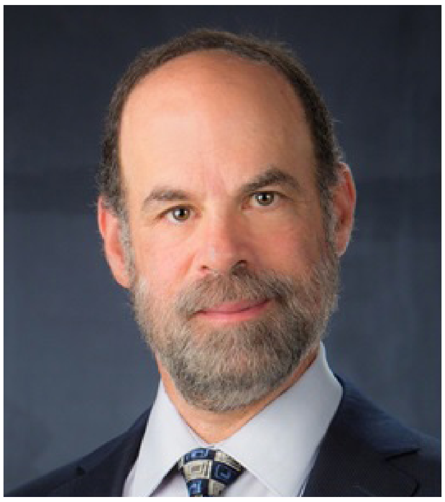
White Paper

Architecture Skills: Abstraction

WP0163 | September 2014



Mike Rosen

Mike Rosen is Chief Scientist at Wilton Mike Rosen is Chief Scientist at Wilton Consulting Group where he provides expert consulting in Business Architecture, Enterprise Architecture, and Service-Oriented Architecture. He is also a founding member of the Business Architecture Guild and Editorial Director for SOA Institute. His current emphasis is on the implementation of Enterprise and Business Architecture and programs. He has years of experience in the architecture and design of solutions for global corporations and 20+ years of product development experience.

Mr. Rosen is an internationally recognized speaker and author of several books including “Applied SOA: Architecture and Design Strategies”. He welcomes your comments at Mike.Rosen@WiltonConsultingGroup.com

One of the key characteristics of architecture is looking at the ‘big picture’, but a major challenge is that we can’t present the big picture on a great big piece of paper – it has to fit on a single sheet or model. In order to do that, we have to come up with new concepts that summarize the overall picture into a small number of elements and relationships. We can do this through a variety of techniques, like divide-and-conquer, categorization, generalization, and so on.

The principles of abstraction are aimed at just these problems. This paper will provide an introduction to abstraction and show how it applies to architectural modeling.

What is Abstraction?

Wikipedia offers several different definitions for abstraction that I’ve adapted below:

1) Abstraction is a conceptual process by which concepts are derived from the usage and classification of signifiers, first principles, or other methods. “An abstraction” is the product of this process—a concept that acts as a super-categorical noun for all subordinate concepts, and connects any related concepts as a group, field, or category. Conceptual abstractions may be formed by reducing the information content of a concept typically to retain only information that is relevant for a particular purpose.

When we examine this definition, we see some important points. Abstractions are derived or inferred based on principles. Abstractions describe related concepts and may be formed by obscuring information that is deemed irrelevant in a given context.

2) Abstraction is a process or result of generalization, removal of properties, or distancing of ideas from objects. This may refer in particular to one of the following:

- Abstraction (computer science), a process of hiding details of implementation in programs and data
    - Abstraction layers, an application of abstraction in computing
    - Hardware abstraction, an abstraction layer on top of hardware
- Abstraction (linguistics)
- Abstraction (mathematics), a process of removing the dependence of a mathematical concept on real-world objects
    - Lambda abstraction, a kind of term in lambda calculus

Again, we see that abstraction is a process of selecting pertinent information, where what is pertinent is determined by the context (and the skillful architect). We are also told that abstraction applies across a broad range of topics, not just to computer science or architecture.

Types of Abstraction

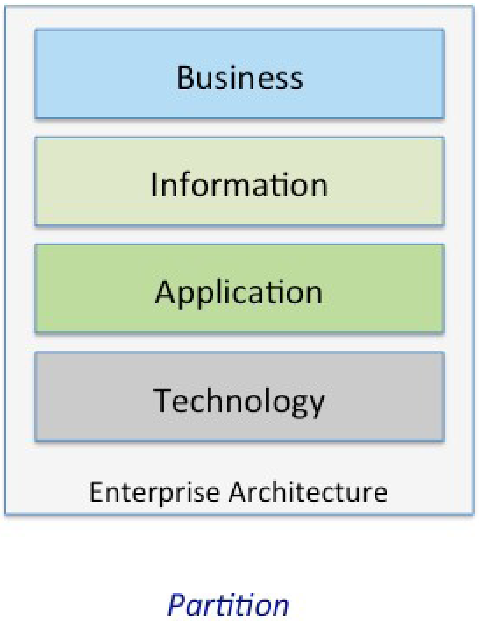
The definition above lists three specific techniques of abstraction that can be applied across a wide range of domains:

- **Generalization** - A generalization is obtained by inference from specific cases of a concept. More precisely, it is an extension of the concept to less-specific criteria. Generalizations describe a domain or set of elements, as well as one or more common characteristics shared by those elements. Verification can be used to determine whether a generalization holds for a given situation:
    - o Of any two related concepts, such as A and B, A is a "generalization" of B, and B is a special case of A, if and only if:
        - Every instance of concept B is also an instance of concept A; and
        - There are instances of concept A which are not instances of concept B.

