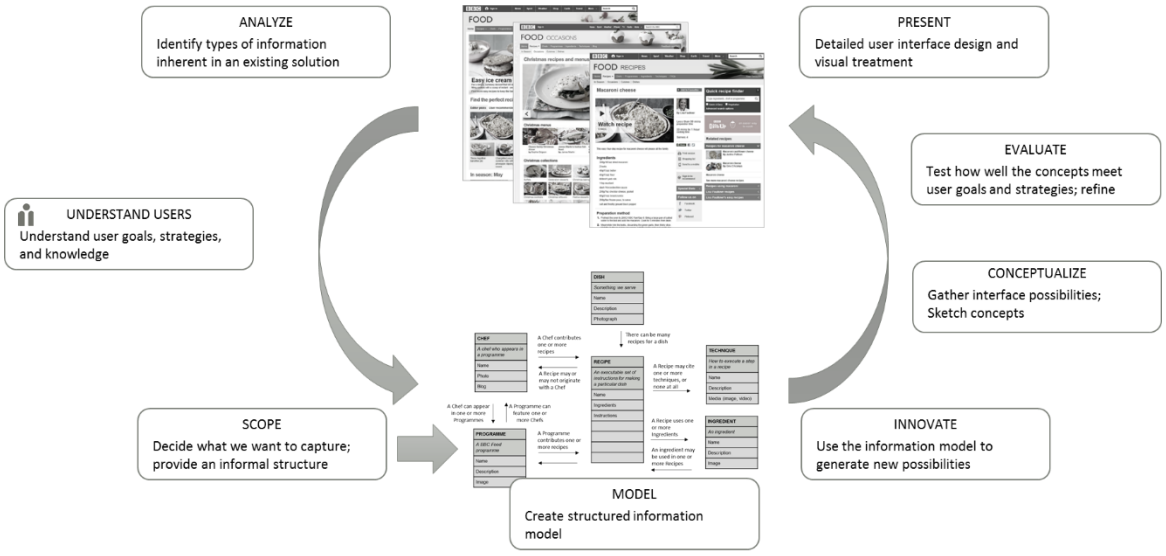
THE INFORMATION ARTICHOKE .COM

Information Modelling

A UNICORN BRIEFING NOTE



Martin Stares

June 2015

Information Modelling

A UNICORN BRIEFING NOTE

Series introduction

This Briefing Note is part of the series "Experiencing + Architecting Information". The series provides intermediate solutions designers with polished and practical insights into designing information rich systems.

It draws upon insights from information architecture and user centered design. As such, it will be useful to front end designers who need to know more about information, and information architects and analysts who need to know more about users.

The series is introduced in the Briefing Note entitled "Experiencing + Architecting Information", where we detail the approach and coverage. It might help to read this first. The remaining briefing notes drill more deeply into each topic, and for the most part can be read in any order.

We sincerely hope this series will help you gain in skill and confidence in our wonderful profession.

Martin Stares
The Information Artichoke

June 2015

Information Modelling

A UNICORN BRIEFING NOTE

Introduction

The series introduction "Experiencing + Architecting Information" introduced two views of a user-facing solution, the user view and the architect view, and explained why they are both important.

