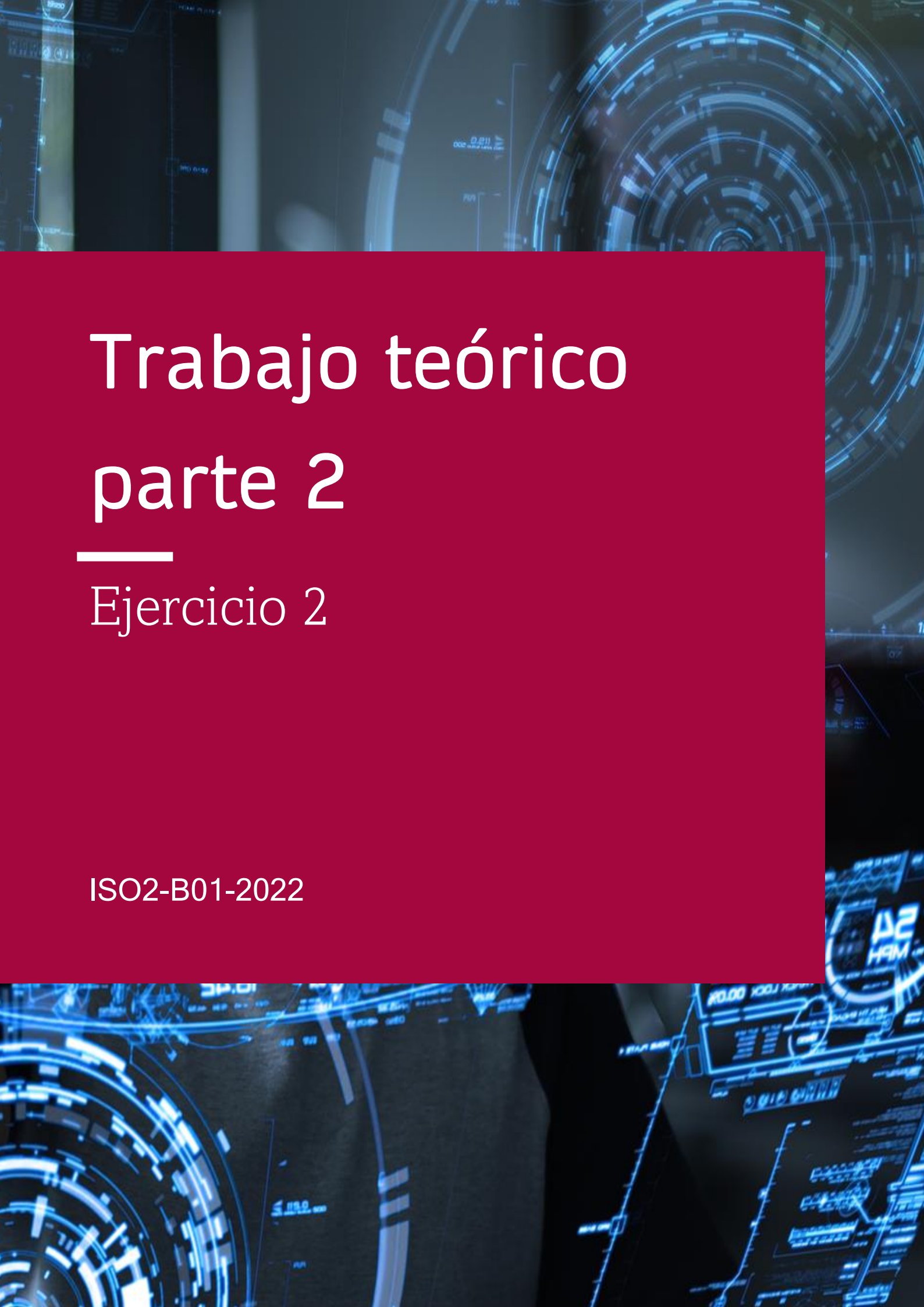

Trabajo teórico parte 2

Ejercicio 2

ISO2-B01-2022

Contenido

1.- Enunciado del ejercicio

Una empresa de certificación concede certificados del nivel de calidad de productos software, si se cumplen una serie de condiciones. Se trata de desarrollar y probar un programa que determine si un producto software puede ser certificable, y si lo tiene qué nivel de certificación obtendría. Sólo se consideran dos características de calidad: Adecuación funcional y mantenibilidad. Para ello se tendrá en cuenta las indicaciones de las siguientes tablas.

