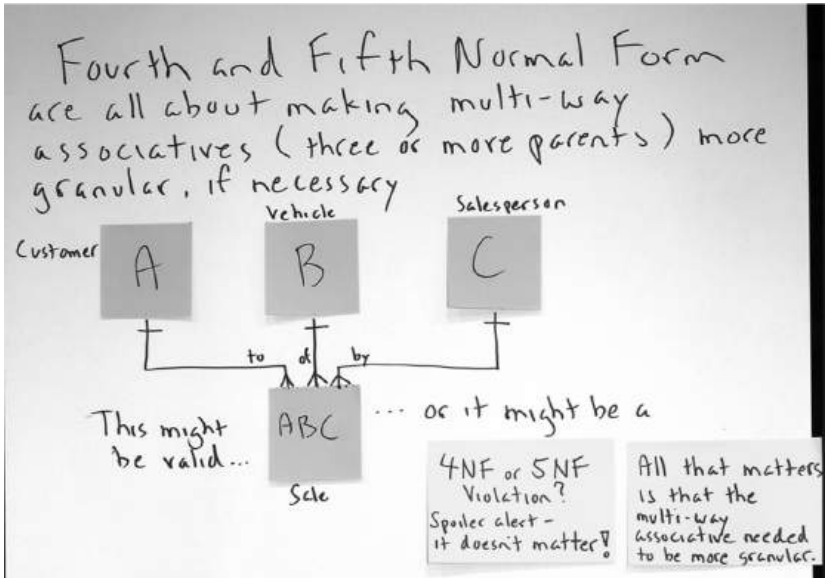


Must go up to the SAME Item, so Item IDs must be the same!

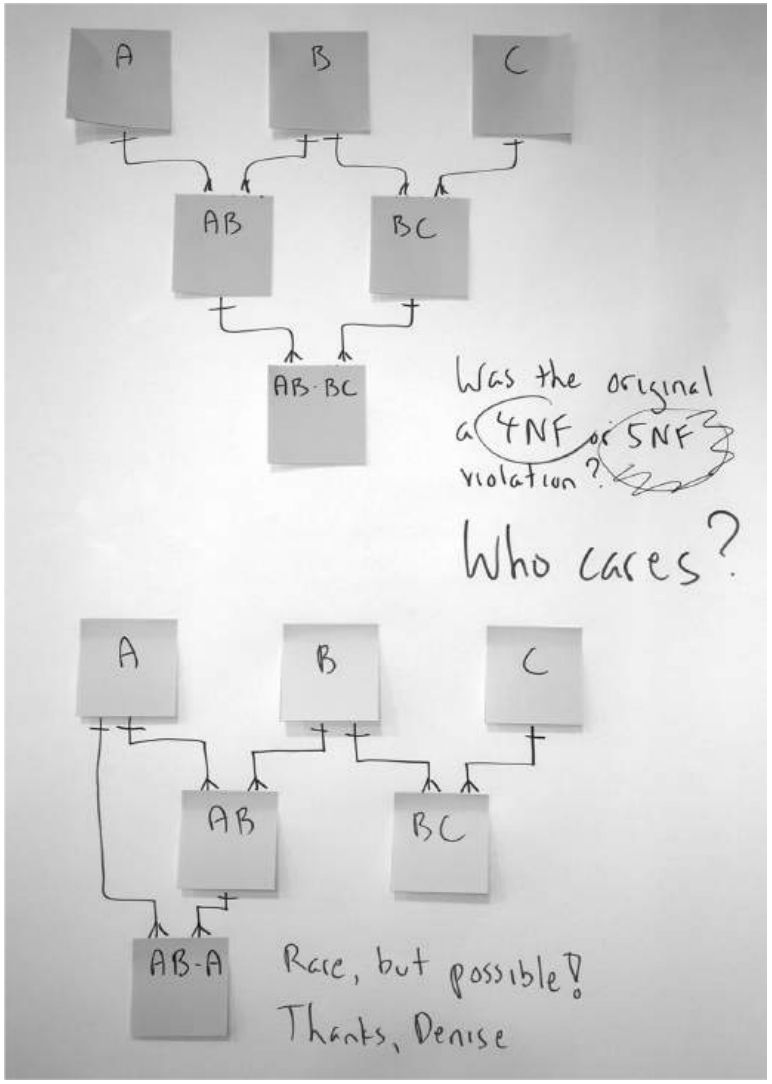
Or use one Item ID as part of two foreign keys