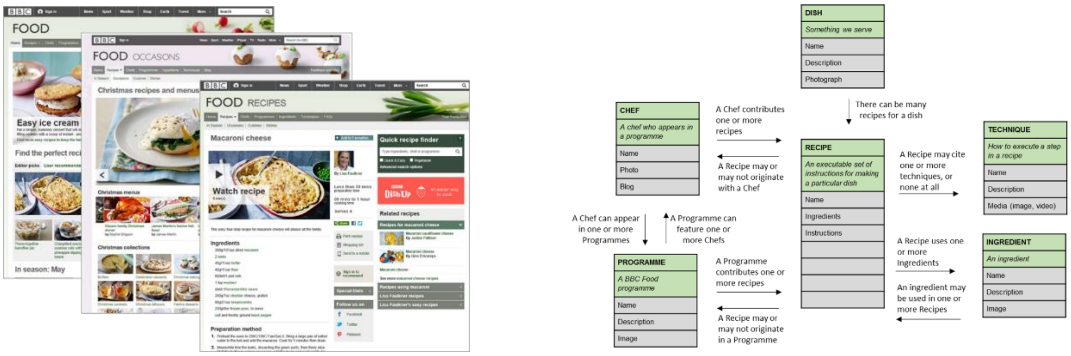


Figure 1: User and architect view of a food site

A user-facing solution always has an underlying information structure, whether we recognize it or not, just as this sentence has a grammatical structure, whether we recognize it or not.

The visual representation of that information structure is called an information model. This Briefing Note explains the mechanics of information modelling.

- The notation – a small expressive visual vocabulary
- The practice the process of creating an information model – examples of the thought process showing initial creation and refinement.

This will be a foundation for the application of models in solution design and innovation.

It's not database design - phew

People often ask whether information modelling is very technical, like database design, and the answer is "not very". Here's why.

Information models can play two roles in solutions delivery:

- Solutioning and interface shaping – using the information model to help us design better experiences and deliver better value to the users
- Implementation – using the information model to help us with implementation approaches that are maintainable, extensible, and reusable, if appropriate.

In our roles, we only do the solutioning and interface shaping. The project team will determine how to implement the solution. Options range from creating data structures similar to the information model

to building the solution in static HTML; the choice will depend on cost, time to delivery, next stage iterations, desire for content reusability, etc.

That's good news for us. We can get away with an approach to information modelling that is simpler than developers need, simple enough in fact that models can be created by many stakeholders and understood by more.

The pieces of an information model

Let's deconstruct the information model we showed previously.

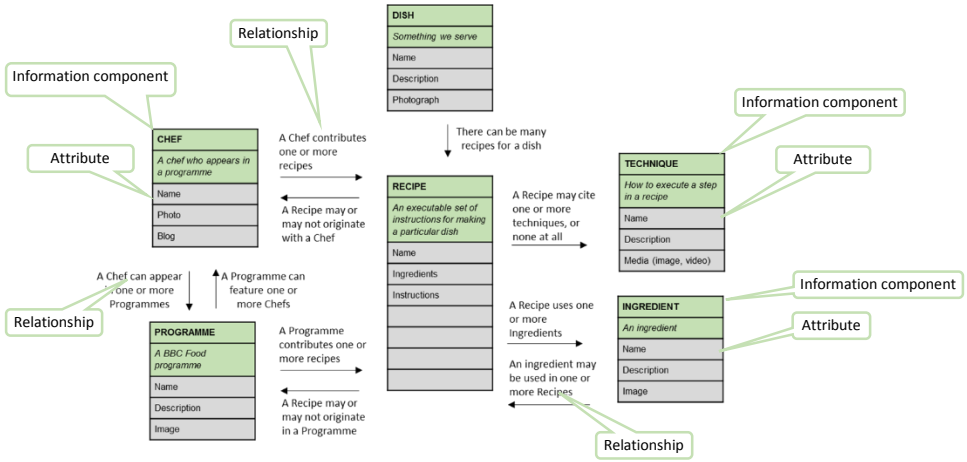


Figure 2: Information model of food site

Two things are apparent; there are boxes and there are arrows.

The boxes list the types of information that are available to our solution: {Recipes, Chefs, etc.}. We call these **information components**. Because there is no box labelled stores, our solution cannot tell us systematically where to purchase ingredients to make a recipe. If this became a requirement, we would have to include a box called STORE.

The boxes have an internal structure specifying what makes up information component. A Recipe is made up of a Name, Ingredients and Instructions. These are called **attributes.** This Recipe specification does not include Preparation Time or Cooking Time attributes, so our solution cannot systematically tell us how long a recipe takes to make, or list recipes from shortest to longest preparation time. If we wanted this, we would add attributes Preparation Time and Cooking Time.

