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

A half-day workshop presented by Adept Events
28 maart, 2024 in Utrecht NL

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

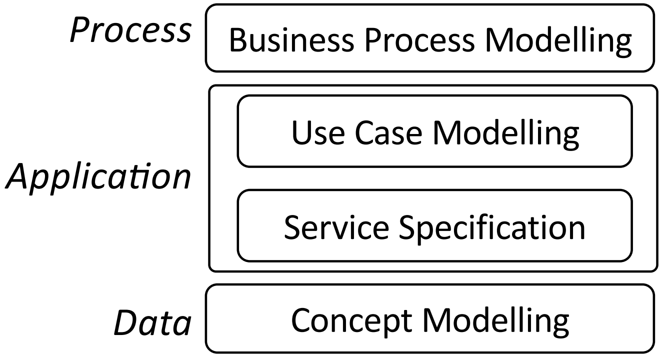
www.adeptevents.nl www.clariteq.com

Instructor / course developer background…

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

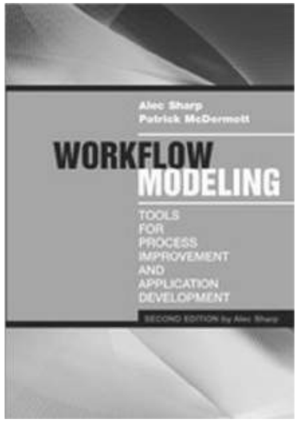
- 40+ years experience as an independent consultant:
  - Business Process Change – discover, model, analyse, and design/redesign processes
  - Application Requirements Specification
  - Data Modelling and Management My roots!

    +
  - Facilitation & Organisational Change
  - Project Recovery



- Consulting, teaching, speaking globally (pre-pandemic)
- Awarded DAMA's global Professional Achievement Award for contributions to "human-friendly" data modelling
- Author of "Workflow Modeling"
  - best-selling book on process modelling & improvement
  - second edition – a complete re-write

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


Alec Sharp
Patrick McDermott
WORKFLOW
MODELING
TOOLS FOR PROCESS IMPROVEMENT AND APPLICATION DEVELOPMENT
SECOND EDITION by Alec Sharp

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	

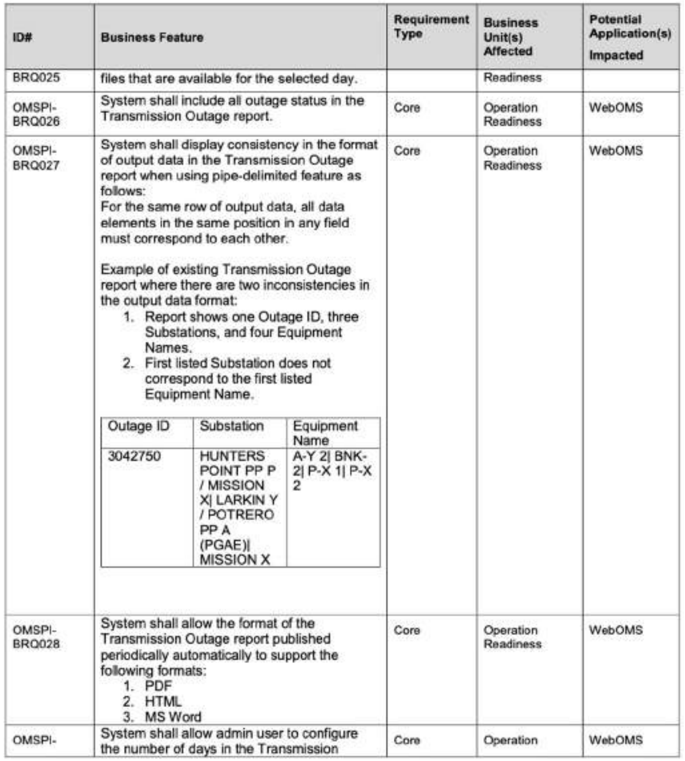
Introductions, if time/numbers permits:

- 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

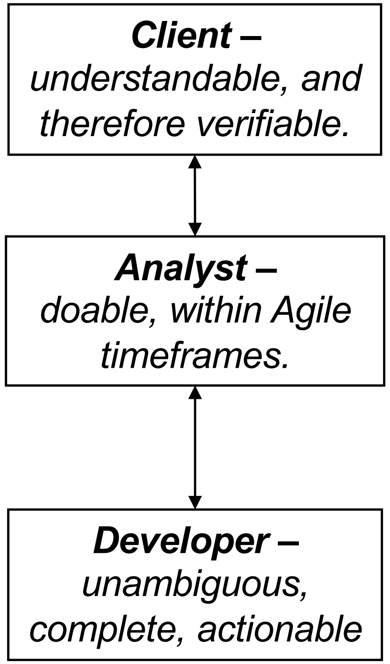
Simplistic methods at one extreme: can do as much harm as good	The goal lies in the middle ground:	Overly complex methods at the other extreme: difficult for businesspeople to verify

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



<table>
<tr><th>ID#</th><th>Business Feature</th><th>Requirement Type</th><th>Business Unit(s) Affected</th><th>Potential Application(s)
Impacted</th></tr>
<tr><td>BRQ025</td><td>files that are available for the selected day.</td><td></td><td>Readiness</td><td></td></tr>
<tr><td>OMSPI-BRQ026</td><td>System shall include all outage status in the Transmission Outage report.</td><td>Core</td><td>Operation Readiness</td><td>WebOMS</td></tr>
<tr><td>OMSPI-BRQ027</td><td>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.
<table>
<tr><th>Outage ID</th><th>Substation</th><th>Equipment Name</th></tr>
<tr><td>3042750</td><td>HUNTERS POINT PP P / MISSION X| LARKIN Y / POTRERO PP A (PGAE)| MISSION X</td><td>A-Y 2| BNK-2| P-X 1| P-X 2</td></tr>
</table>
</td><td>Core</td><td>Operation Readiness</td><td>WebOMS</td></tr>
<tr><td>OMSPI-BRQ028</td><td>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</td><td>Core</td><td>Operation Readiness</td><td>WebOMS</td></tr>
<tr><td>OMSPI-</td><td>System shall allow admin user to configure the number of days in the Transmission</td><td>Core</td><td>Operation</td><td>WebOMS</td></tr>
</table>

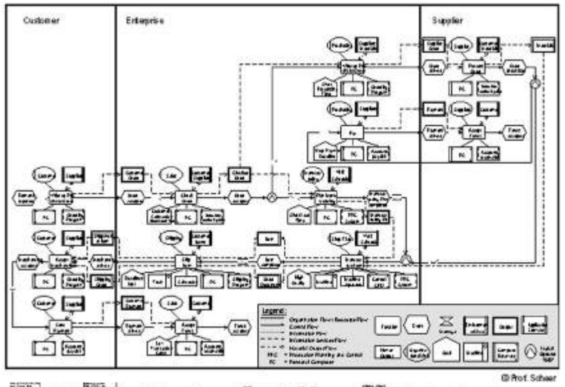


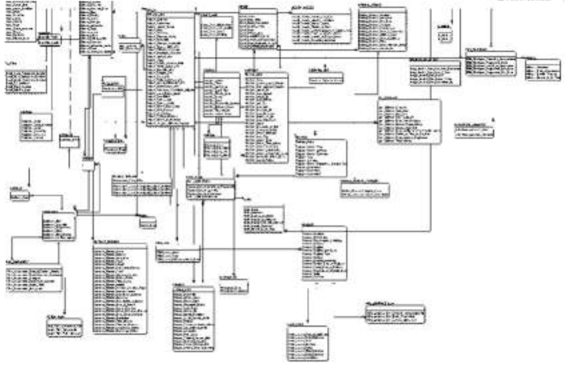
Client – *understandable, and therefore verifiable.*

Analyst – *doable, within Agile timeframes.*

Developer – *unambiguous, complete, actionable*

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





Discussion – the problems 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

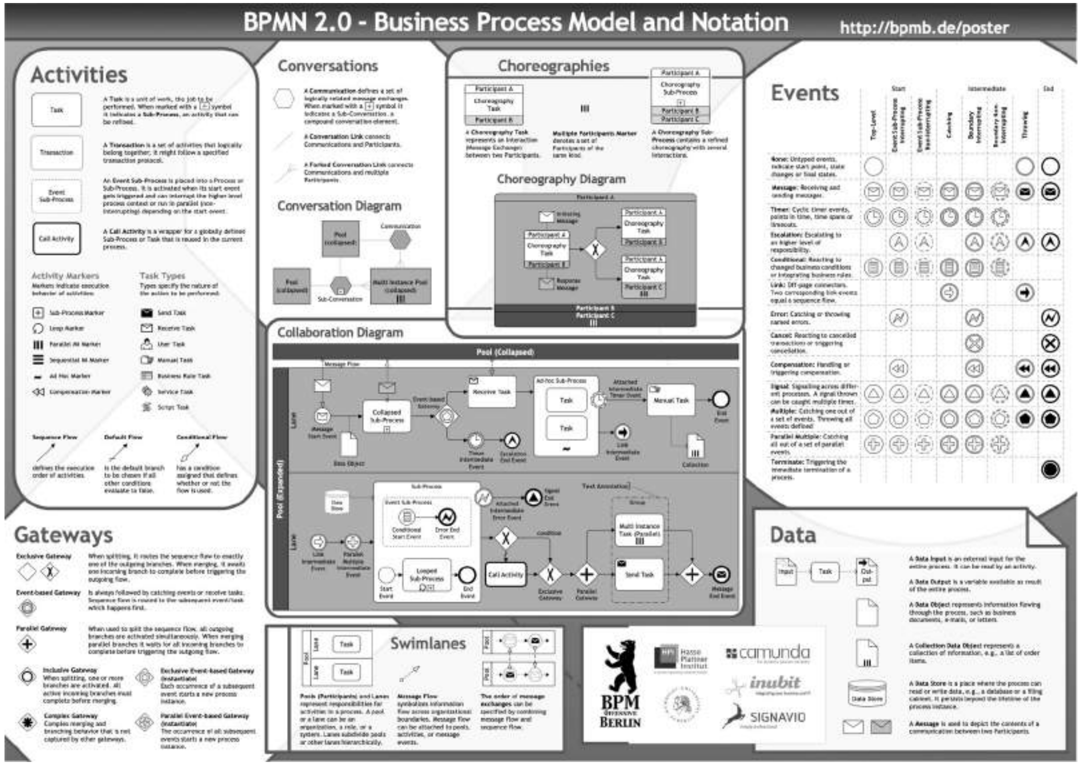
- *Business Process Models* to show where this fits in the end-to-end process
- *Concept Models* to show the required information

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.



A better approach – a model-based framework for Business Analysis

Framework Layer | **Technique sample** | **What it covers**

This is not a sequence!

Goals

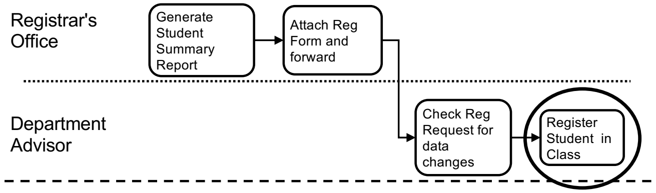
Business Objectives

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

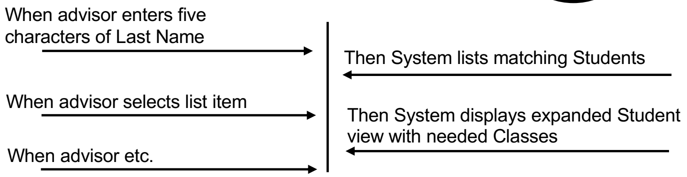

Registrar's Office
Department Advisor
Generate Student Summary Report
Attach Reg Form and forward
Check Reg Request for data changes
Register Student in Class

✓ ***Process Model*** - shows "what" in a Scope Model, then "who & how" in a Workflow Model – the steps done by the actors in the process

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

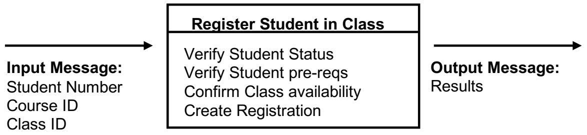
Application

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

Business Services (rules & logic)

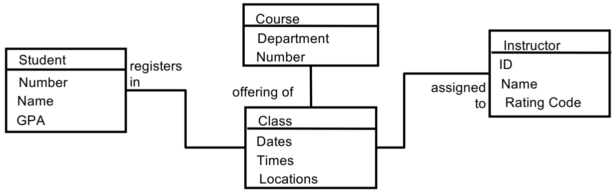


✓ ***Service Specification*** - describes a service – a package of rules and logic – that is triggered to complete or respond to a business event

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

Data

Data Mgmt. Services (databases)

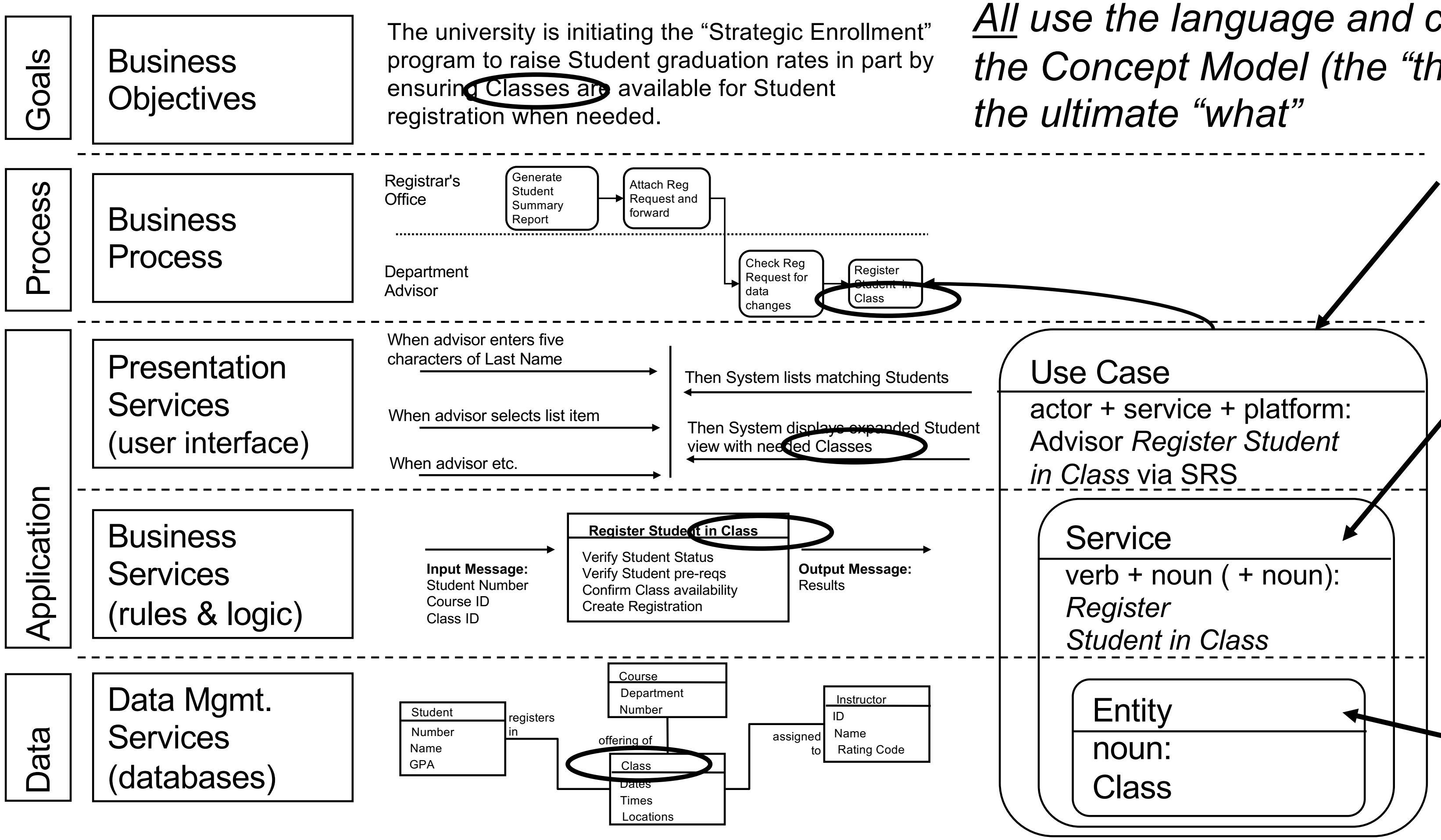


✓ ***Concept Model*** - depicts the things and the facts about things the organisation needs to record; the things (the entities) are what processes and solutions act on.

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

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”



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.

Process

Business Process

Registrar's Office
Generate Student Summary Report
Attach Reg Request and forward
Department Advisor
Check Reg Request for data changes
Register Student in Class

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.