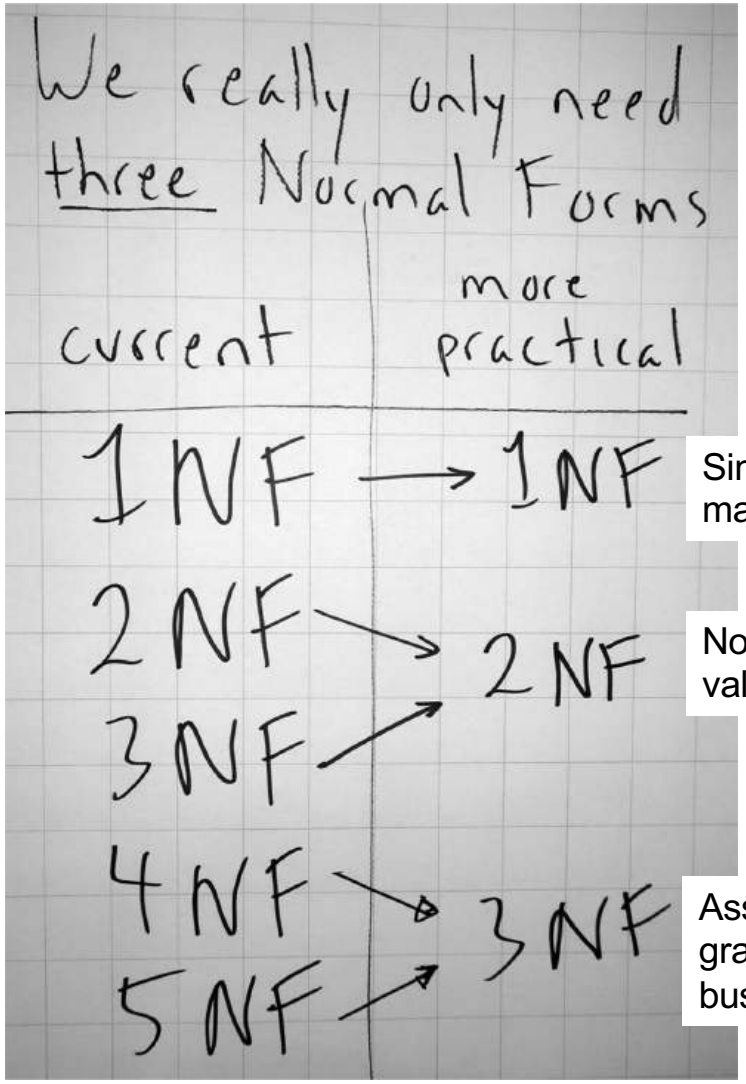
Fourth Normal Form and Fifth Normal Form





More possibilities…





Single-valued attributes make data more "reportable."

Non-redundant – each attribute value is recorded only once.

Associative entities are as granular as necessary to reflect business rules.

This slide added to maintain balance in the universe

Presentation techniques for data modellers

➡	*Outline*

1. Conceptual & Logical models, level-set
2. Interesting structures
3. Rules on relationships and associations
4. Presentation techniques for data modellers

★	*Topics*

- A learning experience
- Storyboarding the presentation
- General guidelines
- A demonstration
- Recap - five key techniques

!	*Themes*

- It's a story, not a data model!
- It's marketing,
  not a data model!
- A few key techniques, not
  hundreds of “tricks”

Presentations – my "new Customer data model" experience…

Road show version 1

"Here's the new Customer model. How do you like it?"

"So what?" "Obvious!" "Yawn." "ZZZZZZzzzzzzzz…."

VP of IS:

"You're dyin' out there, kid! I want you to drag them through all of the pain and misery of our current files and databases.
Then show the new model."

Road show version 2

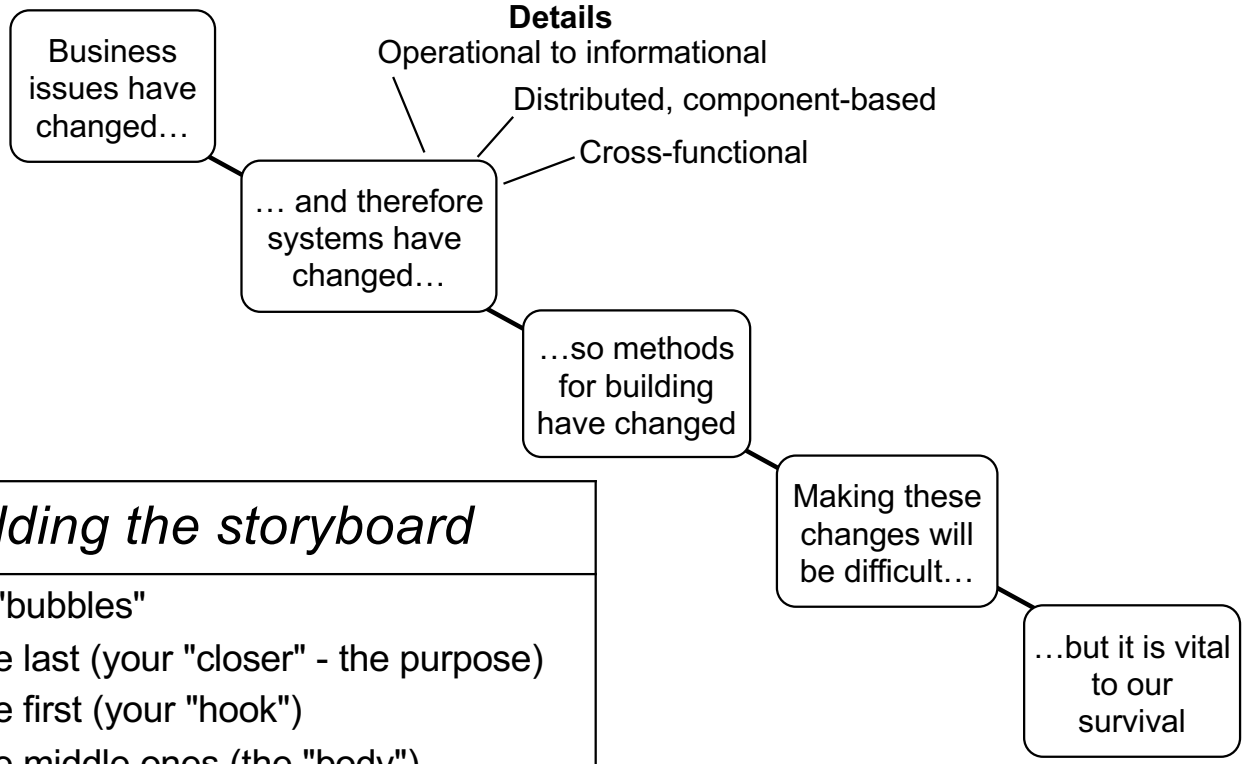
"Let's try answering some important questions using the current model, and then the new model"

