Concept Modelling for Business Analysts –
Making Data Modelling a Vital Technique

A half-day workshop presented by Adept Events
03 april 2025 in Utrecht NL

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

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

Process — Business Process Modelling

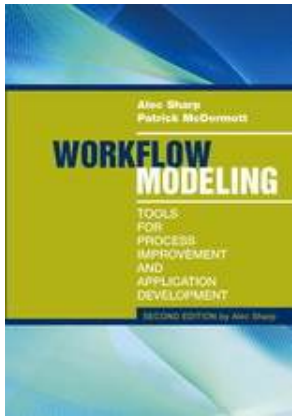
Application — Use Case Modelling, Service Specification

Data — Concept Modelling

- 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

Clariteq – small, husband & wife company, global clients

ABB (ASEA Brown Boveri)
Aflac
American Honda
AMP (Australia Mutual Provident)
BackOffice Associates
Bank of Finland
Bellrock
Brisbane City Council (Australia)
Canadian Natural Resources Ltd.
City of Seattle
Civica UK
Clearwater Paper
Corvias
Dell
DHL Express
Dutch National Bank
Ericsson
Essity
Eurojust (European Justice Comm.)
European Central Bank
Fortum
Helse Vest - Norway
HM Land Registry - UK
Home Depot
Idaho Transportation Dept.
Intel
ISO New England
ING Bank

JP Morgan
Kal Tire
KONE
LGM Financial Services
Liberty Mutual
Livestock Improvement Corp.
MacDonald Dettwiler
Manitoba Public Insurance
Marathon Pipe Line
Microsoft
Ministry of Defence - UK
Ministry of the Interior - Slovakia
MTS Allstream
Nexen
Novo Nordisk
Nusenda Credit Union
OP Bank
Partner Reinsurance
Ritchie Brothers
Phillip Morris
Roche Diagnostics
Salt River Project
Saudi Aramco
Serco
Shell
Sparta Consulting
State Street Bank
SunGard

Synechron
Sysdoc
Talent Base
Teck
The MUSIC Group
The Seattle Times
UK Government
University Med Ctr Groningen
Washington Gas & Light

– Higher Education –
Carnegie Mellon University
Cornell University
Douglas College
Gonzaga University
Humboldt State University
The Jackson Laboratory
The Ohio State University
Portland State University
Salt Lake Community College
Southern NH University
University of Arkansas
University of British Columbia
University of the Fraser Valley
University of Maryland
University of Utah
University of Washington
Utah Valley University

What we'll cover…

Topics

- Concept Modelling within a Business Analysis framework
- Case study – using a Concept Model to discover Use Cases, User Stories, Business Services, and other requirements
- The essential elements of Concept Modelling
- Data model components – "ERA"
- Critical distinctions among Conceptual, Logical, and Physical Models
- Consistency in drawing the model
- The finer points

Introductions, if time/numbers permits:

- Name (how should I address you?)
- Role / job title, organisation, and location
- Is there a topic you are especially interested in?
- *Please try to keep your introduction to one minute or less*

"Analysis" gets criticised because of the extremes

Simplistic methods at one extreme: can do as much harm as good

List-form requirements, typically a Business Requirements Document – "*context-free requirements*"

ID#	Business Feature	Requirement Type	Business Unit(s) Affected	Potential Application(s) Impacted
BRQ025	files that are available for the selected day.		Readiness	
OMSPI-BRQ026	System shall include all outage status in the Transmission Outage report.	Core	Operation Readiness	WebOMS
OMSPI-BRQ027	System shall display consistency in the format of output data in the Transmission Outage report when using pipe-delimited feature as follows: For the same row of output data, all data elements in the same position in any field must correspond to each other. Example of existing Transmission Outage report where there are two inconsistencies in the output data format: 1. Report shows one Outage ID, three Substations, and four Equipment Names. 2. First listed Substation does not correspond to the first listed Equipment Name. Outage ID: 3042750 Substation: HUNTERS POINT PP P / MISSION X| LARKIN Y / POTRERO PP A (PGAE)| MISSION X Equipment Name: A-Y 2| BNK-2| P-X 1| P-X 2	Core	Operation Readiness	WebOMS
OMSPI-BRQ028	System shall allow the format of the Transmission Outage report published periodically automatically to support the following formats: 1. PDF 2. HTML 3. MS Word	Core	Operation Readiness	WebOMS
OMSPI-	System shall allow admin user to configure the number of days in the Transmission	Core	Operation	WebOMS

The goal lies in the middle ground:

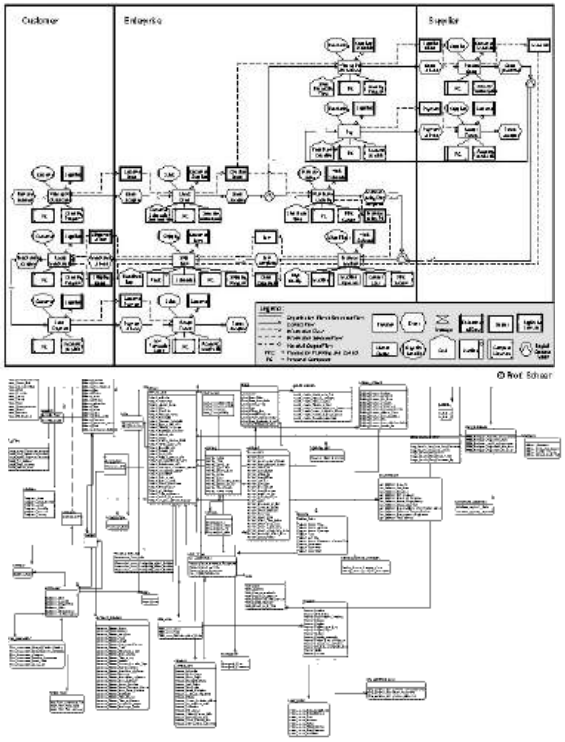
Client – *understandable, and therefore verifiable.*

Analyst – *doable, within Agile timeframes.*

Developer – *unambiguous, complete, actionable*

Overly complex methods at the other extreme: difficult for businesspeople to verify

Thinly-disguised, implementation-level design methods – *not* useful for discovering stakeholder needs



The problem with list-based requirements

Simplistic methods at one extreme:

An actual example, one in a list of 451 individual requirements for the "Provide Scientific Evidence" process at a national forensic science laboratory:

#49 -

The system shall provide a visual mechanism through which to view or amend the sequencing of items for a previously selected case or allocations thereof.

WHAAAT???!!!

List-based approaches to business analysis quickly break down – no way to ensure *completeness, accuracy, consistency, …*

So… what's wrong with this as a requirement? What does it NOT tell us?

What are they really trying to say?

Who? Senior Scientist

What? Schedule a Test (an Allocation) on a Sample from an Item

When? At Item Submission

How? By viewing upcoming workload

Why? To provide a completion date to the Customer (the Police)

Essentially, a Use Case or *User Story*:

As a Senior Scientist, I need the ability to view upcoming workload and schedule a Test on an Item, so I can provide a completion date to the Customer.

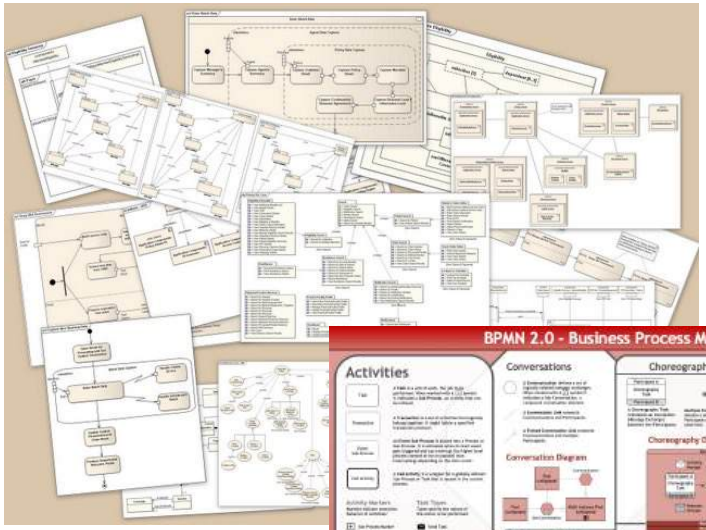
We will also use

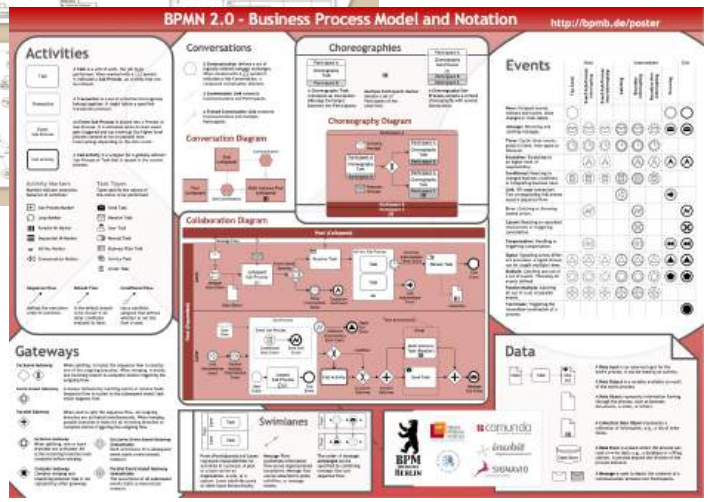
- *Business Process Models* to show where this fits in the end-to-end process
- *Concept Models* to show the required information

Complicated methods at the other extreme

"Can we use UML for Business Analysis?" As the late Michael Hammer said: “You could, but it will be like eating rice with a steak knife – messy, and someone’s going to get hurt.”

From the original UML specification: “The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the *artifacts of a software-intensive system*.”

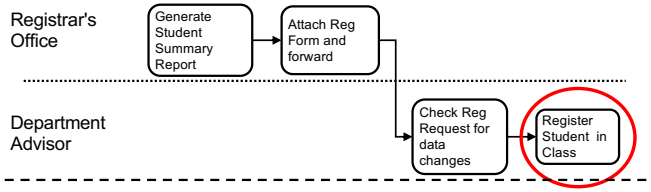
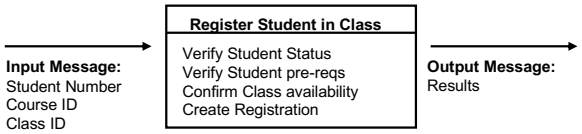
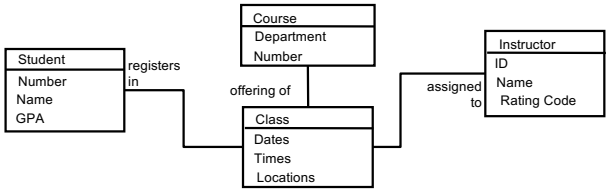




Same story for full BPMN (Business Process Model & Notation) – a platform-independent *visual programming language* for specifying automated workflows.

Best Practice – a business-friendly, model-based framework for Business Analysis

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)*	When advisor enters five characters of Last Name → Then System lists matching Students When advisor selects list item → Then System displays expanded Student view with needed Classes When advisor etc.	✓ ***Use Case*** – models how an actor interacts with a system to obtain (trigger) a service, typically to complete a step in a process
Application	*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*

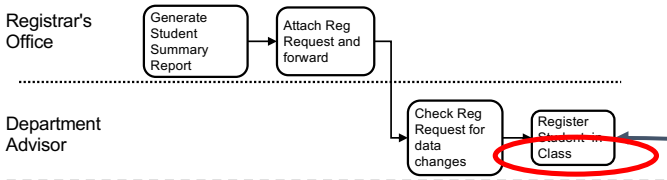
Key point! Everything relies on the Concept Model

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.

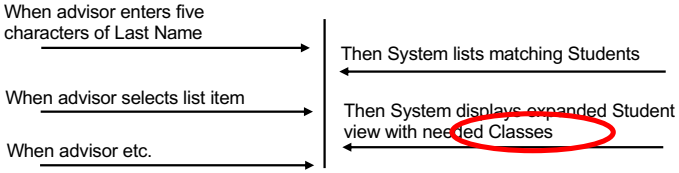
***All** use the language and constraints of the Concept Model (the "thing model") – the ultimate "what"*

Process — Business Process

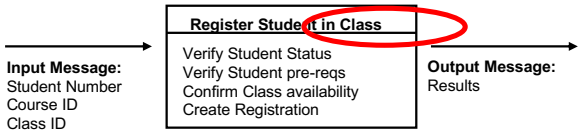


Use Cases/User Stories:
- Who (Actors) needs access to the Services, and how (Platform)?

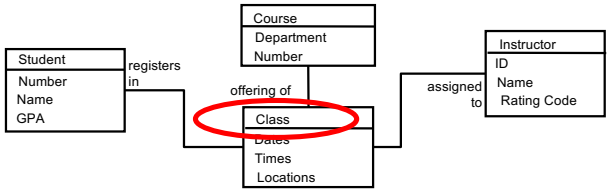
Application — Presentation Services (user interface)

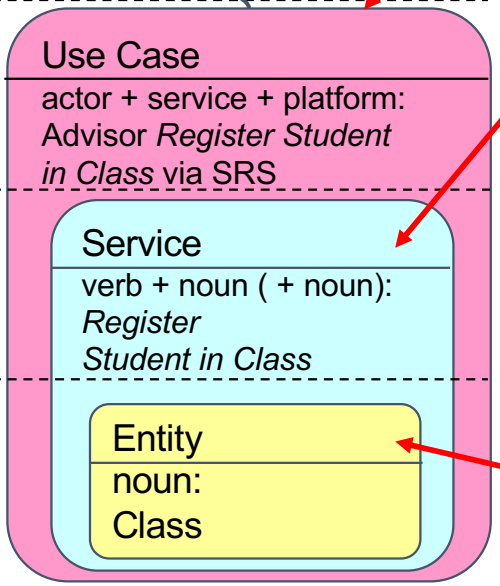


Application — Business Services (rules & logic)



Data — Data Mgmt. Services (databases)





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

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!

