The Lost Art of Conceptual Modelling
Alec Sharp,
Senior Consultant,
Clariteq Systems Consulting Ltd.
asharp@clariteq.com

Overview:
I’ve often spoken at IRMUK conferences about the apparent chasm in practice between architectural (or scoping or contextual) models and implementation (or specification or logical) models. An example was my “Mind the Gap!” presentation at the 2008 Enterprise Architecture Conference in London. This chasm is nicely filled by conceptual models, fleshing out the first three rows of the original Zachman Framework – contextual (the planner’s view,) conceptual (the owner’s view,) and logical (the designer’s view.) This is an important topic for architects because conceptual models establish a communication point between architects and analysts or designers. Unfortunately, this communication often doesn’t take place because our field has seen a loss of skills at conceptual modeling while confusion about what it means has risen. Using data modeling as an example, this article offers some background on the situation, some concrete guidelines on what a conceptual data model is, and some suggestions (with an example) on their use. I hope you enjoy it.

A few years ago, I was part of a panel discussion at a DAMA (Data Administration and Management Association) International Symposium somewhere in the United States—I think I must be travelling too much, because I can’t quite remember where it was. Anyway, the question for the panel was “what can you do to prevent your data management job from being outsourced?” This was a timely question, what with angst spreading over the IT landscape because of the very real prospect that your job might be outsourced to an offshore location.

Having spent a lot of time in Bangalore, a hotbed of offshoring, I had a pretty good idea of why certain jobs were being performed there. The conventional wisdom was that it was all about low cost, which was no doubt the prime factor for many companies, but I could see from the quality of the workforce that other factors would rise to the fore. On my first day of running a Data Modelling workshop there, I was, to put it mildly, blown away—I’d never encountered such a level of enthusiasm and engagement,
coupled with sheer intellectual horsepower. “Wow,” I thought “these folks are going to be tough competition.”

That doesn’t mean it’s a lost cause. Returning to the panel discussion, my suggestion was that very specific communication and engagement skills were going to be paramount, one element being to regain some of our lost skills at conceptual modelling. This, I offered, would be much more difficult if not impossible to outsource. Another panelist (I think it was my friend Michael Scofield) said essentially the same thing, pointing out that “there’s no substitute for being there” and “you can’t outsource empathy.”

From then on, the discussion was quite an eye-opener. The question “what do you mean by conceptual modelling?” came up, and the range of answers was startling. (That’s an embarrassing admission—after close to 30 years of consulting in the field, I should know that whenever you ask that most useful of questions, “what do you mean by…?” you are assured of a surprising range of answers.) Some people had a similar view to mine, which was that a conceptual model was a business-oriented, high-level view focusing on developing a shared understanding of the central concepts in an area. Hence, “conceptual modelling” with an emphasis on terms and definitions, important business constraints, and an avoidance of technical concerns like keys and normalisation. Others, though, felt that a conceptual data model was a fully normalised, fully detailed depiction of data requirements ready for transformation into a physical database design. It might even include some design decisions, such as selective denormalisation or an approach to handling generalisation, as long as there was nothing specific to a particular DBMS. I asked how that was different than a logical data model, and one answer was “There isn’t any difference.” Yikes! Of course, there were other answers, including the idea that they were at the same level of detail, but the conceptual model was a “pure” (read that as “unsullied”) representation of the data, while the logical model included some of the aforementioned design decisions.

My head was swimming, but the best was yet to come. We got into the topic of how our skills at (and, evidently, awareness of) building business-oriented models had declined. There are many factors I could have pointed out, including:

- At colleges and universities, data modelling, if it is treated at all, is most often treated as equivalent to database design;
- Standards like UML (and BPMN on the business process side) have emerged that are very implementation-oriented but have the magic
word “modelling” in their name, so people to use them for purposes they weren’t intended for, like business analysis;

- The lure of instant gratification from purchased applications and various forms of rapid development that seem to eliminate the need for modelling altogether. (Please note that I do a lot of work with package implementation, and with Agile development teams, so I don’t have any negative bias here; the point is that an appropriate amount of modelling is just as useful as ever.)

In this case, I said (and many agreed) that one root cause was the tools. The situation was improving, but I pointed out that most data modelling tools had really been thinly-disguised physical database design tools geared more to making the generation of DDL easier than to supporting the work of the analyst/modeler. On this, there was general agreement, except (not surprisingly) from some of the tool vendors who were present! One stood up and offered that their company’s tool certainly supported conceptual modelling. Immediately, they were challenged by another attendee who said they had been a long-time user of this suite of tools, and it most certainly did not support conceptual modelling. After some back-and-forth on this, the “challenger” asked the vendor “What, exactly, do you mean when you say that “tool X” supports conceptual modelling?” (There’s that question again – “what do you mean by …?”) The answer: “We can print a data model diagram without showing the attributes!” The stunned reply: “You mean if I print out an 800 entity data model, but without showing the attributes, that’s what you mean by a conceptual model?” “Yes.”