Object modelers should be very familiar with the concept of generalization and how it is used to define groups and categories.

- **Removal of Properties** – Abstraction has also been described as the "suppression of irrelevant detail". We remove properties that are not relevant in a particular context, in other words, that are not important in conveying specific concepts to a specific audience.

- **Distancing of Ideas** – Objects contain concrete instantiations of specific concepts and ideas. We can use abstraction to separate the ideas themselves from the objects that reify them.



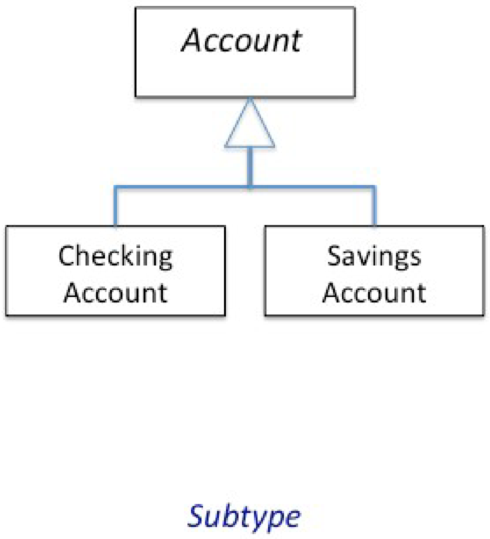


Figure 1 – Examples of Abstraction

Figure 1 shows two typical examples of abstraction. On the left is a common representation of enterprise architecture that illustrates partitioning. In this example, the whole of enterprise architecture is divided (partitioned) into four domains (abstractions) based on subject area. Each domain represents a generalization of a set of related architectural concerns and elements.

On the right is an example of subtyping which illustrates two of the techniques. First, it illustrates the typical generalization / specialization relationship. Account is a generalization of checking and savings accounts. Checking and saving accounts are specializations of account. In this example, I have also illustrated account as an "abstract type" (signified by the italics), meaning that a generalized account cannot be instantiated, only a specialized account can exist. Note that Account is also an example of removal of properties. Only those properties that are important to all types of accounts are relevant in the context of the general account.

Abstraction Levels

More typically, removal of properties is associated with levels of abstraction. Three common levels of architectural abstraction, conceptual, logical, and physical are illustrated on the left side of figure 2.

