

# Data Names Using the "Of" Language

by William Shaffer

This page describes a taxonomy for naming classes and attributes. Data, naming, of language

## Table of contents

1 Purpose.....	2
2 Background.....	2
3 Creating a Data Element Name.....	2
4 Why It Works.....	3
5 Suggested Techniques.....	3

## 1. Purpose

The purpose of the Of Language is to provide a systematic way of naming data elements. Particularly in large systems, there must be a way to easily determine whether a data element exists in the data dictionary or not. Without a taxonomy or system of naming, the developers may end up introducing duplicate data elements under different names. Such duplicates almost always result in defects. Repair the defects involves making changes in the code, often late in the project, or putting in a "hack" to synchronize the two data elements, making the code more confusing.

The Of Language provides a way of easily recognizing whether a data element exists in the data dictionary. The Of Language is a simple, yet powerful concept.

## 2. Background

The Of Language was described by Ken Orr in *Structured System Requirements* [ORR81]. He provided the following history:

*Some years ago, a project team at IBM was assigned the task of building a data dictionary for the entire corporation. One of the by-products of that activity was something called the "of" language. "Of" language is a method of assigning data names from the general to the specific, using certain words as connectors.*

## 3. Creating a Data Element Name

Of Language gets its name from converting a data name to an equivalent name using 'of' as a connecting proposition. It often involves shifting a noun used as an adjective to a possessive phrase. Here is an example:

"insured birthdate" is converted to "date of birth of insured". The *OFs* are then dropped to create "date\_birth\_insured".

Use the these steps to create a data element name in the Of Language:

1. Write out the data element name as you would ordinarily.
2. Select the data element type as the first word. For example, in "insured birthdate", the type is *date*.
3. Convert nouns used as adjectives into possessive expressions using Of. For example, "birthdate" becomes "date of birth".
4. Put adjectives where they make sense. For example, "monthly coverage premium" becomes "premium of coverage monthly".
5. For multiple adjectives, establish and use a standardized convention for the order of the

adjectives. For example, "monthly guaranteed coverage premium" might become "premium of coverage monthly guaranteed". A convention can be established that the timing adjective will come before other adjectives.

6. Remove the OFs and separate the words with underlines or camel case words. For example, "insured birthdate" becomes "date\_birth\_insured" or "dateBirthInsured".

#### 4. Why It Works

Of Language makes finding data elements easier when there is a long list of elements. The reason is that the data element type is usually more easily recognized than the other parts of the name. When the list of data elements is sorted, data elements of the same type cluster together. For example, imagine you are interested in the expiration date. Perhaps the element is named "expiration date", "coverage expiration date", or "policy expiration date". In a list of one thousand data elements, the names would be scattered about. It would be easy for you to miss one or two of them. If the names were "date\_expiration", "date\_expiration\_coverage", "date\_expiration\_policy", the names will be clustered together and are more easily located.

#### 5. Suggested Techniques

These techniques may prove useful as you define your data elements:

- Data elements can be names of classes or attributes.
- Class names begin with a capital letter and use camel case.
- Attributes begin with lower case letters and the words are separated with underlines.
- If the attribute represents a collection of objects, the name should be plural.
- Develop and document a rich set of types related to the domain. For insurance, some useful types include premium, coverage, limit, and deductible. Premium is a preferred to money or number as a type, because it is more descriptive of the usage of the data element.