There are labelled arrows joining some of the boxes. These describe information **relationships** that are useful in this solution. For example, the relationship between Recipe and Chef is useful because it lets us answer questions such as "what recipes has this chef contributed" and "who contributed this recipe (and what else did they contribute)".

There are not arrows between all boxes. This is because not all relationships are meaningful or useful in this solution. For example, there is no relationship between Ingredient and Technique. This means that

the current solution cannot answer questions such as “what are all the techniques that can be applied to an onion” or “tell me all the things that can be chopped”.

Where information components come from

Information components come from two different processes, both of which might be present in a given initiative.

- Information Analysis – this is where you have an existing user-facing solution and want to understand its information structure. See “Experiencing + Architecting Information – Information Analysis” for details and the following example.

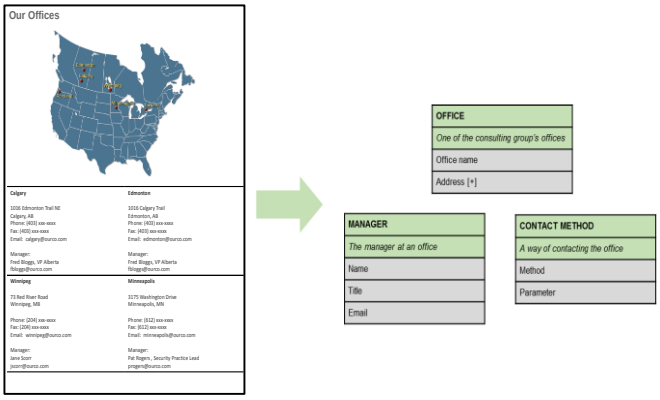


Figure 3: Result of information analysis for a consulting group's offices

- Scoping – this is where you have a net new initiative; from everything that could possibly be said, you have to select that information deemed most useful to the users. See “Experiencing + Architecting Information – Scoping” for details and the following example.

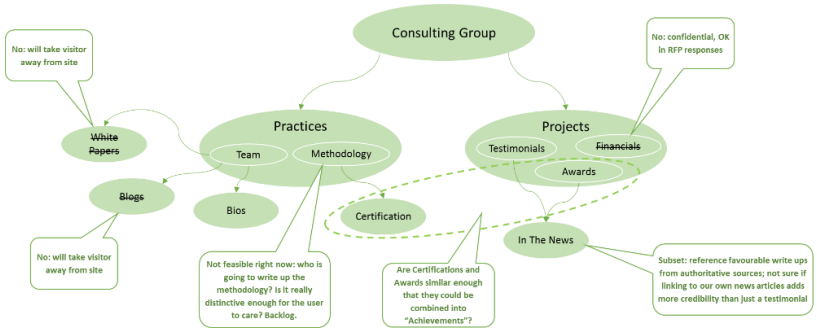


Figure 4: Results of scoping exercise for a consulting group

Now that we understand the structure of information models, let’s look at the process for creating them.

Case study

A consulting group wants to tell its potential customers about its practices and track record. The scoping exercise raised a lot of possibilities, but for the first phase it was decided to simply present information about projects and practices.

Create information components

Create boxes for the information components in scope.

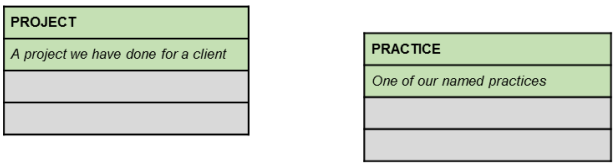


Figure 5: Initial information components for consulting group

Label each box with the name of the information type participating in the solution, as well as a short description. The short description sometimes raises interesting issues. For example, when you look at the Practice component, you may wonder whether the accounting department is a practice. You may find that Service or Offering feels more natural.