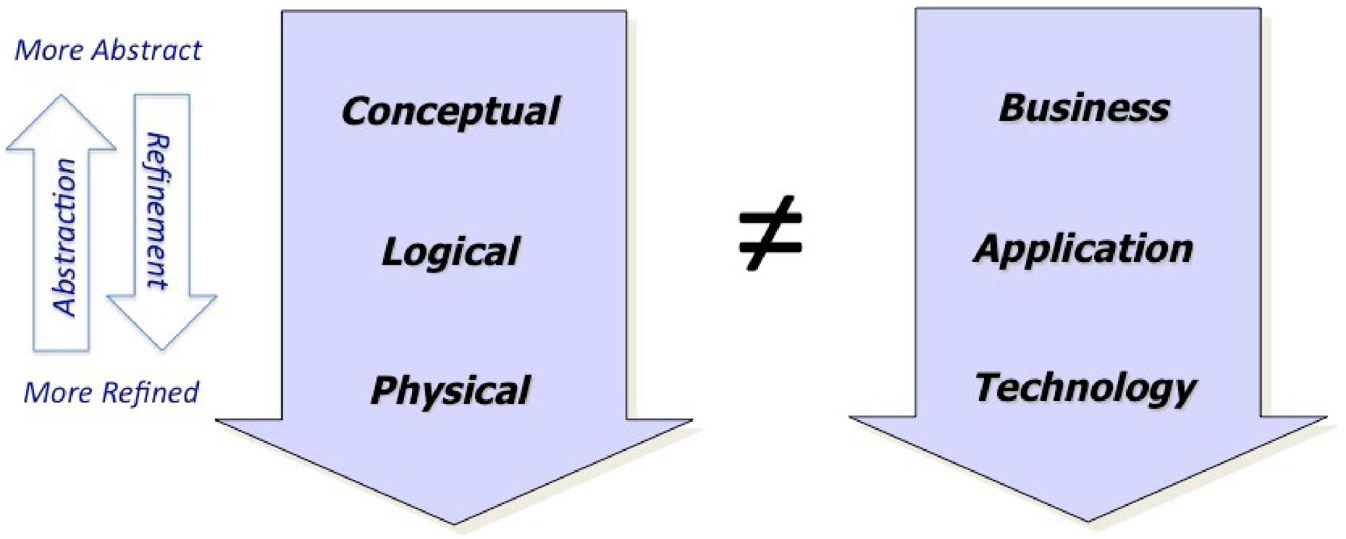


Figure 2 –Abstraction Levels

While the definitions of each level can be a little fuzzy we can provide some guidelines:

- **Conceptual** - Conceptual models focus on the key concepts and relationship of the whole solution, not on how the system works. As such, they are generally static models where connectors, if present, show relationships, not flows.
- **Logical** – Logical models describe how a solution works, in terms of function and logical relationships between the resources, activities, outputs, and outcomes. They can show a static view or a dynamic view.
- **Physical** – Physical models refer to specific products, protocols, data representation, network capabilities, server specifications, hardware requirements, and other information related to deploying the proposed system.

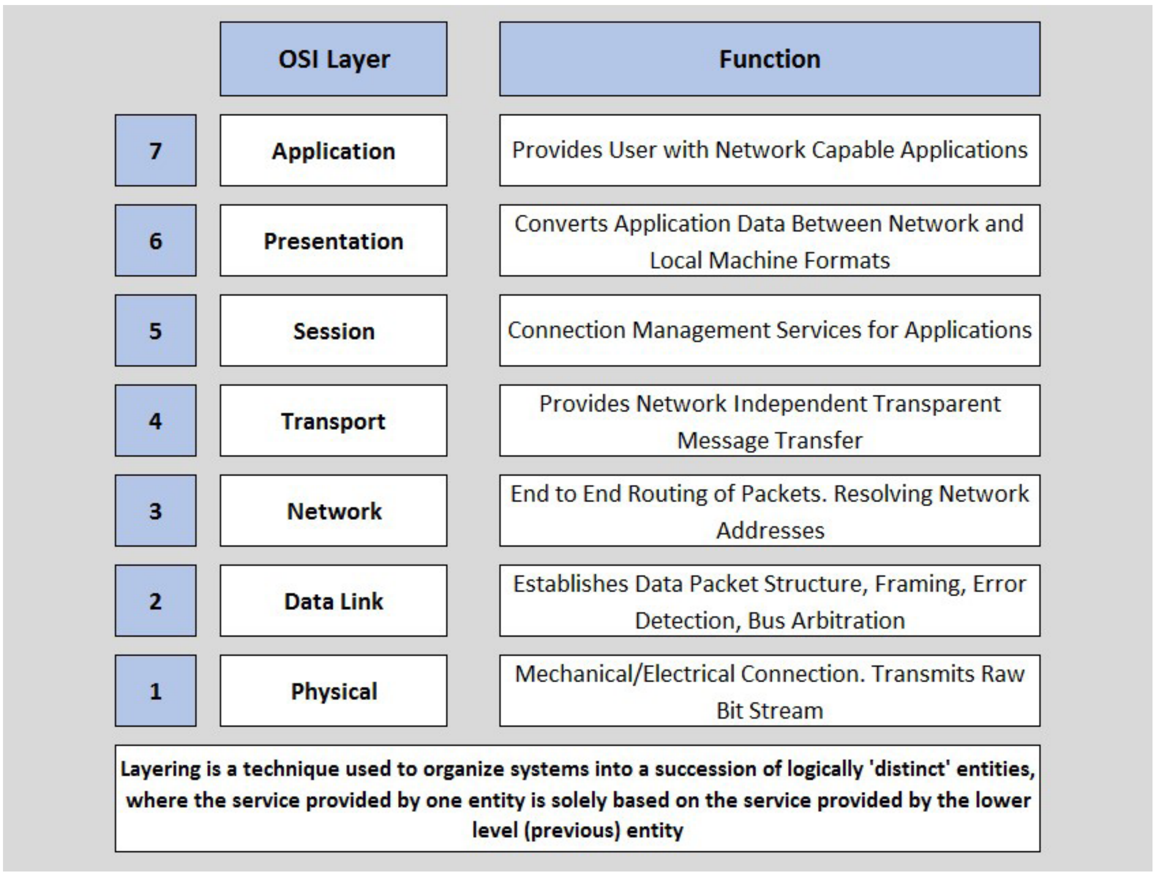
Conceptual models are more abstract than logical models, which are more abstract than physical models. We describe the process of transforming one model to another as refinement when we reduce the level of abstraction. Note that the transformation of models between levels involves more than just adding detail. Abstract concepts are transformed into more concrete concepts during transformation. For example, the logical concept of a ‘customer’, may be transformed into a logical customer entity, and then transformed into a set of tables and joins at the physical level. We can also transform models in the other direction, going from physical (more refined) to logical, to conceptual (less refined). We call this process abstraction.