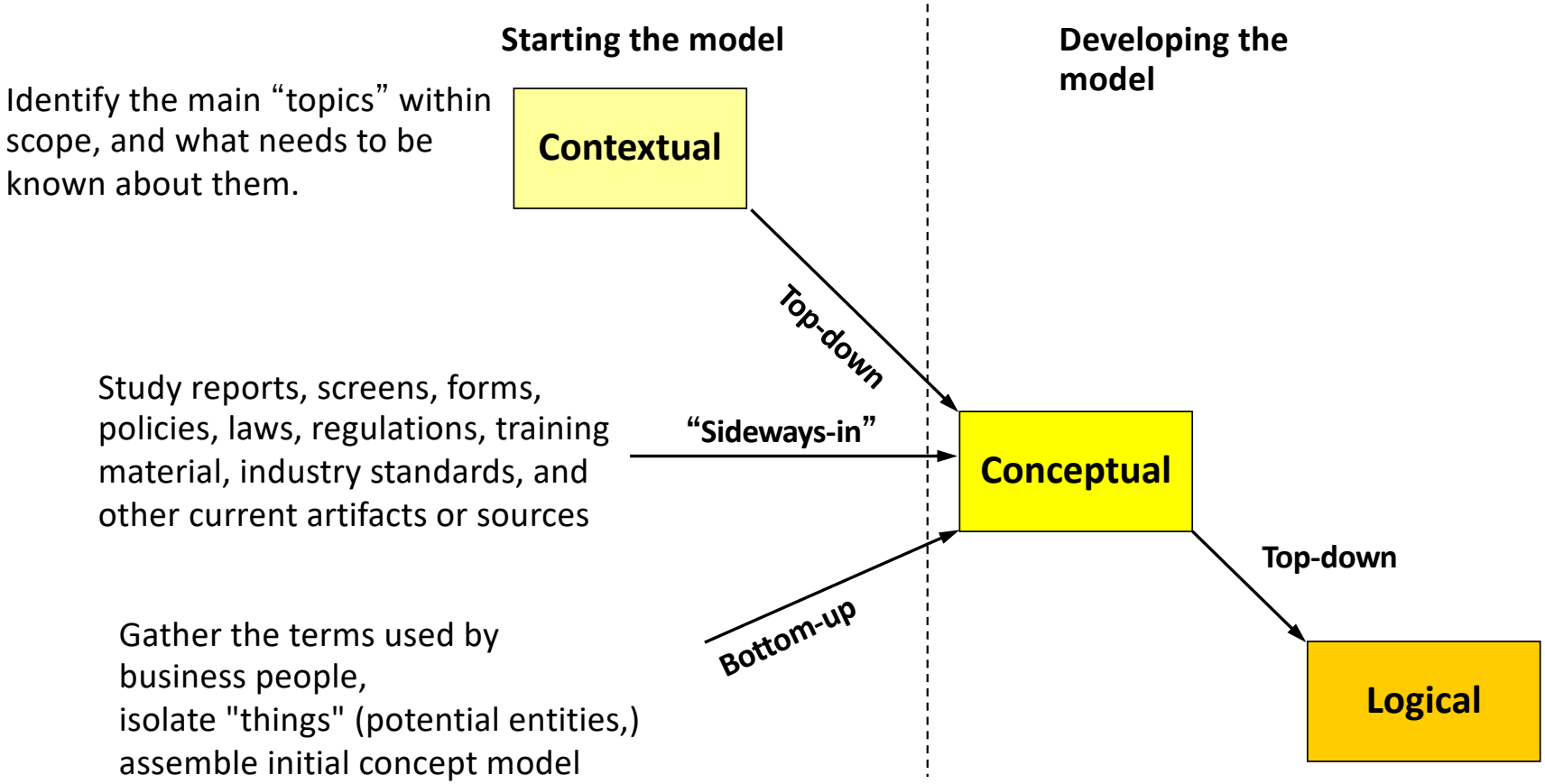
Case study – Concept Model, Services, Use Cases

- *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*

Different ways to get started



Starting a Concept 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" includes artifacts (forms, spreadsheets, reports…,) systems, mechanisms, job titles, organisational structures, work (processes, activities, steps…,) and anything else that isn't a basic thing or fact about a thing
   - As needed, introduce criteria to be a"thing" (an entity)

Always start with terminology (the "things")

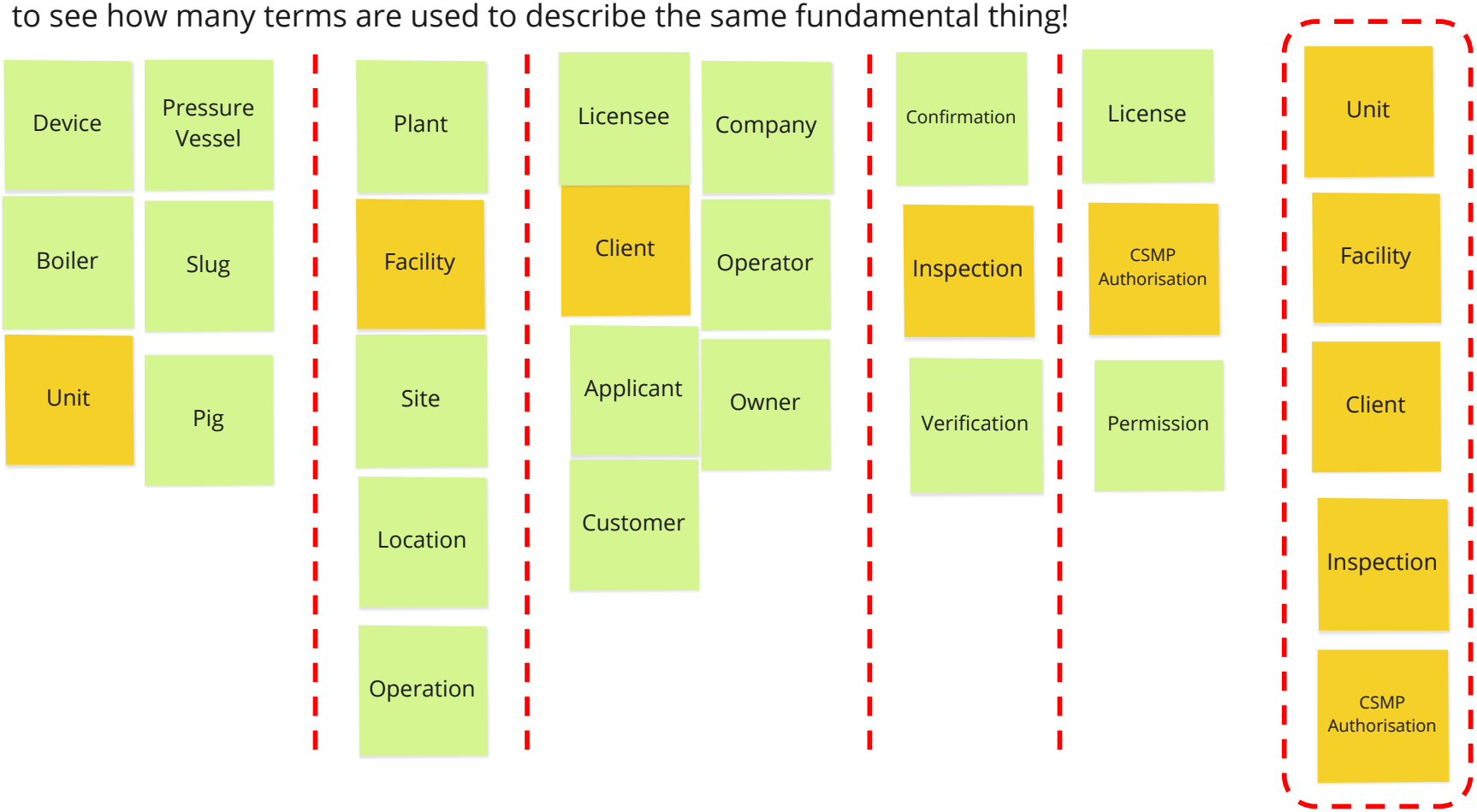
From one-on-one interviews with 10-12 key stakeholders we gathered ~200 terms related to CSMP (Client Safety Management Program) – "anything that went by a name." Here are 24 that met the criteria to be a "thing" – an entity in a Concept Model.

Tools like Miro and Lucidchart / Lucidspark are ideal virtual "Post-it work"

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

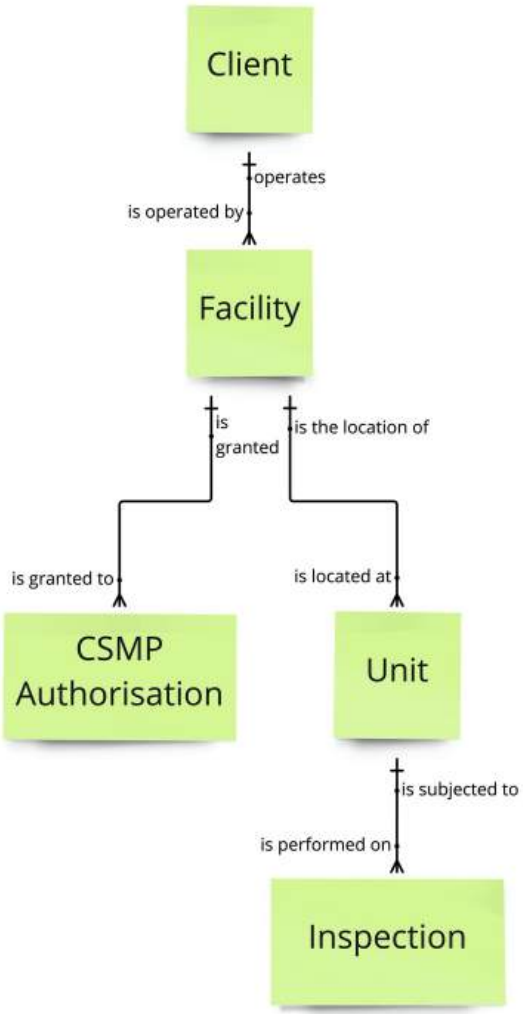
Review of a Miro example – Terminology Analysis

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

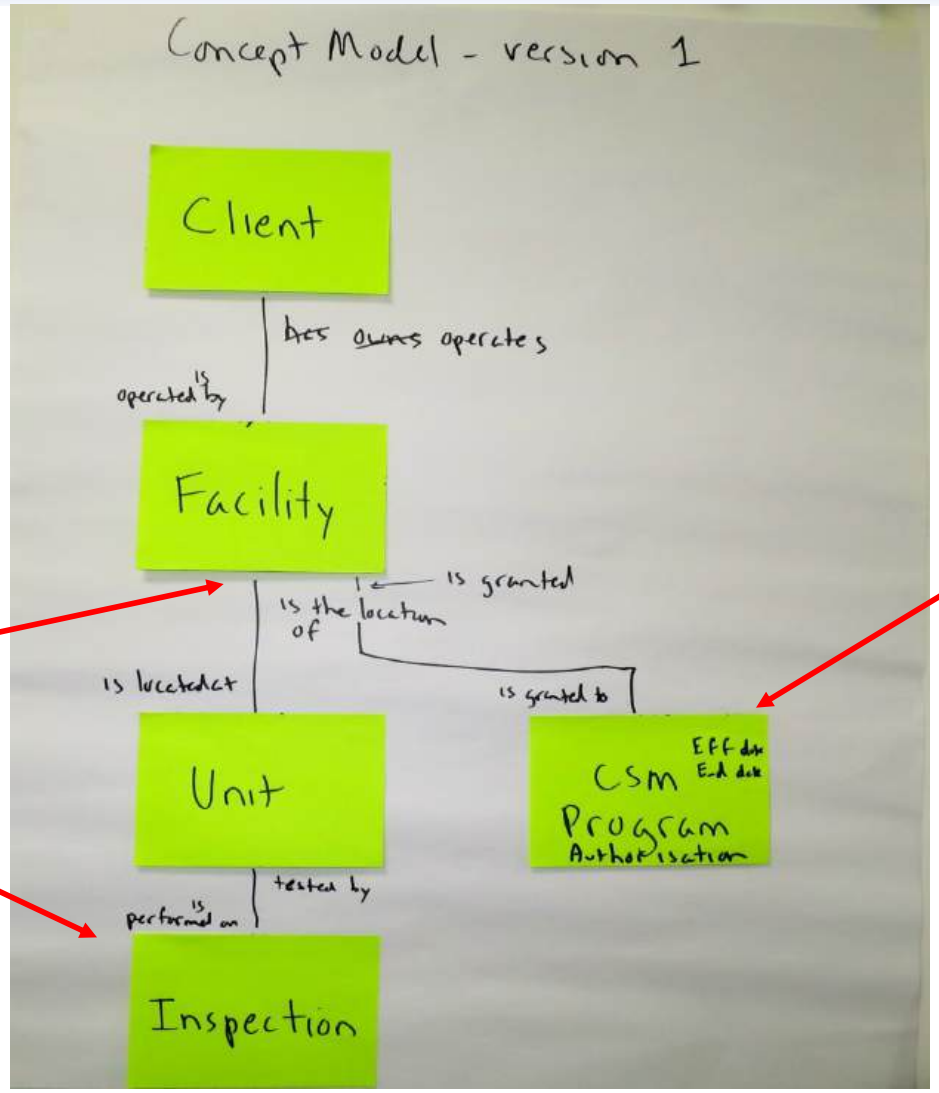
1. We arranged the entities / business objects by dependency
2. Then we drew relationship lines
3. Then we added a relationship name in each direction
4. Only then did we state (in words) the cardinality (1:1, 1:M, M:M) and then update the diagram with hash marks (┼) and crowsfeet (⅄)



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?
- the Facility?
- the Unit

What do we issue the Authorisation to?
- the Client?
- the Facility?
- some set of Units?

Concept Model Version 1; state Assertions and challenge them

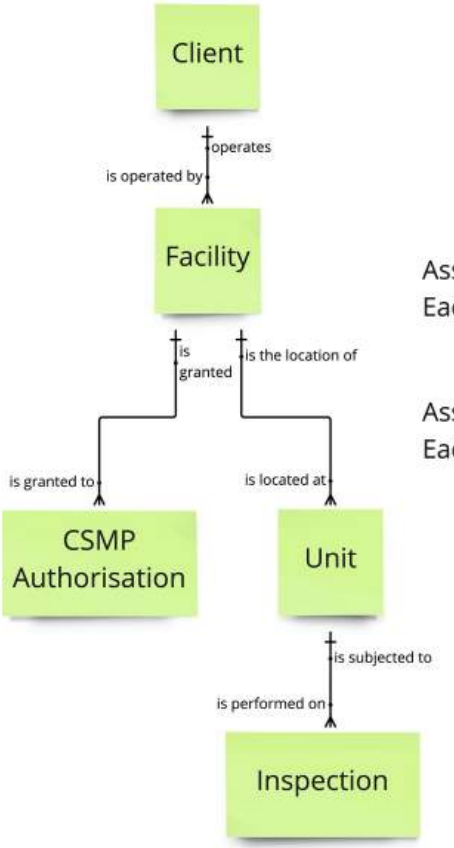
Now, state the relationships ***emphatically*** as Assertions. ***Each*** Client operates ***one or more*** Facilities! Then, ***challenge*** them!
Again, don't worry yet about ***optionality*** – whether the relationship ***must be*** or ***may be*** be present.
We only care now about the ***maximum*** – 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.

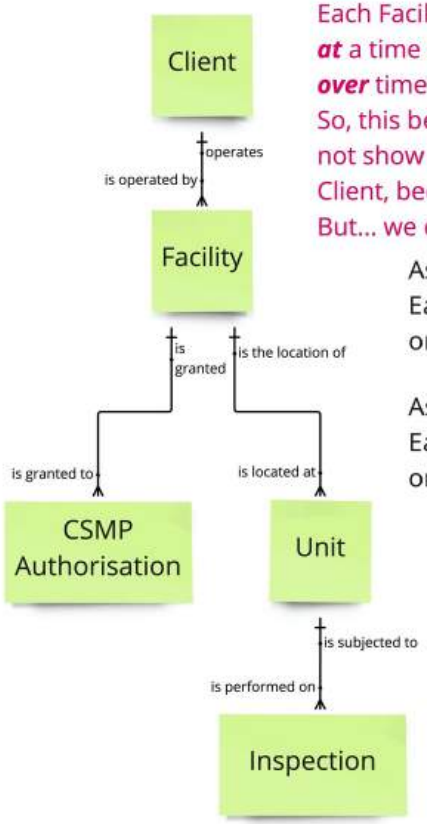
Assertion:
Each Client operates
one or more Facilities

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

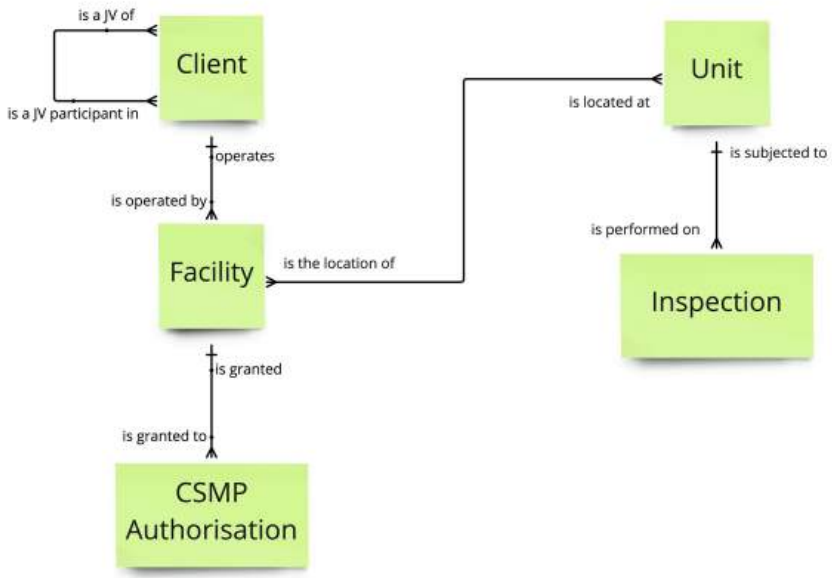
Assertion:
Each Unit is located at
one Facility