Business Services (rules & logic)

Register Student in Class

Input Message: Student Number, Course ID, Class ID

Verify Student Status
Verify Student pre-reqs
Confirm Class availability
Create Registration

Output Message: Results

Data

Data Mgmt. Services (databases)

Use Case

actor + service + platform:
Advisor *Register Student in Class* via SRS

Service

verb + noun (+ noun):
Register Student in Class

Entity

noun:
Class

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

Client –

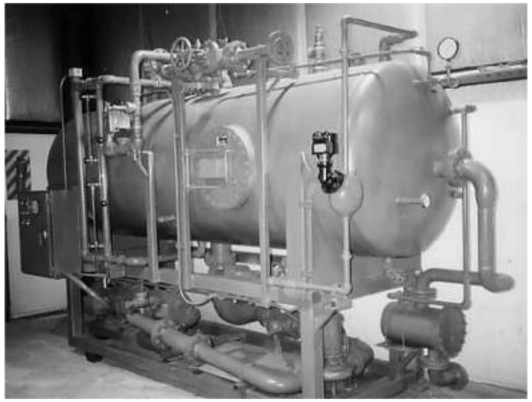
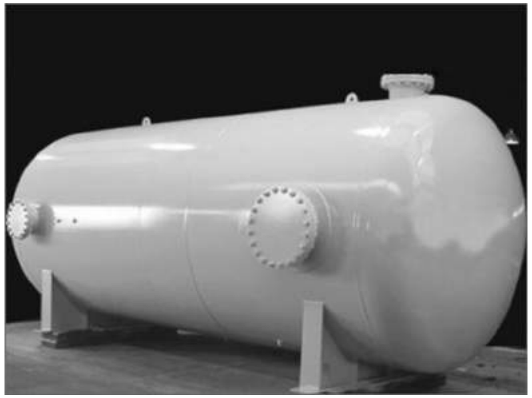
- Regulatory agency ensuring the safe design, installation, and use of technical equipment
- Natural gas systems, electrical systems, boilers and pressure vessels, elevating devices, & many more

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

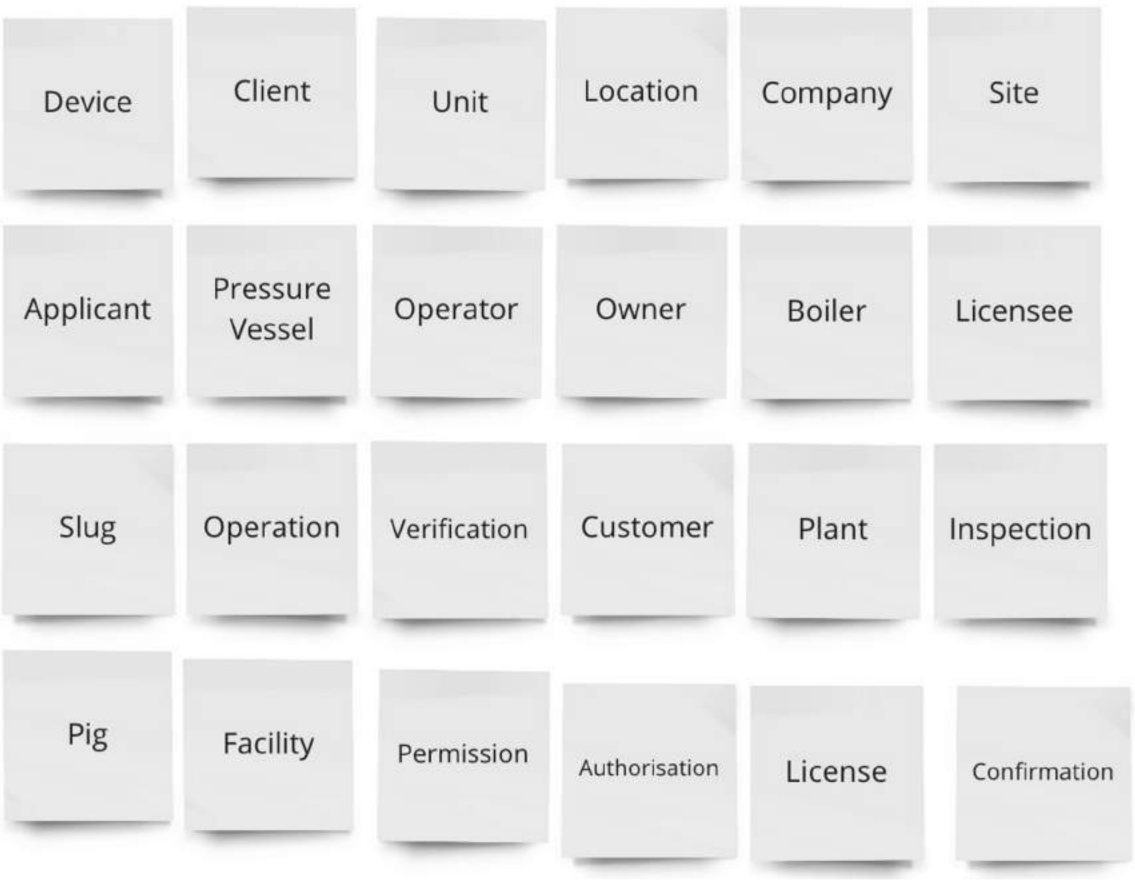
- *Business Development chooses Pilot Program –*
  boilers and pressure vessels in Oil & Gas fields

- 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 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.

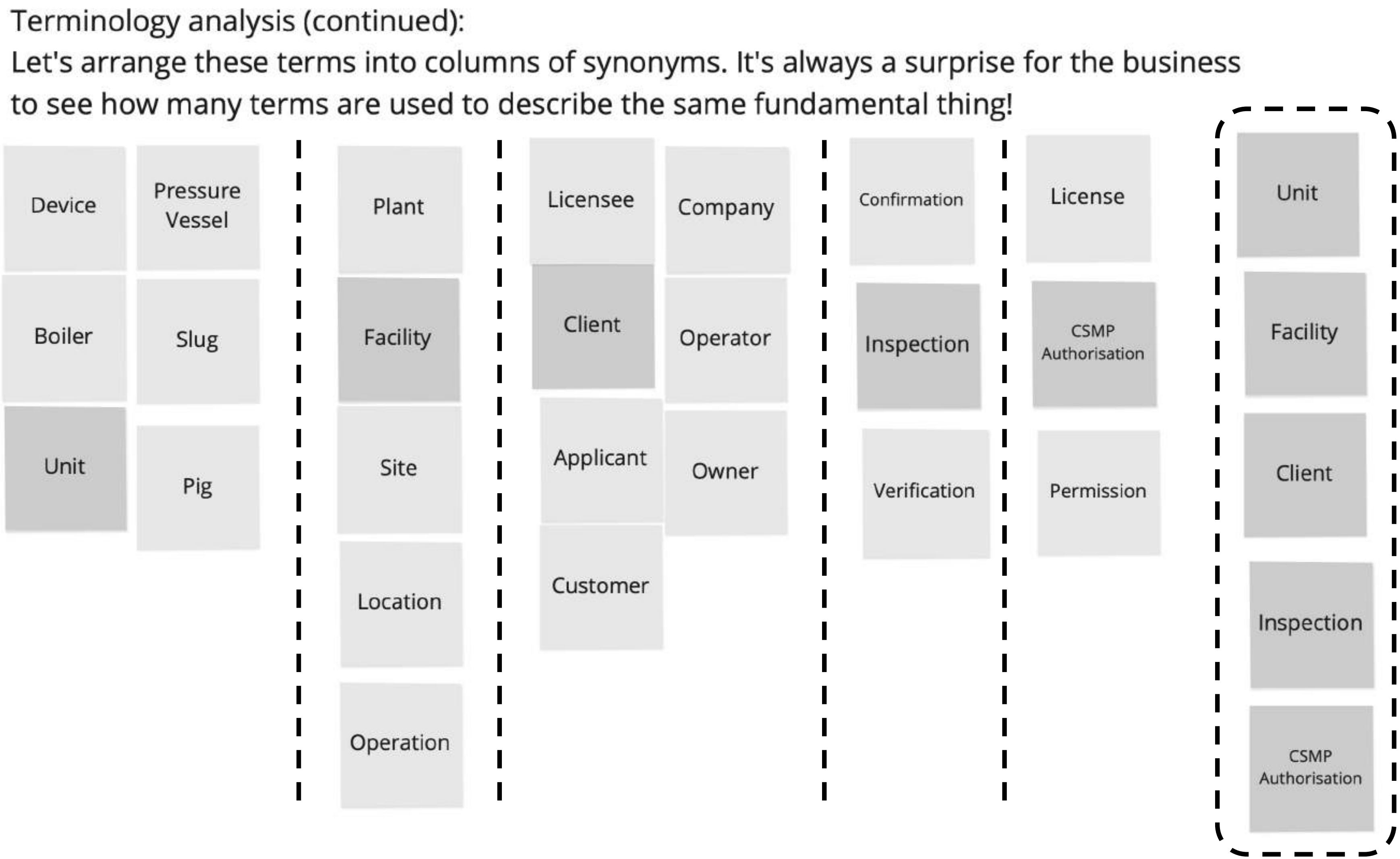


Device	Client	Unit	Location	Company	Site
Applicant	Pressure Vessel	Operator	Owner	Boiler	Licensee
Slug	Operation	Verification	Customer	Plant	Inspection
Pig	Facility	Permission	Authorisation	License	Confirmation

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?

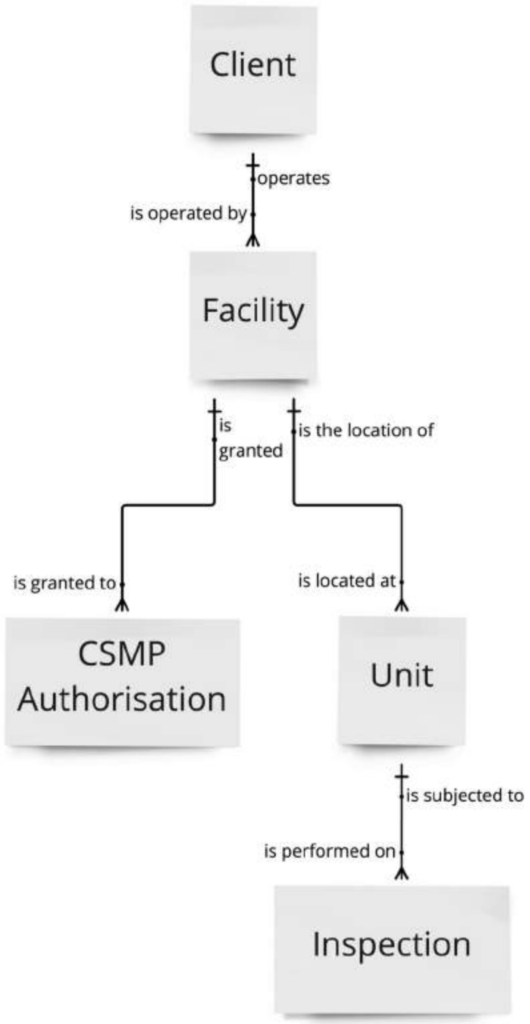
Review from an example on Miro – Terminology Analysis



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

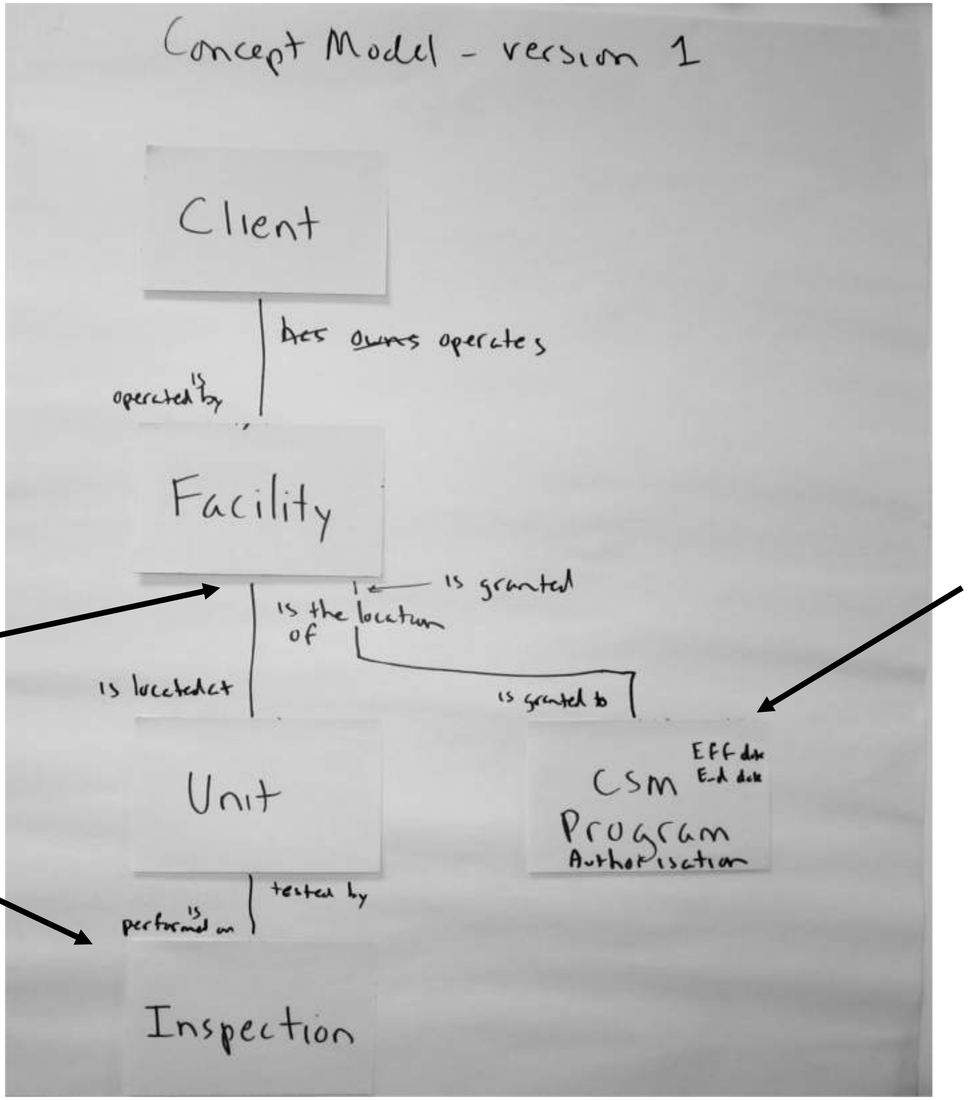
1. We arranged the entities / business objects by dependency
2. Then we drew relationship lines
3. Then we added a relationship name in each direction
4. Only then did we state (in words) the cardinality (1:1, 1:M, M:M) and then update the diagram with hash marks (┼) and crowsfeet (⋔)



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

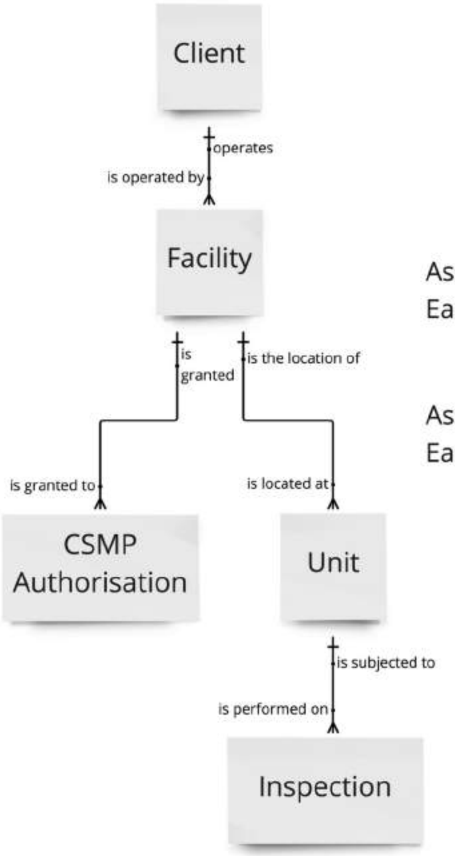
Now, state the relationships ***emphatically*** as Assertions. ***Each*** Client operates ***one or more*** Facilities! Then, ***challenge*** them!
Again, don't worry yet about ***optionality*** – whether the relationship ***must be*** or ***may be*** be present.
We only care now about the ***maximum***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

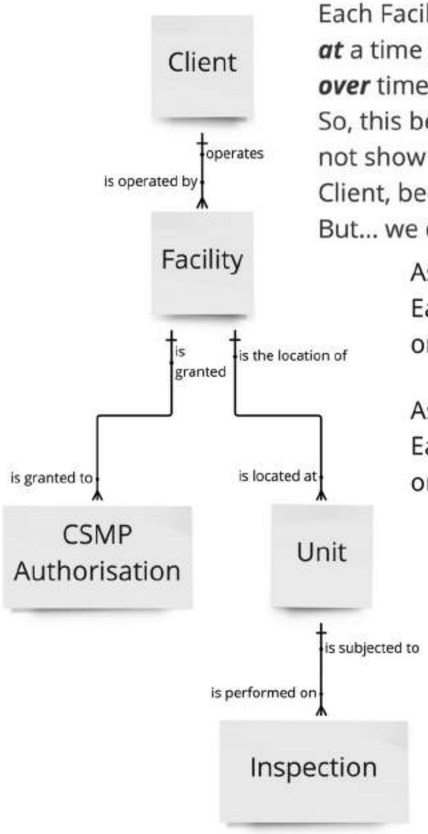
Assertion:
Each Client operates
one or more Facilities

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

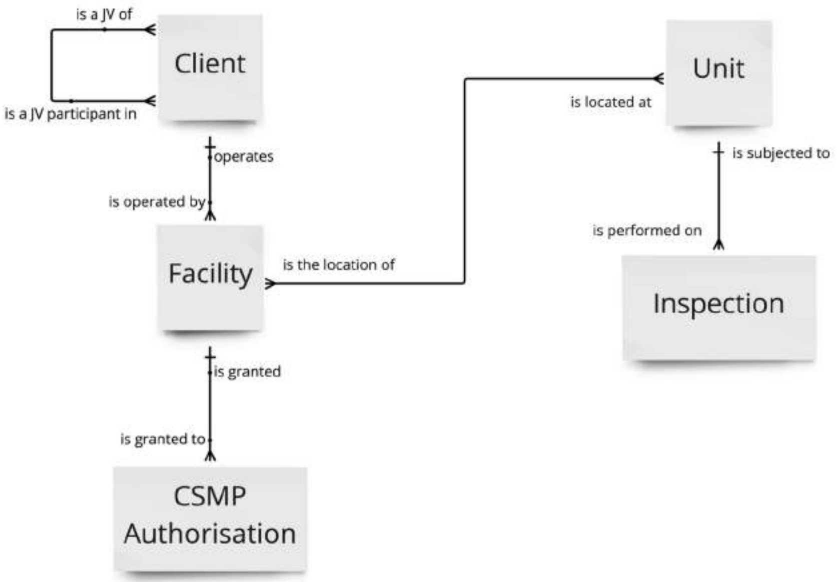
Assertion:
Each Unit is located at
one Facility

Assertion:
Each Unit is subjected to
one or more Inspections

Assertion:
Each Inspection is performed on
one Unit

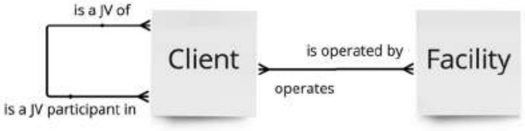
Concept Model Version 2; revised from challenging Assertions