Next, see if you need a top level container for the whole solution. In this case, we recognize that projects and practices are parts of an entity called Consulting Group.

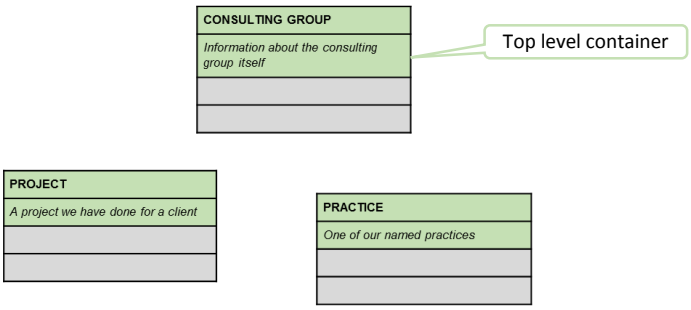


Figure 6: Information components for consulting group with top level component

Sometimes, as in this case, a top level container has extensive content in its own right, and we need to model it; other times it is just a place to hook other things onto and we don't explicitly model it (as in Figure 1).

You can arrange the boxes in any way that makes sense. In Figure 6, the container is placed at the top. In Figure 2, recipes are placed in the centre because of their central role in the site, and because it makes it easier to draw in all the relationships.

Look for containers

For components that have multiple instances of the same type, see if a container is appropriate.

With projects, for example, we might want to have some general statements about our project work that cover all projects (e.g. “With experience in some of the toughest environments in the world ….”). This content does not reside with any one project; rather it is part of a container component, which we can call “Our Projects”.

We might want a similar container for Practices, called Our Practices, containing general statements like “The following diagram shows how our individual practices collaborate synergistically …” These container correspond to landing pages in the user interface, among other things.

The information model gets extended like this.

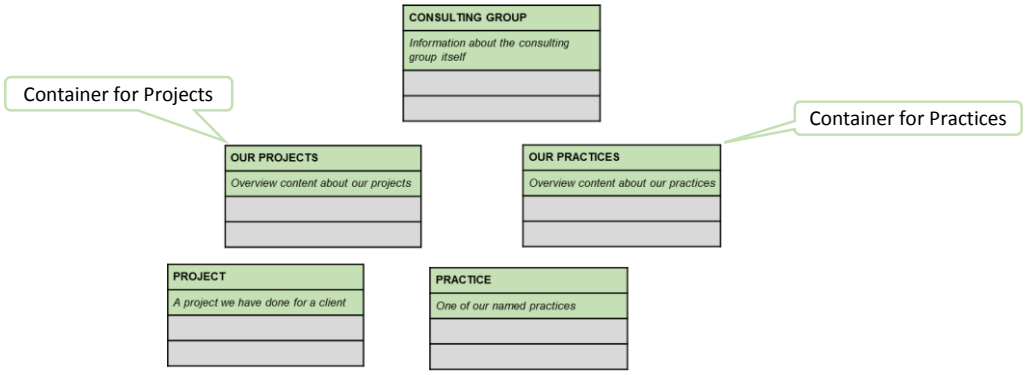


Figure 7: Information model with containers

Names like “Our Projects” or “All Projects” are good architect-view default names for containers of repeating elements. If there are existing names in common use, use them in user-views (e.g. “The House of Lords”, “The Harry Potter Series”).

Add attributes

Start to flesh out the information components, like this.

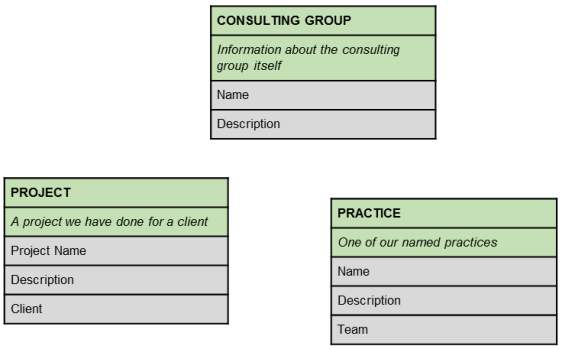


Figure 8: Adding attributes to the information model

Your understanding of attributes will change over time, so just get started with your best guess. Name and Description are common and a good place to start. Client seems appropriate for a project.