Assertion:
Each Unit is subjected to
one or more Inspections

Assertion:
Each Inspection is performed on
one Unit

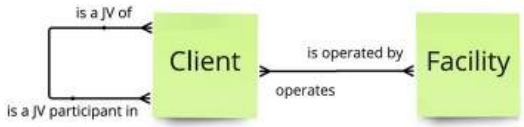
Concept Model Version 2; revised from challenging Assertions

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

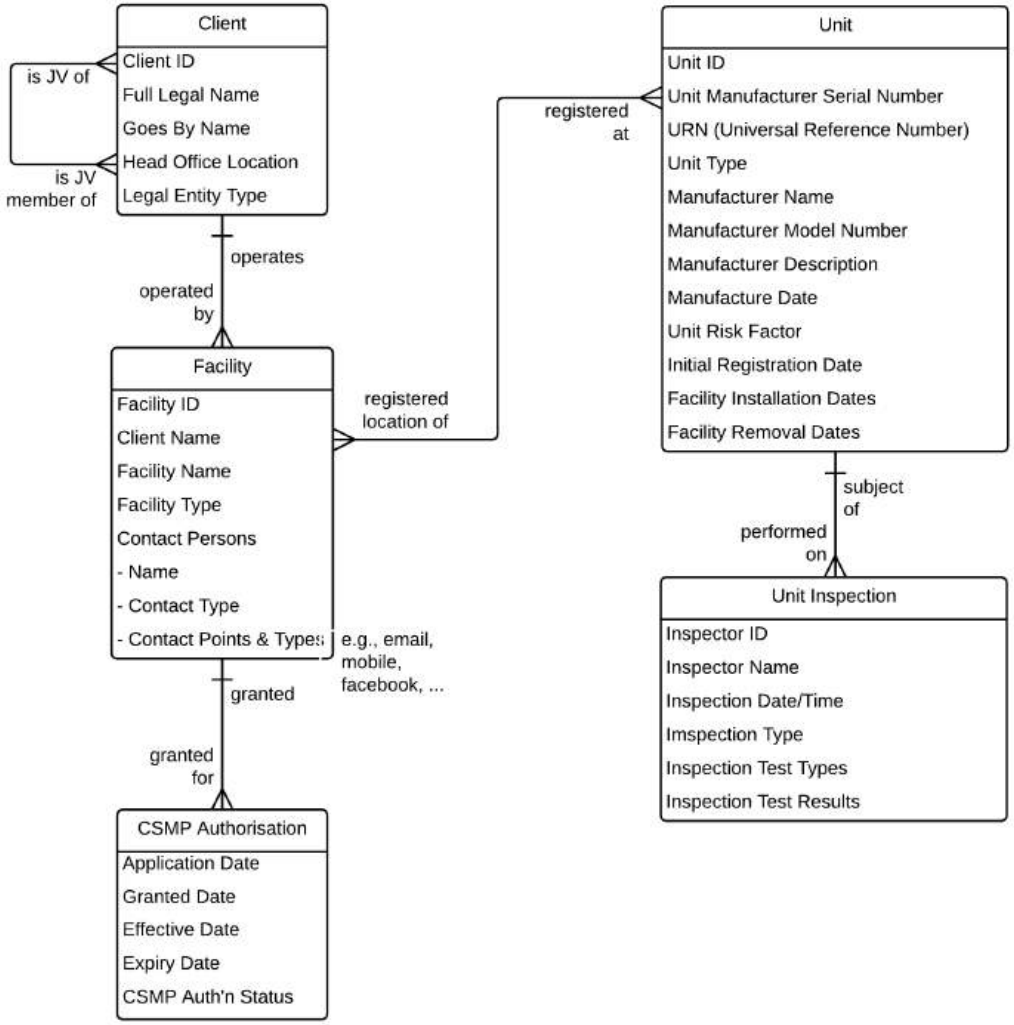


Note:
You don't always get what you *want* or what you think is the *right* thing in Concept Modelling. In this case the client (the Regulator) said they always wanted a Facility to be operated by ONE AND ONLY ONE Client.
If a Facility was operated by multiple Clients, they would require the Clients 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.

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

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

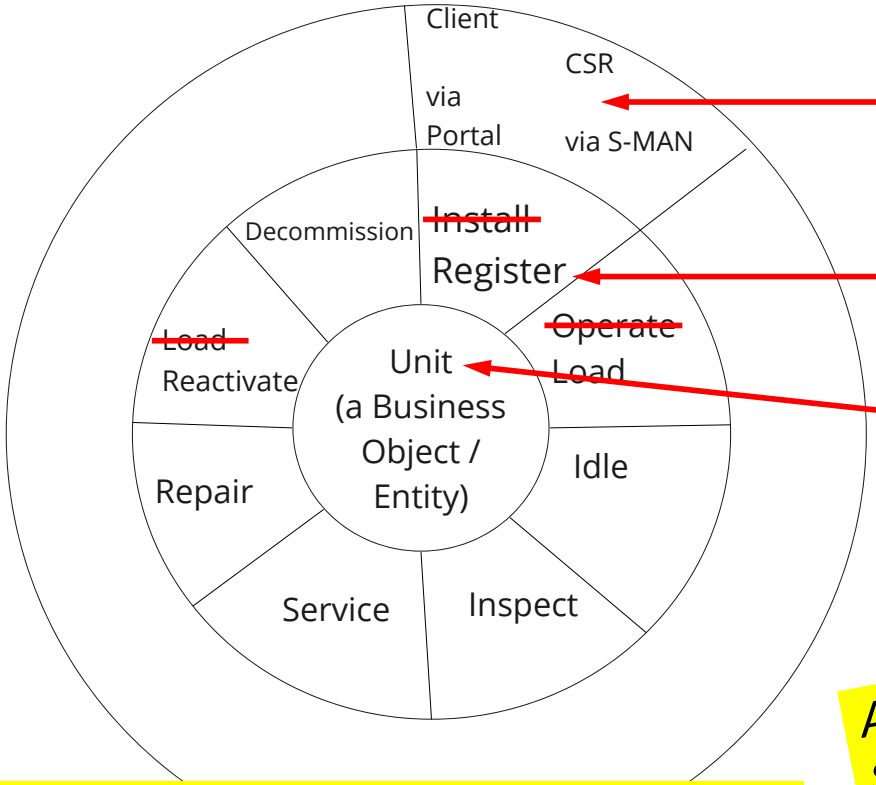
Model took ~90 minutes

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*

Reminder – what an analyst can do with a Concept Model

First, clarify language. (A platform)

Second, establish policies and rules.

And then, identify events or 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
- …

These are the *essential* capabilities

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

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

Develop high-level services then high-level use cases

Service: *Register Unit*

- Check for presence of properly formatted UR Number
- Determine if Unit UR Number is previously known
- If known, has it (a) moved (b) changed ownership (c) …?

Use Case: *CSR Registers Unit via S-MAN*

- CSR will select "spreadsheet" of all Units covered by CSMP app
- S-MAN will highlight all that can proceed immediately
- For each category of Units requiring intervention…

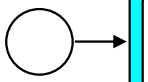
Use Cases and User Stories begin in different formats but soon become the same

Note:

Services and Use Cases at the "upper conceptual" level to provide vendor with key elements of requirements and avoid the usual bulleted list requirements document.

Clarify scope of the new process and identify participants

Trigger:
Client submits request to enter a CSMP

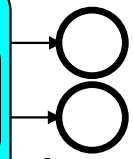


Grant Client Safety Management Program

- 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.

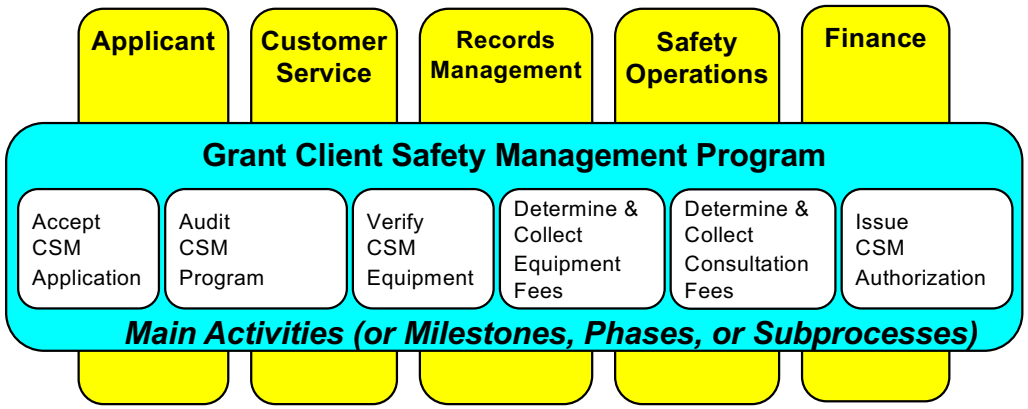


Agency Result:
Revenue collected. New participant in CSMP; confirmation that regulations are satisfied

Cases:

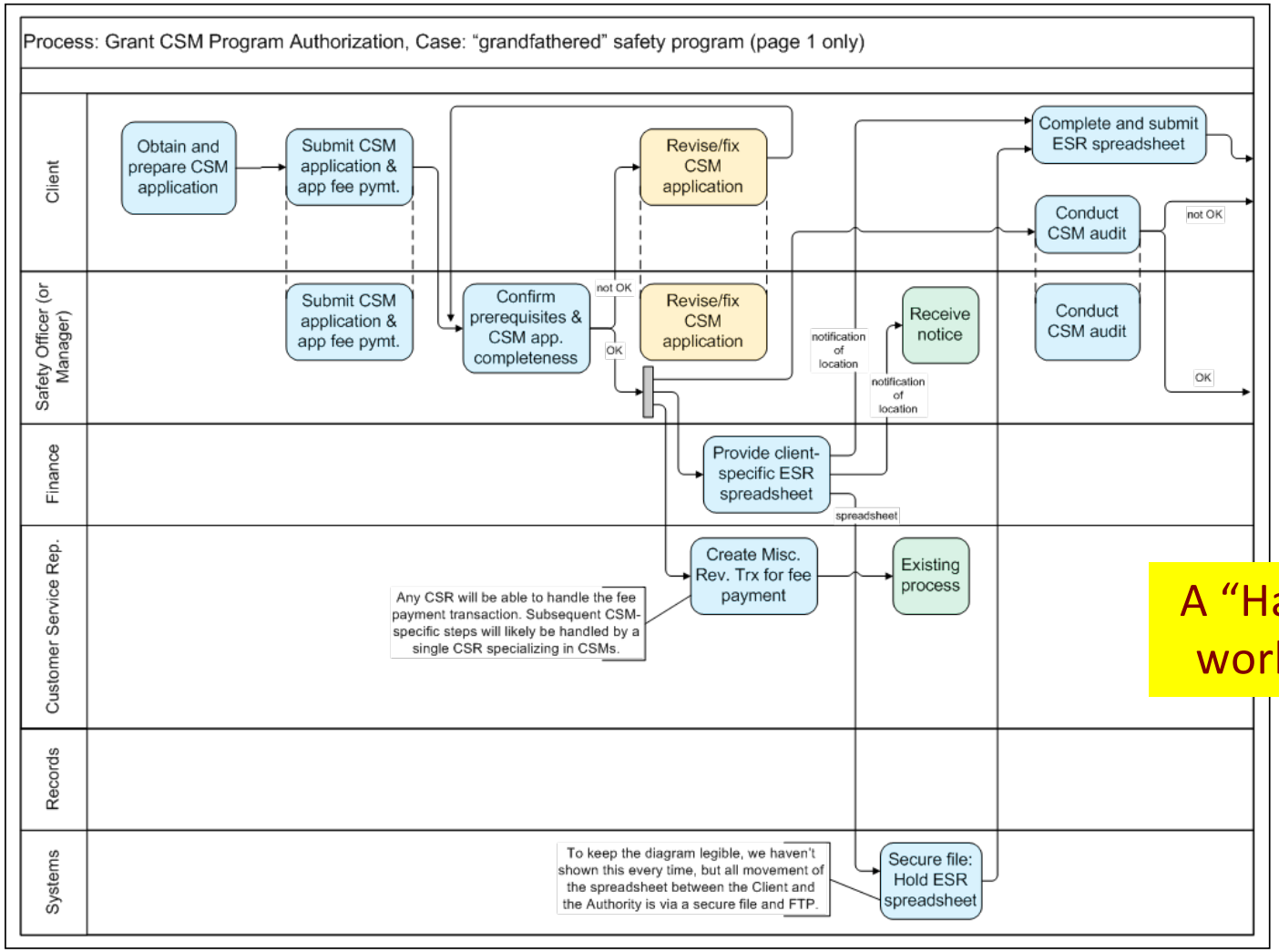
- New
- Legacied
- Ownership Change

Process Scope Model – pure "what"…



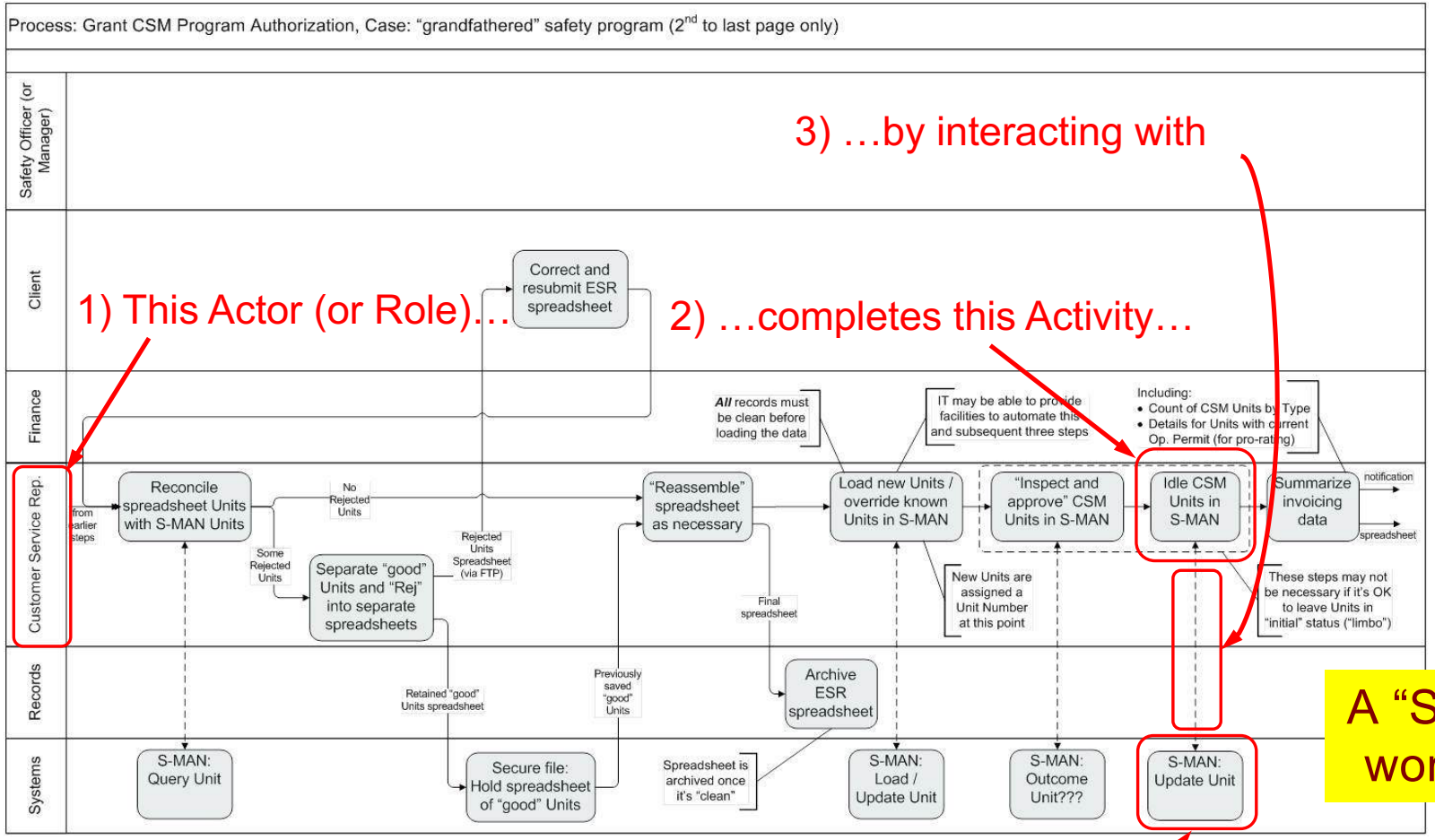
Process Summary Chart – simplified "what," plus "who"