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

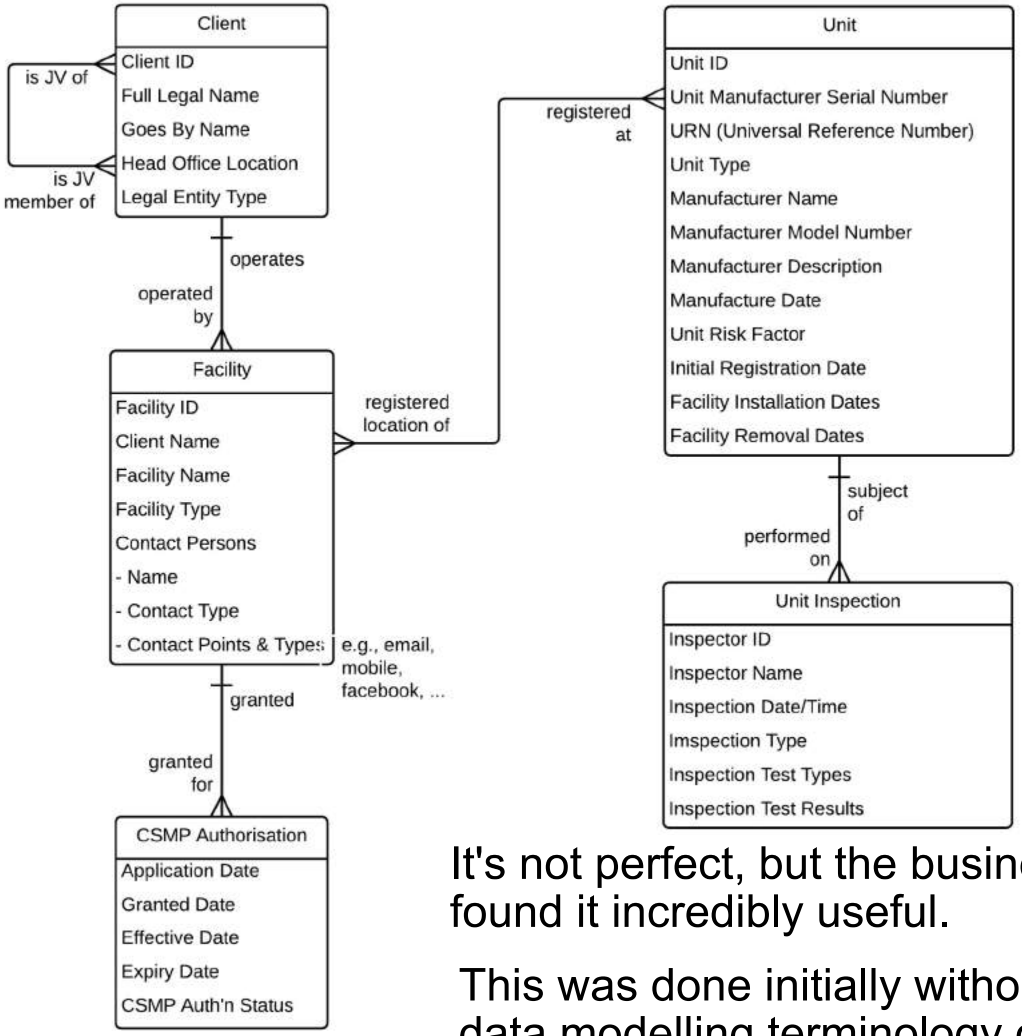


Note:
You don't always get what you *want* or what you think is the *right* thing in Concept Modelling. In this case the client (the Regulator) said they always wanted a Facility to be operated by ONE AND ONLY ONE Client.
If a Facility was operated by multiple Clients, they would require the Clients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



Client
- Client ID
- Full Legal Name
- Goes By Name
- Head Office Location
- Legal Entity Type

Facility
- Facility ID
- Client Name
- Facility Name
- Facility Type
- Contact Persons
- - Name
- - Contact Type
- - Contact Points & Types

e.g., email, mobile, facebook, ...

CSMP Authorisation
- Application Date
- Granted Date
- Effective Date
- Expiry Date
- CSMP Auth'n Status

Unit
- Unit ID
- Unit Manufacturer Serial Number
- URN (Universal Reference Number)
- Unit Type
- Manufacturer Name
- Manufacturer Model Number
- Manufacturer Description
- Manufacture Date
- Unit Risk Factor
- Initial Registration Date
- Facility Installation Dates
- Facility Removal Dates

Unit Inspection
- Inspector ID
- Inspector Name
- Inspection Date/Time
- Imspection Type
- Inspection Test Types
- Inspection Test Results

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

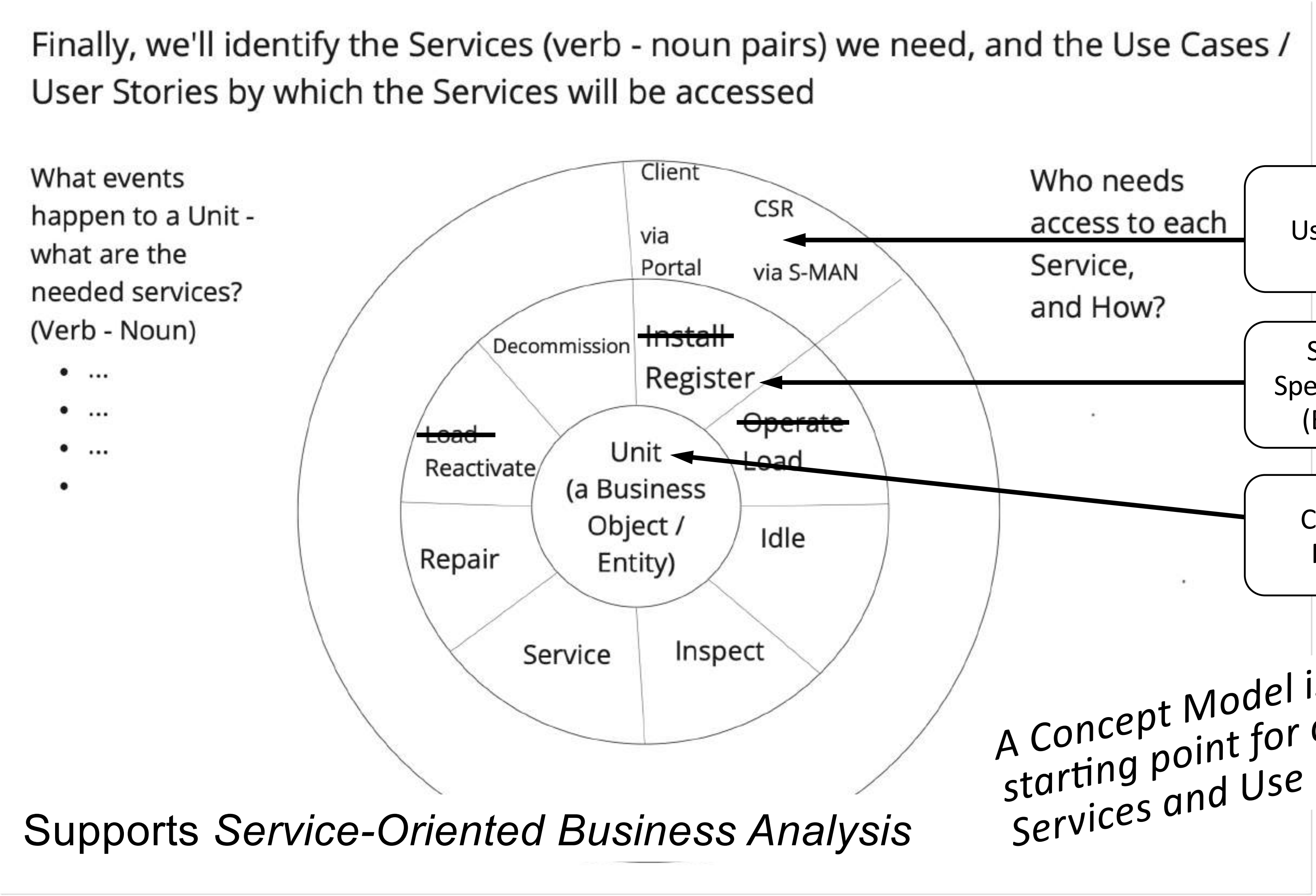
- Is there one CSMP per Client, per Facility, or some other basis?
- Do Units frequently relocate, or even turn up at another Client?
- What is inspected – the Facility or the Unit?
- Does the CSMP cover all or some Units at a Facility?
- …and MANY more…

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

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 Cases

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…

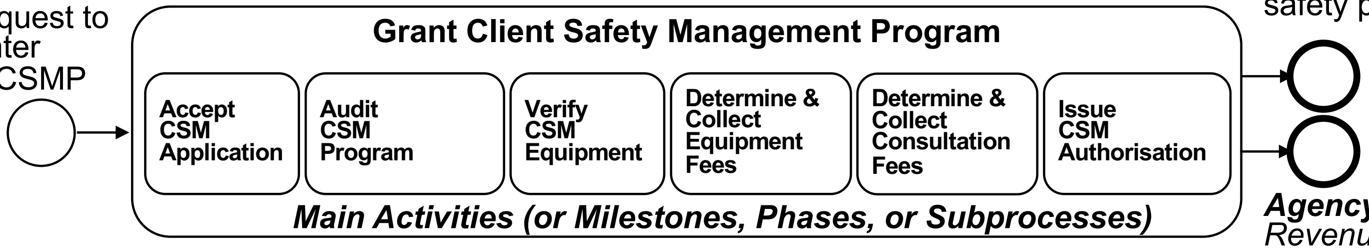
Later we'll clarify that
Use Cases and User Stories
are essentially the same

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)

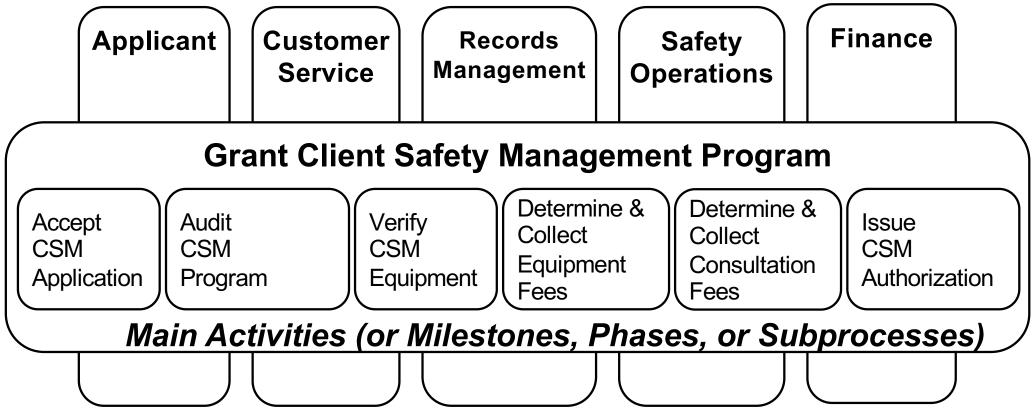
Client Result:
Approval granted for a self-managed safety program.

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

Cases:

- New
- Legacied
- Ownership Change

Process Scope Model – pure “what”…



Applicant

Customer Service

Records Management

Safety Operations

Finance

Grant Client Safety Management Program

Accept CSM Application

Audit CSM Program

Verify CSM Equipment

Determine & Collect Equipment Fees

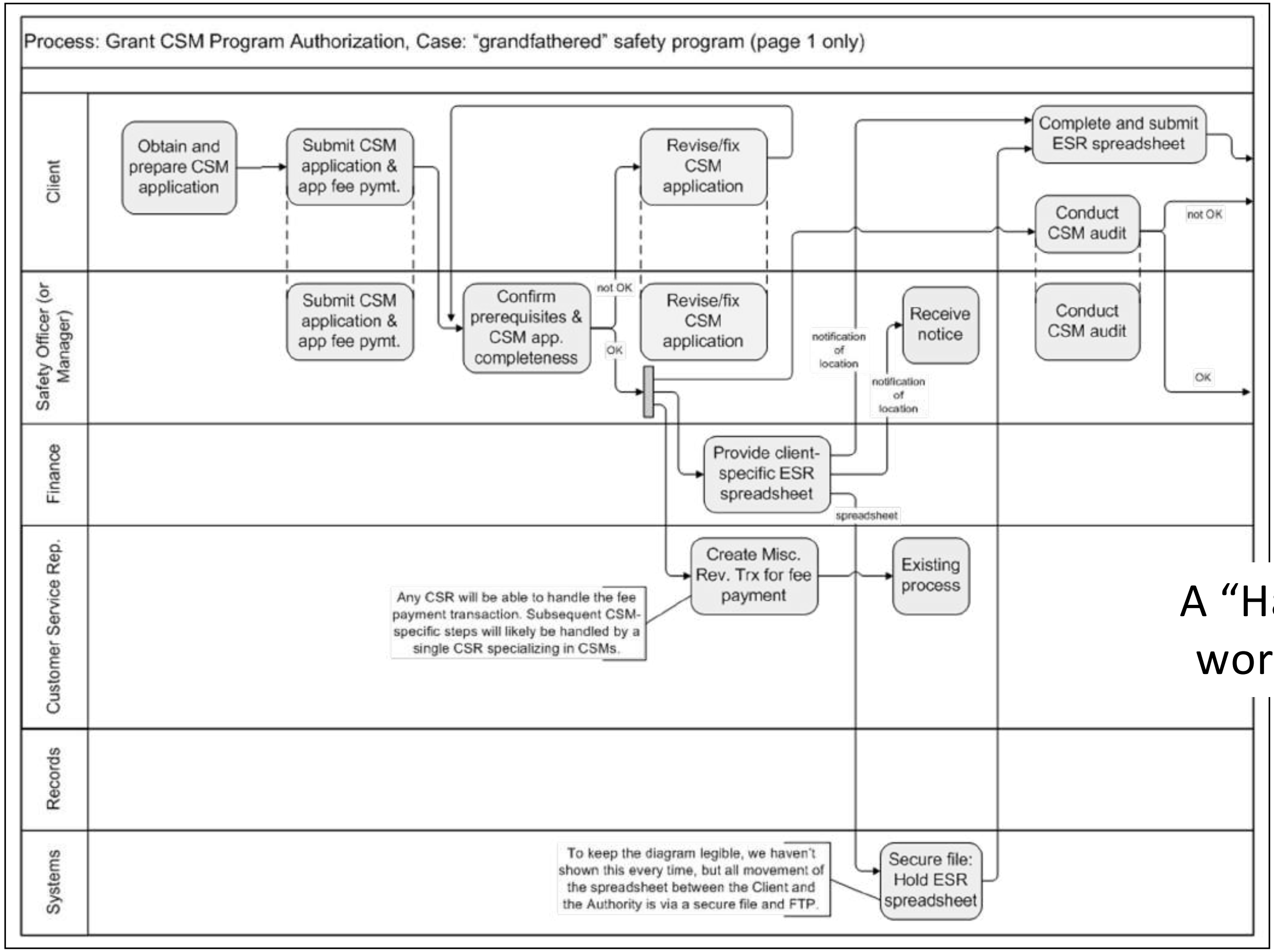
Determine & Collect Consultation Fees

Issue CSM Authorization

Main Activities (or Milestones, Phases, or Subprocesses)

Process Summary Chart – simplified “what,” plus “who”

The initial, business-friendly workflow model

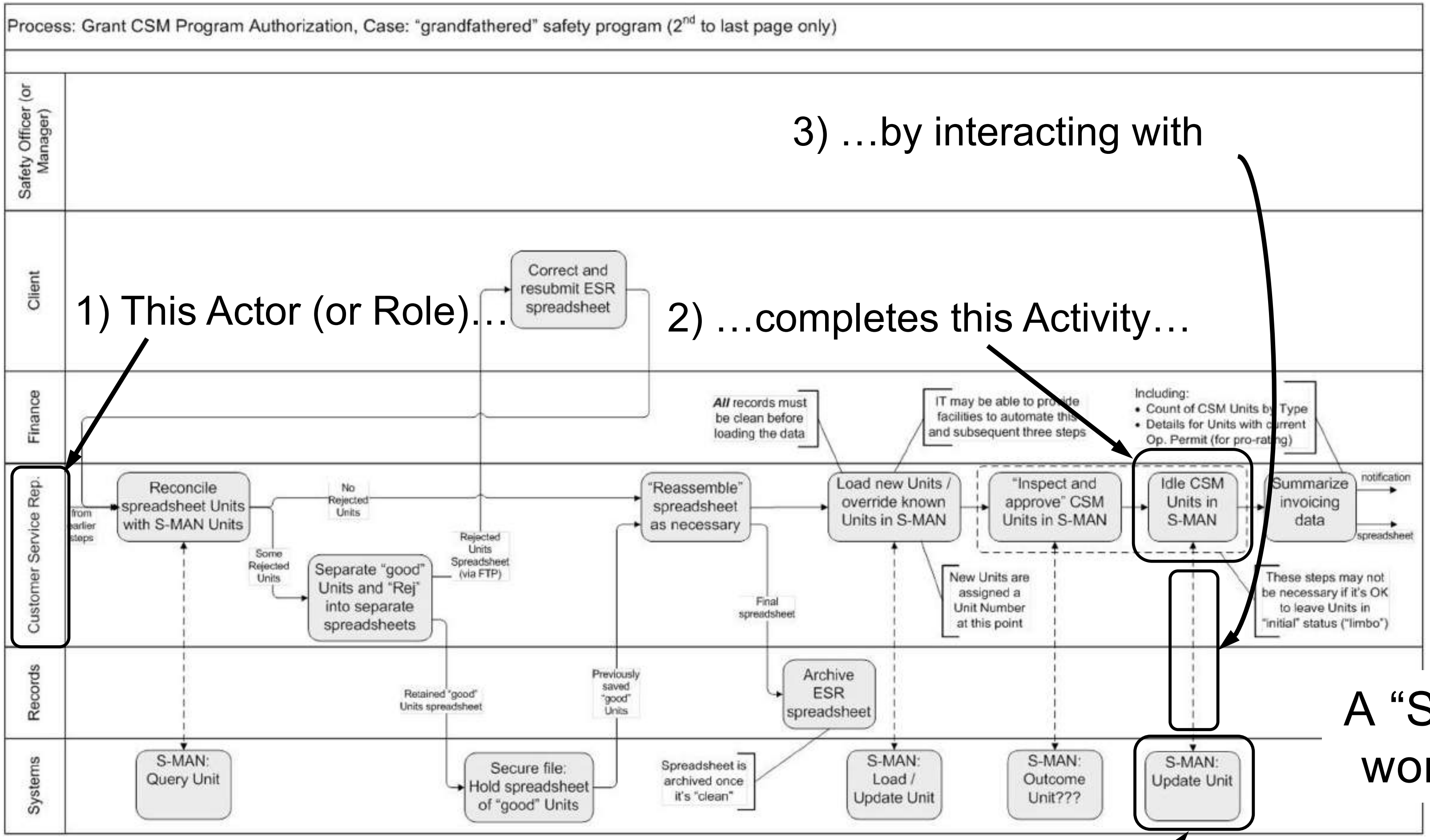

Process: Grant CSM Program Authorization, Case: "grandfathered" safety program (page 1 only)
Client
Safety Officer (or Manager)
Finance
Customer Service Rep.
Records
Systems
Obtain and prepare CSM application
Submit CSM application & app fee pymt.
Submit CSM application & app fee pymt.
Confirm prerequisites & CSM app. completeness
not OK
OK
Revise/fix CSM application
Revise/fix CSM application
notification of location
Receive notice
notification of location
Complete and submit ESR spreadsheet
Conduct CSM audit
Conduct CSM audit
not OK
OK
Provide client-specific ESR spreadsheet
spreadsheet
Create Misc. Rev. Trx for fee payment
Existing process
Secure file: Hold ESR spreadsheet

Any CSR will be able to handle the fee payment transaction. Subsequent CSM-specific steps will likely be handled by a single CSR specializing in CSMs.