"Fantastic!"

"When do we get it?"

"Do you need funding? We can help!"

Point 1 - it's a story, so storyboard it



Building the storyboard

1. Draw 5 "bubbles"
2. Fill in the last (your "closer" - the purpose)
3. Fill in the first (your "hook")
4. Fill in the middle ones (the "body") –
   add or subtract bubbles as needed
5. Allocate details to bubbles
6. Iterate until it flows and builds properly

Only include detail that matters!

Point 2 – remember the basics of presenting data models

Try not to call it a *data model*

- I often call it a "world view"
- Or… "This is how Application XYZ sees the world."

Start simple, and add details in layers

- Begin with two or three fundamental things
- Work "across" the model, not a "deep dive" in one area
- Draw the model on a whiteboard as you speak to it
- Save detail like optionality until later, and primary/foreign keys until *much* later

Speak exclusively in the language of the *business*

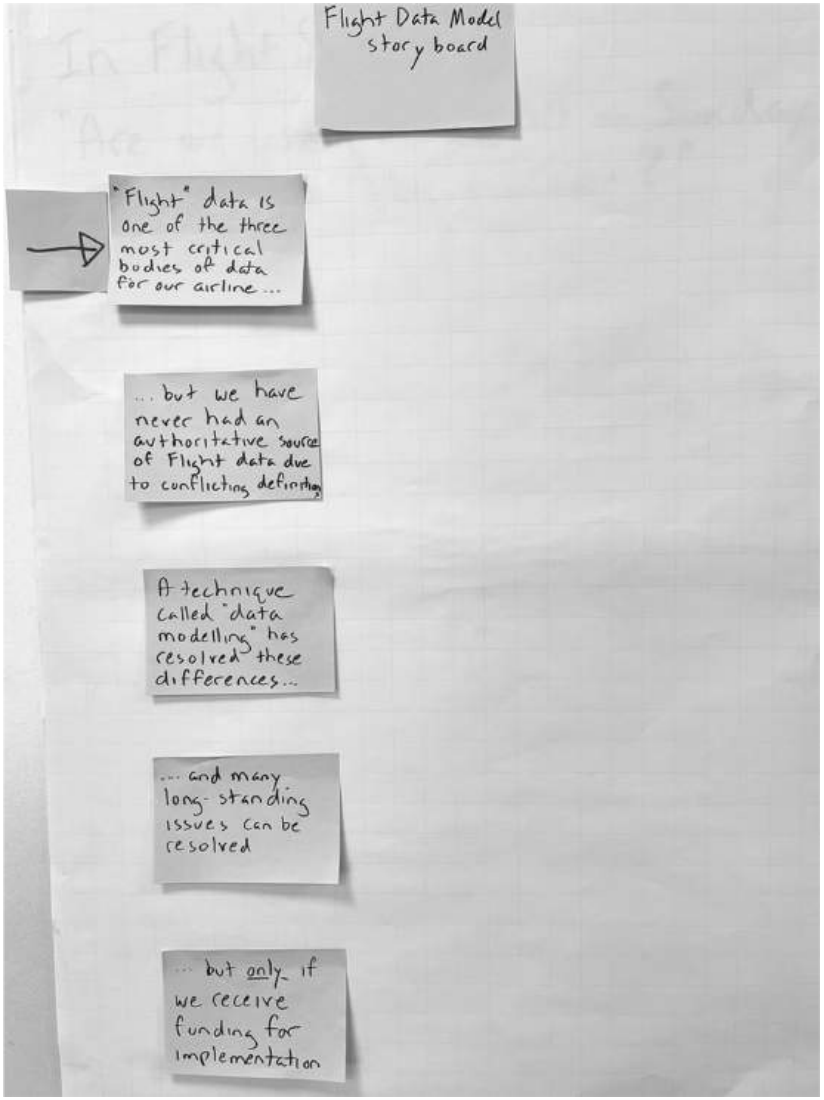
- Don't use terms like entity, relationship, attribute, optionality, supertype, subtype, recursion, etc. Remember, you're describing a *business,* not a *database.*
- Point to the relevant entity while addressing a concept
- Someone overhearing your presentation should not realise you are presenting a data model