The initial, business-friendly workflow model



A "Handoff Level" workflow model

Eventually, detail showing where use cases & services fit



1) This Actor (or Role)…

2) …completes this Activity…

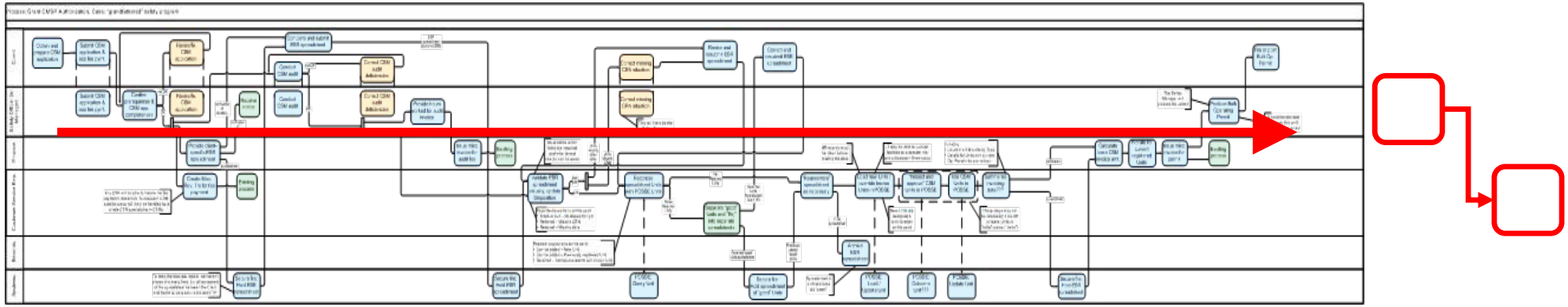
3) …by interacting with

4) … this Service offered by a System (which collectively is a Use Case)

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!***

Progressive detail for all techniques

Clariteq framework for analysis and architecture

Project Charter: primarily "Scope" level - may evolve

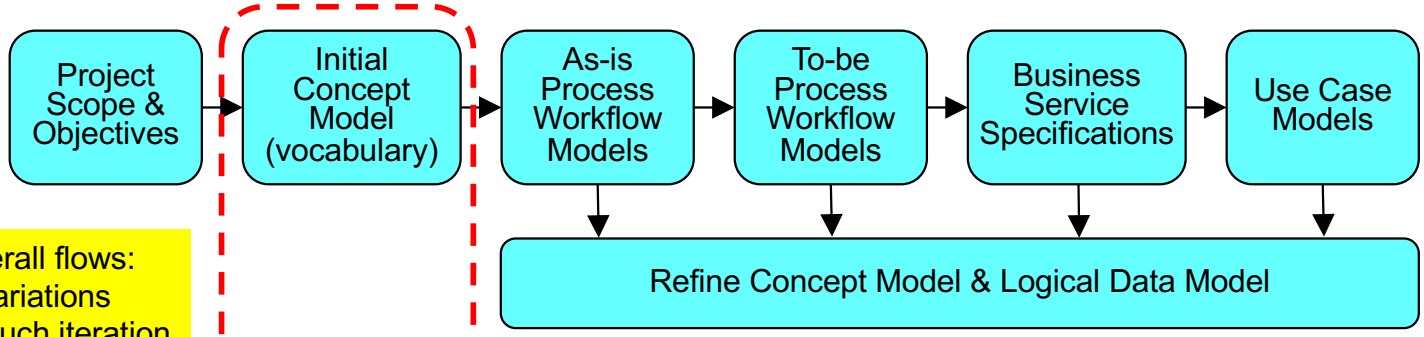
		Scope	Concept	Detail	
Goals	Business Objectives				
Process	Business Process	Process Landscape showing target and related processes, Process Scope Model, initial assessment and goals.	As-is (and later, to-be) Workflow Models for the process' main variations (cases) to the Handoff level.	As-is Workflow Models to the appropriate detail, and to the Service level for to-be. Optionally, document procedures for manual to-be steps.	Process Modelling
Application	Presentation Services	List of the main Use Cases in the form: Actor + Service + (optionally) Technology / Platform (named only.)	Initial Use Case description (goal, stakeholder interests, use case abstract) for each Use Case. May include initial dialogs.	Use Case dialogs in "when-then" format, annotated, and including alternate sequences. Optionally, Use Case Scenarios.	Use Cases
Application	Business Services	List of main Business Services (named only.)	Initial Service description - result, main actions, cross-referenced to Concept Model	Each service fully documented, including input/output messages, validation, business rules, and data updates to the attribute level.	Service Specification
Data	Data Management Services	Contextual Model (optional) and a glossary defining the main entities and other important terms.	Concept Model (Business Object Model, Conceptual Data Model) with main entities, relationships, attributes, and rules.	Fully normalised Logical Data Model with all attributes fully defined and documented.	Concept Modelling
		Plan	*Understand*	*Specify*	

The "Agile Zone"

Techniques and methodologies

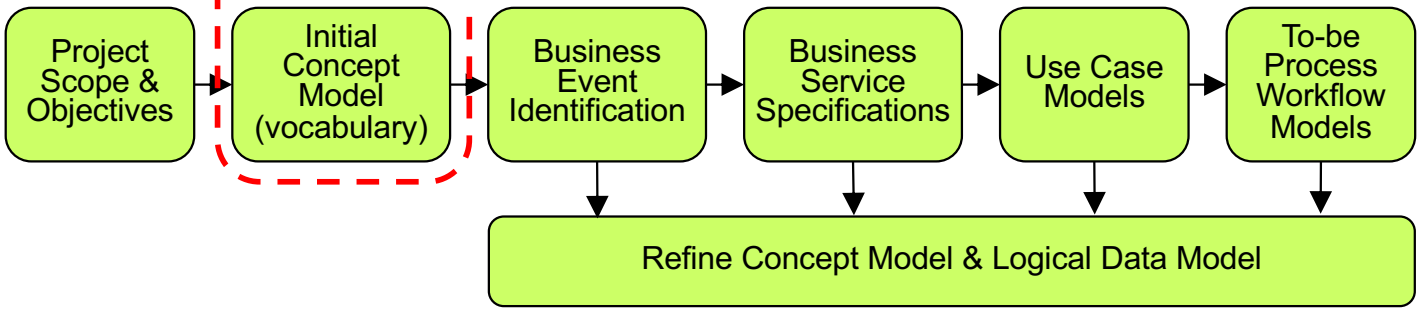
- The same techniques are used in different sequences, with different emphasis, in different methodologies.
- Concept Modelling to clarify language is a great starting point.

Larger project: process-oriented / "outside-in" –



These are typical overall flows:
- there are many variations
- there is always much iteration

Smaller project: service or use case-oriented / "inside-out" –



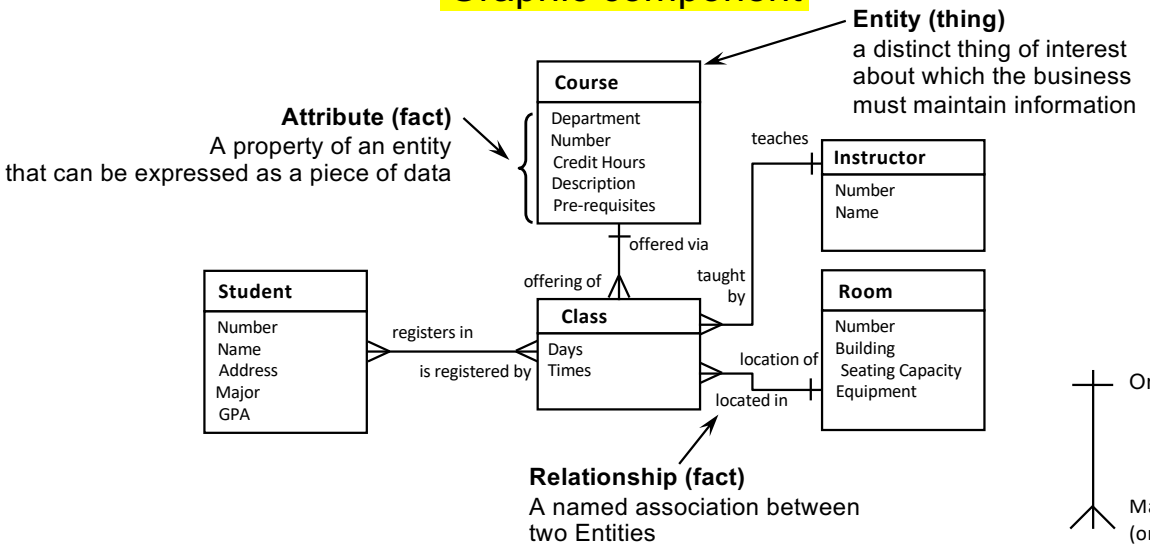
What is a Concept Model / Business Object Model / Domain Model…?

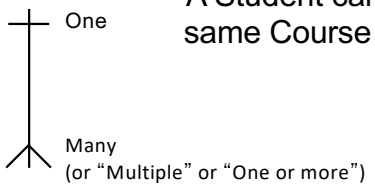
- A *description of a business* in terms of
  - **things** it needs to maintain records of – *Entities*
  - **facts about those things** – *Relationships & Attributes*
  - **policies & rules** *governing those things and facts*
- Models a view of the **real world**, not a technical design (therefore, 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

Graphic component





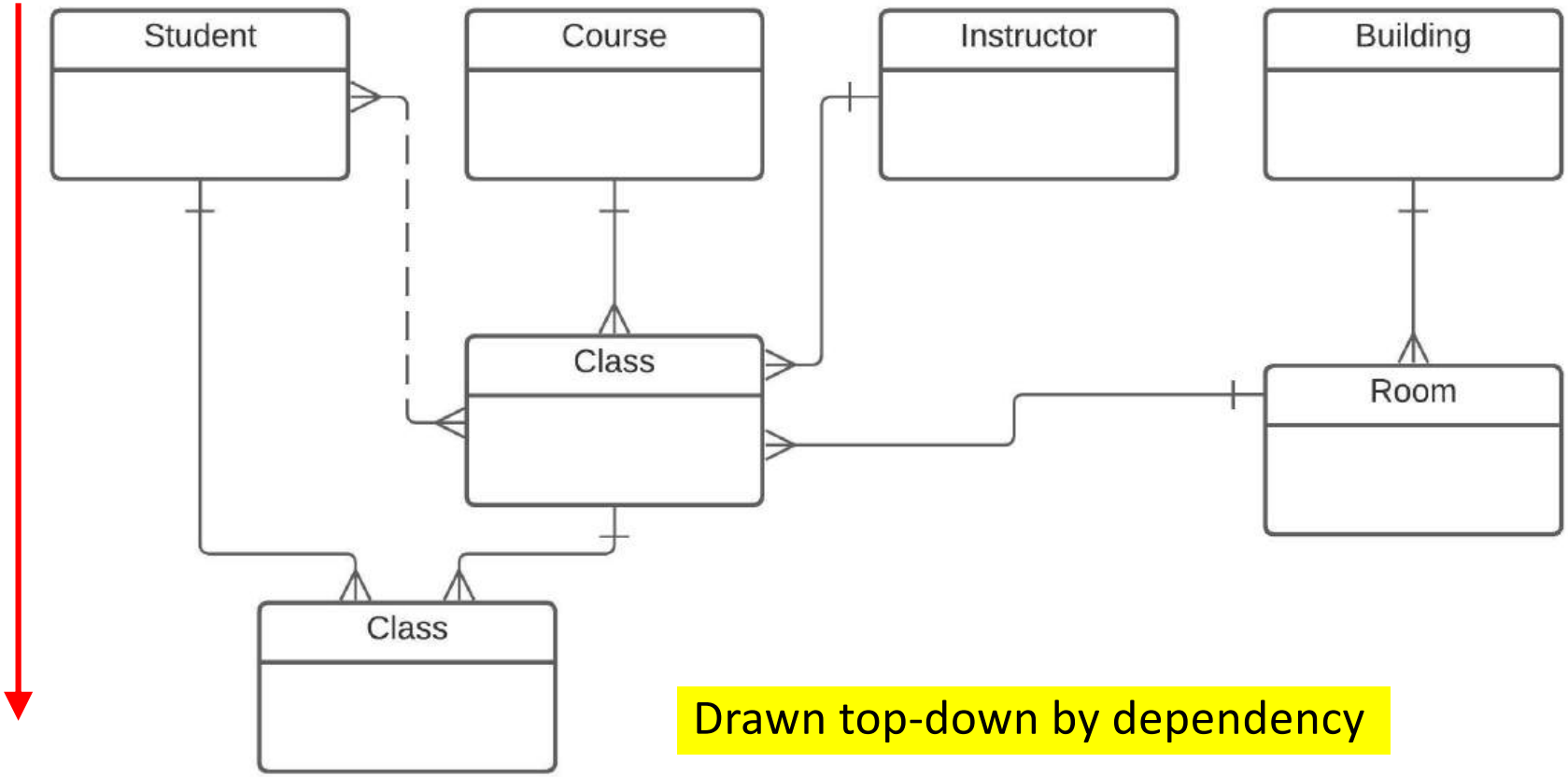
Student definition:
A Student is any person who has been admitted to the University, has accepted, and has enrolled in a course within a designated time. Faculty and staff members may also be Students

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

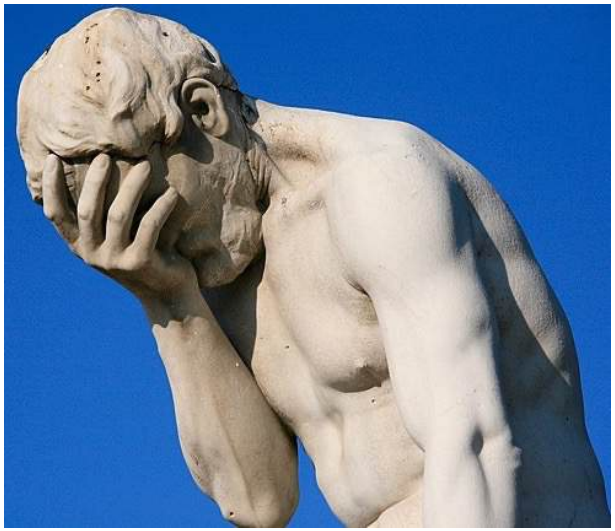
A better looking version of the model on the previous slide

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 and IoT ("It's schema-less!")
- 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, …?)