To keep the diagram legible, we haven't shown this every time, but all movement of the spreadsheet between the Client and the Authority is via a secure file and FTP.

A "Handoff Level" workflow model

Eventually, detail showing where use cases & services fit


Process: Grant CSM Program Authorization, Case: "grandfathered" safety program (2nd to last page only)
Safety Officer (or Manager)
Client
Finance
Customer Service Rep.
Records
Systems
Correct and resubmit ESR spreadsheet
from earlier steps
Reconcile spreadsheet Units with S-MAN Units
No Rejected Units
Some Rejected Units
Separate "good" Units and "Rej" into separate spreadsheets
Rejected Units Spreadsheet (via FTP)
"Reassemble" spreadsheet as necessary
Final spreadsheet
Retained "good" Units spreadsheet
Previously saved "good" Units
All records must be clean before loading the data
IT may be able to provide facilities to automate this and subsequent three steps
Including:
• Count of CSM Units by Type
• Details for Units with current Op. Permit (for pro-rating)
Load new Units / override known Units in S-MAN
"Inspect and approve" CSM Units in S-MAN
Idle CSM Units in S-MAN
Summarize invoicing data
notification
spreadsheet
New Units are assigned a Unit Number at this point
These steps may not be necessary if it's OK to leave Units in "initial" status ("limbo")
Archive ESR spreadsheet
S-MAN: Query Unit
Secure file: Hold spreadsheet of "good" Units
Spreadsheet is archived once it's "clean"
S-MAN: Load / Update Unit
S-MAN: Outcome Unit???
S-MAN: Update Unit

3) …by interacting with

1) This Actor (or Role)…

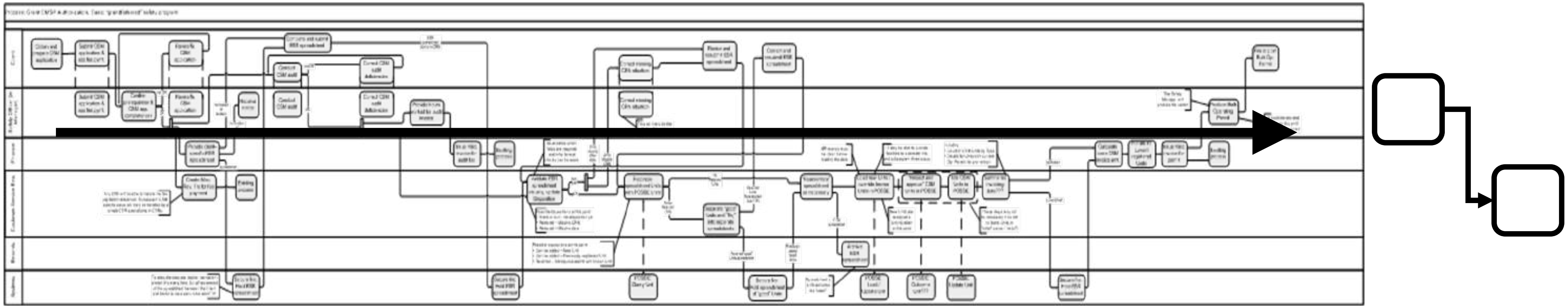
2) …completes this Activity…

A "Service Level" workflow model

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

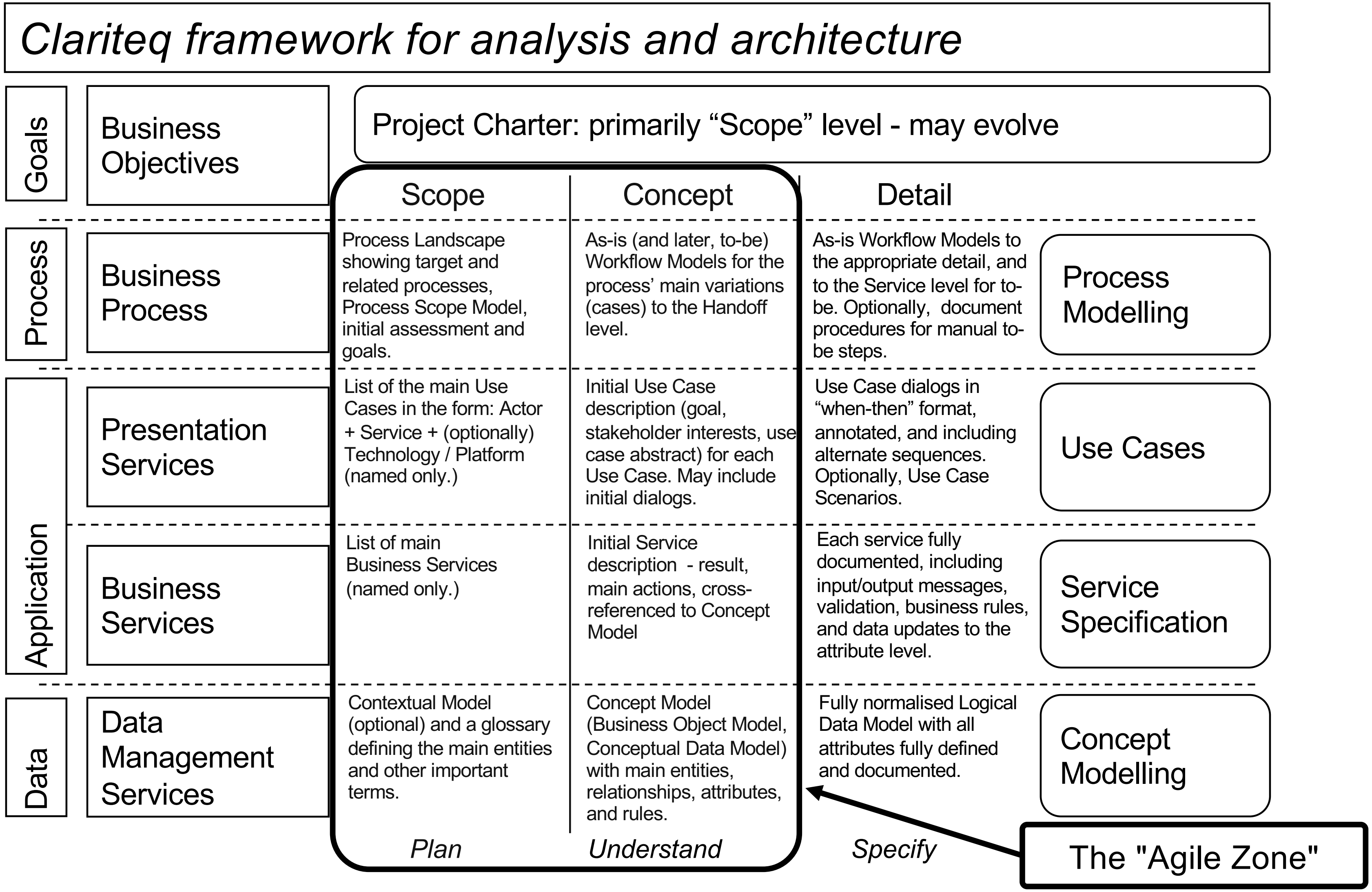
Mission accomplished! Conclusions:

- "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

Goals	Business Objectives	Project Charter: primarily "Scope" level - may evolve			
		Scope	Concept	Detail	
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
	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*	

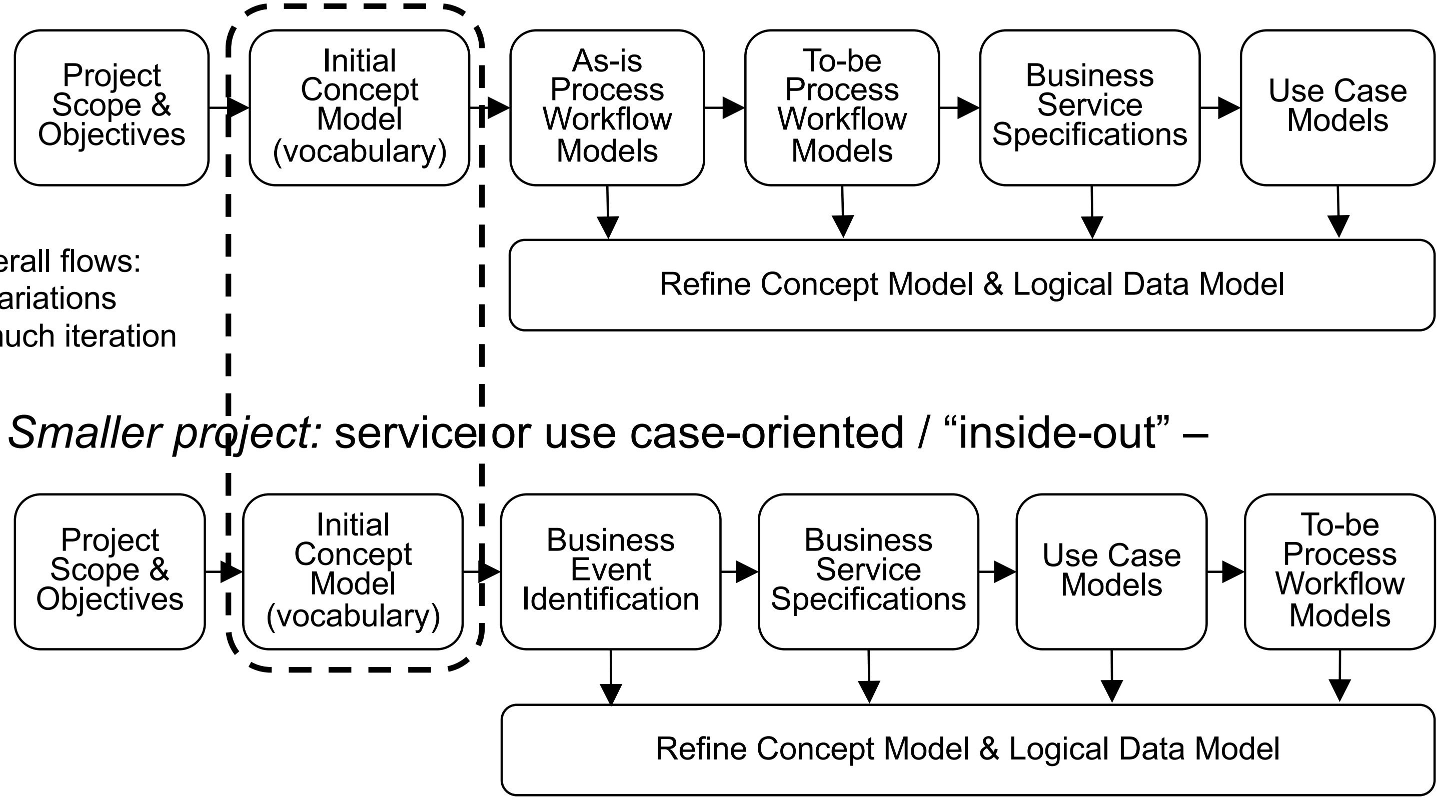
The "Agile Zone"

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


Project Scope & Objectives
Initial Concept Model (vocabulary)
As-is Process Workflow Models
To-be Process Workflow Models
Business Service Specifications
Use Case Models
Refine Concept Model & Logical Data Model
Project Scope & Objectives
Initial Concept Model (vocabulary)
Business Event Identification
Business Service Specifications
Use Case Models
To-be Process Workflow Models
Refine Concept Model & Logical Data Model

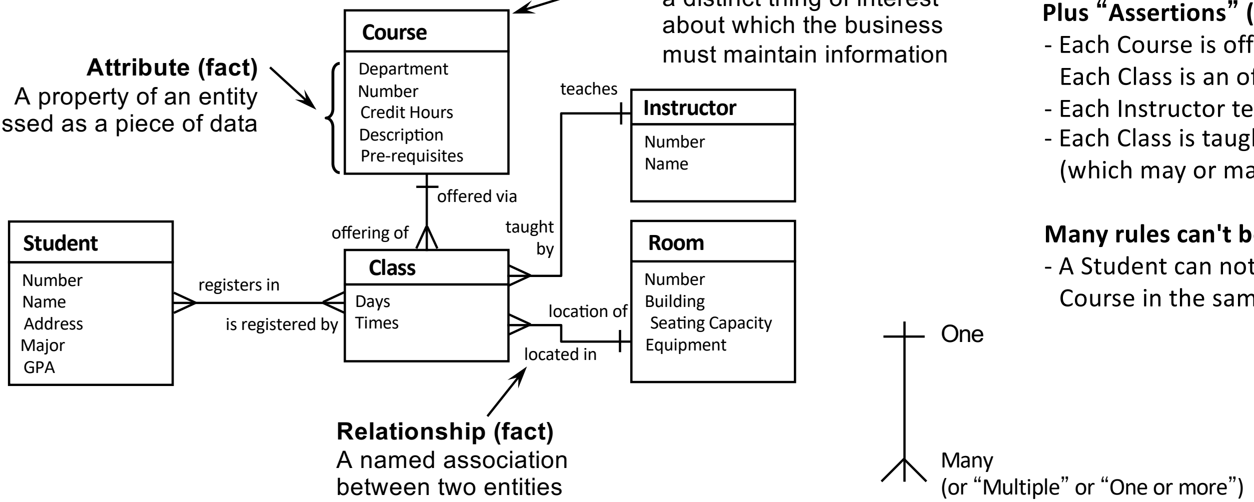
Smaller project: service or use case-oriented / “inside-out” –

What actually is a Concept Model / Data 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!

Graphic component



Entity (thing)
a distinct thing of interest about which the business must maintain information

Attribute (fact)
A property of an entity that can be expressed as a piece of data

Relationship (fact)
A named association between two entities

Narrative component

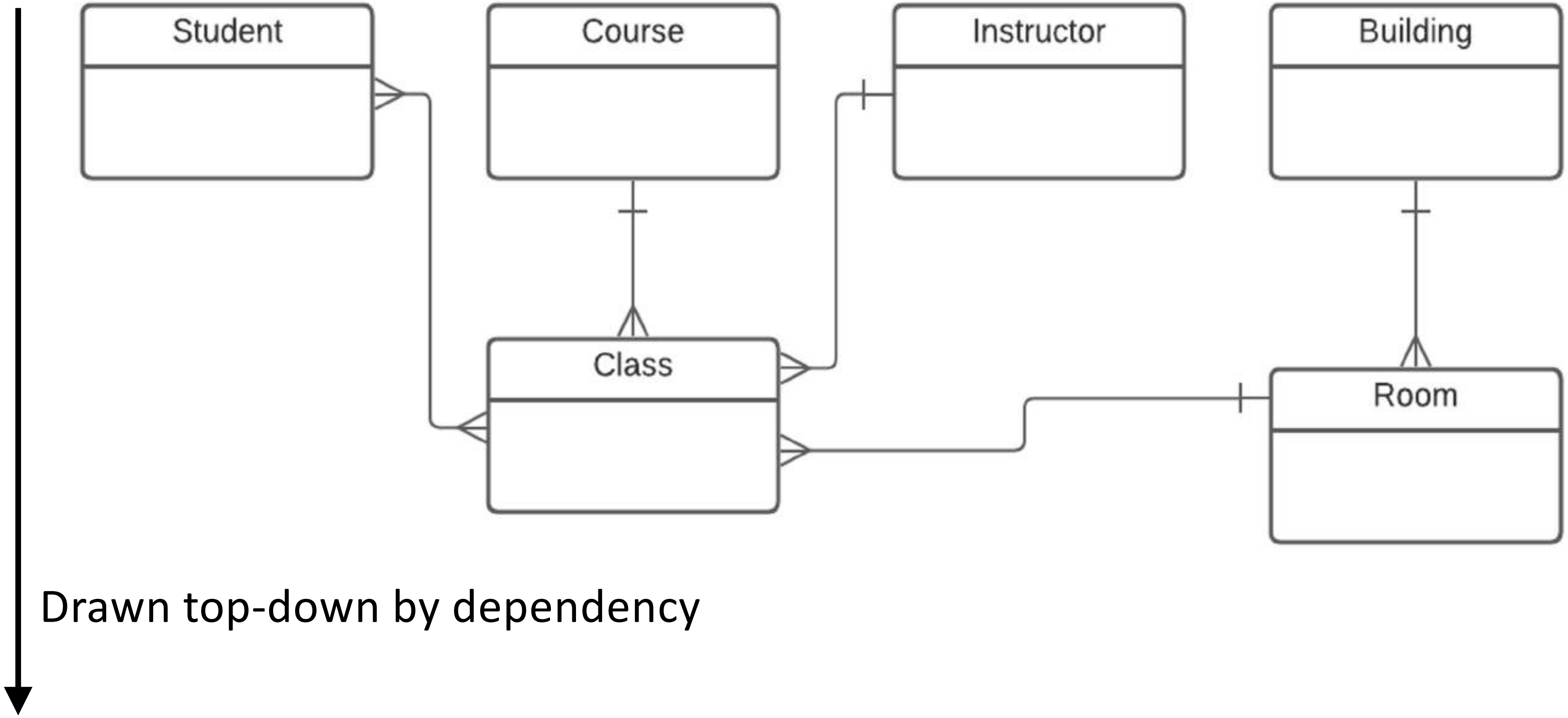
Student definition:
A Student is any person who has been admitted to the University, has accepted, and has enrolled in a course within a designated time. Faculty and staff members may also be Students

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

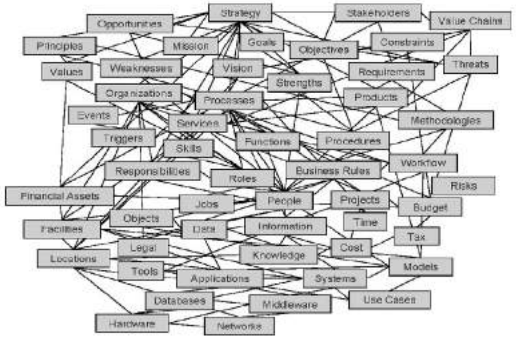
Independent Entities at the top



Drawn top-down by dependency

A few central ideas…

- Confusing *concept modelling* with detailed *database design* discourages the use of concept modelling



- We don't call it "data modelling" because, initially, “data” is not the issue – we model:
  - the things / objects / concepts the business cares about:
    - terms and definitions – ***language first!***
    - policies and rules
  - “things first, data later”

- A concept model provides a great platform for:
  - requirements discovery
  - package selection
  - business process change
  - business architecture, etc.