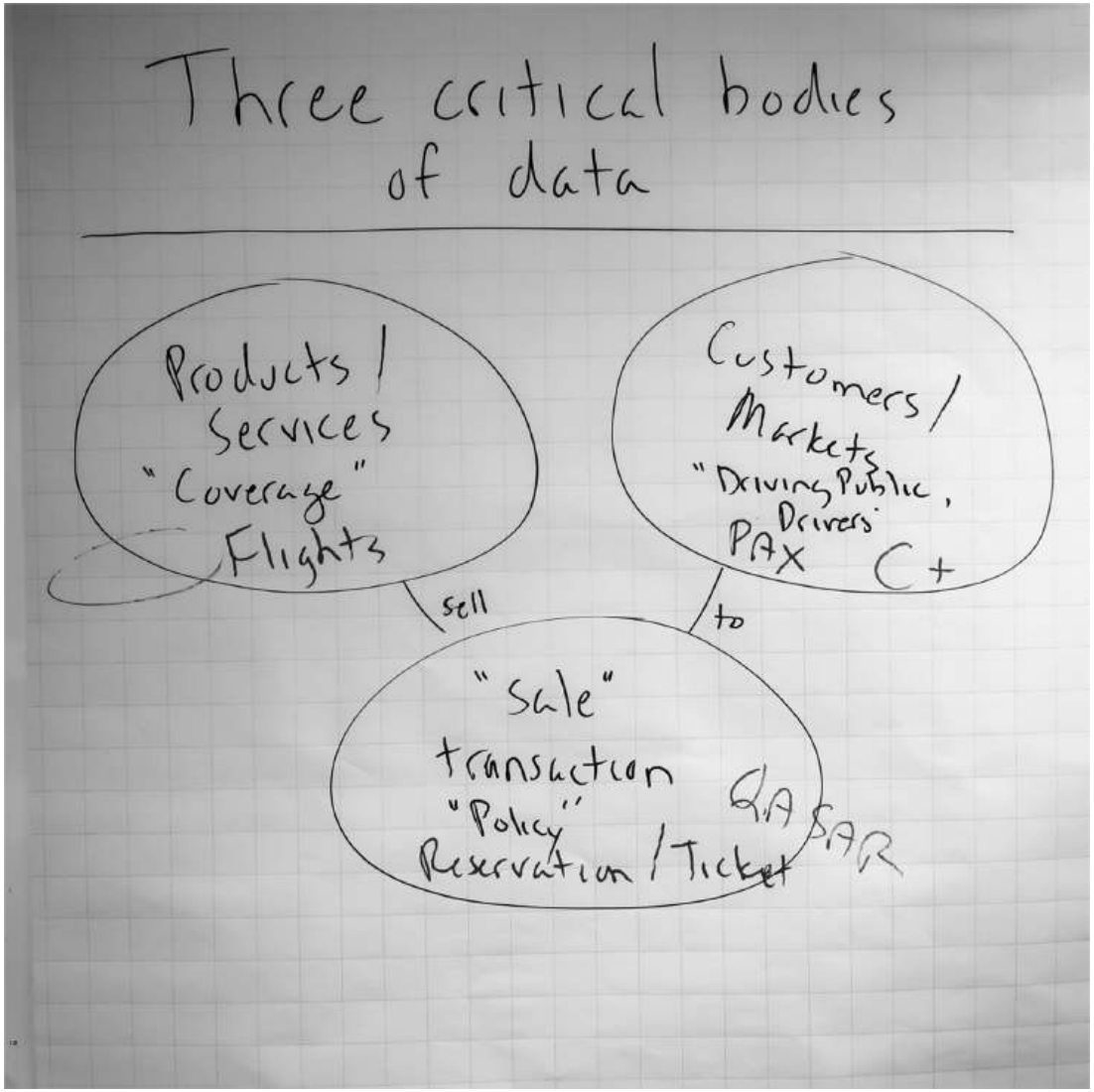
Make it *real*

- Back it up with sample data, queries, and scenarios
- Identify specific business issues or opportunities, and show how the data model helps