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!*"
- At Big Data London 2024 Concept Modelling was the hot topic

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…

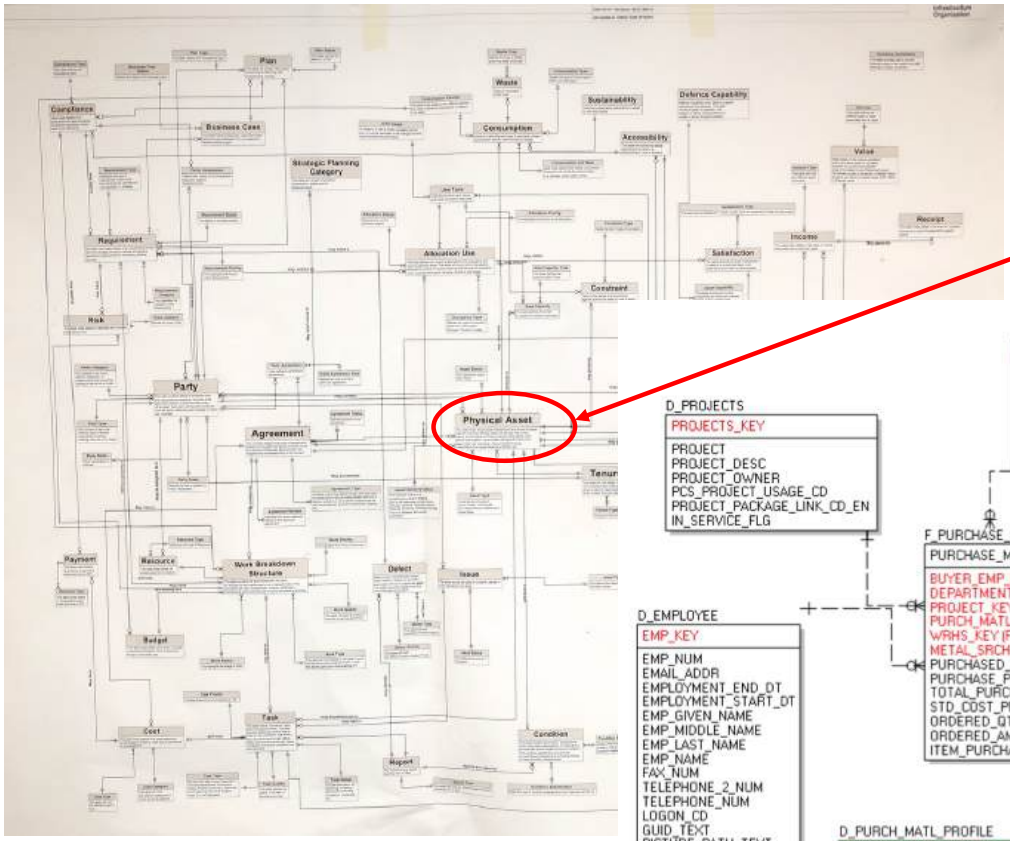
"Help – everyone hates our data model!"

2 – "Data people" can make "data" far too difficult

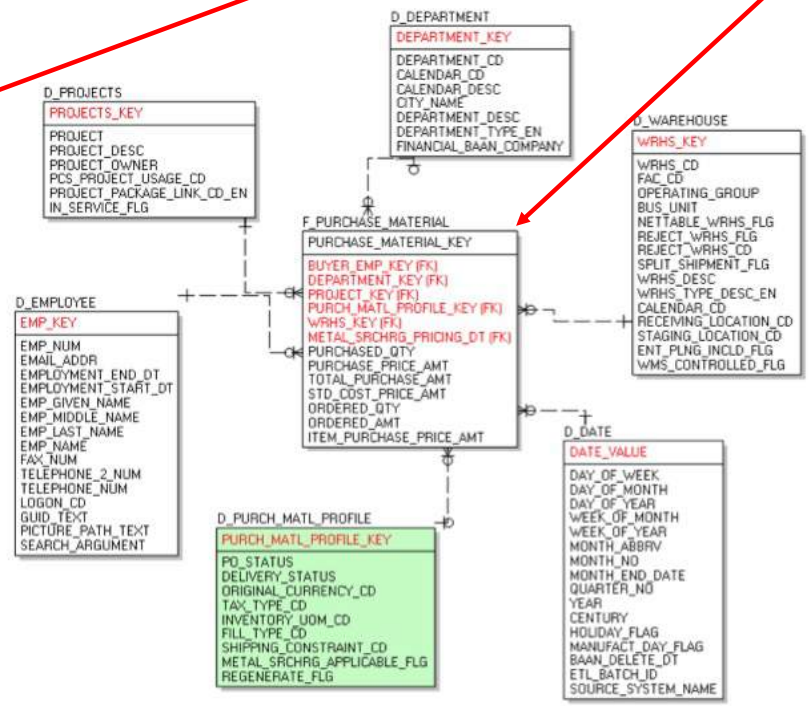
2 – Terrible diagramming…
A common error – *"the most important entity should go in the centre of the diagram."*

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



3 – No clarity on different model levels 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

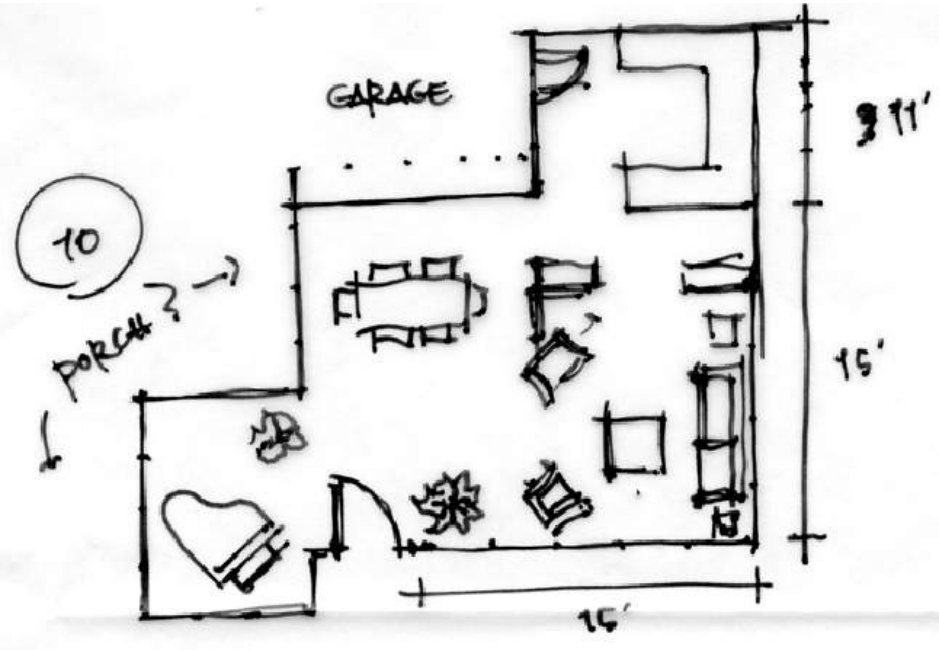
Some important differences

✓ Main ("recognisable") entities only - a singular noun used daily	✓ All granular entities – many too detailed to come up daily
✓ Main attributes only, many are non-atomic	✓ All attributes included, all are atomic
✓ M:M relationships	✓ All M:M resolved
✓ Doesn't show keys	✓ Shows primary & foreign keys
✓ Not normalised	✓ Fully normalised
✓ A "one-pager"	✓ Five times as many entities

My most plagiarised slide!
More details later.

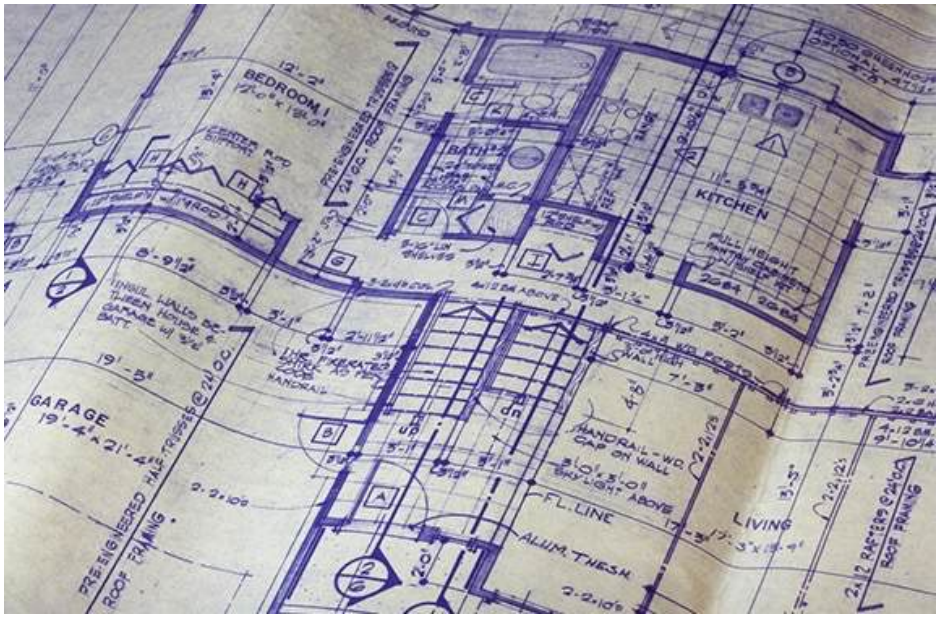
An analogy for Conceptual and Logical models

A Concept Model is like a sketch of a floor plan



Easy to understand, and provides enough detail for the homeowner to decide if the layout will work for them.

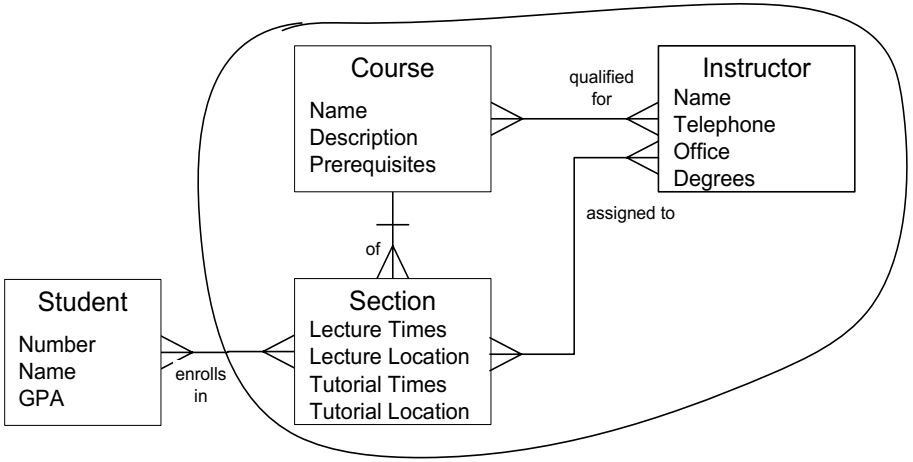
A Logical Data Model is like construction blueprints



Requires specialised skills to interpret and provides enough detail (along with other design views and callouts) for the builder to construct the building.

Concept model

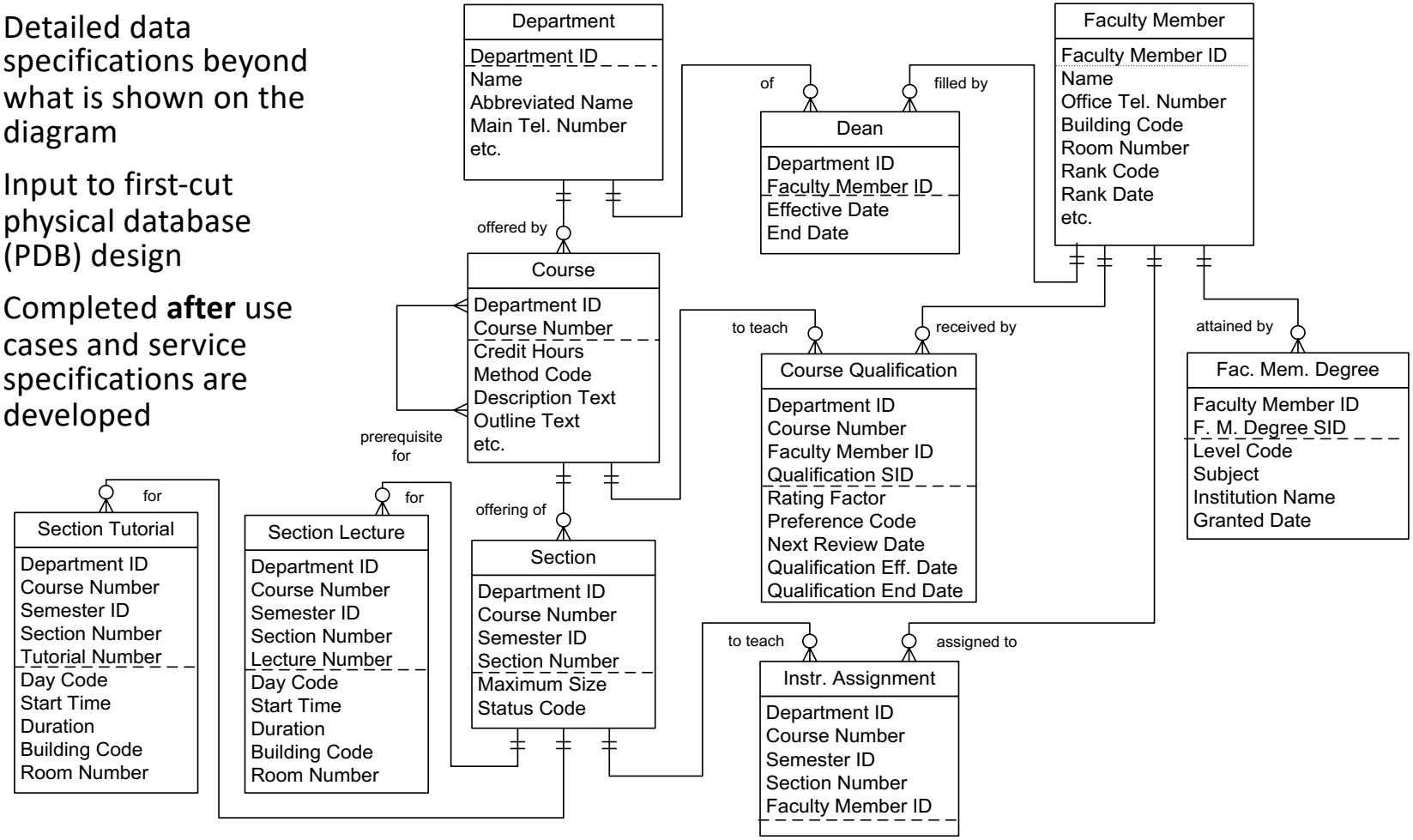
- Shows main or core entities, relationships, attributes, and rules
- Gets the “concept” across
- Great for communication, but not for detailed specifications
- Best done ***before*** any significant process modelling or application requirements (use cases and service specifications)



Let's see what happens when we take these three entities to the "Logical" level...

Logical data model

- Detailed data specifications beyond what is shown on the diagram
- Input to first-cut physical database (PDB) design
- Completed **after** use cases and service specifications are developed



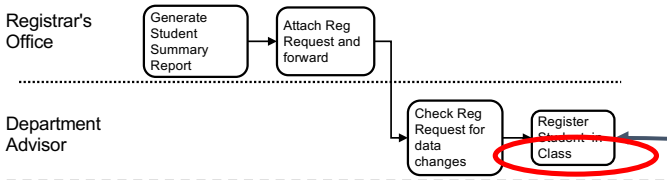
4 – Not clarifying that all other analysis relies on the Concept Model

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

Goals — Business Objectives

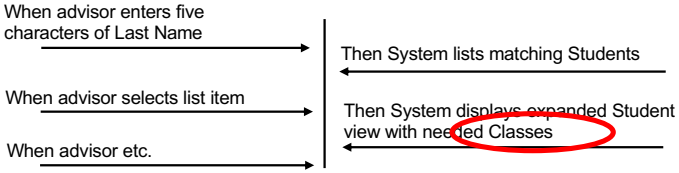
The university is initiating the "Strategic Enrollment" program to raise Student graduation rates in part by ensuring Classes are available for Student registration when needed.