The basics: ERA – *Entities*

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

Criteria –

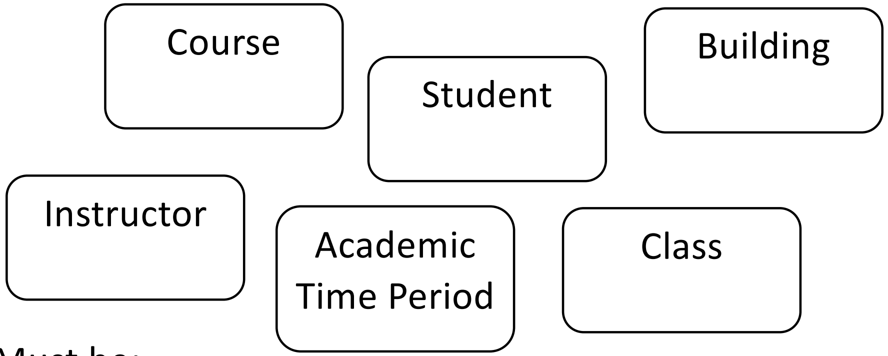
- *singular noun* – we can talk about *one of them* ("Employee," not "Staff")
- *multiple* instances
- must *need to* and be *able to* keep track of *each* instance
- has *facts* (attributes & relationships) that must be recorded
- makes sense in a *"verb-noun"* pair
- *NOT* an *artifact* like a spreadsheet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



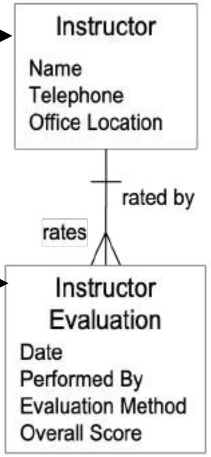
Must be:

- named: business-oriented noun / noun phrase

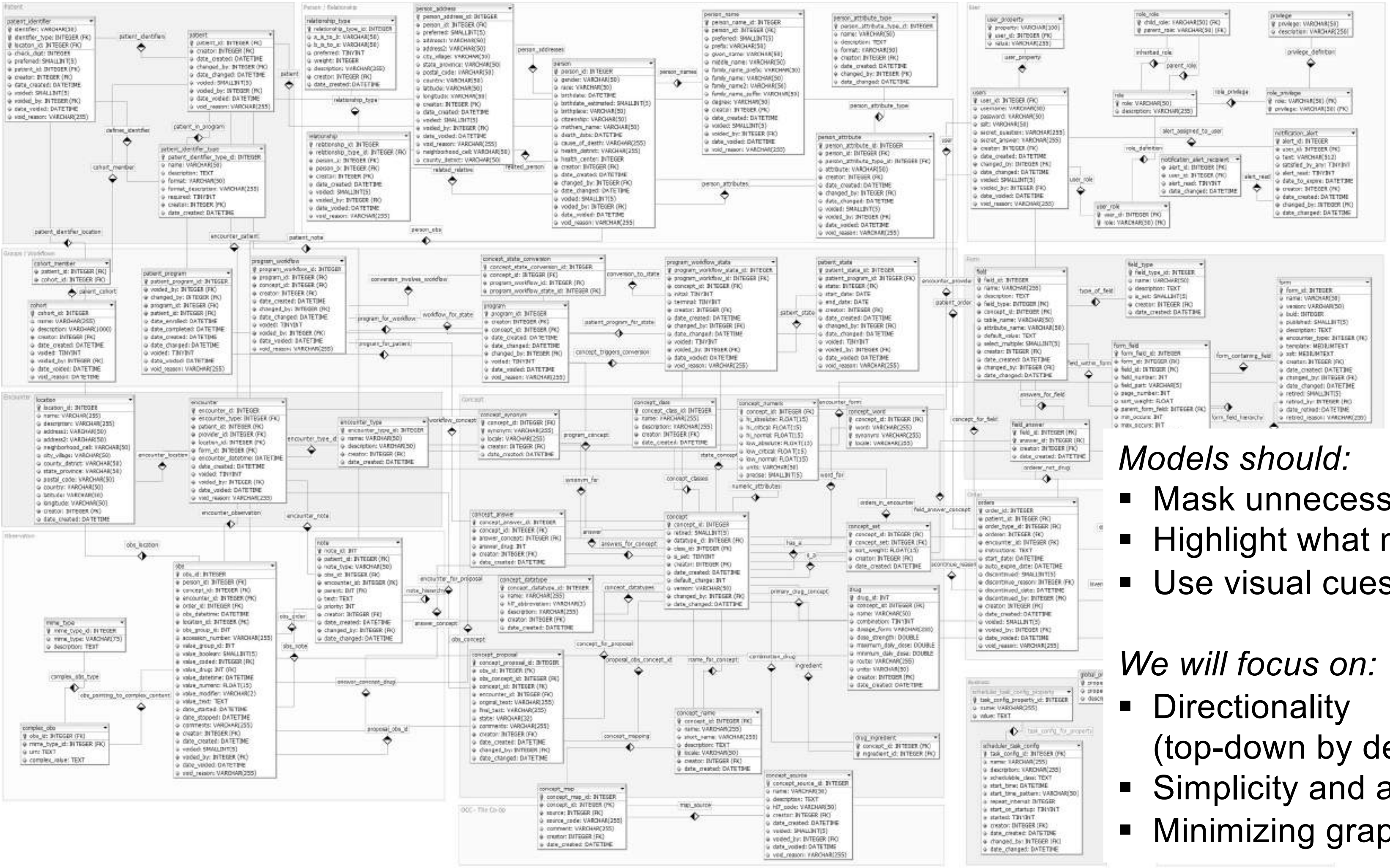
- defined: "What is one of these things?"
  or "What do you mean by ________?"

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

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



Entity-Relationship Modelling principles



Models should:

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

We will focus on:

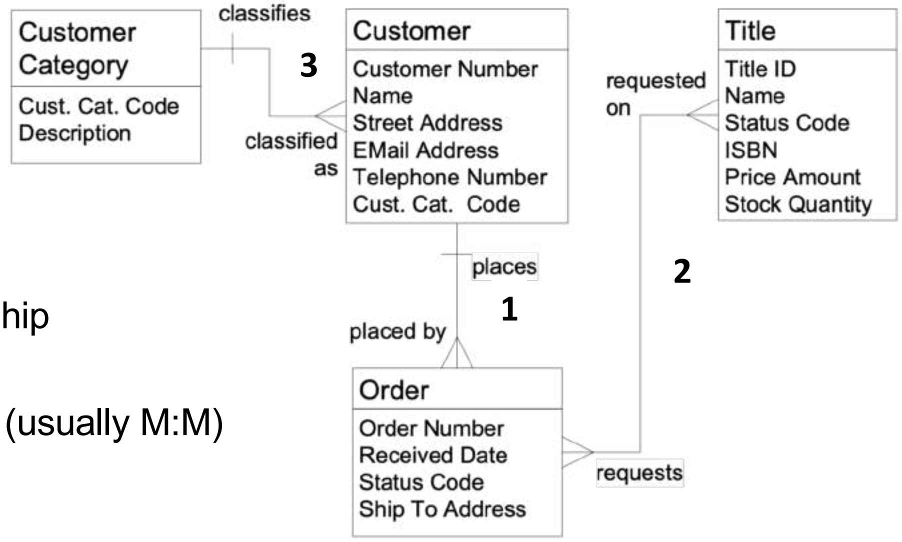
- Directionality
  (top-down by dependency)
- Simplicity and abstraction
- Minimizing graphic "widgets"

The basics – ERA – *Relationships*

An association between Entities that the business must keep track of

Named in both directions

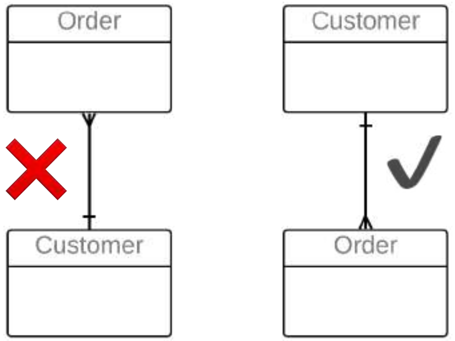
- verb-based phrase
- the line tells us they *are* related,
  the name tells us *how*



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

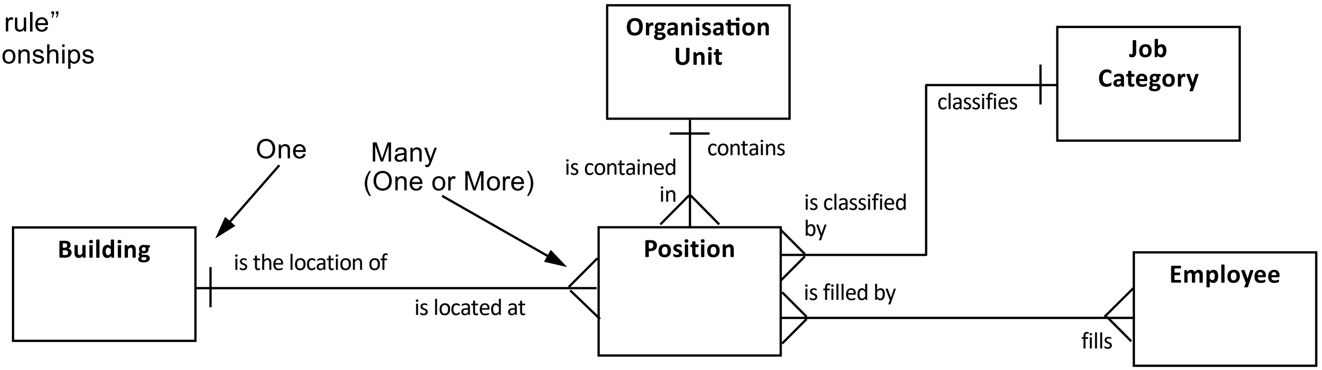


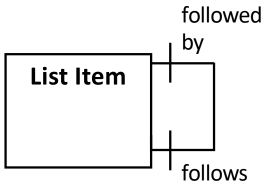
Relationships have rules

- 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:*

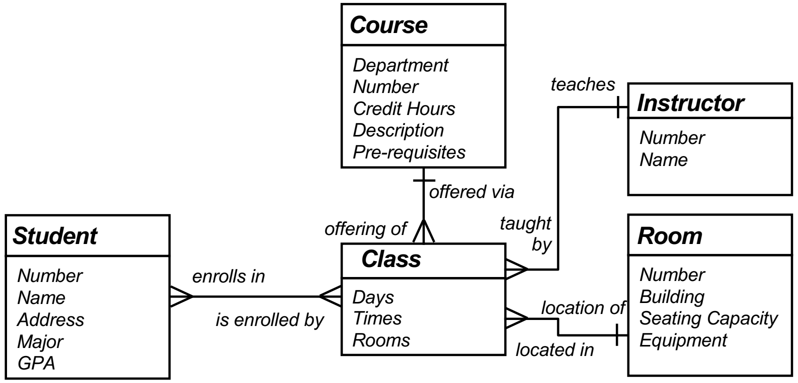
- for each entity, ask
  “Can one of these be related to a *maximum* of *One* of the other or a *maximum* of *Many* of the other?”

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

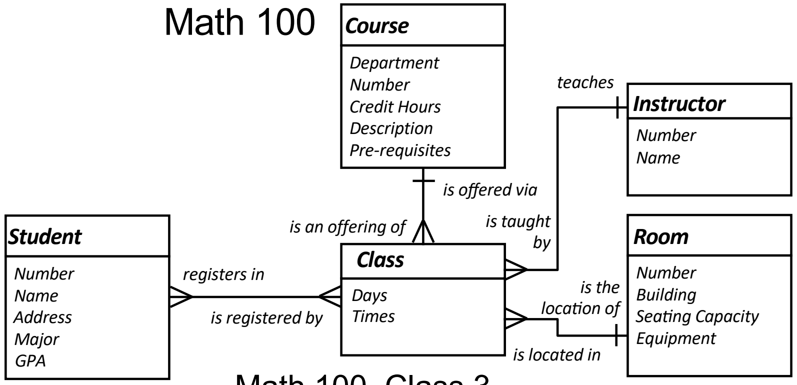
1. Student-Class
2. Course-Class
3. Instructor-Class
4. Room-Class

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.

Math 100



Math 100, Class 3,
Spring Semester 2022

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

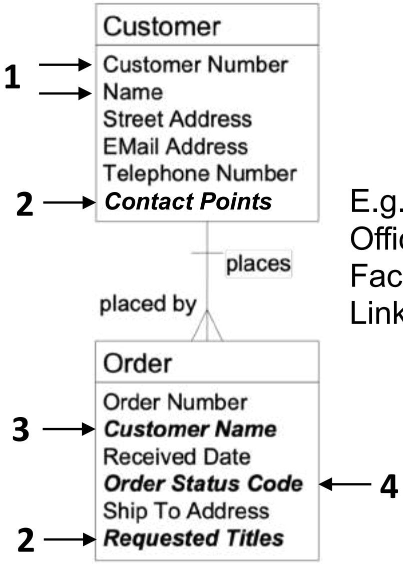
Have properties that we address during the transition from Concept Model to Logical Data Model

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, …

Summary – three types of data models

Different levels of detail support different perspectives

1 Contextual (Scope)

- ✓ *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

2 Conceptual (Overview)

- ✓ *Concept Model*
- ✓ Agreements on basic concepts, vocabulary, and rules

3 Logical (Detail)

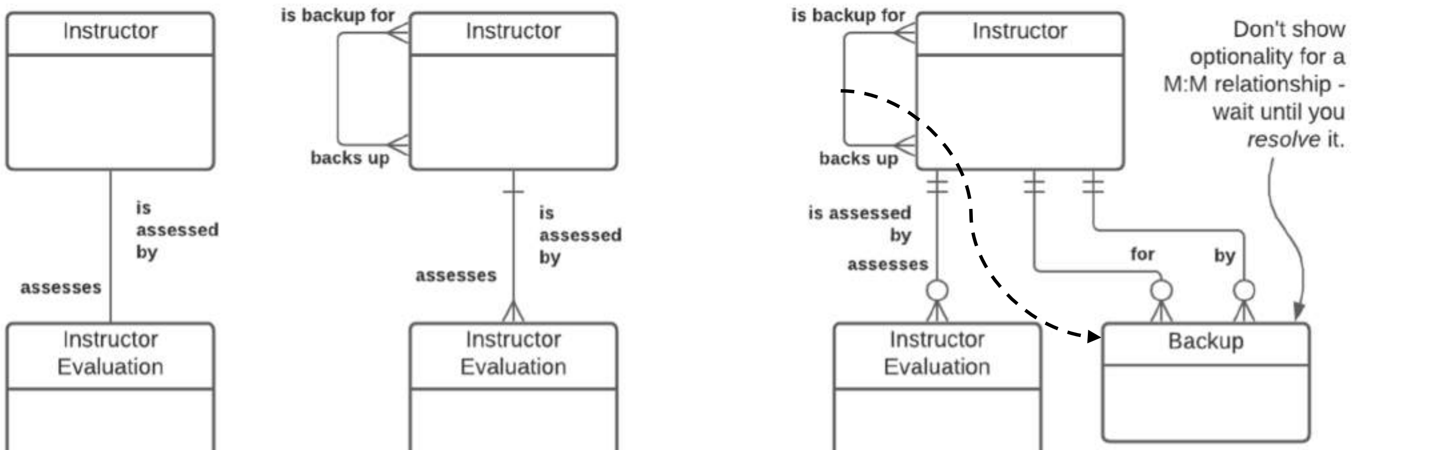
- ✓ *Logical Data Model*
- ✓ Complete detail for physical design

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

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.



Don't show optionality for a M:M relationship - wait until you *resolve* it.