Para Adecuación funcional se tiene la siguiente tabla:

Rango Mediciones	Completitud Funcional	Corrección Funcional	Pertinencia Funcional
[0,10)	0	0	0
[10, 35)	1	1	2
[35,50)	2	1	2
[50, 70)	2	2	3
[70, 90)	3	3	4
[90,100]	4	5	5

La forma de interpretar esta tabla es considerando una función de mínimos:

$$Adecuación\ Funcional = Min\{Completitud\ Funcional, Corrección\ Funcional, Pertinencia\ Funcional\}$$

Por ejemplo, se han medido la completitud Funcional y se ha obtenido un nivel de 55, la corrección funcional se ha obtenido un nivel de 86, y se ha medido la pertinencia funcional de 19. Mirando las tablas, se puede ver que para completitud funcional se obtiene un valor equivalente de 2, de corrección funcional sería 3, y de pertinencia funcional de 1. Por tanto, la Adecuación Funcional sería Min {2, 3, 1} = 1. Para la mantenibilidad se tendría una matriz similar con cada una de las características correspondientes.

$$Mantenibilidad = Min\{Modularidad, Reusabilidad, Analizabilidad, Capacidad\ de\ ser\ modificado, Capacidad\ de\ ser\ probado\}$$

Rango Mediciones	Modularidad	Reusabilidad	Analizabilidad	Capacidad de ser modificado	Capacidad de ser probado
[0,10)	0	0	0	0	0
[10,35)	1	1	0	1	1
[35, 50)	2	2	1	2	1
[50, 70)	2	2	2	3	2
[70, 90)	3	3	3	4	4
[90, 100]	4	5	5	5	4

Finalmente, para calcular el nivel de calidad global del producto software con respecto a las dos características propuestas (solo se puede certificar cuando se obtiene un nivel 3) se usa la siguiente tabla:

		Mantenibilidad				
		1	2	3	4	5
Adecuación Funcional	1	1	1	1	1	1
	2	1	2	2	2	2
	3	2	2	3	3	3
	4	3	3	3	3	4
	5	3	3	4	4	5

Si por alguna razón, no se pudiera obtener algunas de las mediciones base para las subcaracterísticas, se lanzará la excepción correspondiente (no hace falta crear excepciones para cada tipo de subcaracterística)