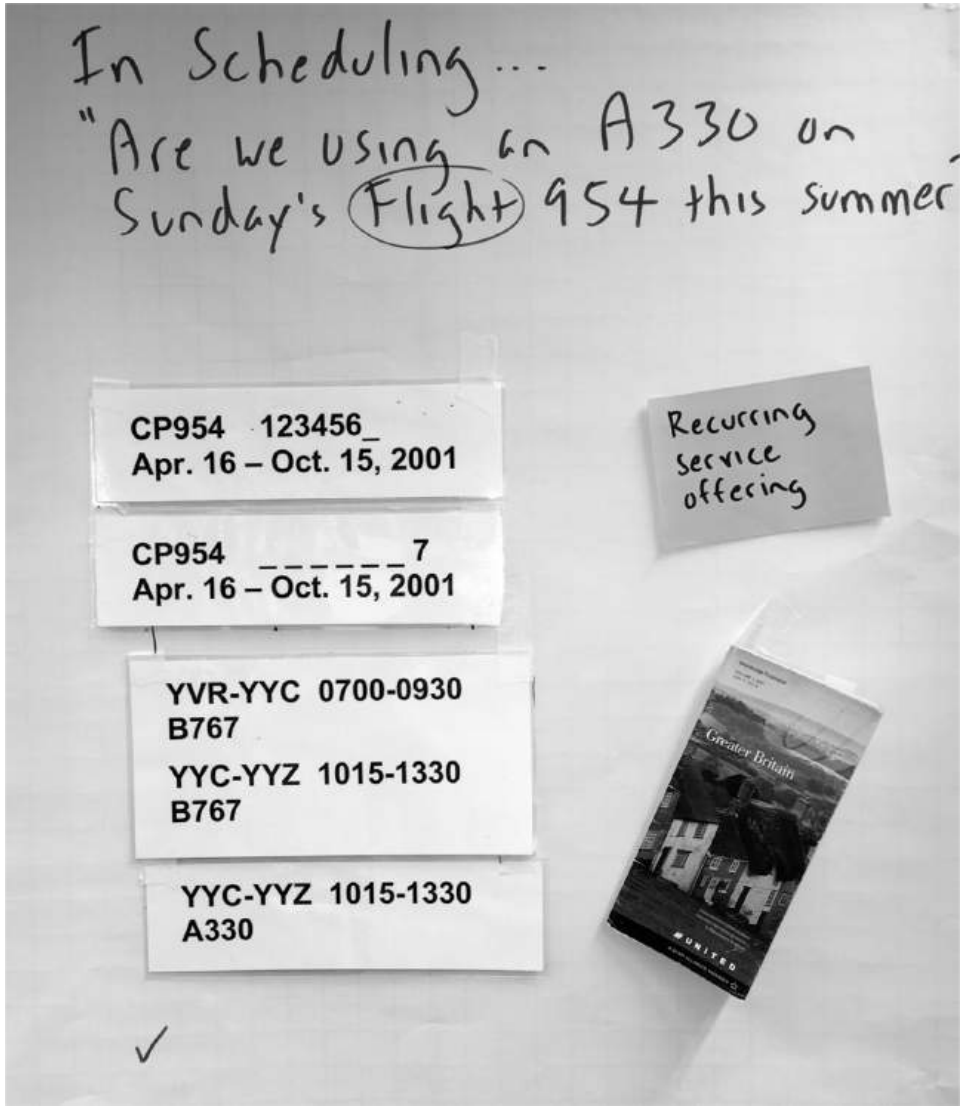
We'll now walk through a successful data model presentation, followed by a discussion of key points