2 Process — Business Process

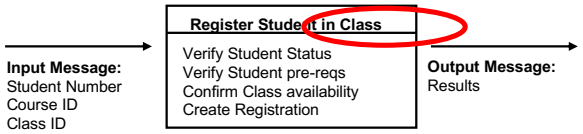


Application

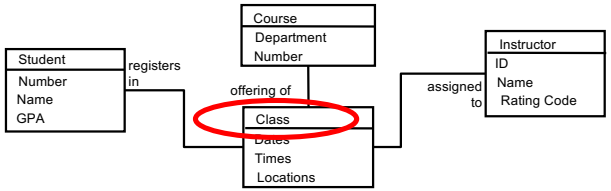
4 — Presentation Services (user interface)

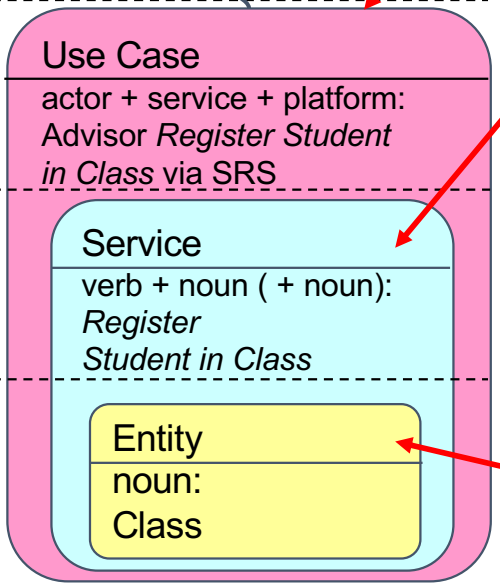


3 — Business Services (rules & logic)



1 Data — Data Mgmt. Services (databases)





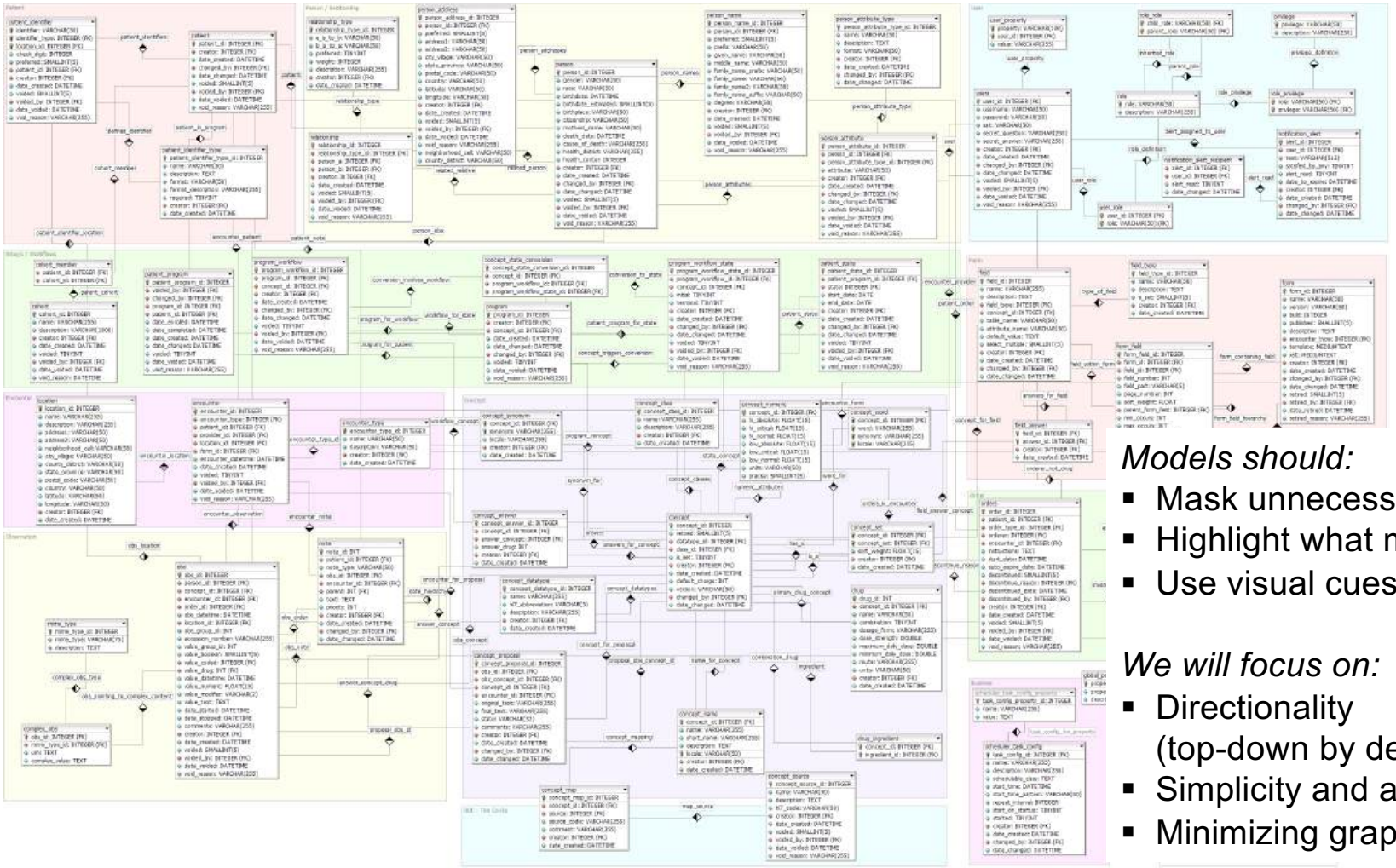
Use Cases/User Stories:
- Who (Actors) needs access to the Services, and how (Platform)?

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.*

My usual sequence

Entity-Relationship 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 –

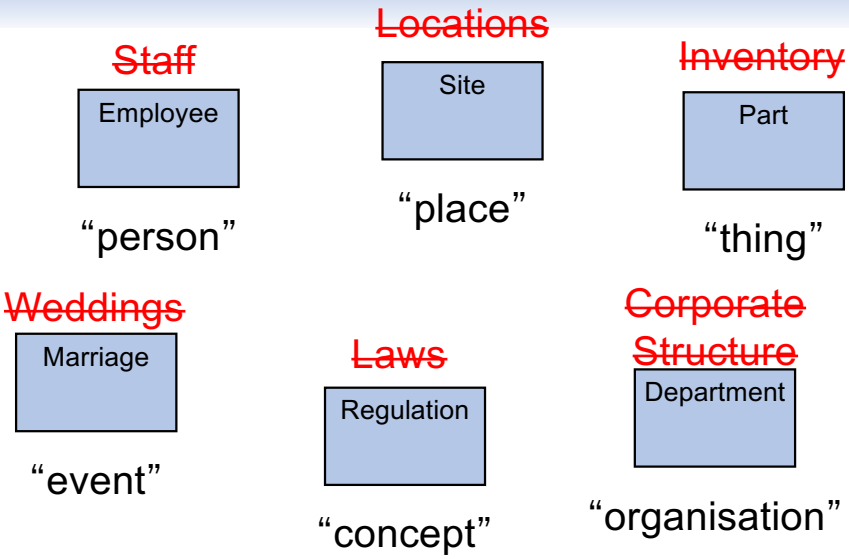
- *singular noun* – we can talk about *one of them* ("Employee," not "Staff")
- *multiple* instances
- must *need to* and be *able to* keep track of *each* instance
- has *facts* (attributes & relationships) that must be recorded
- makes sense in a *"verb-noun"* pair
- *NOT* an *artifact* like a spreadsheet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

Must be:

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

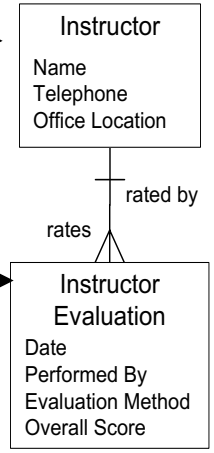
Two basic types:

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



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

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

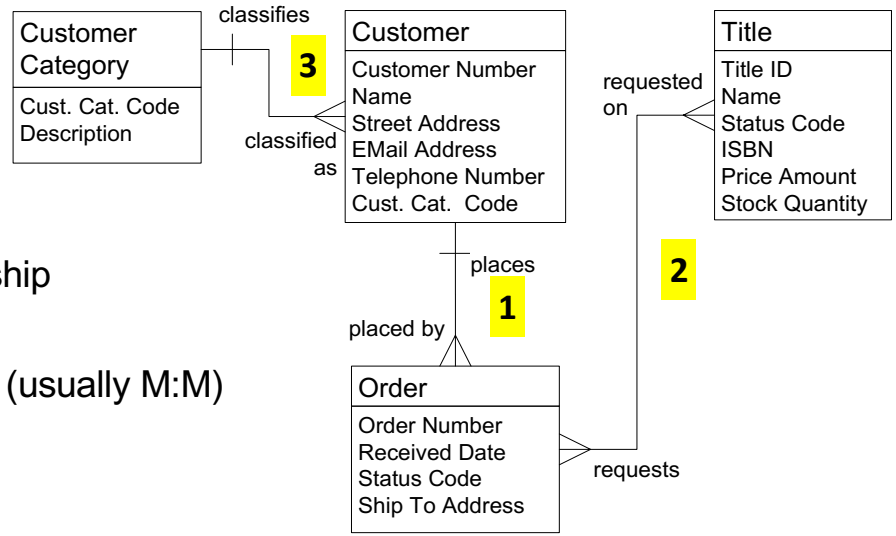


The basics – ERA – *Relationships*

An association between Entities that the business must keep track of

Named in both directions

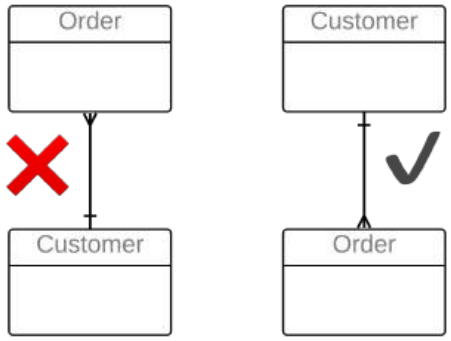
- verb-based phrase
- the name tells us *how* they are related,



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

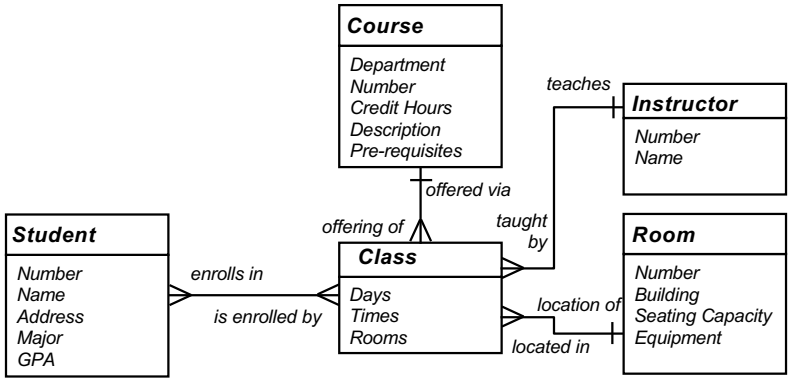


Relationships have rules

- 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)
- *always* state relationships as *assertions* and *challenge* them!

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…



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

Each Class is taught by one Instructor…

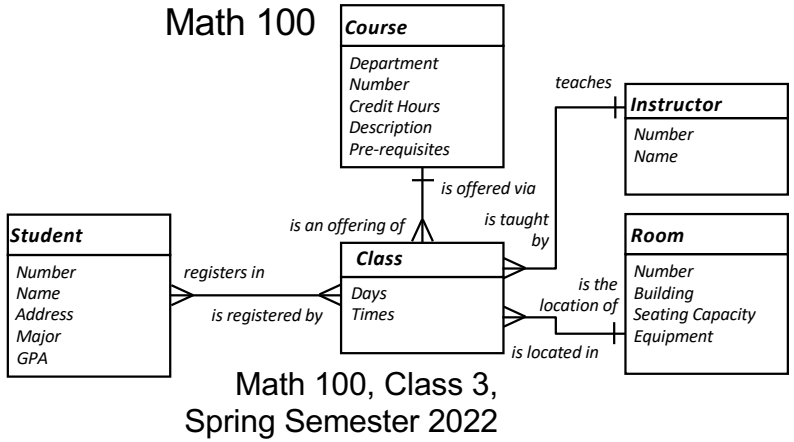
1. Student-Class
2. Course-Class
3. Instructor-Class
4. Room-Class

Which ones might be *incorrect?*

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.)

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

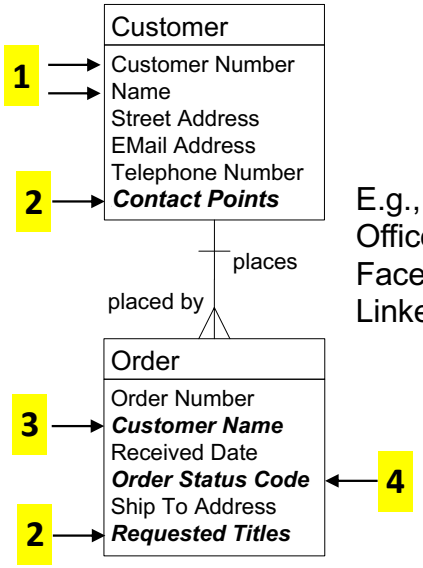
Have properties that we address during the transition from Concept Model to Logical Data Model

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, …

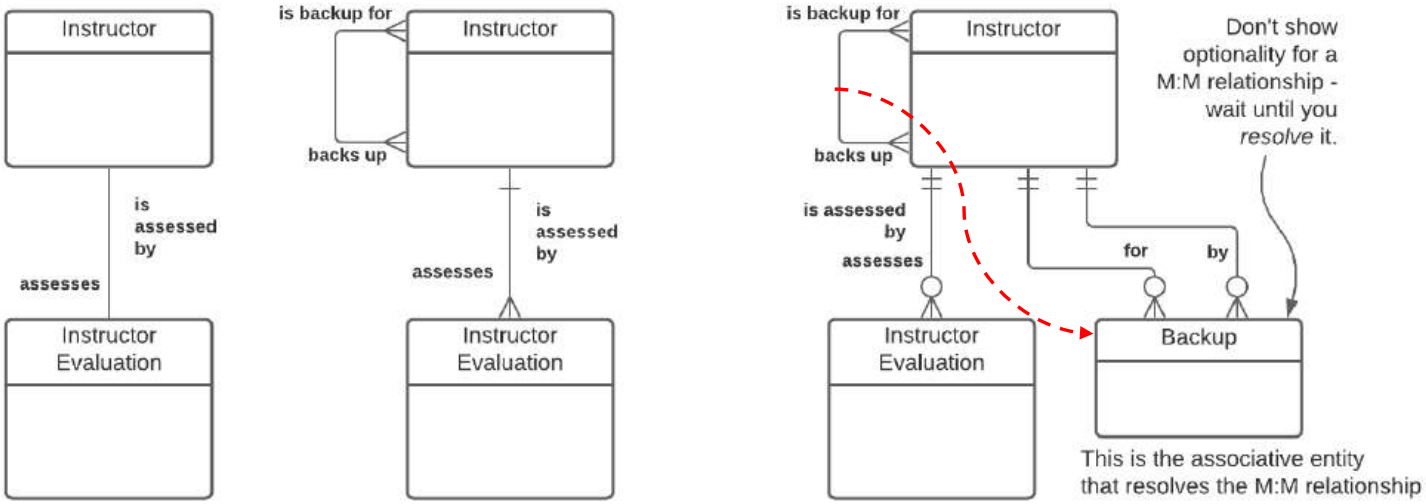
The details – Contextual, Conceptual, & Logical models

1 Contextual (Scope – Planner's View)	2 Conceptual (Overview – Owner's View)	3 Logical (Detail – Designer's View)
Agree on context or "big picture"	**Agree on basic concepts and rules**	**Full detail for physical design**
• 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	• 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	• 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

My most plagiarised slide ever!

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.