2.- Pseudocódigo

Este es el código fuente que se ajustaría al enunciado

```
obtenerCompFunc (medicionuno) {

      res: entero

      res = -1;

      si (medicionuno < 0) {
            print("El valor introducido no es valido")
            exit(-1);
      } si no si (medicionuno => 0 AND medicionuno < 10) {
            res =  0;
      } si no si (medicionuno => 10 AND medicionuno < 35) {
            res =  1;
      } si no si (medicionuno => 35 AND medicionuno < 50) {
            res =  2;
```

```
        } si no si (medicionuno => 50 AND medicionuno < 70) {
              res =  2;
        } si no si (medicionuno => 70 AND medicionuno < 90) {
              res =  3;
        } si no si (medicionuno => 90 AND medicionuno <= 100) {
              res =  4;
        } si no si (medicionuno > 100) {
              print("El valor introducido no es valido")
              exit(-1);
        }

        return res;

}

obtenerCorFunc (mediciondos) {

        res: entero

        res = -1;

        si (mediciondos < 0) {
              print("El valor introducido no es valido")
              exit(-1);
        } si no si (mediciondos => 0 AND mediciondos < 10) {
              res =  0;
        } si no si (mediciondos => 10 AND mediciondos < 35) {
              res =  1;
        } si no si (mediciondos => 35 AND mediciondos < 50) {
              res =  1;
        } si no si (mediciondos => 50 AND mediciondos < 70) {
              res =  2;
        } si no si (mediciondos => 70 AND mediciondos < 90) {
              res =  3;
        } si no si (mediciondos => 90 AND mediciondos <= 100) {
              res =  5;
        } si no si (mediciondos > 100) {
              print("El valor introducido no es valido")
              exit(-1);
        }

        return res;

}

obtenerPertFunc (mediciontres) {

        res: entero

        res = -1;

        si (mediciontres < 0) {
              print("El valor introducido no es valido")
              exit(-1);
        } si no si (mediciontres => 0 AND mediciontres < 10) {
              res =  0;
        } si no si (mediciontres => 10 AND mediciontres < 35) {
              res =  2;
        } si no si (mediciontres => 35 AND mediciontres < 50) {
              res =  2;
        } si no si (mediciontres => 50 AND mediciontres < 70) {
              res =  3;
        } si no si (mediciontres => 70 AND mediciontres < 90) {
              res =  4;
        } si no si (mediciontres => 90 AND mediciontres <= 100) {
```

```
            res =  5;
      } si no si (mediciontres > 100) {
            print("El valor introducido no es valido")
            exit(-1);
      }

      return res;

}


obtenerAdecuacionFuncional (compfunc, corfunc, pertfunc): entero {

      si (compfunc < 0 OR corfunc < 0 OR pertfunc < 0) {
            println("Error: Un parametro tiene un valor por debajo de lo
permitido");
            exit(-1);
      }

      si (compfunc > 4 OR corfunc > 5 OR pertfunc > 5) {
            println("Error: Un parametro tiene un valor por encima de lo
permitido");
            exit(-1);
      }

      minimo: entero

      minimo = min (compfunc, corfunc, pertfunc);

      return minimo;
}

obtenerModu (medicioncuatro) {

      res: entero

      res = -1;

      si (medicioncuatro < 0) {
            print("El valor introducido no es valido")
            exit(-1);
      } si no si (medicioncuatro => 0 AND medicioncuatro < 10) {
            res =  0;
      } si no si (medicioncuatro => 10 AND medicioncuatro < 35) {
            res =  1;
      } si no si (medicioncuatro => 35 AND medicioncuatro < 50) {
            res =  2;
      } si no si (medicioncuatro => 50 AND medicioncuatro < 70) {
            res =  2;
      } si no si (medicioncuatro => 70 AND medicioncuatro < 90) {
            res =  3;
      } si no si (medicioncuatro => 90 AND medicioncuatro <= 100) {
            res =  4;
      } si no si (medicioncuatro > 100) {
            print("El valor introducido no es valido")
            exit(-1);
      }

      return res;

}

obtenerReusa (medicioncinco) {

      res: entero
```

```
	res = -1;

	si (medicioncinco < 0) {
		print("El valor introducido no es valido")
		exit(-1);
	} si no si (medicioncinco => 0 AND medicioncinco < 10) {
		res =  0;
	} si no si (medicioncinco => 10 AND medicioncinco < 35) {
		res =  1;
	} si no si (medicioncinco => 35 AND medicioncinco < 50) {
		res =  2;
	} si no si (medicioncinco => 50 AND medicioncinco < 70) {
		res =  2;
	} si no si (medicioncinco => 70 AND medicioncinco < 90) {
		res =  3;
	} si no si (medicioncinco => 90 AND medicioncinco <= 100) {
		res =  5;
	} si no si (medicioncinco > 100) {
		print("El valor introducido no es valido")
		exit(-1);
	}

	return res;

}

obtenerAna (medicionseis) {

	res: entero

	res = -1;

	si (medicionseis < 0) {
		print("El valor introducido no es valido")
		exit(-1);
	} si no si (medicionseis => 0 AND medicionseis < 10) {
		res =  0;
	} si no si (medicionseis => 10 AND medicionseis < 35) {
		res =  0;
	} si no si (medicionseis => 35 AND medicionseis < 50) {
		res =  1;
	} si no si (medicionseis => 50 AND medicionseis < 70) {
		res =  2;
	} si no si (medicionseis => 70 AND medicionseis < 90) {
		res =  3;
	} si no si (medicionseis => 90 AND medicionseis <= 100) {
		res =  5;
	} si no si (medicionseis > 100) {
		print("El valor introducido no es valido")
		exit(-1);
	}

	return res;

}

obtenerCapModificacion (medicionsiete) {

	res: entero

	res = -1;

	si (medicionsiete < 0) {
		print("El valor introducido no es valido")
```

```
            exit(-1);
      } si no si (medicionsiete => 0 AND medicionsiete < 10) {
            res =  0;
      } si no si (medicionsiete => 10 AND medicionsiete < 35) {
            res =  1;
      } si no si (medicionsiete => 35 AND medicionsiete < 50) {
            res =  2;
      } si no si (medicionsiete => 50 AND medicionsiete < 70) {
            res =  3;
      } si no si (medicionsiete => 70 AND medicionsiete < 90) {
            res =  4;
      } si no si (medicionsiete => 90 AND medicionsiete <= 100) {
            res =  5;
      } si no si (medicionsiete > 100) {
            print("El valor introducido no es valido")
            exit(-1);
      }

      return res;
}

obtenerCapSerProbado (medicionocho) {

      res: entero

      res = -1;

      si (medicionocho < 0) {
            print("El valor introducido no es valido")
            exit(-1);
      } si no si (medicionocho => 0 AND medicionocho < 10) {
            res =  0;
      } si no si (medicionocho => 10 AND medicionocho < 35) {
            res =  1;
      } si no si (medicionocho => 35 AND medicionocho < 50) {
            res =  1;
      } si no si (medicionocho => 50 AND medicionocho < 70) {
            res =  2;
      } si no si (medicionocho => 70 AND medicionocho < 90) {
            res =  4;
      } si no si (medicionocho => 90 AND medicionocho <= 100) {
            res =  4;
      } si no si (medicionocho > 100) {
            print("El valor introducido no es valido")
            exit(-1);
      }

      return res;
}



obtenerMantenibilidad (int modularidad, int reusabilidad, int analizabilidad,
int capmodificacion, int capserprobado): entero {

      si (modularidad < 0 OR reusabilidad < 0 OR analizabilidad < 0 OR
capmodificacion < 0 OR capserprobado < 0) {
            println("Error: Un parametro tiene un valor por debajo de lo
permitido");
            exit(-1);
      }

      si (modularidad > 4 OR reusabilidad > 5 OR analizabilidad > 5 OR
capmodificacion > 5 OR capserprobado > 4) {
```

```
            println("Error: Un parametro tiene un valor por encima de lo
permitido");
            exit(-1);
      }

      minimo: entero

      minimo = min (modularidad, reusabilidad, analizabilidad, capmodificacion,
capserprobado);

      return minimo;
}

evaluacionCalidad (int mantenibilidad, int adfuncional): boolean {

      res: entero

      res = min (mantenibilidad, adfuncional);

      if (res => 3) {
            return true;
      } else {
            return false;
      }

}
```