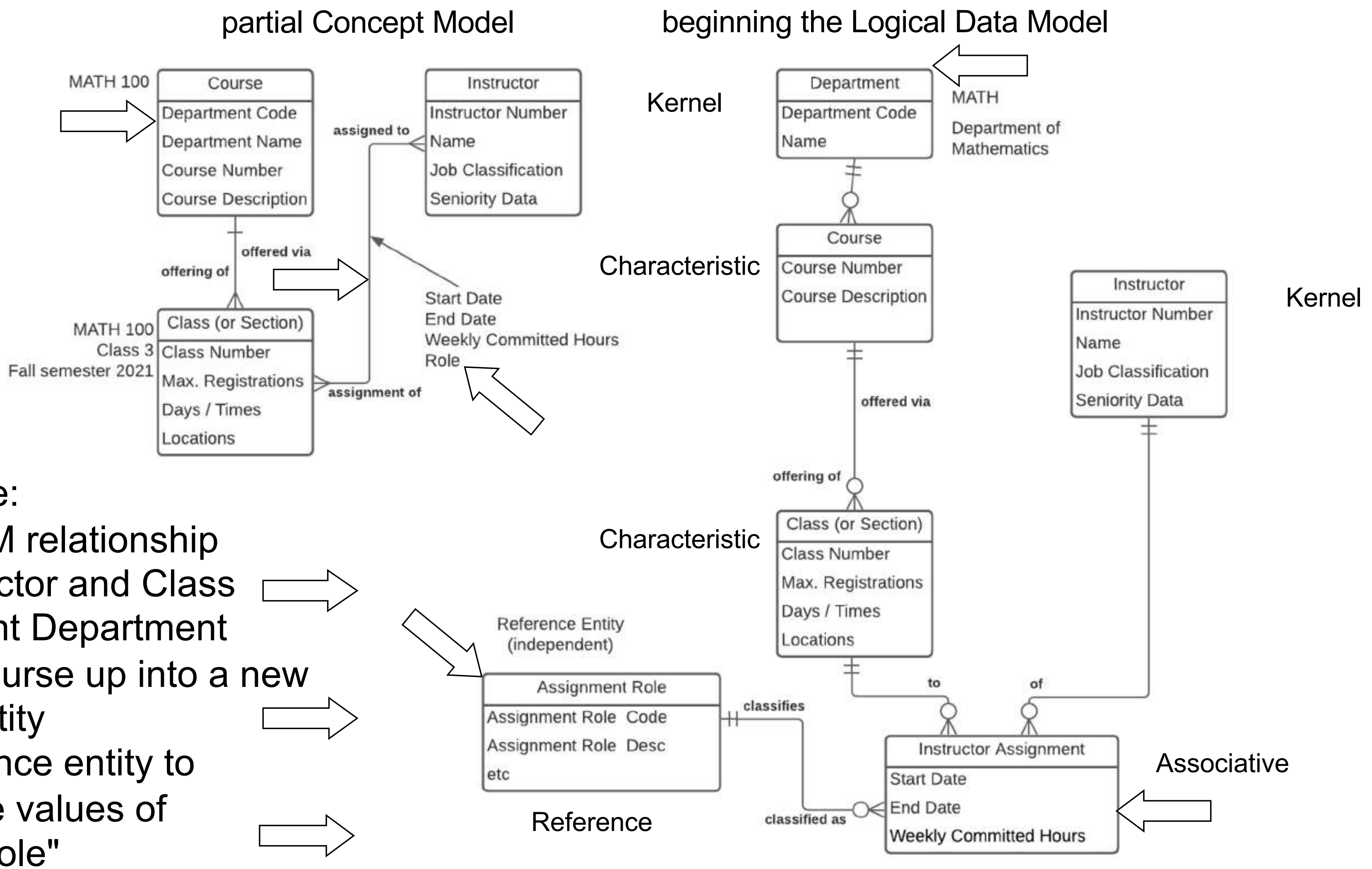
This is the associative entity that resolves the M:M relationship

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 quick example – from Concept Model to Logical Data Model



In this example we:

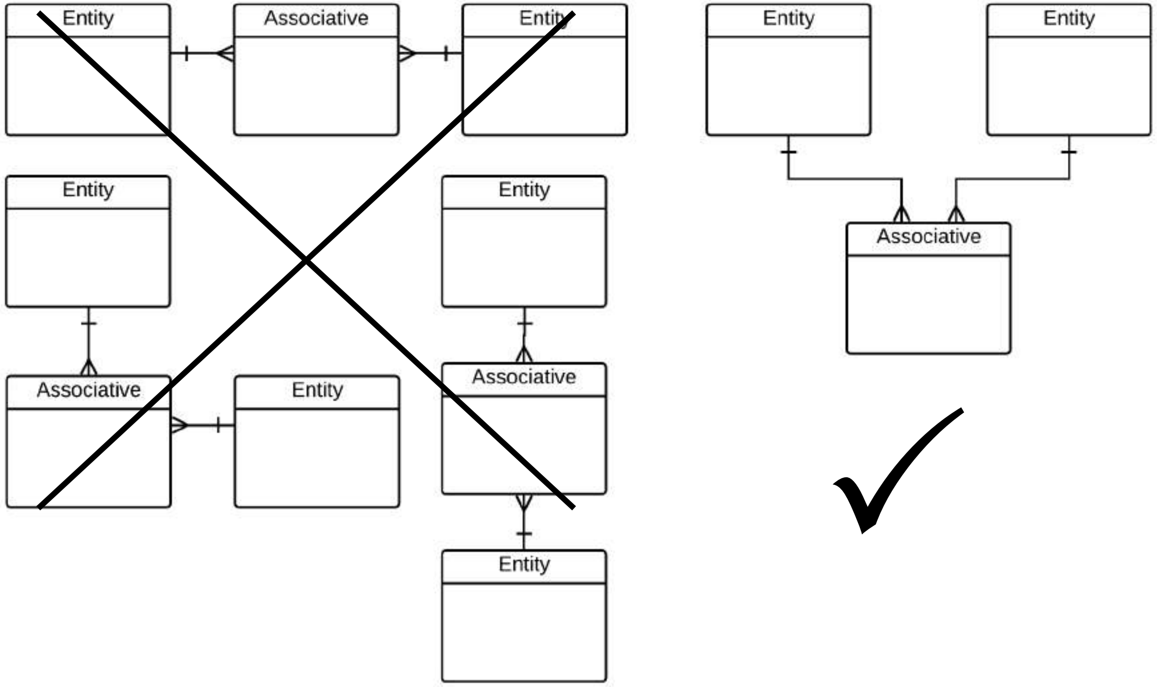
- resolve the M:M relationship between Instructor and Class
- move redundant Department attributes in Course up into a new Department entity
- create a reference entity to standardise the values of "Assignment Role"

Consistency is a virtue

People pick up data modelling without training if you…

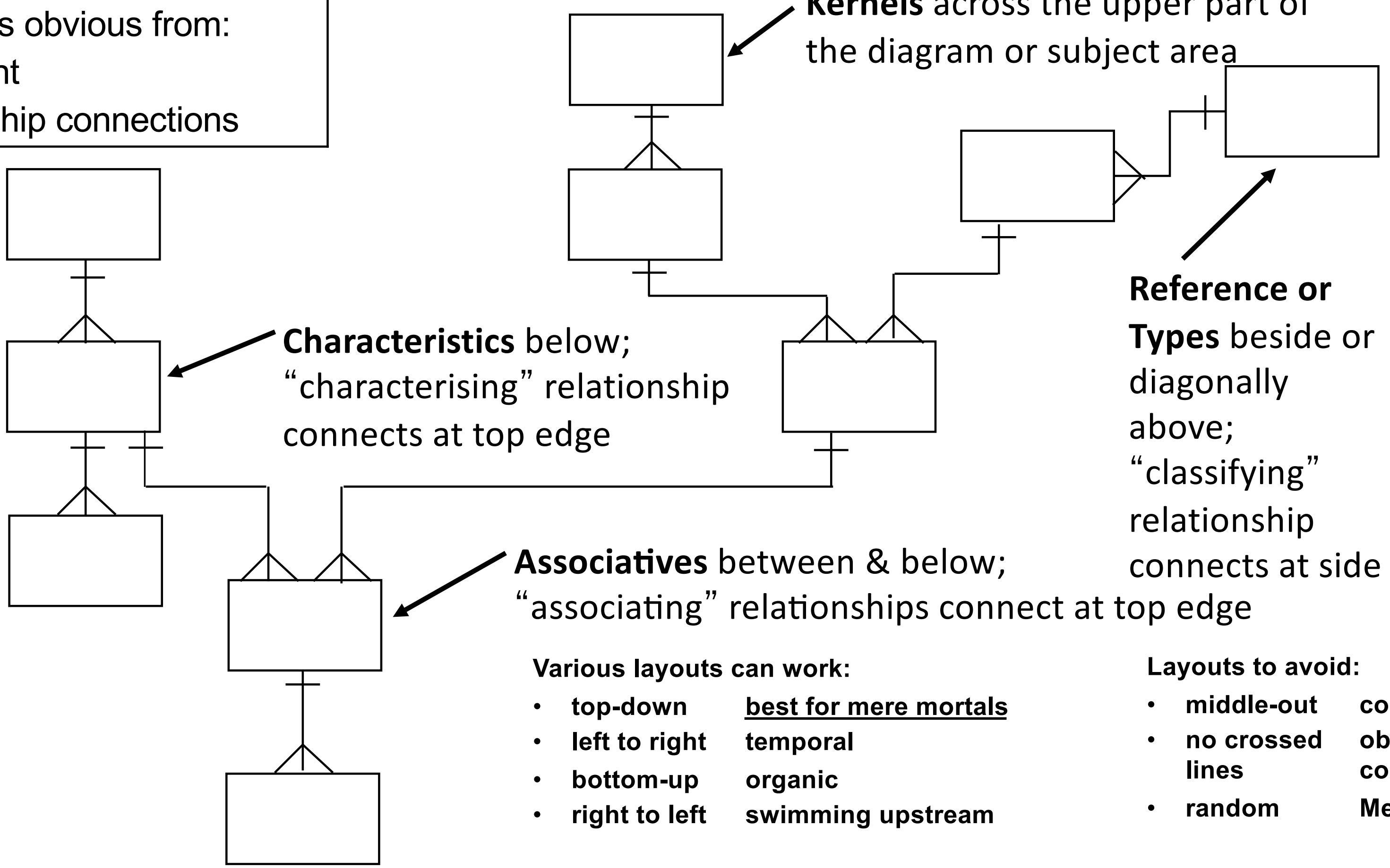
- treat it as a natural way to describe a business,
  not a new technique being imposed on them
- draw the same kinds of things the same way every time

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

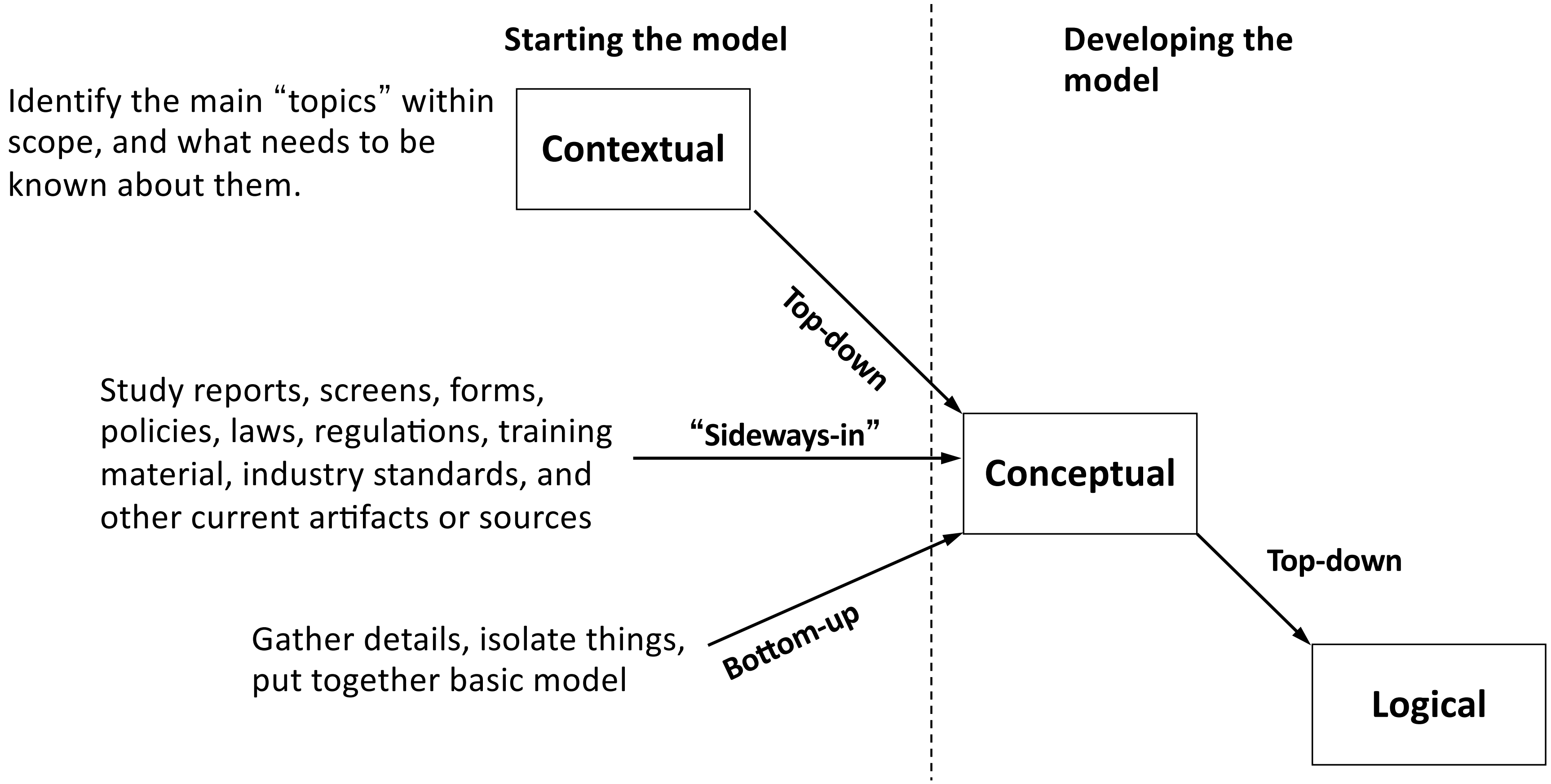
Various layouts can work:

• 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

Different ways to get started



Starting the model

Developing the model

Identify the main “topics” within scope, and what needs to be known about them.

Contextual

Top-down

Study reports, screens, forms, policies, laws, regulations, training material, industry standards, and other current artifacts or sources

“Sideways-in”

Conceptual

Top-down

Gather details, isolate things, put together basic model

Bottom-up

Logical

Some advice on starting the concept model

What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!
What they hear
blah blah GINGER blah blah blah blah blah blah blah blah blah GINGER blah blah blah blah blah...

__Don't__ 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 __nouns__ –
at the outset of every project.
This was demonstrated earlier in the Client Safety Management example.

For reference – 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"

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.

For reference – starting a concept data model bottom-up (exercise)

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.

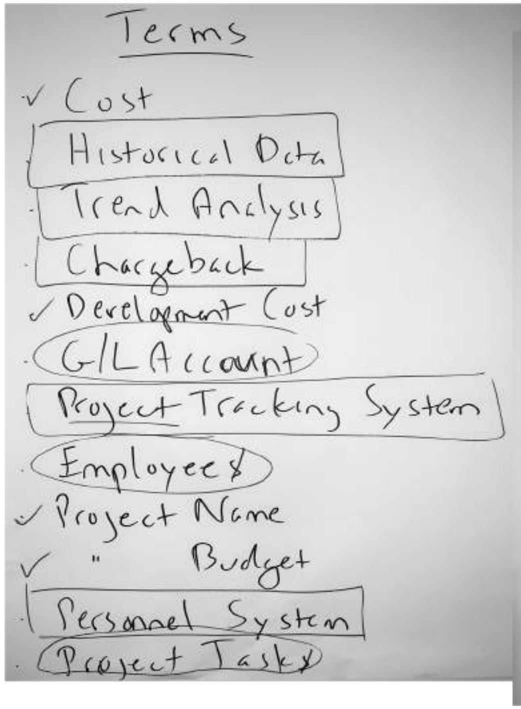
When a development project is initiated, a project name and a short description are recorded, among other things. Soon, before any further work is done on the project, a new account is created on the G/L System, identified by a G/L account number. Project costs will be charged to this account, and the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

Worked example from in-person two-day workshop



Terms

✓ Cost

Historical Data

Trend Analysis

Chargeback

✓ Development Cost

G/L Account

Project Tracking System

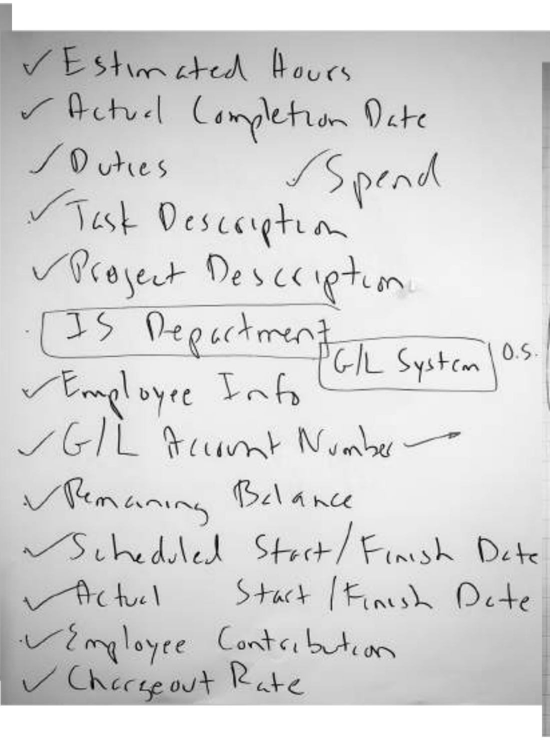
Employee~~s~~

✓ Project Name

✓ " Budget

Personnel System

Project Task~~s~~



✓ Estimated Hours

✓ Actual Completion Date

✓ Duties ✓ Spend

✓ Task Description

✓ Project Description

IS Department G/L System O.S.