I didn’t know whether to laugh or cry, but I’m certainly grateful I was there. As I said, it was a real eye-opener. What does it mean, I wondered, when at the world’s largest gathering of data management professionals there is such divergence on the meaning of an important term like “conceptual?” That question is even more surprising when you realize how many of the participants employed John Zachman’s “Framework for Enterprise Architecture.” Earlier versions of that explicitly used the subtitles contextual, conceptual, and logical to describe, respectively, the first three rows of the framework: the planner’s view, the owner’s view, and the designer’s view. The current version simply calls them “scope contexts,” “business concepts,” and “system logic.” In any case, many people had a definition of “conceptual” that was much closer to what John’s framework would describe as “logical.” Is it any wonder that so many data management professionals and architects complain about difficulties in communicating with other stakeholders?
At this point, I encourage you to stop reading for a moment, and think about a couple of questions.

1. What does conceptual data modelling mean to you, and to others at your organisation? Could they have the wildly different interpretations in evidence at that DAMA conference?

2. How do you, or could you, use conceptual data models?

Actually, do more than just think about it—write your answers down. (I’ll wait here while you tackle those questions.)

Welcome back. The point of all this is to encourage everyone to be more precise about what is meant by terms like “logical” and “conceptual,” and I’ll close this article by offering some of my thoughts. Your ideas might be different, but my purpose isn’t to get you to agree with my ideas, it’s just to have you think about a topic that’s so fundamental we often gloss over it.

In defining a “conceptual data model” one of the challenges is that we know that eventually databases have to be designed, and in that realm properties such as normalisation, primary and foreign keys, and atomicity of attributes are crucial. The problem is that they’re irrelevant in conceptual modelling, but many people find it very difficult to draw a data model that doesn’t display these properties. Certainly when I started doing conceptual modelling I still had my DBA mindset and had to learn to continually “grab myself by the scruff of the neck” and pull myself out of the weeds of detail. It took a lot of effort, and I know it’s difficult for others. To illustrate this point, in my Advanced Data Modelling workshop I often show a fragment of a conceptual model like the following, and ask the group “what bothers you about this, or what would bother a more detail-oriented modeler?” (Note that a conceptual model is certainly going to have more than 2 entities in it, perhaps 15, 20, or even 25, but this is enough to make the point.)

<table>
<thead>
<tr>
<th>Section</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code (Dept.)</td>
<td>Student Number</td>
</tr>
<tr>
<td>Course Number</td>
<td>Name</td>
</tr>
<tr>
<td>Course Name</td>
<td>Addresses and Types</td>
</tr>
<tr>
<td>Section Number</td>
<td>Admission Date</td>
</tr>
<tr>
<td>Lecture Days</td>
<td>Admission Category</td>
</tr>
<tr>
<td>Lecture Times</td>
<td></td>
</tr>
<tr>
<td>Lecture Room Number</td>
<td></td>
</tr>
</tbody>
</table>
The answers include:

- Missing entities because these two are unnormalised (e.g., Department, Course, Lecture Day-Time, Building, Room, etc.)
- Redundant attributes (e.g., about Course)
- Multivalued attributes (e.g., Addresses, or Lecture Days and Times)
- Non-atomic attributes (e.g., Name and Address)
- No primary, foreign, or candidate keys are indicated
- An unresolved M:M relationship where there should be an associative entity
- No optionality shown on the relationship
- Missing reference entities (e.g., Address Type)
- The ungeneralised Student entity could be a Person entity
- …and various others

These could all be problematic if this was a completed, logical data model, but they are perfectly okay in a conceptual model, especially one that is still under development. This conceptual model would be extended to include the Course entity (as a parent of Section) but not much more. As it is, it satisfies the main purpose of a conceptual model—to illustrate the important things we are interested in, what we call them, and the main facts we want to know about them. Remember, the focus in a conceptual model should be much more on the things than all the data we eventually need to record about these things.

To summarise, here’s a table illustrating how I differentiate conceptual and logical data models. Please keep in mind that most books and courses on data modelling don’t say anything specific about the differences—you might not agree with my guidelines, but at least they’re a starting point.
### Conceptual Data Model | Logical Data Model
---|---
Ensures that everyone is on the same wavelength before diving into the details | Provides all detail for first-cut physical database design and requirements specification
Overview: *main* entities, attributes, and relationships | Detail: approximately 5 times as many entities as the conceptual model
Lots of M:M relationships | All M:M relationships resolved
Relationships show multiplicity | Relationship optionality added
No keys | Primary, foreign, alternate keys
No reference ("lookup") entities except a few critical ones that “tie together” parts of the model | Lots of reference entities
Many attributes will be non-atomic and multi-valued | Fully normalised – no multi-valued, redundant, or non-atomic attributes. All attributes defined and properties described
Verified by direct inspection (client sees the model) plus narrative assertions, scenarios, etc. | May be verified by other means: sample data, report mockups, UI layouts, etc.
A “one-pager” | Likely to be partitioned into multiple diagrams by subject area, business process, or other.
20% of the modelling effort | 80% of the modelling effort