Storyboard & 1st point for the "Flight Data Model" presentation

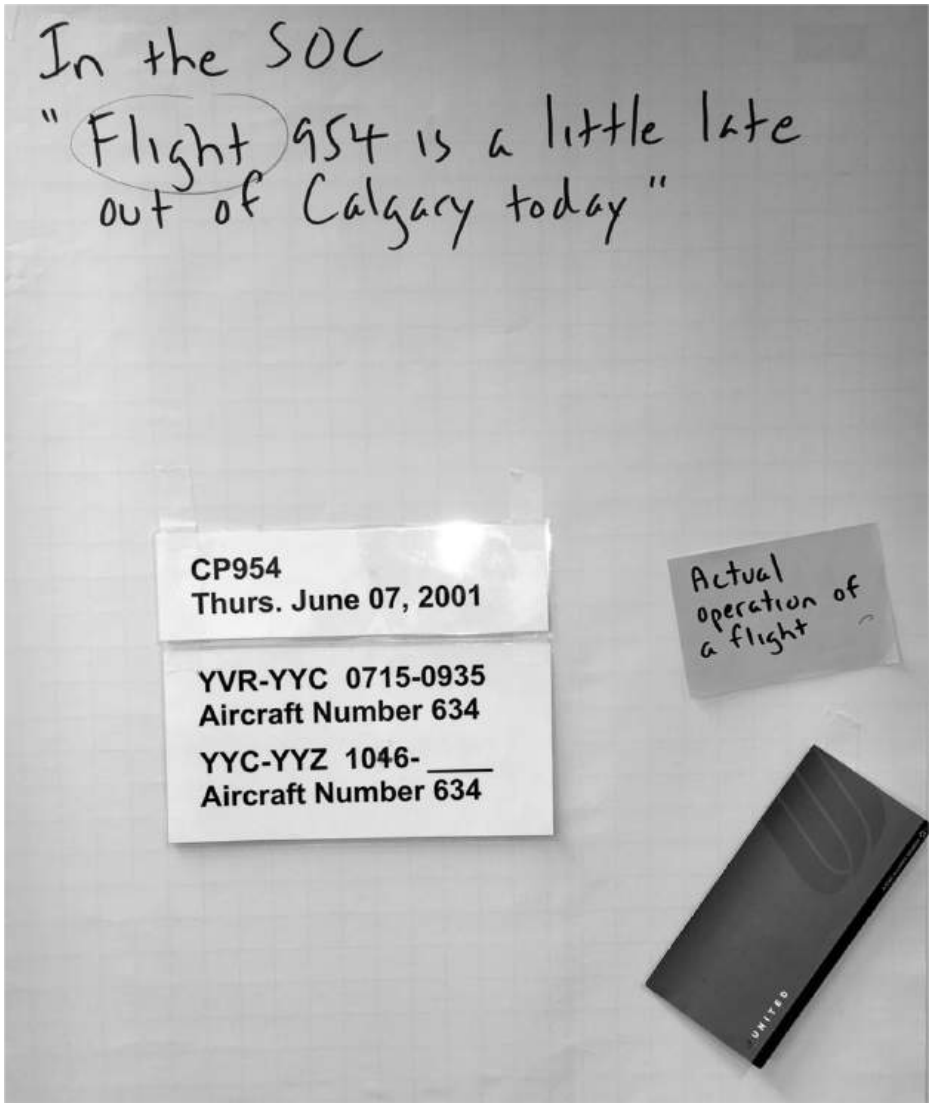




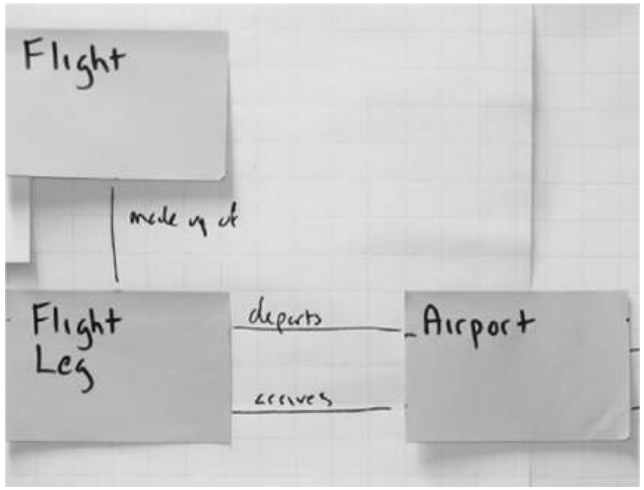
"Flight Data Model"

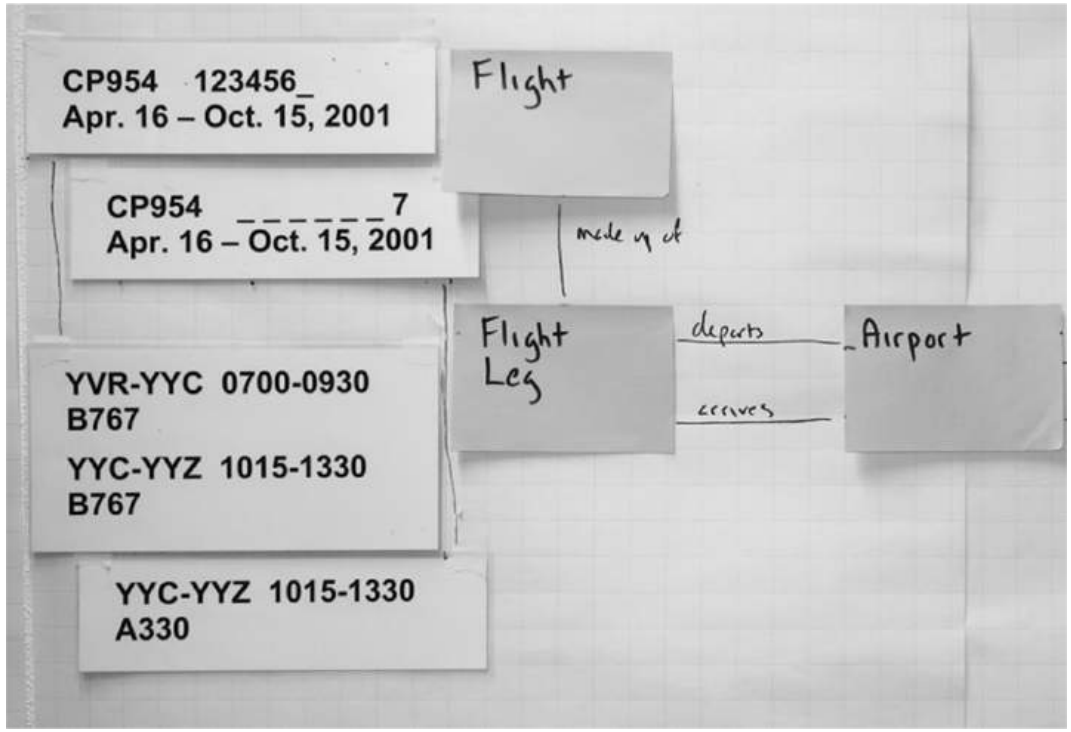


"Flight Data Model"