Don’t confuse abstraction and domains

A common mistake is to equate the levels of abstractions with architectural domains as shown on the right side of figure 2. While it is coincidental that many business models are at the conceptual level, application models are more logical, and technology models may be physical, it is not always the case. For example, many organizations with have conceptual, logical, and physical levels of information / data models, or of models in the application domain. Abstraction levels affect the scope, concepts, and details that are represented in a model, not the subject matter.

Abstraction Layers

Another common application of abstraction is to partition complexity into cohesive layers that interact through interfaces. Figure 3 illustrates perhaps one of the best known and long lasting examples of architectural layers, the ISO Open System Interconnect (OSI).



	OSI Layer	Function
7	Application	Provides User with Network Capable Applications
6	Presentation	Converts Application Data Between Network and Local Machine Formats
5	Session	Connection Management Services for Applications
4	Transport	Provides Network Independent Transparent Message Transfer
3	Network	End to End Routing of Packets. Resolving Network Addresses
2	Data Link	Establishes Data Packet Structure, Framing, Error Detection, Bus Arbitration
1	Physical	Mechanical/Electrical Connection. Transmits Raw Bit Stream

Layering is a technique used to organize systems into a succession of logically 'distinct' entities, where the service provided by one entity is solely based on the service provided by the lower level (previous) entity

Figure 3 –Abstraction Layers

In this case, the layers are an abstraction of the role that is responsible for executing specific functions in the communications of data. Solely

Tips for Abstraction

As architects, models are one of our major work products, and abstraction is one of our most important tools in creating these models. Here are some tips for using abstraction to create successful architectural models:

- **Make the model fit for purpose** – Every model is intended to inform a particular stakeholder, or set of stakeholders, with information that is relative to their role and that is useful in supporting their decision making. The first step in a good model is to determine who the stakeholders are, and what information you are trying to convey to support what type of decision. This will tell you what level of abstraction (conceptual, logical, physical) and detail is appropriate. If you can't answer these questions about your models, then chances are good that the models will not be of much use to anyone.

- **Capture the view of the big picture. Put things in context** – Understand the context that the system or solution must fit within. Is the context the enterprise, a business strategy, a channel, a subsystem? Make sure the context is consistent with the level of abstraction.

- **Determine the fundamental elements and relationships** – Given the context, what are the fundamental concepts (elements) that relate to the stakeholder role? What are the relationships between concepts? Use generalization to create abstract concepts. Use removal of properties to suppress irrelevant details.

- **A good model must be clear and easily understandable** – In general, a model should not have more than 5-8 different concepts (although you may have several instances of each concept). The model should be consistent in terms of level of abstraction and detail across all of the concepts and relationships. Typically, the more abstract a model is, the fewer total elements it will contain. As a rule of thumb, a conceptual model may have 25-50 total elements whereas a physical model could contain well over 100.

- **Explore different perspectives to capture the right elements and relationships** – Be agile. Try initial models from a few different perspectives with different stakeholder to see what works, and then go back and add polish and detail to the best ones. Better yet, engage the stakeholders in helping to create the initial models.

- **The diagram must appeal to the stakeholder and highlight their viewpoint** – Again, this is reinforcing the first point about being fit for purpose. If a stakeholder does not find the model useful, interesting, or appealing, then they probably won't use it. If it's not used, you have not only wasted your time, but also failed to have your architecture implemented.

Conclusion

Architects need to collect information and generate solutions across a broad context, and then distil that information in a way that presents the important concepts and relationships, in the big picture context, to a range of different stakeholders. Typically, architects create models and diagrams as a primary means of communication. Abstraction provides the architect with a set of techniques to determine what concepts, information, and detail to include in order to create models which influence and impact decisions.

Orbus Software
3rd Floor
111 Buckingham Palace Road
London
SW1W 0SR
United Kingdom

+44 (0) 870 991 1851
enquiries@orbussoftware.com
www.orbussoftware.com