The following approaches will help refine your choice of attributes,

Provide illustrative examples

Information components are very abstract, being a template or specification for real world content. Check them against actual content to understand the real world complexities and any regularities we can exploit.

For each information component, provide an illustrative example showing or describing typical content; it may already exist or we may have to invent plausible content. For a Project, we might have.

PROJECT	**Illustrative Example**
A project we have done for a client	
Project Name	Online banking web presence
Description	WBW Bank required an online presence and support for online application for financial products. We introduced agile user centered design capabilities, with which we generated a stream of innovative products in a repeatable fashion [1]. "They made our vision a reality" – Bart Davis VP Marketing "Excellent knowledge transfer" – Linda Chow, Sales
Client	*WBW Bank* [2]

Notes:

[1] The full version of description is richly formatted, with a summary table and images.
[2] The client name should be consistent across projects.

Figure 9: Illustrative content for project component

This example shows a mixture of actual content and notes. The notes indicate areas where there might be implications for the information model.

To go further, use the information analysis techniques described in the Briefing Note "Experiencing + Architecting Information – Information Analysis". Although they were introduced as tools for analyzing actual content, they can be applied to the illustrative content, regardless of whether it is real or imagined.

The next sections show the application of a couple of techniques.

Analyze large content pieces

Large content pieces are great candidates for applying the information analysis techniques. As an example, consider applying "Look for Different Information Sources" to the description above. We see that most of the content was provided by someone in the consulting group, but testimonials are provided by the client.

Logically, then, Testimonials could be split out from Description.

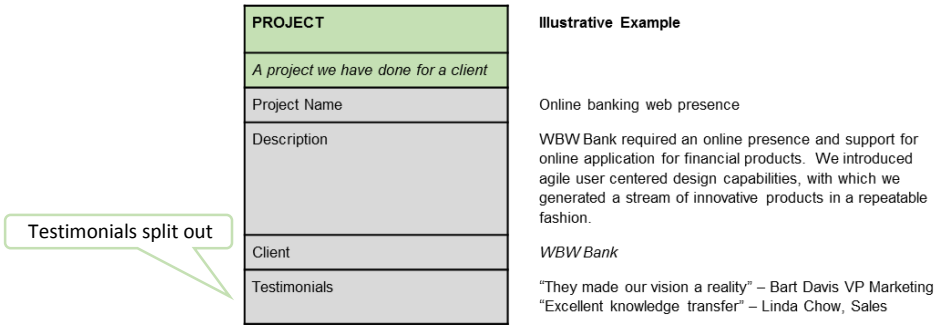


Figure 10: Testimonials split out from description

This can be justified by both business requirements and user interface considerations.

- Business requirement: "project description may include or exclude testimonials in different circumstances"
- User interface considerations: "if there are testimonials, they will be displayed in big gold letters in a prominent place on the screen or document".

Both of these considerations are much easier to accommodate if Testimonials are a separate attribute of Project, so we will split out Testimonials from Description.

To be completely rigorous, we could make the case for creating a separate Testimonial component.

TESTIMONIAL	Illustrative Example
An individual testimonial for a project	
Name	Bart Davis
Title	VP Marketing
Quote	They made our vision a reality.

Figure 11: Information component for a testimonial

The logical justification is:

- Testimonials have a fixed structure
- There could be a variable number of testimonials for a project, including zero.

Once again, look for justification from business requirements and user interface considerations.

One justification would be if we wanted to access or display individual testimonials.