The other question I asked you to think about was “How could you use a conceptual data model?” The obvious answer is that as a data modeler on a project you can use it between initial discussions and final design to ensure that all participants—business subject matter experts, business analysts/data modelers, enterprise architects, and technical resources—share the same understanding (concepts) of the area being modeled. You can then add detail, precision, and rigor, eventually completing a logical data model that will be the foundation for a stable database design. As an enterprise data architect, you can use conceptual level models as the “bridge” between the contextual data models (“scope contexts”) that make up the highest levels of your enterprise data architecture, and the logical data models (“system
logic”) that precede implementation. This establishes a mechanism for better alignment between the two.

The problem is that you might not have the opportunity to do this, because there is less and less “green field” (starting with a blank whiteboard) modelling being done. Either of the following might describe your situation:

- “We just don’t have the time and the support to start new modelling initiatives;”
- “Our operational systems are a mix of legacy and purchased applications, so modelling doesn’t seem appropriate.”

In these situations, one of the most useful modelling activities is to get into the existing files and databases and reverse engineer a conceptual model out of them. This can then be shared with the business to illustrate more clearly the strengths and limitations of a current or proposed system.

In one memorable case, a company had purchased an application for tracking employment candidates, which is more complex than you might think, especially due to federal reporting requirements. After spending ten times the purchase price of the application on customisation, they were still very unsatisfied with the support it provided. When the vendor released the extensively redesigned version 2.0, my client faced a stark choice—forego the attractive features of 2.0, or implement it and lose over $1M worth of customisation. I was asked if I could come in and help them decide what to do. By analyzing their complaints about the current implementation, I suspected that the root cause of their dissatisfaction was the underlying data model – it was fundamentally a mismatch with the client’s business, and was unchanged in 2.0. For instance, if a complaint was “the application allows multiple hires per requisition” it wasn’t hard to find out that their rule was “one requisition, one hire.”

I needed to find out what the application’s underlying data model was, so I contacted the vendor to request it. That was a humorous conversation! They weren’t even sure what I was talking about, but they eventually sent me a printed stack of SQL DDL. It took a day or two of effort, but I was able to reverse-engineer a logical data model out of that, and then simplify it until it was more or less a conceptual model. In the meantime, I’d taken my understanding of the client’s business and developed a conceptual model for that. However, I couldn’t just dump this on the client with the advice “you have a data model mismatch,” so I decided to convey my findings in a presentation to the human resource staff and management who had retained
me. I began by drawing, on a large whiteboard, the data model that I felt represented their business. The key to this example is that I never called it a data model, drew it while I described it, and never used any data modelling terms such as entity, relationship, attribute, key, or anything like that. Instead, I started by saying “Here’s how I think you see the world.” For instance, “when a Position in a Department becomes vacant, management might issue a Requisition. This might be advertised to locate Applicants for the Requisition, or you might match it with Resumes on file to determine who might be an Applicant.” All the while, I’d be drawing the appropriate boxes and lines on the whiteboard, keeping it straightforward and (I’ll stress this again) never using any data modelling terminology. At the end of this part of the presentation, the clients said that this was the clearest description of their business they’d ever seen, and wondered if I could provide it for inclusion in their training material.

That was a gratifying moment, but we weren’t done yet. I explained that the application’s “world view” (a term I often use instead of “data model”) was substantially different than their own. On another large whiteboard, I went through the same routine, but this time illustrating the application’s underlying data model. The reaction was “shock and awe” as one mismatch after another became evident, and I’ll never forget the senior executive in the room saying “this has been a revelation.” It occurred to me that for “a revelation” I should charge more!

In the end, the client realised that they could redesign screens and rewrite application code, but they were still going to be unhappy with the application unless the underlying database was redesigned, which was just not an economical course of action. The application was abandoned altogether, which might not seem like a happy ending, but the client was happy to have avoided throwing good money after bad. Eventually, a more suitable solution located using the client’s data model as one of the evaluation criteria. Conceptual data models, even if we didn’t call them that, were fundamental to helping them understand the situation and make this important decision.

With information systems becoming more complex and more important than ever, I find that conceptual data models are more useful than ever, as long as you keep that “business-oriented overview” guideline in mind. I hope this article has given you some food for thought on conceptual models, and some ideas on how to use them. Please let me know what you think!
Alec Sharp

Clariteq Systems Consulting Ltd. | The convergence of strategy, business process, and systems development
2478 Nelson Avenue, West Vancouver, BC, Canada V7V 2R4
Mobile: +1 604 418-3352
asharp@clariteq.com
www.clariteq.com - site
www.twitter.com/alecsharp - microblog
www.erwin.com/expert_blogs - ERwin data modelling blog
www.bptrends.com - search on "Alec Sharp" for "Practitioner's Perspective" columns