3.- Parámetros de entrada

Los parámetros de entrada son los siguientes:

- medicionuno con Dominio (-inf, inf)
- mediciondos con Dominio (-inf, inf)
- mediciontres con Dominio (-inf, inf)
- medicioncuatro con Dominio (-inf, inf)
- medicioncinco con Dominio (-inf, inf)
- medicionseis con Dominio (-inf, inf)
- medicionsiete con Dominio (-inf, inf)
- medicionocho con Dominio (-inf, inf)
- compfunc con Dominio (-inf, inf)
- corfunc con Dominio (-inf, inf)
- pertfunc con Dominio (-inf, inf)
- modularidad con Dominio (-inf, inf)
- reusabilidad con Dominio (-inf, inf)
- analizabilidad con Dominio (-inf, inf)
- capmodificacion con Dominio (-inf, inf)
- capserprobado con Dominio (-inf, inf)
- mantenibilidad con Dominio (-inf, inf)
- adfuncional con Dominio (-inf, inf)

4.- Máximo número de casos de prueba

Un caso de prueba es un caso donde se asigna un valor para cada parámetro de entrada. No se puede decir cuántos casos de prueba exactamente se pueden realizar ya que todos los parámetros de entrada pueden tener valores infinitos.

5.- Conjunto de valores de prueba

Esta es la tabla con todos los parámetros de entrada, sus clases de equivalencia y los valores elegidos para realizar las pruebas de testeo.