- Business requirement: “testimonials for a project shall be accessible individually”
- User interface considerations: “if there are testimonials, one will be selected randomly and displayed when the project page loads, and fade into the next after a few seconds”.

But if we do not have such requirements, we will content ourselves with lumping all testimonial as an attribute of Project.

Promote attributes to components

Look for places where an attribute can be promoted to an information component. This is the Attributes vs. Components technique.

Consider the Project:Client attribute, with illustrative value “WBW Bank”. The bank certainly has more things that could be said about it, for example its logo and its industry.

Look for business requirements or user interface reasons for providing more information about a client; there are some common reasons.

- Provide filtering (“the user shall be able to query all projects for clients in a given industry”)
- Provide strong information scent (“a project will display the client logo to provide fast recognition”)
- Allow new user-facing functionality (“the landing page will display a list of client names and logos”).

If we want to support this functionality, we will promote Project:Client to an information component like this.

CLIENT	**Illustrative Example**
A client for whom we have done work	
Name	WBW Bank
Logo	*Logo*
Industry	Financial services

Figure 12: Information component for a client

If we don’t want to support this functionality, we can leave Client as an attribute of Project.

Add relationships

Add labelled arrows between the boxes, where appropriate.

For the basic information model for the consulting group, we have.

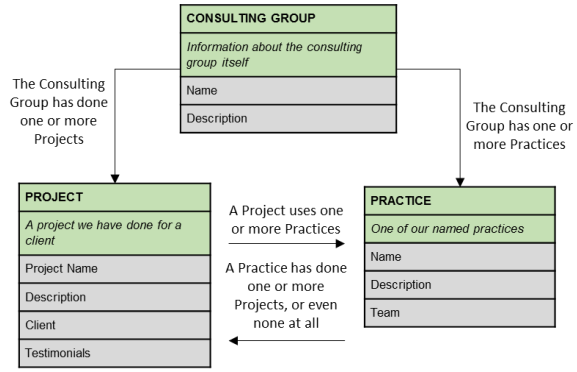


Figure 13: Relationships for basic consulting group information model

There is a direction to the arrows – the relationship between Project and Practice is distinct from the relationship between Practice and Project.

The description on the arrow is a sentence with a distinctive structure.

- It includes the name of both information components that it connects
- It first names the component at the un-pointed end of the arrow, and then names the component at the pointed end
- It includes a numeric specification such as “one or more”, “at least one”; this numeric specification is called the **multiplicity** of the relationship.

Consider all possible relationships between the boxes, and keep only those that make sense from a business point of view.

Keep only relationships that make sense

Decide which relationships you want to keep; this is based on business requirements and what you are trying to provide your users. There are some complexities here, which we start to explore below/

Revisiting the information model for the food site, not all pairs of boxes have a relationship.

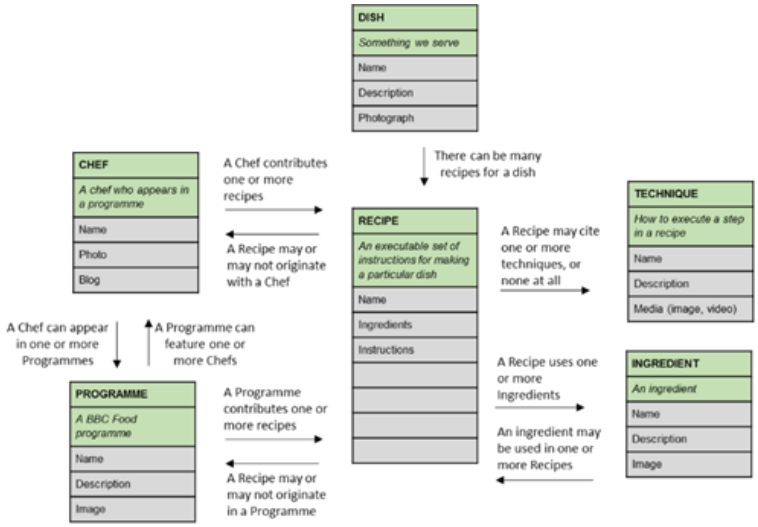


Figure 14: Information model for food site (again)

Consider the relationship between Recipe and Chef; this lets us answer questions such as “what recipes has this chef contributed” and “who contributed this recipe (and what else did they contribute)”. The product team decided that this is useful functionality.