✓ Employee Info

✓ G/L Account Number →

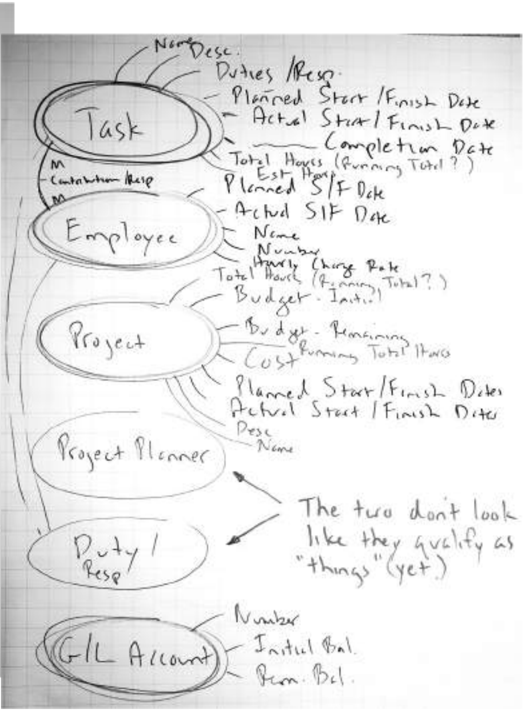
✓ Remaining Balance

✓ Scheduled Start/Finish Date

✓ Actual Start / Finish Date

✓ Employee Contribution

✓ Chargeout Rate



We have demonstrated there are four main entities (it's a very simple example)

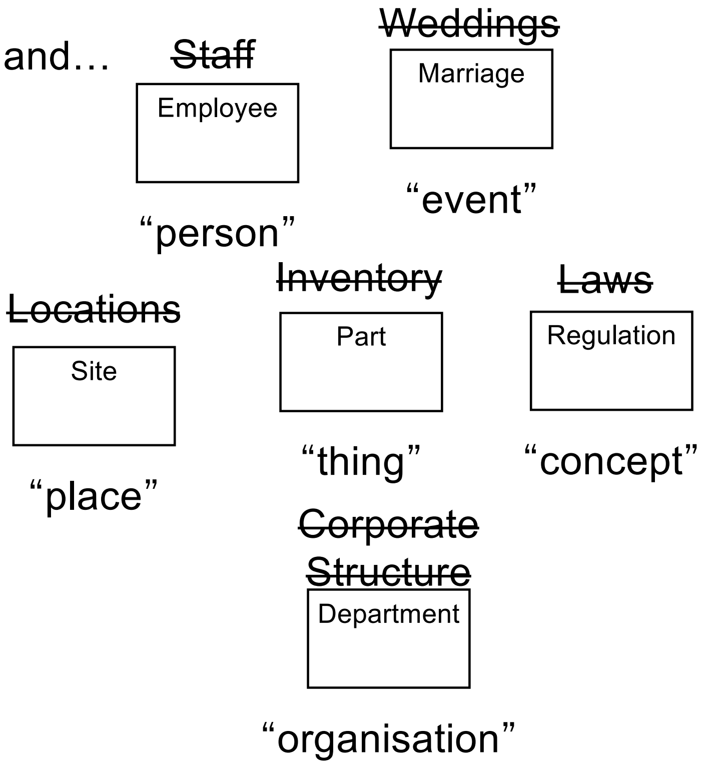
- Project
- Employee
- G/L Account
- Task

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
- *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…



~~Staff~~

Employee

"person"

~~Weddings~~

Marriage

"event"

~~Locations~~

Site

"place"

~~Inventory~~

Part

"thing"

~~Laws~~

Regulation

"concept"

~~Corporate Structure~~

Department

"organisation"

- 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"
- 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 – four common errors

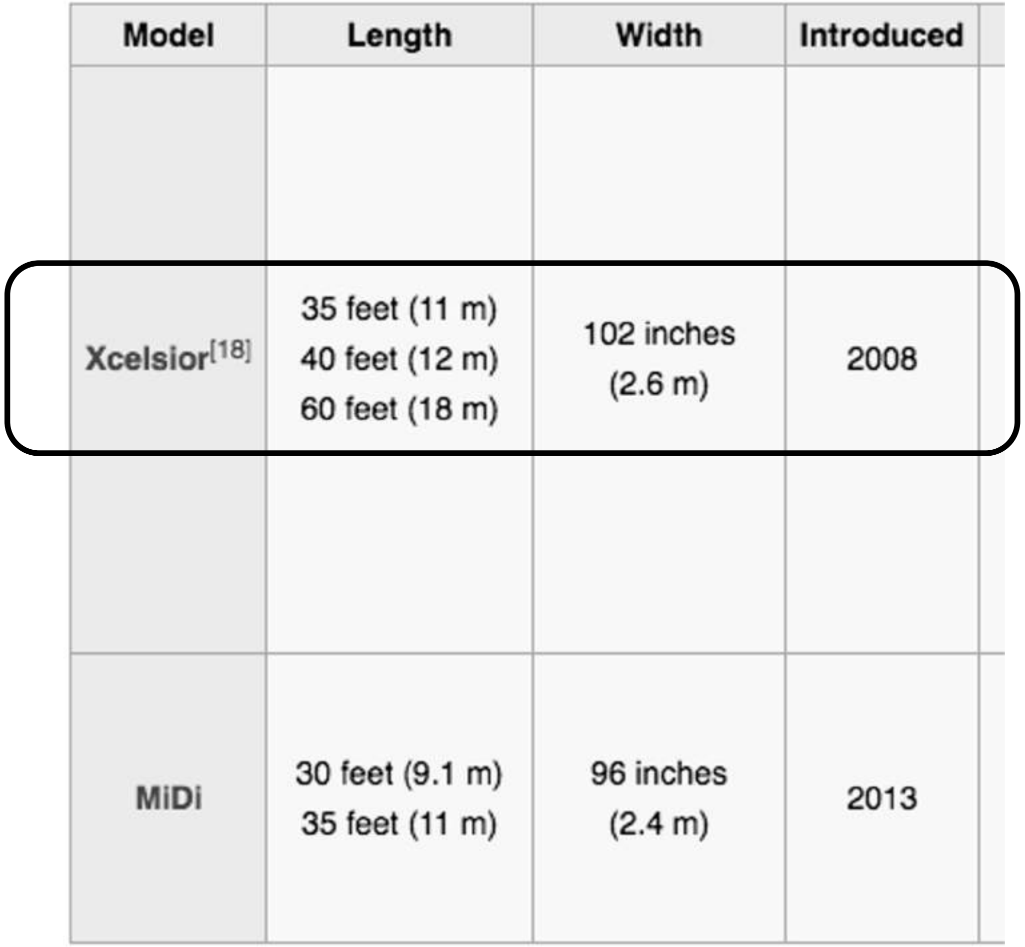
1. Treating an “artifact” (a spreadsheet, report, web page, form, etc.) as an Entity – an Entity is a fundamental thing – *“what”* – with no reference to *“who or how.”* Artifacts typically contain attributes from *multiple* Entities
   e.g., *“Admission Request* *<u>Form</u>”* or *“Orders Summary* *<u>Spreadsheet</u>”*
   or *“Daily Call* *<u>Log</u>”* or *“Class* *<u>Roster</u>”* *or “Materials List* *<u>Fax</u>”* 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., “*Vehicle”*
   (An example of this is coming up.)

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

4. Identifying Entities that are simply too vague, or are just a "fact of life;"
   that is, the name doesn‘t imply a single *instance*
   e.g., *“Weather”* or *"the Environment"* or *"the Economy"* or *"Society"*

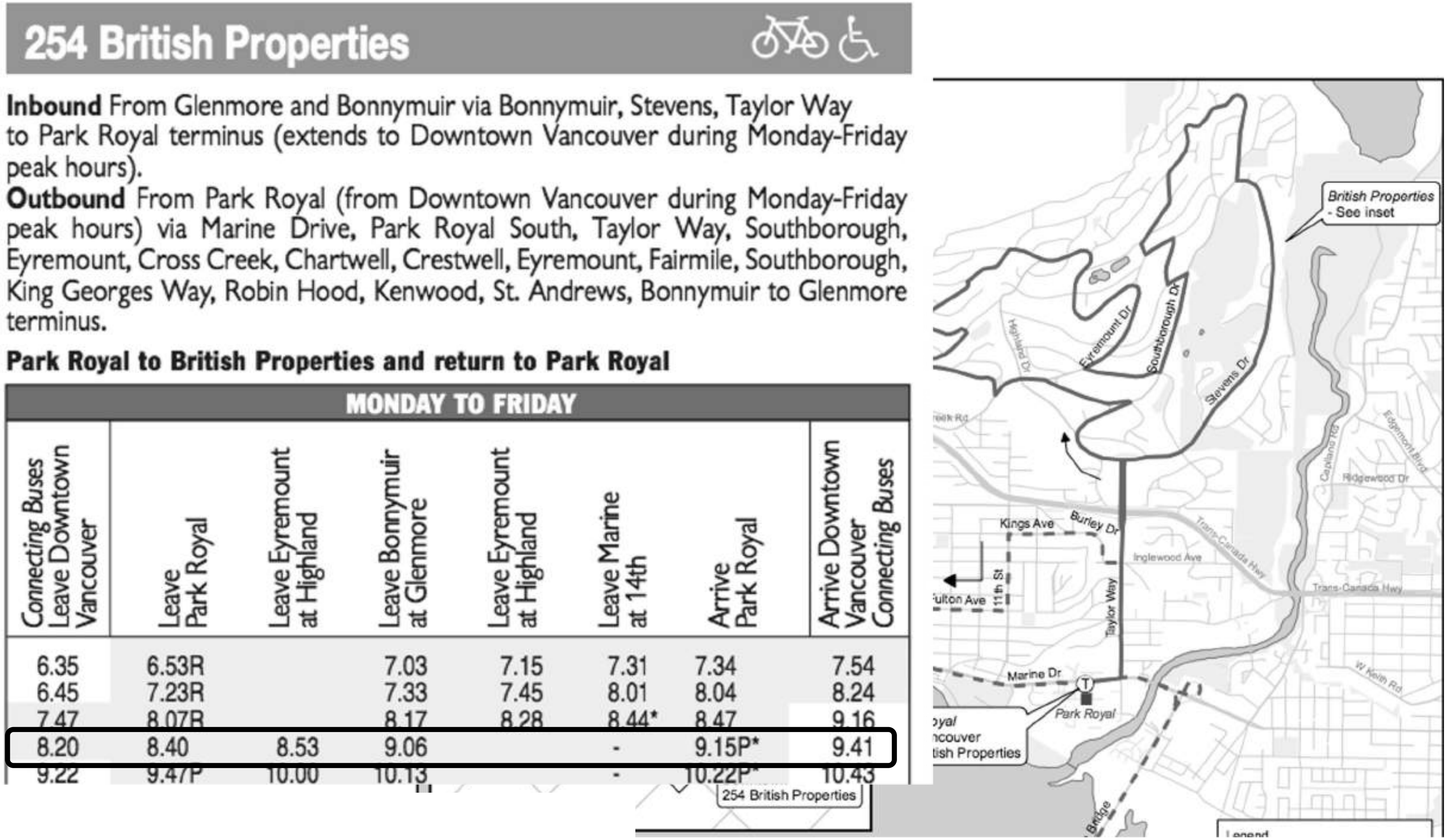
Types vs. Instances – “What do you mean by a Bus?”



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

A category of Bus – a "meta-Type?"
A Make and Model of Bus – a Type?
An individual Vehicle? – an Instance?

“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?

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, "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?”

- 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.~~

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...")

Customer

1. A Customer is a person or organisation that is a past, present, or potential user of our products or services.

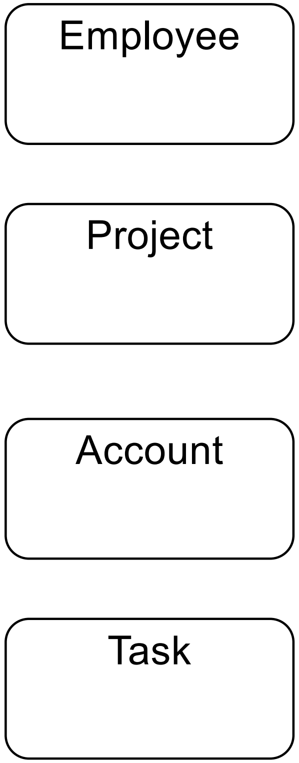
2. Current examples include Solectron (contract manufacturer,) Cisco Systems (OEM,) Arrow Electronics (distributor,) Best Buy (retailer,) M&P PCs (assembler,) and individual consumers.

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.)

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)?

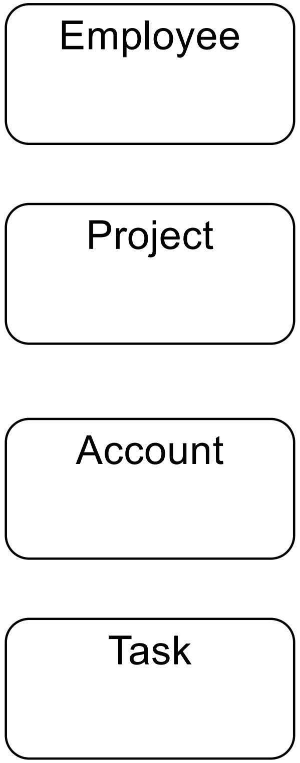
Onboarded? on probation? active? retirees?

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?



Employee

Project

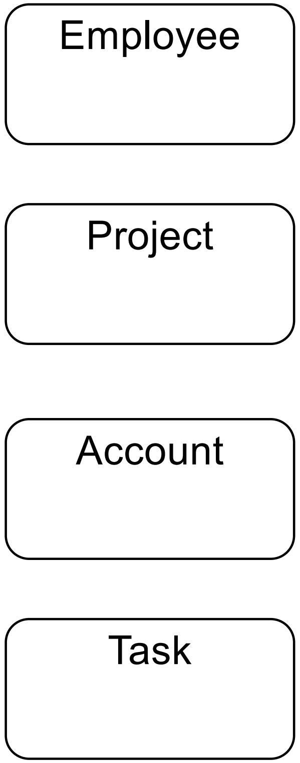
Account

Task

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:

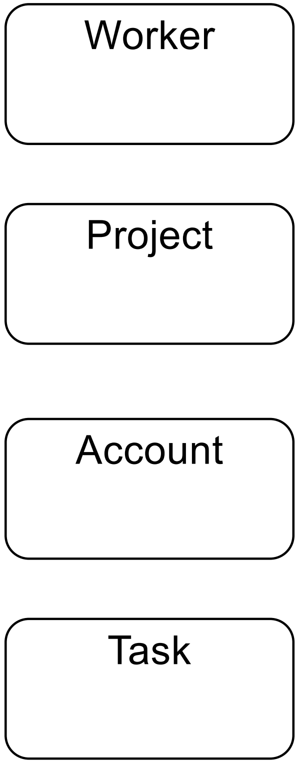
- Full or Part-time Employees who have been onboarded, including Probation, Active, Seconded, Suspended, Retired…
- Contractors
- Consultants
- Student Interns
- Vendor Staff Persons
- Company Owners and Managers

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?



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?
- …

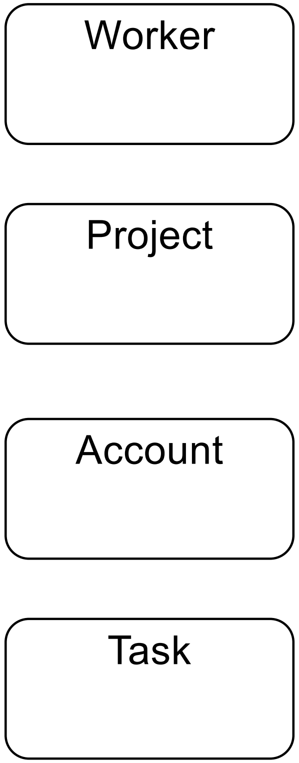
A ***Task*** is a specific, time-bounded, unit of work, within a single Project, intended to be performed by one or more Workers, that produces an intended deliverable or achieves a specific state.

Examples:

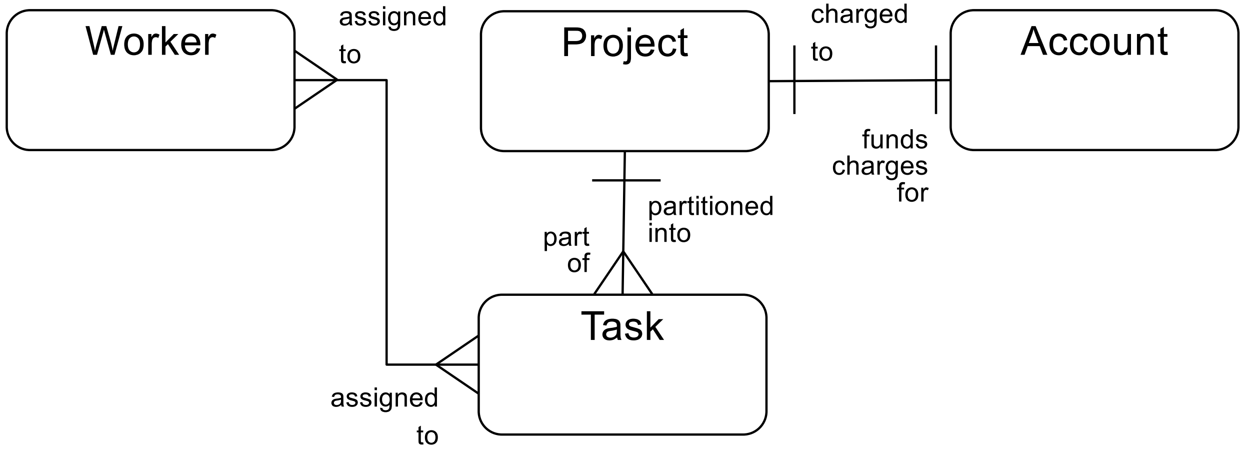
- Code *Place Order* service
- Test *Place Order* service

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



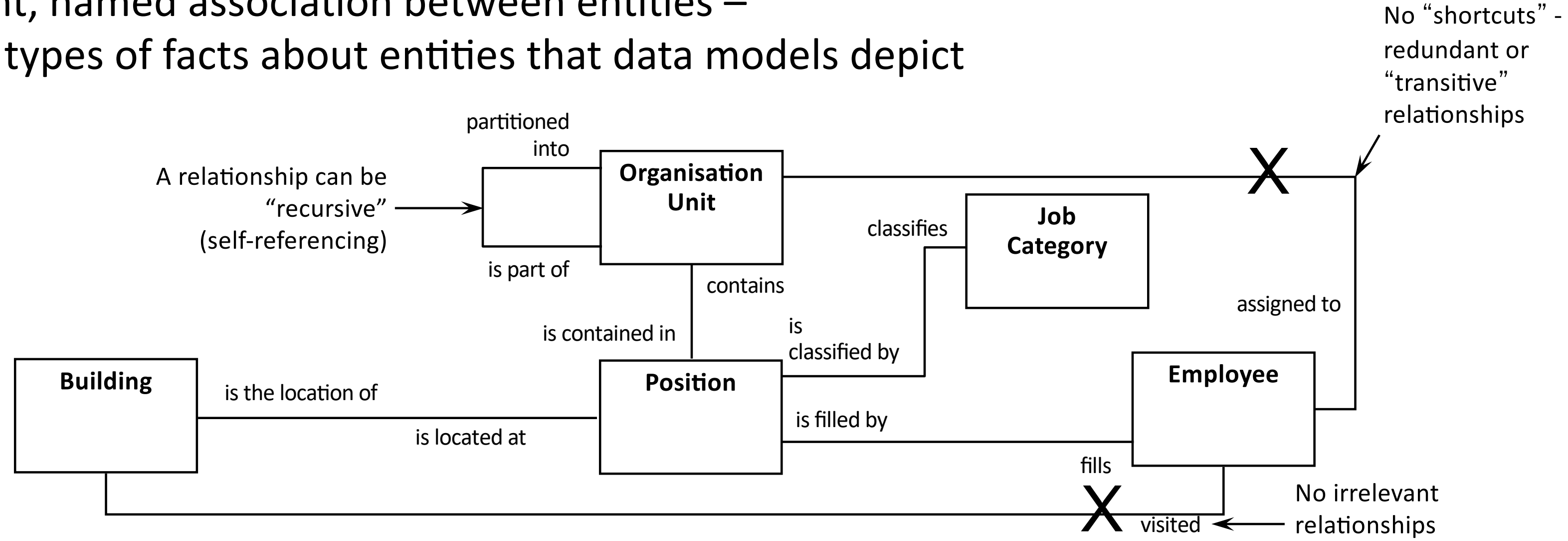
First arrange entities top-down by dependency.