Parámetros	Clases de Equivalencia	Valores elegidos	
medicionuno	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101
mediciondos	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101

mediciontres	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101
medicioncuatro	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101
medicioncinco	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101

medicionseis	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101
mediciensiete	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101
medicionocho	(-inf, 0) [0, 10) [10, 35) [35, 50) [50, 70) [70, 90) [90, 100] (100, +inf)	Valores dentro de las clases de Equivalencia	-5, 5, 15, 40, 55, 75, 80, 95, 105
		Valores límite entre clases	0, 10, 35, 50, 70, 90, 100
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 9, 11, 34, 36, 49, 51, 69, 71, 89, 91, 99, 101

compfunc	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6
corfunc	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6
pertfunc	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6

modularidad	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6
reusabilidad	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6
analizabilidad	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6

capmodificacion	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6
capserprobado	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6
mantenibilidad	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6

adfuncional	(-inf, 0) [0, 5] (5, +inf)	Valores dentro de las clases de Equivalencia	-1, 3, 7
		Valores límite entre clases	0, 5
		Valores variante pesada (uno mas y uno menos de los valores limites)	-1, 1, 4, 6

6.- Mínimo número de casos de prueba

Suponiendo que al menos se usa un valor de cada variable, hemos calculado que son necesarios realizar como mínimo, 30 casos de prueba, número que coincide con el número de valores elegidos para las variables “medición”.

7.- Casos de prueba “each-use”

Each use es un criterio de cobertura que defiende que, cada valor debe utilizarse, al menos, en un caso de prueba. De manera que, como la variable con mayor número de valores de prueba posee 30 valores en total, debemos representar 30 conjuntos de casos de prueba.

Esto se debe a que, de esta manera, utilizamos al menos 1 vez cada uno de los valores de la variable mediciónuno en un conjunto diferente. Esto queda representado de la siguiente manera:

- Conjunto 1: (-5, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 2: (5, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 3: (15, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 4: (40, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 5: (55, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 6: (75, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 7: (80, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 8: (95, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 9: (105, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)

- Conjunto 10: (0, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 11: (10, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 12: (35, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 13: (50, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 14: (70, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 15: (90, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 16: (100, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 17: (-1, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 18: (1, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 19: (9, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 20: (11, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 21: (34, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 22: (36, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 23: (49, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 24: (51, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 25: (69, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 26: (71, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 27: (89, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 28: (91, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 29: (99, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)
- Conjunto 30: (101, 5, 40, 55, 80, 105, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1)

8.- Casos de prueba “Pairwise”

Pairwise es un criterio de cobertura que defiende que, los casos de prueba deben visitar, al menos una vez, todos los pares de valores de dos parámetros cualesquiera. De manera que, cada uno de los valores obtenidos para las variables anteriores, deben relacionarse al menos una vez, con los valores obtenidos para el resto de las variables.

Al ser muchísimos casos de uso, hemos previsto crear una tabla que refleje este criterio de cobertura,

En nuestro caso, hemos utilizado intervalos para representar este criterio de cobertura ya que, tenemos una gran cantidad de valores para las variables. Por tanto, hemos pensado que esta es la forma más ordenada y visual de representar dicho criterio de cobertura.

Las diferentes tablas que aparecen son una única tabla, pero se han partido por razones de publicación

medicion uno	medicion dos	medicion tres	medicionc uatro	medicionc inco	medicion seis	medicions iete	mediciono cho
(-inf, 0)	(-inf, 0)	(-inf, 0)	(-inf, 0)	(-inf, 0)	(-inf, 0)	(-inf, 0)	(-inf, 0)
[0,10)	[0,10)	[0,10)	[0,10)	[0,10)	[0,10)	[0,10)	[0,10)
[10,35)	[10,35)	[10,35)	[10,35)	[10,35)	[10,35)	[10,35)	[10,35)
[35,50)	[35,50)	[35,50)	[35,50)	[35,50)	[35,50)	[35,50)	[35,50)
[50,70)	[50,70)	[50,70)	[50,70)	[50,70)	[50,70)	[50,70)	[50,70)
[70,90)	[70,90)	[70,90)	[70,90)	[70,90)	[70,90)	[70,90)	[70,90)
[90,100]	[90,100]	[90,100]	[90,100]	[90,100]	[90,100]	[90,100]	[90,100]
(100, +inf)	(100, +inf)	(100,+inf)	(100, +inf)	(100, +inf)	(100, +inf)	(100, +inf)	(100, +inf)

compf unc	corf unc	pertf unc	modula ridad	reusabi lidad	analizab ilidad	capmodifi cacion	capserpr obado	manteni bilidad	adfunc ional
(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)	(-inf,0)
[0,5]	[0,5]	[0,5]	[0,5]	[0,5]	[0,5]	[0,5]	[0,5]	[0,5]	[0,5]
(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)	(5,+inf)

9.- Cobertura de Decisiones

En primer lugar, hemos designado como variables A, B, C, D, E, F, G y H como las variables que representan los condicionales en los métodos que obtienen las variables que deciden la Mantenibilidad y la Adecuación Funcional

De las 256 filas existentes, solo son correctas en las que una de las variables esté a Verdadero y el resto a Falso, ya que no se pueden cumplir todas las condiciones a la vez.