There is no relationship between Ingredient and Technique. This means that the current solution cannot answer questions such as “what are all the techniques that can be applied to an onion” or “tell me all the things that can be chopped”. The product team decided that this is not useful functionality.

The decision to include a relationship is a business decision influenced by what we are trying to provide to users. While the food site does not value a relationship between Ingredient and Technique, this may be a very important relationship in a site that teaches Culinary Skills.

Another thing to remember is that, when a data-driven solution is implemented, someone has to maintain the relationship, which requires software and human effort.

Consulting group with more structure

In previous sections, we talked about splitting Client and Testimonial from Project. Here's what the information model could look like.

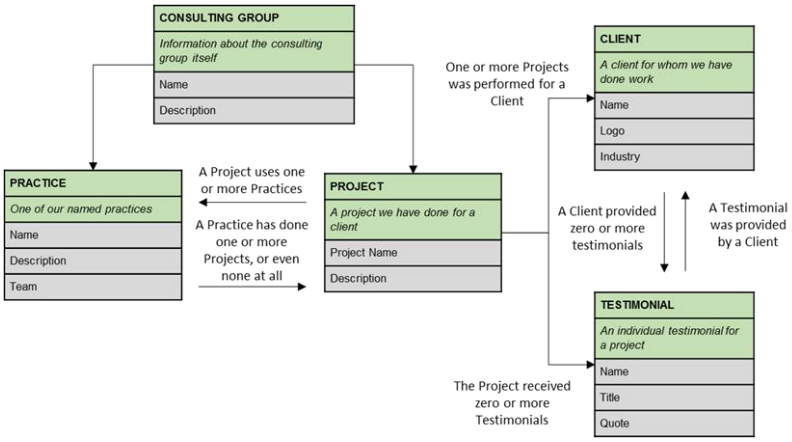


Figure 15: Elaborated information model for consulting group

It now shows the information components for Client and Testimonial; we rearranged things somewhat for clarity, which almost always happens at some point.

We have added a few new relationships; we will discuss some of these, and see if we need others.

Relationship: Project-Client

"One or more projects was performed for a client".

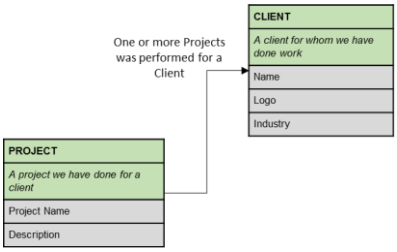


Figure 16: Project-client relationship

This is OK. It was an early requirement that projects identified their clients. It just happened that we promoted client to its own component.

What about the other direction, from Client to Project? We don't have a specification, but wonder about "A Client has had one or more Projects done for them".

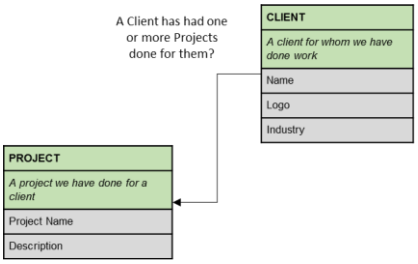


Figure 17: Client-project relationship

It sounds reasonable, but we have to check the multiplicity with the business. The conversation takes us by surprise.

> Us: Could you have a Client for whom you have done no Projects?
>
> Business: Happens all the time, like with service or maintenance agreements.
>
> Us: Aargh, do we have to include Service and Maintenance Agreements in scope?