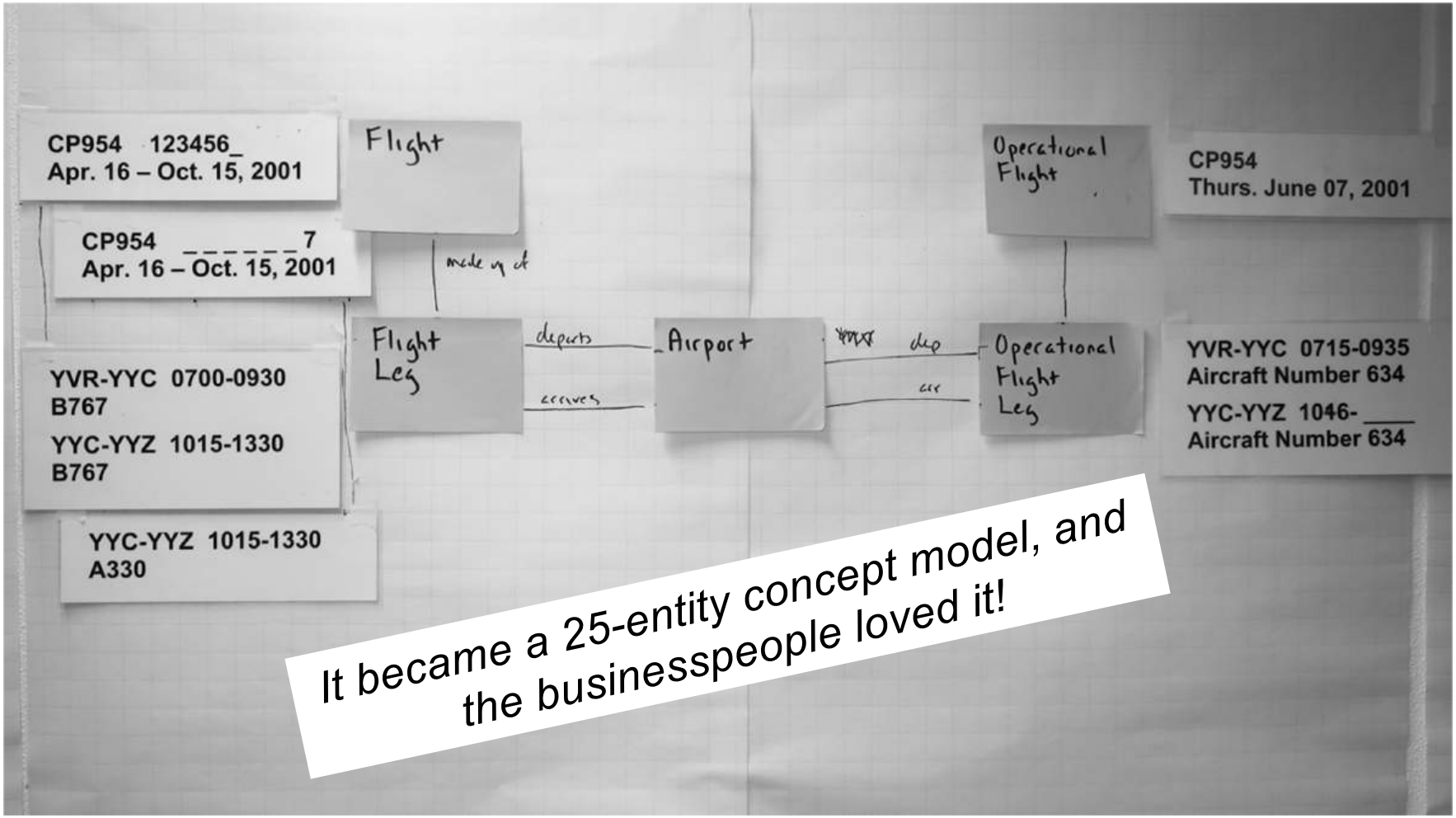


"Flight Data Model"





"Flight Data Model"



