ASSIGNMENT NO: 2

NAME: MUHAMMAD KAMAL KHAN

ID : 13126

SEMESTER: 6TH

SUBJECT: SOFTWARE VERIFICATION AND VALIDATION

ASSIGNMENT TOPIC:

What is Z specification , why it is use for , also give example?

Z Specification

The Z notation is a formal specification language used for describing and modelling computing systems. It is targeted at the clear specification of computer programs and computer-based systems in general.

History

In 1974, Jean-Raymond Abrial published 'Data Semantics'. He used a notation that would later be taught in the University of Grenoble until the end of the 1980s. While at EDF (Electricite de France), Abrial wrote internal notes on Z notation is used in the 1980 book Methodes de programmation.

Z was originally proposed by Abrial in 1977 with the help of Steve Schuman and Bertrand Meyer. It was developed further at the Programming Research Group at Oxford University, where Abrial worked in the early 1980s, having arrived at Oxford in September 1979.

Abrial has said that Z is so named 'Because it is the ultimate language' although the name 'Zermelo' is also associated with the Z notation through its use of Zermelo-Fraenkel set theory.

Z Specification use for:

Z is based on the standard mathematical notation used in axiomatic set theory, lambda calculus , and first-order predicate logic. All expressions in Z notation are typed, thereby avoiding some of the paradoxes of naïve set theory. Z contains a standardized catalogue (called the mathematical toolkit) of commonly used mathematical functions are predicates, defined using Z itself.

Although Z notation (just like the APL language, long before it) uses many non-ASCII symbols, the specification includes suggestions for rendering the Z notation symbols in ASCII and in LaTeX. There are also Unicode encodings for all standard Z symbols.

Example:

Example: Banking System

```
— WithdrawMoney ——————————————————
ΔBankAccount
dollarAmount? : ℕ
centAmount? : ℕ
——————————————————————————————————
dollarAmount? ≤ dollars
dollarAmount? = dollars ⇒ centAmount? ≤ cents
centAmount? > cents
  ⇒ (    dollars'  = dollars − dollarAmount? − 1
      ∧  cents'    = cents − centAmount? + 100 )
centAmount? ≤ cents
  ⇒ (    dollars'  = dollars − dollarAmount?
      ∧  cents'   = cents − centAmount?     )
```

Example: Data dictionary entry

[NAME, DATE]
sem_model_types = { relation, entity, attribute }

```
— DataDictionaryEntry ————————————————
name: NAME
type:
sem_model_types
creation_date: DATE
description : seq Char
——————————————————————————————————
#description ≤ 2000
```

A specification to convert requirements written in natural language to Z formalspecification language method is given below. The specification depicts small operation to add, students details such as rollno, name, class, section, address into school database.

Add and lookup operations

```
┌─ Add_Error ──────────────────────────────────┐
│  Ξ DataDictionary                             │
│  entry?: DataDictionaryEntry                  │
│  error!: seq char                             │
├───────────────────────────────────────────────┤
│  entry?.name ∈ dom ddict                      │
│  error! = "Name already in dictionary"        │
└───────────────────────────────────────────────┘
```

```
┌─ Lookup_Error ───────────────────────────────┐
│  Ξ DataDictionary                             │
│  name?: NAME                                  │
│  error!: seq char                             │
├───────────────────────────────────────────────┤
│  name? ∉ dom ddict                            │
│  error! = "Name not in dictionary"            │
└───────────────────────────────────────────────┘
```

Example: Data dictionary entry

[NAME, DATE]
sem_model_types = { relation, entity, attribute }

```
┌─ DataDictionaryEntry ────────────────────────┐
│  name: NAME                                   │
│  type:                                        │
│  sem_model_types                              │
│  creation_date: DATE                          │
│  description : seq Char                        │
├───────────────────────────────────────────────┤
│  #description ≤ 2000                           │
│                                                │
└───────────────────────────────────────────────┘
```

THE END