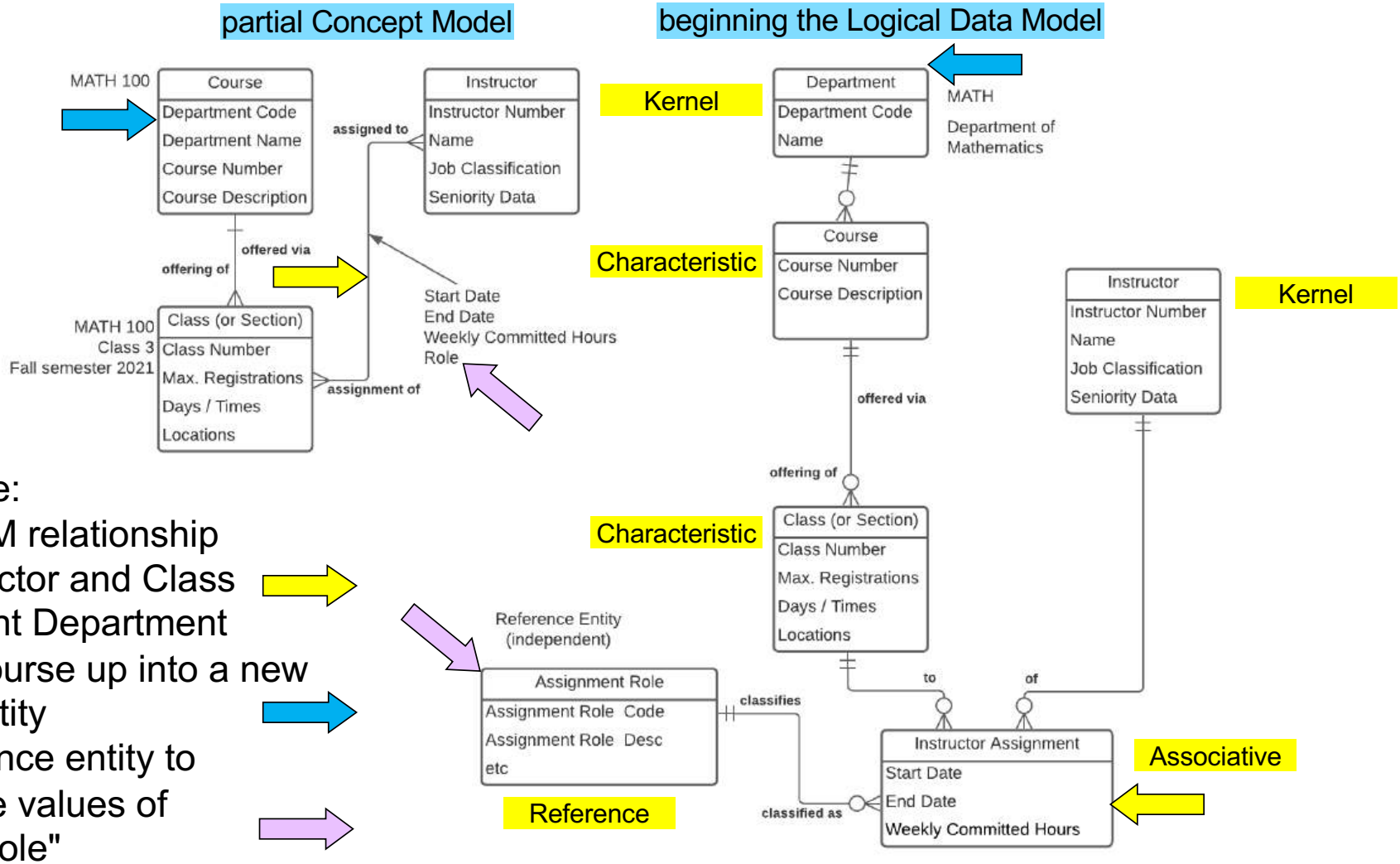
There is a 1:M relationship from parent to child, *optional* for the parent and *mandatory* for the child. (The parent *may* have a child, the child *must* have a parent.) This is by far the most common relationship in a logical model.

For reference – from Concept Model 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 quick example – from Concept Model to Logical Data Model



In this example we:

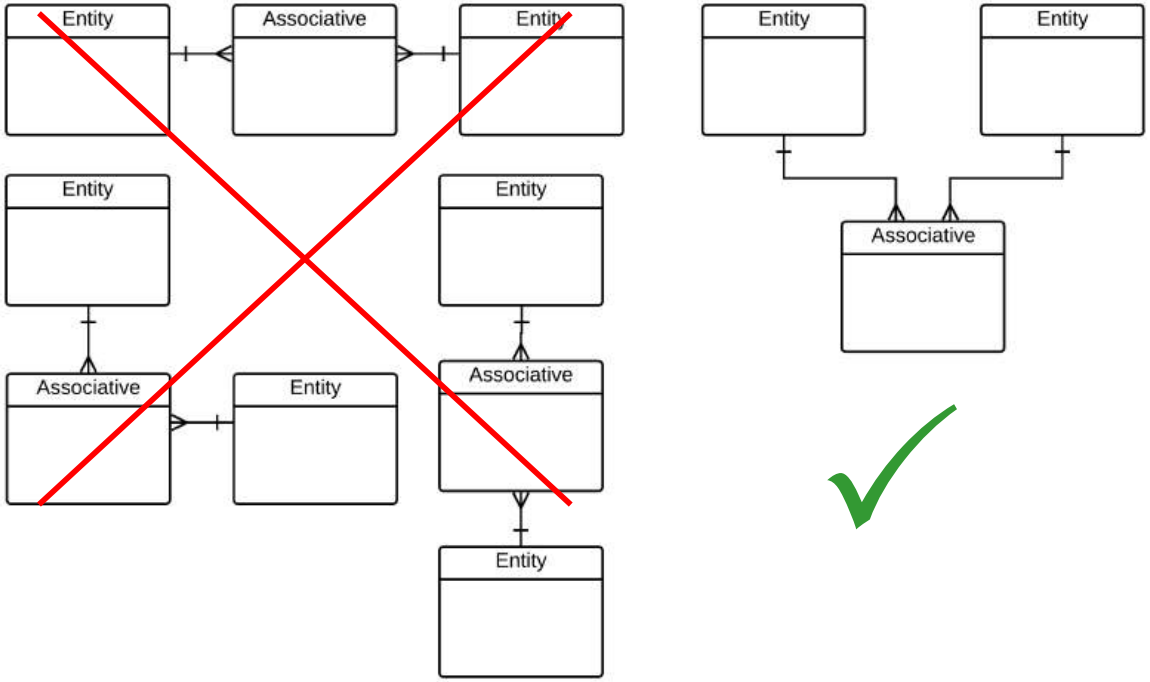
- resolve the M:M relationship between Instructor and Class
- move redundant Department attributes in Course up into a new Department entity
- create a reference entity to standardise the values of "Assignment Role"

Consistency is a virtue

People pick up data modelling without training if you…

- treat it as a natural way to describe a business, not a new technique being imposed on them
- draw the same kinds of things the same way every time

E.g., when drawing an associative entity…

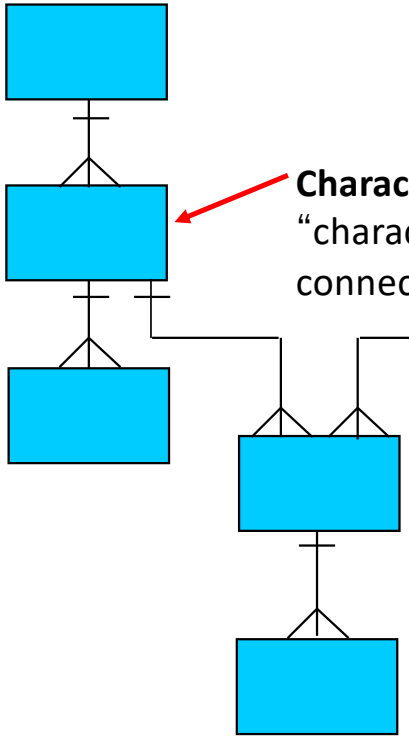


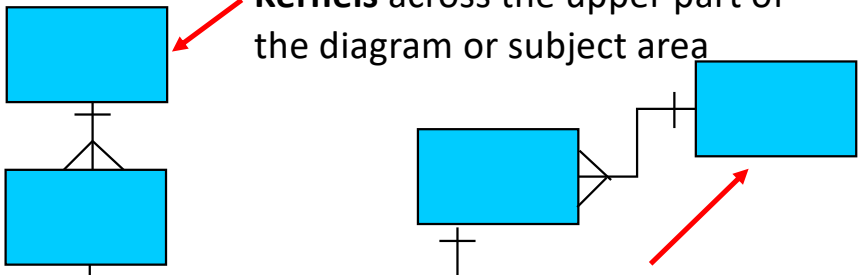
Graphic guidelines – the "no dead crows" principle

! *Key point*

Entity type is obvious from:

- Placement
- Relationship connections





Kernels across the upper part of the diagram or subject area

Characteristics below; "characterising" relationship connects at top edge

Reference or Types beside or diagonally above; "classifying" relationship connects at side

Associatives between & below; "associating" relationships connect at top edge

Various layouts can work:

- **top-down** — **<u>best for mere mortals</u>**
- **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**

Identifying Entities – three common errors

1. Treating an "artifact" (a spreadsheet, report, 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

254 British Properties

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

Never 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			

Answers – good Entity or not?

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

- ✗ Transcript — a report
- ✓ Student
- ✓ Building
- ✗ Student Directory — a report
- ✓ Faculty Member
- ✗ Instructor History — a list, of what?, "history" is not singular
- ✓ Department
- ✓ Course
- ✗ Organisation Chart — a visual report
- ✗ Prerequisite List — a list
- ✓ Payment
- ✗ Student Body — not singular
- ✗ Class Roster — a report
- ✓ Scholarship
- ✓ Faculty
- ✗ Assistant Dean — a Job Title
- ✗ Admission Date — an attribute
- ✗ Phillips Building — an instance
- ✓ Registration
- ✓ Section
- ✗ Course Catalogue — a report
- ✗ Physics — an instance
- ✓ Class
- ✗ Professor — a Job Title
- ✗ Admission Request Form — a form (artifact)

Entity definition basics

Definitions must focus on what a single instance is:

- Not "how they're used" or "how they're created" or "why we care" or "how the process works" or "interesting problems and tidbits" etc.
- They simply answer the question "What is *one* of these things?"

Key Point

"What is one of these things?"

The most useful questions:

"Can anyone think of examples that might surprise someone else – that is, anomalies or potential sources of confusion?" E.g., to define *Customer…*

- "In our area, other divisions are treated as customers"
- "We record recipients of charitable donations as customers."

"Could we list some examples?" e.g.,
Rita Smith, Acme Auto, Ministry of Finance, homeowners… (aha!)

"Does this deal with "kinds of things" or "specific things?"

- "kind" - Customer Category vs. "specific" – an individual Customer
- if it's a specific thing, still ask if there are recognised types (e.g., Personal, Corporate, Government; Lead, Prospect, Active)

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.~~

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?

-
-
-
-
-
-
- ...

Employee

Project

Account

Task

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?

F/T vs. P/T?

Only IS Department?

Include management,
or only individual contributors?

Still in recruitment (an applicant)?

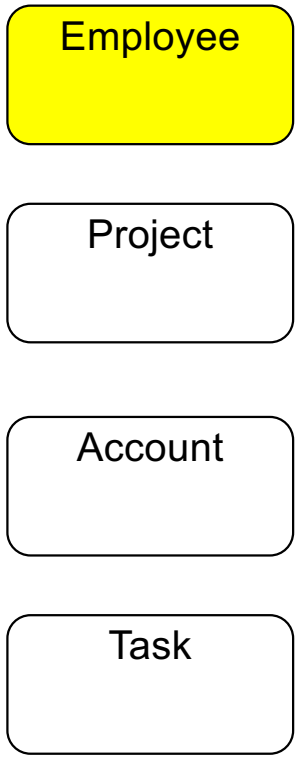
Onboarded? on probation? active? retirees?

Include contractors, student interns, vendor staff, etc.?

Volunteers?

A type of worker (DBA or tester) or a specific person?

A robotic, automated, or AI agent?



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*

Employee

Project

Account

Task

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?
- 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?
- …

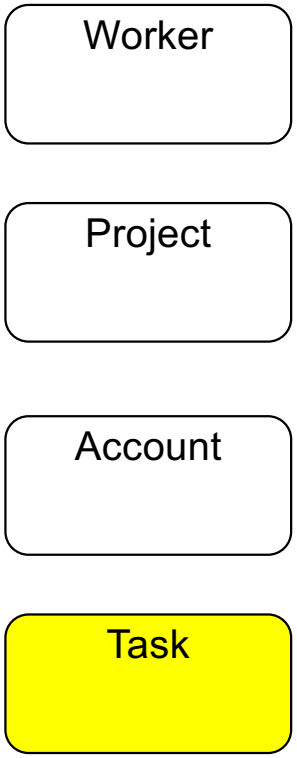
A ***Task*** is a specific, time-bounded, unit of work, within a single Project, intended to be performed by one or more Workers, that produces an intended deliverable or achieves a specific state.

Examples:

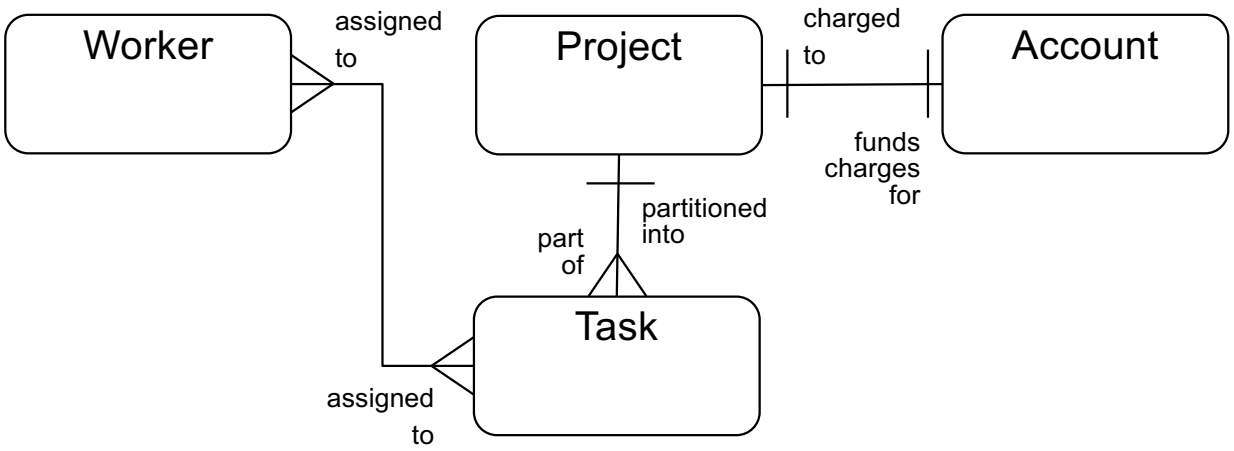
- Code *Place Order* service
- Test *Place Order* service

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.

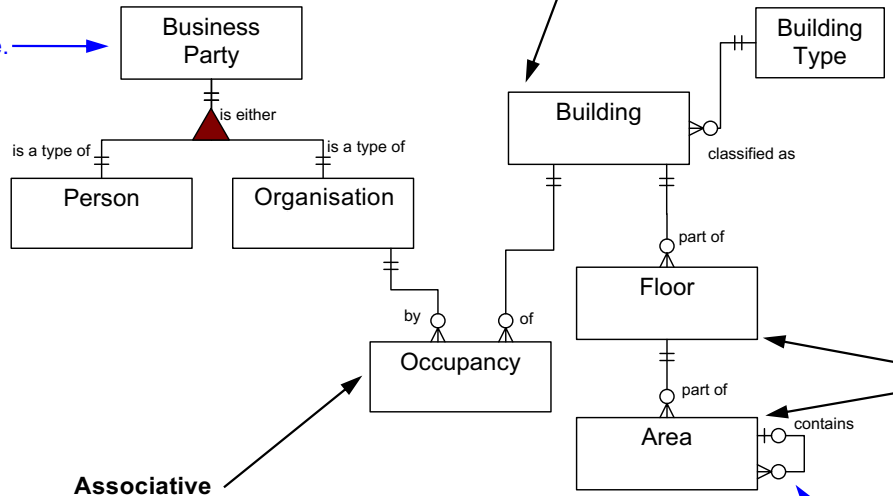
Then add relationships with a verb-based phrase.