Then add relationships with a verb-based phrase.

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

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



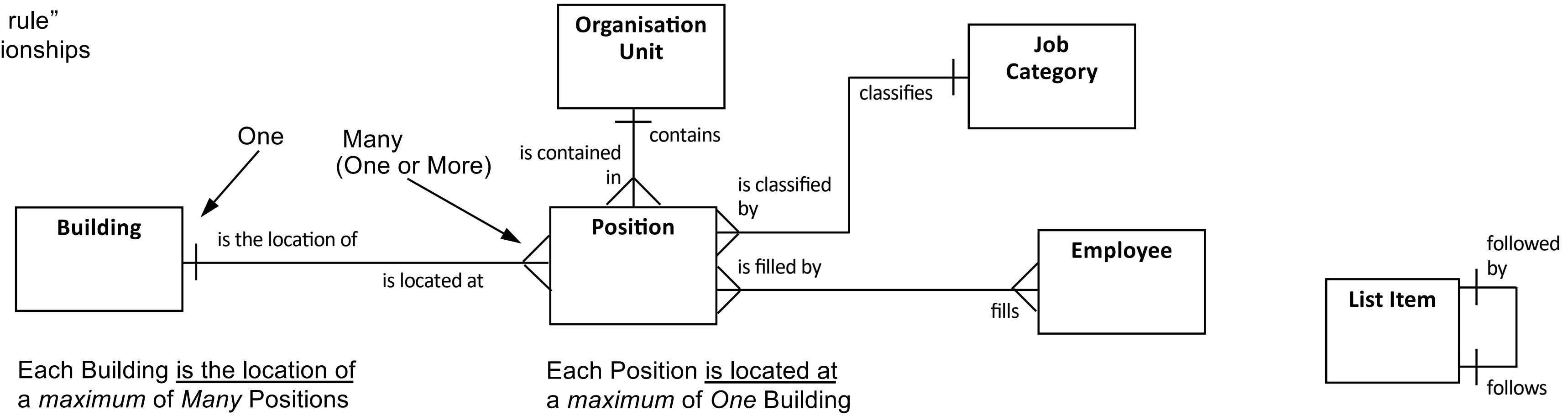
Guidelines

- 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 68

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:*

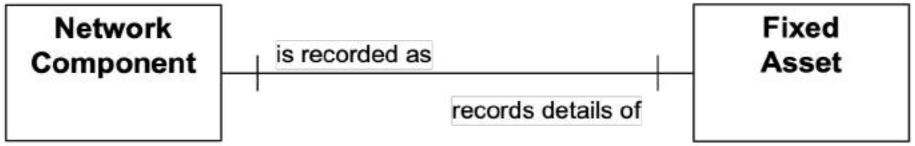
- for each entity, ask
  "Can one of these be related to a *maximum* of *One* of the other or a *maximum* of *Many* of the other?"

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

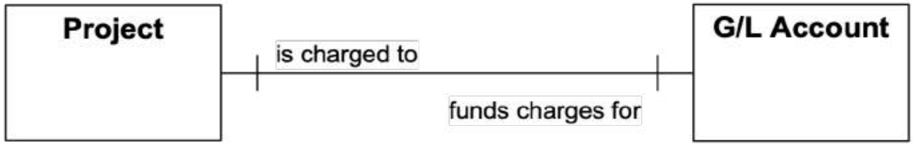
- possibilities – 1:1 (error), 1:M (common), M:M (more work, eventually)

1:1 relationships – almost always an error!

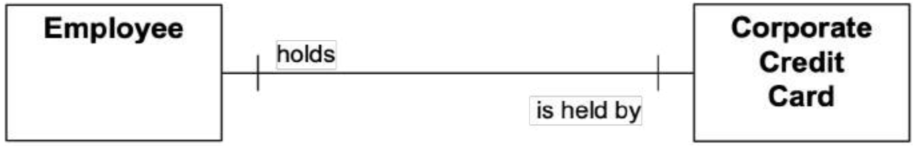
- Note – a 1:1 relationship might be necessary in the Physical Database Design
  e.g., "Fixed Asset" records financial data about a "Network Component" but they are in two separate systems (the G/L System and the Configuration Management System) connected by a 1:1 relationship



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

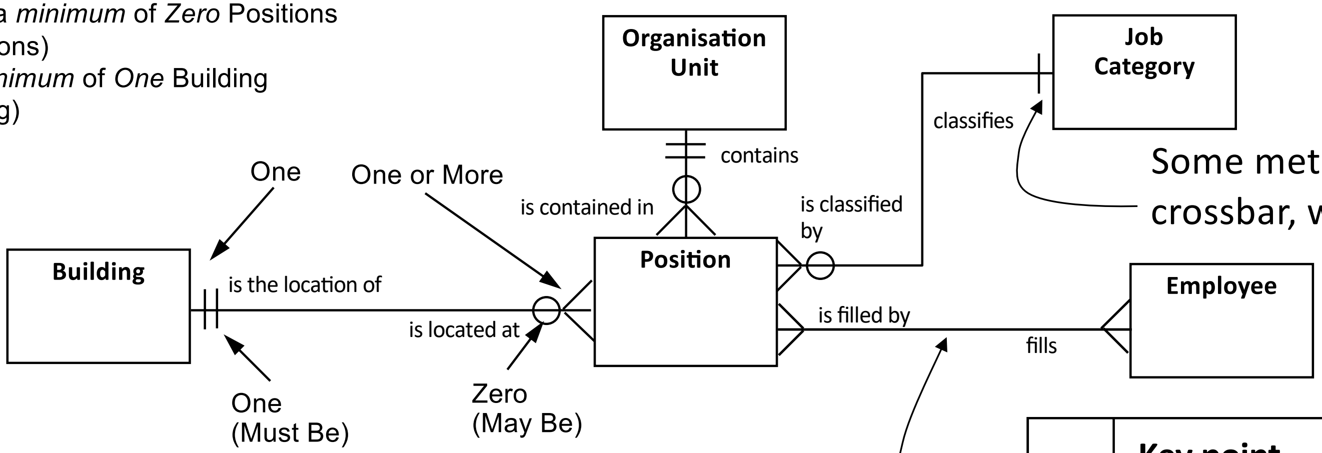


✘ Failing to account for changes over time
e.g., an Employee may hold only *one* Credit Card at a time, but *many over time,* and we virtually always want history.
The most common written constraint in Concept Modelling is
"one <u>at</u> a time but many <u>over</u> time."



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

Key point

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

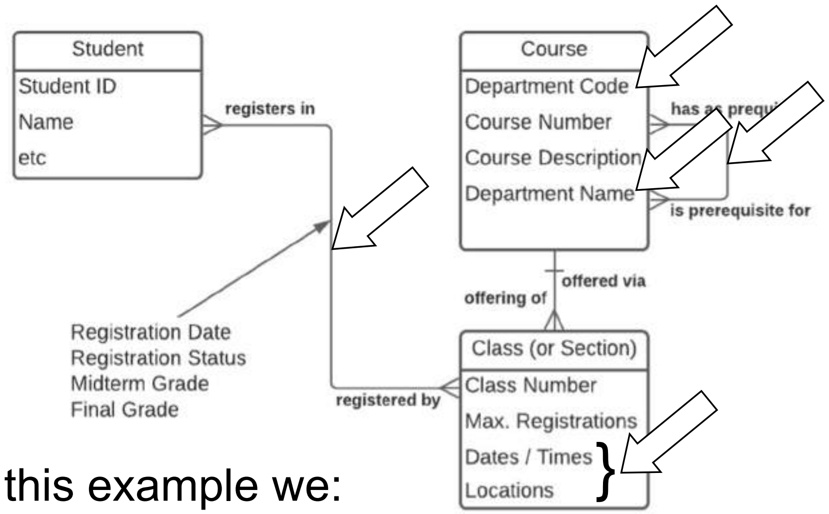
or,
Each Building *May Be* <u>the location of</u> *One or More* Positions
Each Position *Must Be* <u>located at</u> *One* Building

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

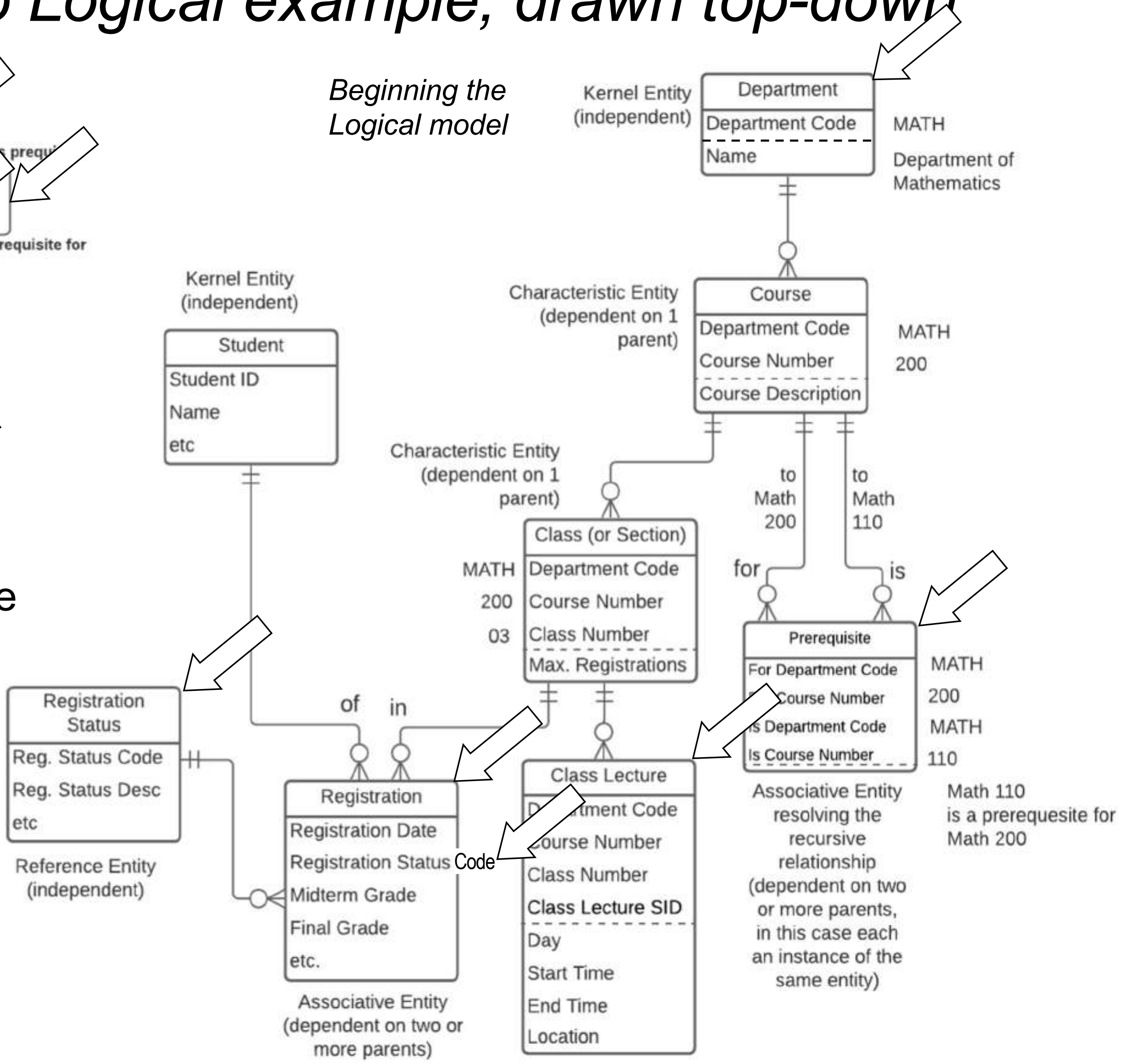
- for each entity, ask
  "Can one of these be related to a *minimum* of *Zero* <u>or</u> a *minimum* of *One* of the other entity?"
- record the answer – 0 <u>or</u> 1 – at the "other" end
  "zero" means an optional relationship (*May Be*) and "one" means a mandatory relationship (*Must Be*)
- easier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

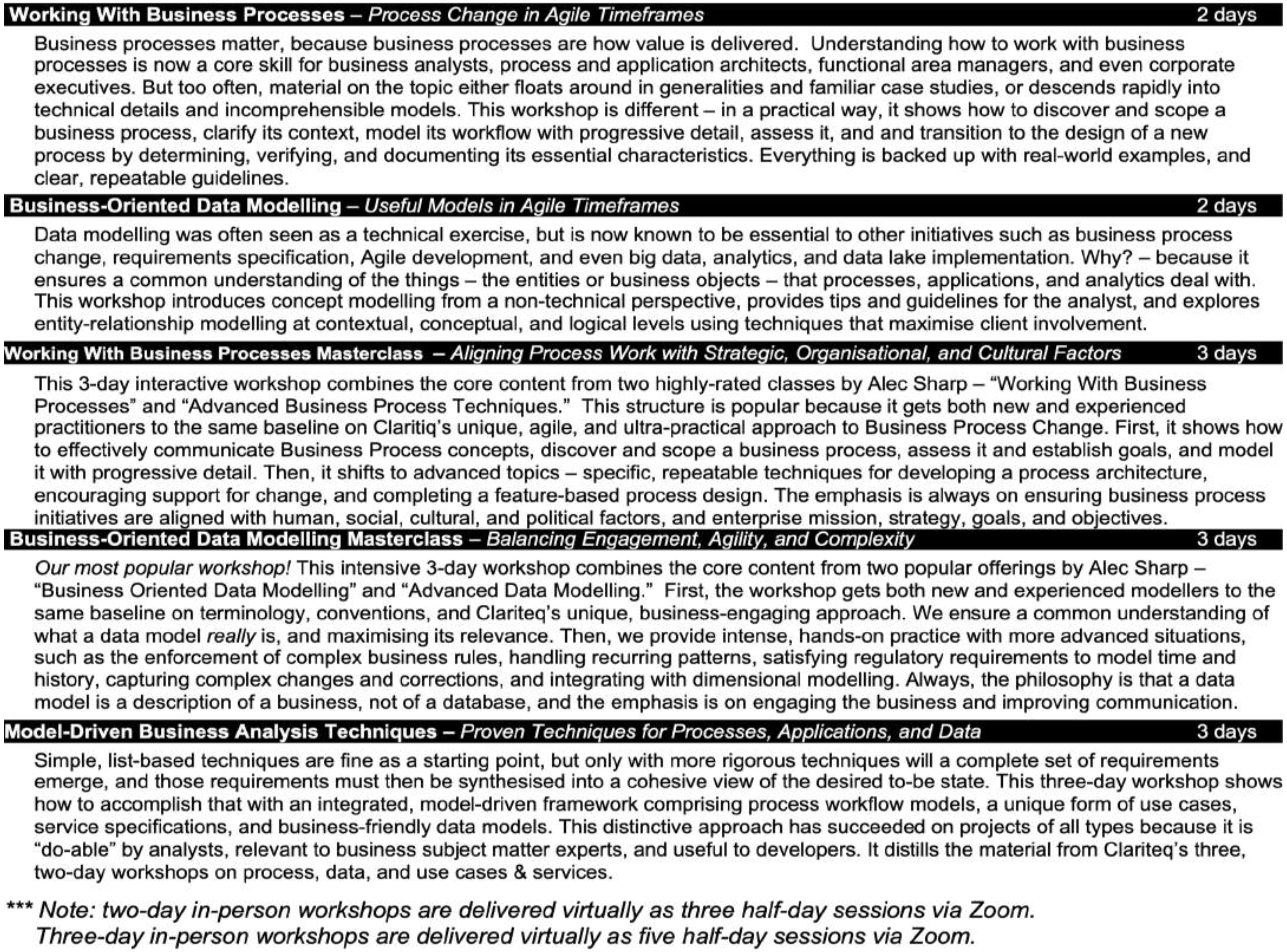
4 Demonstration

- 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.

*** *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.

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