By enquiring about multiplicity, we have uncovered a situation that may impact the solution. At the very least, we have to change the multiplicity in the relationship to "A Client has had zero or more Projects done for them".

<u>Relationship: Client – Testimonial; Project-Testimonial</u>

Look at the Testimonials component.

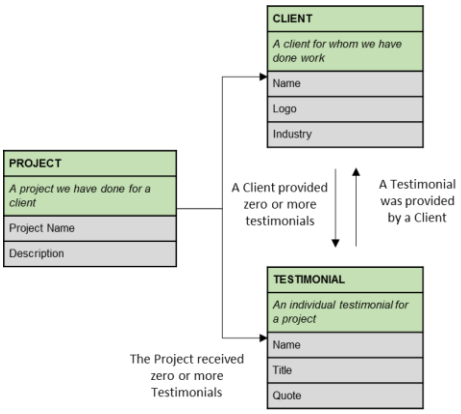


Figure 18: Neighbourhood of testimonials in the information model

There are two relationships terminating at the component.

> "The Project received zero or more Testimonials"
> "A Client provided zero or more Testimonials"

We feel they are saying the same thing, so let's explore this.

First, they are not saying exactly the same thing because they relate different components (Projects-Testimonials; Client-Testimonials respectively).

But they are related, and may convey the same information depending on the information model.

- If the Client can give Testimonials only for Projects, then one relationship implies the other. Here's how. When a client gives a testimonial, they have a project in mind, so a relationship between testimonial and project is inferred. Conversely, when a testimonial is given for a project, the client is known, so a relationship between client and testimonial is inferred.
- The situation is different if the Client can give Testimonials for other things, for example Service and Maintenance Agreements, or simply for being a good group to work with. In this case, project testimonials imply a client testimonial, but there can be client testimonials that do not imply a project testimonial because a project is not involved.

Relationship: Practice-Testimonial

The elaborated model for the consulting group (Fig 17.) does not show a relationship between Practice and Testimonial. Perhaps we should consider a relationship like this.

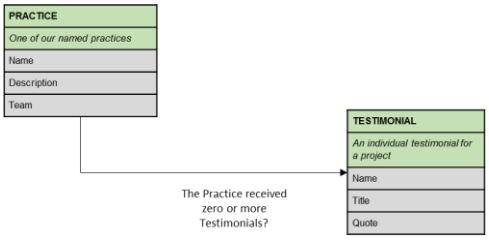


Figure 19: Possible relationship between practice and testimonial

There is no right or wrong answer from an information point of view.

Some considerations from a business point of view are:

- Are individual Practices something we sell independently of project work; for example, hiring out information architects for general purpose staff augmentation
- Are individual Practices merely part of the consulting group's project capabilities, to be selected and deployed as needed without fanfare
- Is it easy or meaningful to obtain practice-specific testimonials from the client
- How powerful are practice-specific testimonials; would they enhance a practice page
- Are resources available to enter practice-specific testimonials into a system?

What's next?

This material has introduced the notation and practice of information modelling.

The focus is the structure of the information being considered for a solution. It is less concerned with how the information is to be used, found, managed and recombined.

As soon as we consider who consumes the information, then we need to explore how the information will be accessed, and may need to enrich the model with **access metadata**.

Likewise, as soon as we start building the system in a contemporary manner, we will need **admin metadata** to control creation, review and disposition processes.

As part of the "Experiencing + Architecting Information" series, this Briefing Note "Information Modelling" has many touchpoints to other documents in the series.

- Scoping – shows how to generate informal information scoping diagrams when there is no existing solution; these are inputs into the formal information models described here
- Information Analysis – shows how to generate information models from user-facing artefacts
- Innovation – shows how to use an information model to explore user access possibilities; shows how to relax constraints in the model to generate brand new possibilities.
- Conceptualizing – shows how the various aspects of an information model are reflected (in a one-to-many fashion) in user interface constructs, and demonstrates this using wireframes.

Good designing!