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

For reference – summary of 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.



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

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

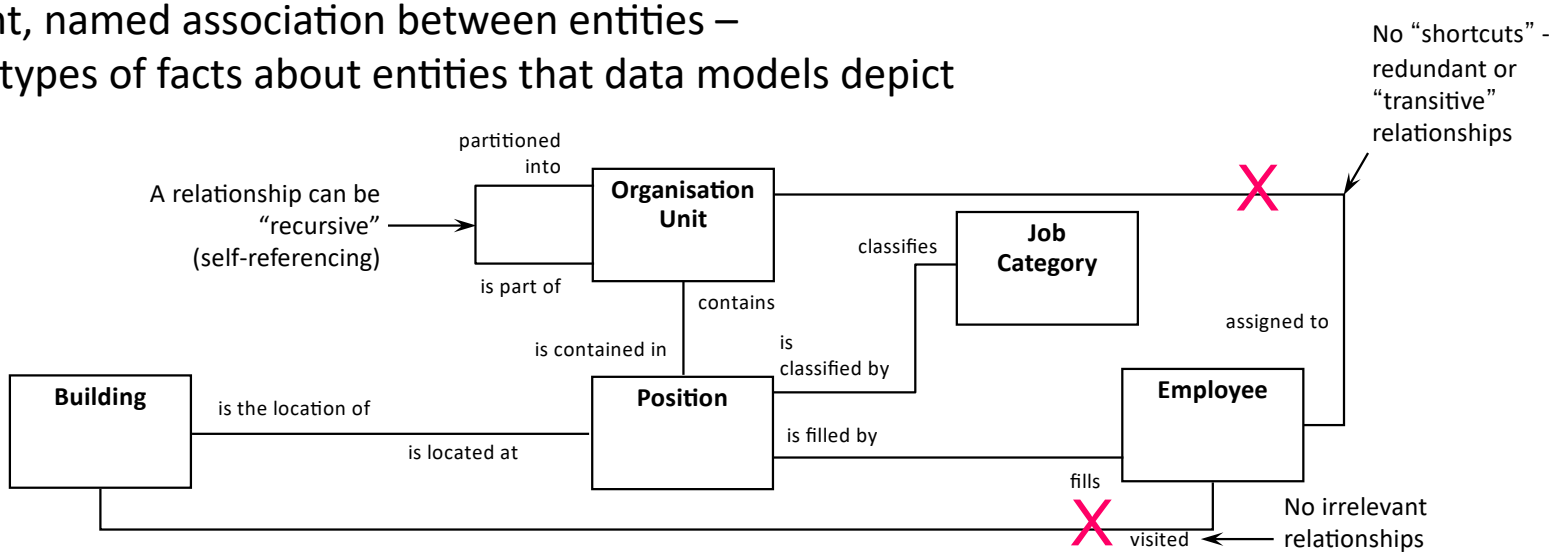
Characteristic
Dependent on one parent
Records multi-valued facts about a parent entity that have been "cast out" from that entity
Drawn below parent

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

Optional – the finer points, beginning with relationships

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



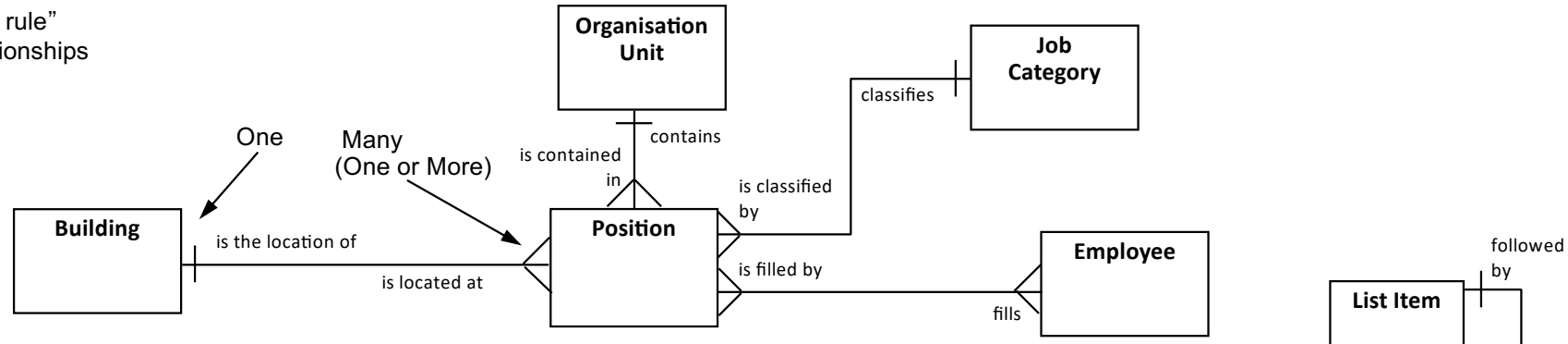
Guidelines

- 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"

Jump ahead to 73

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:*

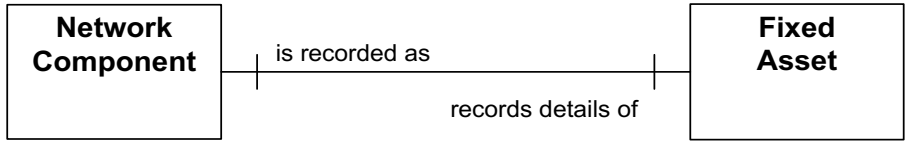
- for each entity, ask
  "Can one of these be related to a *maximum* of *One* of the other or a *maximum* of *Many* of the other?"

- record the answer (One or Many) at the "other" end;
  later, "One or More" will be better than "Many"

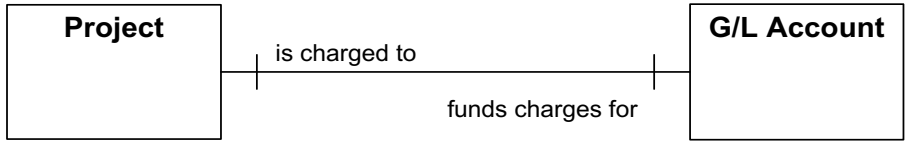
- possibilities – 1:1 (error), 1:M (common), M:M (more work, eventually)

1:1 relationships – almost always an error!

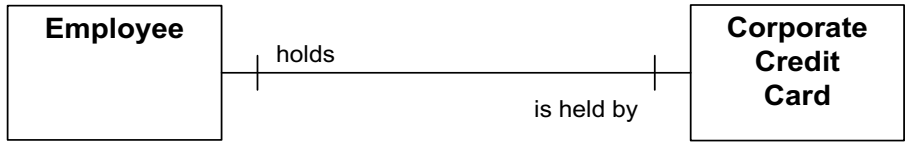
- Note – a 1:1 relationship might be necessary in the Physical Database Design
  e.g., "Fixed Asset" records financial data about a "Network Component" but they are in two separate systems (the G/L System and the Configuration Management System) connected by a 1:1 relationship



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

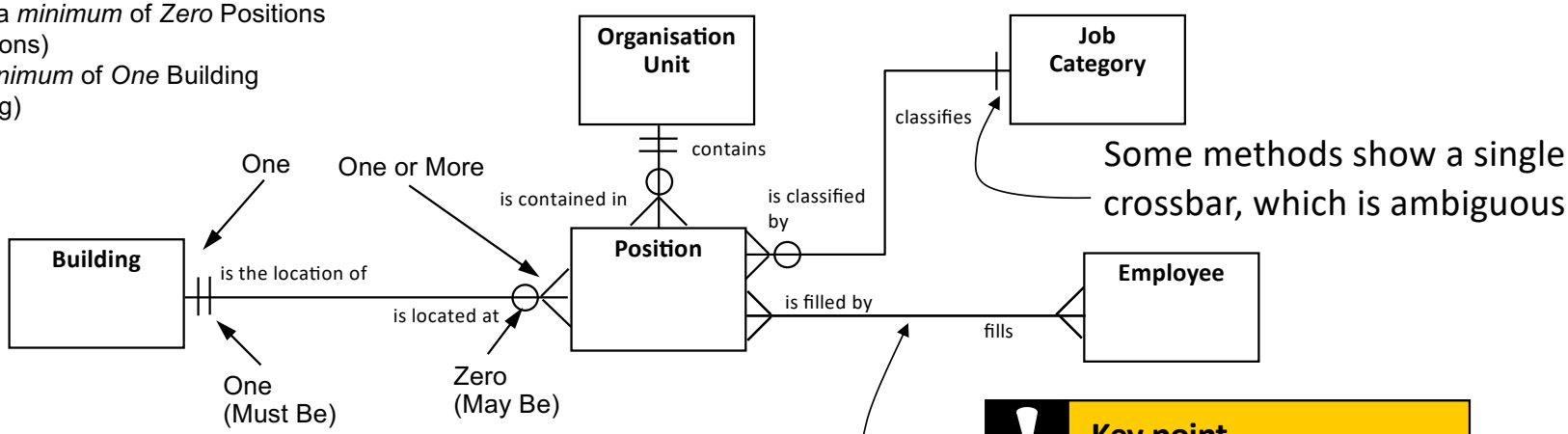


✘ Failing to account for changes over time
  e.g., an Employee may hold only *one* Credit Card at a time, but *many over time,* and we virtually always want history.
  The most common written constraint in Concept Modelling is
  "one <u>at</u> a time but many <u>over</u> time."



Relationship optionality (logical models only)

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



Some methods show a single crossbar, which is ambiguous

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

Key point

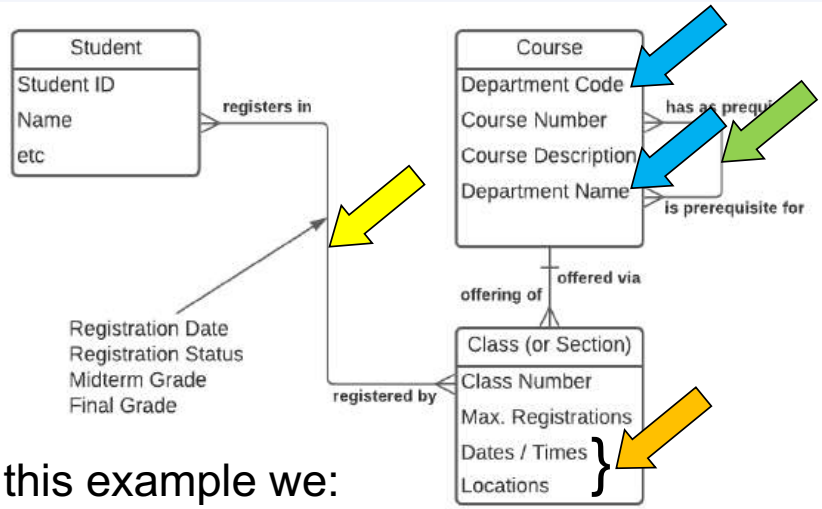
Typically only shown in logical data models – don't bother with optionality on M:Ms

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

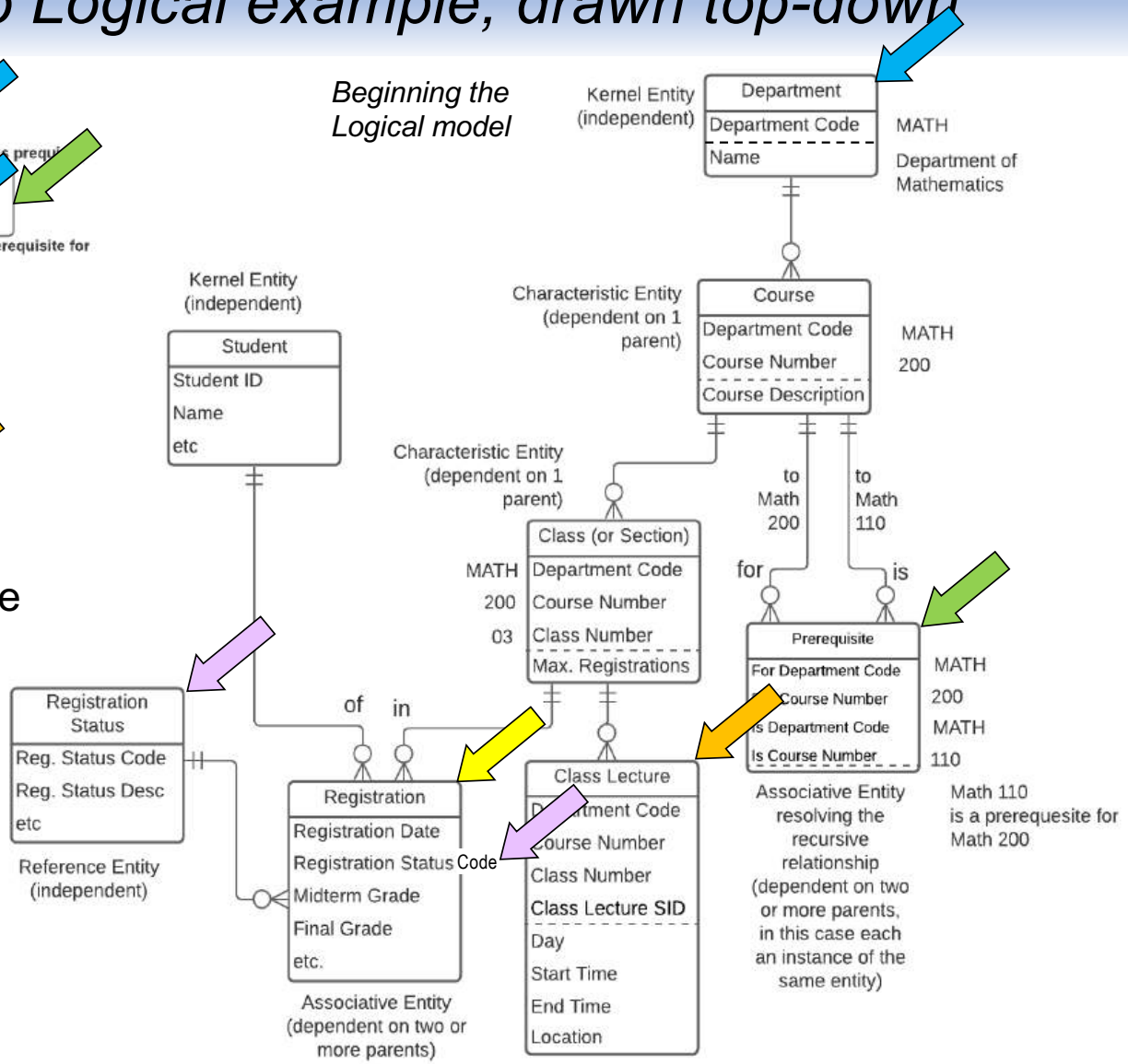
- for each entity, ask
  "Can one of these be related to a *minimum* of *Zero* <u>or</u> a *minimum* of *One* of the other entity?"
- record the answer – 0 <u>or</u> 1 – at the "other" end
  "zero" means an optional relationship (*May Be*) and "one" means a mandatory relationship (*Must Be*)
- easier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

One more Conceptual 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

Don't forget the four Ds of Concept 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?

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)

Wrap-up discussion

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

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.

Thanks again!

Alec Sharp, West Vancouver, BC, Canada

If you have questions or comments…
don't be shy, get in touch!

- e: asharp@clariteq.com
- ig: @alecsharp01
- m: +1 604 418-3352