Department of Electrical and Computer Engineering
Dalhousie University

The Data Dictionary

Dr. Larry Hughes

Introduction

In Introduction to Dataflow Diagrams, a dataflow diagram is represented as a series of well-defined symbols. One of these symbols, the *flow*, illustrates the data that flows between all other objects (i.e., the terminators, processes, and stores). In this document, the *data dictionary* is described; the data dictionary is a technique for representing the flow between the other objects.

The data dictionary is an organized list of the data elements that pertain to the system. The data elements allows data to be described in terms of:

- the *meaning* of the flows and stores.
- the *composition* of the data that moves along the flows.
- the *composition* of the data in the stores.
- the *values* and *units* of the data elements in the flows and stores.

In short, the data dictionary allows complex data objects to be defined in terms of simplier objects. For example, a *house* could be defined as a *foundation*, *floors*, *door*, *windows*, *roof*, ...

Data Dictionary Notation

The data dictionary is a tool that allows the system analyst to describe the user's data in a formal way. Using English (or any other human language) to describe data can be confusing, or lead to confusion, or both.

The following data dictionary notation is in common usage:

Symbol	Meaning
=	is composed of, or is defined as
+	and
( )	optional
{ }	iteration
[ ]	selection
|	choice (used with selection)
-	through (used with selection)
**	comment

"=" - Composed of

The "=" symbol indicates that the data element on the lefthand side of the "=" is composed of or is defined as or means, whatever is on the righthand side. For example, if A is composed of B and C, one would write (see below for a description of the "+" operator):

```
A = B + C
```

This can be interpreted as:

- A is composed of B and C, or
- A means B and C, or
- A is defined as B and C.

If a symbol, such as A is defined as several other data elements, it is said to be a *composite* symbol. A symbol that can be decomposed no further is said to be a *terminal* symbol.

The definition should also include comments (see below).

"+" - And

"+" is the "and" operator, indicating that a composite data element consists of several data elements (each of which could be a composite or a terminal). For example, a house could be defined as:

```
House = Basement + Walls + Roof
```

Similarly, a person's name could be defined as:

```
Name = FirstName + LastName
```

"( )" - Optional

The "( )" symbol indicates that the data elements within the "( )" are optional (i.e., they can be used or they can be ignored). For example, a person's name could have an optional initial:

```
Name = FirstName + (Initial) + LastName
```

The Name data element can be interpreted as either:

```
Name = FirstName + LastName
```

or

```
Name = FirstName + Initial + LastName
```

"{ }" - Iteration

The "{ }" symbol denotes iteration, which means that the data elements within the "{ }" can be repeated a number of times. For example, a house could have walls, a roof, and a number of windows:

```
House = Walls + Roof + {Window}
```

By definition, the *lower bound* of an iteration is 0 (zero); there is no *upper bound*. Using this definition, the above house could have no windows or have an infinite number of windows!

Limits can be specified. The lower bound is given on the left brace, while the upper bound is on the right brace:

Lower bound{ }*Upper bound*

Both the bounds are optional; for example:

1{Item}12	1 through 12 Items
1{Item}	A minimum of 1 Item
{Item}12	0 through 12 Items
{Item}	0 through infinite Items

For example, a student could be defined as a name and the number of courses being taken (from 1 to 6):

```
Student = Name + Courses
Name    = FirstName + (Initial) + LastName
Courses = 1{Course}6
```

"[|]" - Selection and Choice

The "[ ]" and "|" symbols are combined to list possible choices that are associated with a data element, typically a terminal. For example, a house could be defined as having one or more doors; each door could be defined as one of:

```
House = Walls + Roof + {Window} + Doors
Doors = 1{Door}
Door  = [Sliding | French | Hollow core | Steel]
```

The defined symbol can assume one of the meanings within the "[ ]"; in the above example, a `Door` could be one of `Sliding`, `French`, `Hollow core`, or `Steel`.

If there is a range of choices, not all of which need be displayed (i.e., their values are self-evident), one can use the "-" (through) symbol. For example, a person's name can consist of a number of characters (upper and lower case alphabetics, hyphens, apostrophes, or numbers):

```
Name = FirstName + (Initial) + LastName
FirstName = 1{Character}32
Initial = Character
LastName = 1{Character}
Character = [ A-Z | a-z | 0-9 | ' | - ]
```

In the above example, `FirstName`, `Initial`, and `LastName` are made up of `Character`s. Each `Character` can be an upper or lower case character, a number, an apostrophe, or a hyphen. Note that the order or spelling of a name is not taken into account here.

"**" - Comment

By rights, all data elements should be associated with the following:

- the *composition* of each data element. For example:

  ```
  Name = FirstName + (Initial) + LastName
  ```

  In this case, `Name` is composed of several other data elements.

- the *values* that are associated with a terminal data element. For example:

```
Character = [ A-Z | a-z | 0-9 | ' | - ]
```

In addition to the above, the data dictionary should also include the *meaning* of the data element. The meaning is a verbal description of the data element, enclosed in the "**" comment symbol. For example:

```
Name = FirstName + (Initial) + LastName
       * The person's name *
```

In certain cases, the name associated with the data element being defined is considered *self-defining*, and does not require a descriptive comment associated with it; in these cases, the comment should be written as "**". For example:

```
PersonsWeight = * *
```

Furthermore, in some cases, the value of terminal data elements can be shown in comments:

```
Gender = * Person's gender *
         * Values: Male or Female *
```

Whether the terminal data element is a comment or a choice "[|]" is at the discretion of the system analyst; the following would have been equally acceptable:

```
Gender = [Male | Female]
         * Person's gender *
```

In addition to the value, it is often useful to include the *units* and possible *range* of values; for example:

```
PersonsWeight = * *
                * Units: kilograms  Range: 1 to 200 *
```

An **alias** is a comment stating that one data element is synonymous with another; for example:

```
Instructor = * alias for Teacher *
```

Aliases can be used in situations where several different users of the same system refer to similar items with different names. If possible, assigning aliases should be avoided since they can lead to confusion. For example, what data element should be used on a dataflow diagram, the actual term or the alias?

Checking the Data Dictionary

The data dictionary should always be checked against the dataflow diagram. The following points should be noted:

- Have all the flows been defined?
- Have all components of composite data elements been defined?
- Are there any multiple definitions?
- Has the notation been correctly used?
- Have all data elements been defined?