The five techniques that really matter

	Technique	Why?	How?
1	Organise their minds to receive the presentation	· Otherwise, you're just "noise" · "Why is this person telling me these things?"	· "Here's the point I want to make." · "This is why you care, and how I know." (even if it's obvious) · "These are the caveats and limitations." · "This is how I'll make my point." (storyboard!)
2	Big picture first	· Provides context and perspective · Makes subsequent detail understandable	· Show contextual data model first, build up detailed models later · Process context first, process flow later · Describe 5 problem areas first, specifics of each area later
3	Do it live	· Focuses, demands that they watch · Involves them / you · It means 'attending has value'	· Use memory triggers, not a script · Build up content progressively on white board, flip chart, or screen · Add brainstorming, discussion, or questions · Have them physically "do stuff"
4	Present information in various forms	· Adds interest · Different forms have different strengths	· Supplement PowerPoint slides with flip charts, white boards, Post-Its, handouts, etc. · Use props – the thing itself, not a description · Use visual, auditory, and kinesthetic approaches
5	Show, then tell	· Point is more meaningful if experienced firsthand · Saves time, simplifies	· Scenario / example first, then concept / abstraction · Problem first, solution second · Thing first, description / discussion second

Other courses for analysts by Alec Sharp

Working With Business Processes – *Process Change in Agile Timeframes* 2 days

Business processes matter, because business processes are how value is delivered. Understanding how to work with business processes is now a core skill for business analysts, process and application architects, functional area managers, and even corporate executives. But too often, material on the topic either floats around in generalities and familiar case studies, or descends rapidly into technical details and incomprehensible models. This workshop is different – in a practical way, it shows how to discover and scope a business process, clarify its context, model its workflow with progressive detail, assess it, and and transition to the design of a new process by determining, verifying, and documenting its essential characteristics. Everything is backed up with real-world examples, and clear, repeatable guidelines.

Business-Oriented Data Modelling – *Useful Models in Agile Timeframes* 2 days

Data modelling was often seen as a technical exercise, but is now known to be essential to other initiatives such as business process change, requirements specification, Agile development, and even big data, analytics, and data lake implementation. Why? – because it ensures a common understanding of the things – the entities or business objects – that processes, applications, and analytics deal with. This workshop introduces concept modelling from a non-technical perspective, provides tips and guidelines for the analyst, and explores entity-relationship modelling at contextual, conceptual, and logical levels using techniques that maximise client involvement.

Working With Business Processes Masterclass – *Aligning Process Work with Strategic, Organisational, and Cultural Factors* 3 days

This 3-day interactive workshop combines the core content from two highly-rated classes by Alec Sharp – "Working With Business Processes" and "Advanced Business Process Techniques." This structure is popular because it gets both new and experienced practitioners to the same baseline on Claritiq's unique, agile, and ultra-practical approach to Business Process Change. First, it shows how to effectively communicate Business Process concepts, discover and scope a business process, assess it and establish goals, and model it with progressive detail. Then, it shifts to advanced topics – specific, repeatable techniques for developing a process architecture, encouraging support for change, and completing a feature-based process design. The emphasis is always on ensuring business process initiatives are aligned with human, social, cultural, and political factors, and enterprise mission, strategy, goals, and objectives.

Business-Oriented Data Modelling Masterclass – *Balancing Engagement, Agility, and Complexity* 3 days

Our most popular workshop! This intensive 3-day workshop combines the core content from two popular offerings by Alec Sharp – "Business Oriented Data Modelling" and "Advanced Data Modelling." First, the workshop gets both new and experienced modellers to the same baseline on terminology, conventions, and Clariteq's unique, business-engaging approach. We ensure a common understanding of what a data model *really* is, and maximising its relevance. Then, we provide intense, hands-on practice with more advanced situations, such as the enforcement of complex business rules, handling recurring patterns, satisfying regulatory requirements to model time and history, capturing complex changes and corrections, and integrating with dimensional modelling. Always, the philosophy is that a data model is a description of a business, not of a database, and the emphasis is on engaging the business and improving communication.

Model-Driven Business Analysis Techniques – *Proven Techniques for Processes, Applications, and Data* 3 days

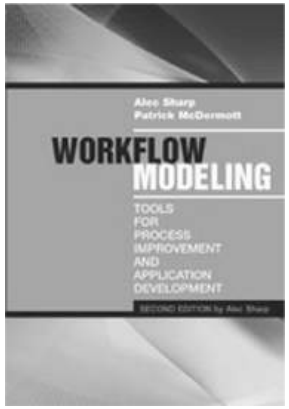
Simple, list-based techniques are fine as a starting point, but only with more rigorous techniques will a complete set of requirements emerge, and those requirements must then be synthesised into a cohesive view of the desired to-be state. This three-day workshop shows how to accomplish that with an integrated, model-driven framework comprising process workflow models, a unique form of use cases, service specifications, and business-friendly data models. This distinctive approach has succeeded on projects of all types because it is "do-able" by analysts, relevant to business subject matter experts, and useful to developers. It distills the material from Clariteq's three, two-day workshops on process, data, and use cases & services.

**** Note: two-day in-person workshops are delivered virtually as three half-day sessions via Zoom.*
Three-day in-person workshops are delivered virtually as five half-day sessions via Zoom.

Thank you – stay in touch!

Alec Sharp
Clariteq Systems Consulting Ltd.
West Vancouver, BC, Canada

- asharp@clariteq.com
- ig: @alecsharp01
- www.clariteq.com
- http://amzn.to/dHun1o



And most of all, if you have questions or comments…
don't be shy – send me a note!