Once upon a time - way back in the historical vaults of computer science, information engineering and data modelling – the terms “physical” and “logical”, and “conceptual”, were very popular.

Without going into too much detail, the three terms were intended to show a progression in the data design process:

- **Conceptual** described the theoretical notions and thoughts that provided the foundation for the design.
- **A logical design**, which translated the conceptual plan into a coherent model that would deliver the conceptual intent.
- **The physical model** provided any adjustments that were needed to make the concepts and the logical design work in practice.

The three terms made it easier to model data by outlining a way of thinking about data constructs – starting with abstract principles and working towards the practical considerations required in implementation. Conceptual models were highly normalized, giving very versatile and adaptive data constructs, but because of processing limitations, these normalized models were compromised at both the logical and physical levels as design and implementation constraints were considered. The idea is very similar to the levels of architectural understanding.

Some people found the three terms very confusing! I personally remember having difficulty making sense of the words and their differences when I first started learning about information engineering.

1. This is partly why I’ve produced an online course to answer these two questions: **What is Enterprise Architecture, and why do we need it?**
The terms were very popular for a time, and so they were adopted in other areas. A common example was the notion of Logical and Physical Application components.

Which brings me to the present day... and to the TOGAF® metamodel, where we find the following constructs:

**Application Component**
An encapsulation of application functionality aligned to implementation structure. For example, a purchase request processing application.

**Logical Application Component**
An encapsulation of application functionality that is independent of a particular implementation. For example, the classification of all purchase request processing applications implemented in an enterprise.

**Physical Application Component**
An application, application module, application service, or other deployable component of functionality. For example, a configured and deployed instance of a Commercial Off-The-Shelf (COTS) Enterprise Resource Planning (ERP) supply chain management application.

**Logical Data Component**
A boundary zone that encapsulates related data entities to form a logical location to be held, for example, external procurement information.

**Physical Data Component**
A boundary zone that encapsulates related data entities to form a physical location to be held, for example, a purchase order business object comprising a purchase order header and item business object nodes.

**Logical Technology Component**
An encapsulation of technology infrastructure that is independent of a particular product. A class of technology product; for example, supply chain management software as part of an Enterprise Resource Planning (ERP) suite, or a Commercial Off-The-Shelf (COTS) purchase request processing enterprise service.

**Physical Technology Component**
A specific technology infrastructure product or technology infrastructure product instance; for example, a particular product version of a Commercial Off-The-Shelf (COTS) solution, or a specific brand and version of server.

OK. Well, I'm glad that's all clear now. It's not? Let me try to simplify by explaining the things that I would like to capture in an EA metamodel. In stick to the application side of things, but the general points that I'm going to make would also apply to data or technology.

---

100'S of FREE Downloads Available!

Posters, Whitepapers, Case Studies, Videos, Quizzes, Articles

Download Now!
The TOGAF construct, Application Component, is part of the core metamodel. When you adopt the TOGAF metamodel extensions, the core Application Component construct is extended to include Logical Application Components (a class of application) and Physical Application Components (an actual application).

There are many other things that I might want to include in an effective EA metamodel. For example, I might want to classify applications depending on who develops or owns the application. I might want to keep track of deployed instances of a Software Application, and some of these instances might be of different versions of the software.

More importantly, I probably want to relate these various components to constructs that make their architectural structure explicit. For example, I might want to detail the modules within an application; I might want to explicitly show the layering within an application; or describe in detail the interfaces between the application and data, hardware or process components. And I might want to describe the patterns that show how these components come together.

When I started writing this article I wanted to show how words like conceptual, logical and physical are inadequate terms for describing the realities of our architectural landscapes. I was also concerned that the terminologies inherent in the TOGAF metamodel are not clear that are achieved by, for example, metamodeling activities required for an EA practice.

But at the very least, I hope that I’ve got you thinking about what you need to describe in your metamodel, and how you need to do that. The key takeaway is that your EA metamodel must work for you; that’s why it’s so important to adapt any pre-defined metamodel, such as the one provided in TOGAF, to meet your exact needs. Give us your feedback, and perhaps we can cover further issues in subsequent blogs.

100'\text{S} \text{ of FREE Downloads Available!}

Posters, Whitepapers, Case Studies, Videos, Quizzes, Articles

\text{Download Now!}