A	B	C	D	E	F	G	H	A or B or C or D or E or F or G or H
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	1	1
0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	1	1
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	0	0	1
0	0	0	0	1	0	0	1	1
0	0	0	0	1	0	1	0	1
0	0	0	0	1	0	1	1	1
0	0	0	0	1	1	0	0	1
0	0	0	0	1	1	0	1	1
0	0	0	0	1	1	1	0	1
0	0	0	0	1	1	1	1	1
0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	0	1	0	0	1	0	1
0	0	0	1	0	0	1	1	1
0	0	0	1	0	1	0	0	1
0	0	0	1	0	1	0	1	1
0	0	0	1	0	1	1	0	1
0	0	0	1	0	1	1	1	1
0	0	0	1	1	0	0	0	1
0	0	0	1	1	0	0	1	1
0	0	0	1	1	0	1	0	1
0	0	0	1	1	0	1	1	1
0	0	0	1	1	1	0	0	1
0	0	0	1	1	1	0	1	1
0	0	0	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	0	0	0	1	1	1

0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	1	1
0	0	1	0	0	1	1	0	1
0	0	1	0	0	1	1	1	1
0	0	1	0	1	0	0	0	1
0	0	1	0	1	0	0	1	1
0	0	1	0	1	0	1	0	1
0	0	1	0	1	0	1	1	1
0	0	1	0	1	1	0	0	1
0	0	1	0	1	1	0	1	1
0	0	1	0	1	1	1	0	1
0	0	1	0	1	1	1	1	1
0	0	1	1	0	0	0	0	1
0	0	1	1	0	0	0	1	1
0	0	1	1	0	0	1	0	1
0	0	1	1	0	0	1	1	1
0	0	1	1	0	1	0	0	1
0	0	1	1	0	1	0	1	1
0	0	1	1	0	1	1	0	1
0	0	1	1	0	1	1	1	1
0	0	1	1	1	0	0	0	1
0	0	1	1	1	0	0	1	1
0	0	1	1	1	0	1	0	1
0	0	1	1	1	0	1	1	1
0	0	1	1	1	1	0	0	1
0	0	1	1	1	1	0	1	1
0	0	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	1
0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	0	1
0	1	0	0	0	1	0	1	1
0	1	0	0	0	1	1	0	1
0	1	0	0	0	1	1	1	1
0	1	0	0	1	0	0	0	1

0	1	0	0	1	0	0	1	1
0	1	0	0	1	0	1	0	1
0	1	0	0	1	0	1	1	1
0	1	0	0	1	1	0	0	1
0	1	0	0	1	1	0	1	1
0	1	0	0	1	1	1	0	1
0	1	0	0	1	1	1	1	1
0	1	0	1	0	0	0	0	1
0	1	0	1	0	0	0	1	1
0	1	0	1	0	0	1	0	1
0	1	0	1	0	0	1	1	1
0	1	0	1	0	1	0	0	1
0	1	0	1	0	1	0	1	1
0	1	0	1	0	1	1	0	1
0	1	0	1	0	1	1	1	1
0	1	0	1	1	0	0	0	1
0	1	0	1	1	0	0	1	1
0	1	0	1	1	0	1	0	1
0	1	0	1	1	0	1	1	1
0	1	0	1	1	1	0	0	1
0	1	0	1	1	1	0	1	1
0	1	0	1	1	1	1	0	1
0	1	0	1	1	1	1	1	1
0	1	1	0	0	0	0	0	1
0	1	1	0	0	0	0	1	1
0	1	1	0	0	0	1	0	1
0	1	1	0	0	0	1	1	1
0	1	1	0	0	1	0	0	1
0	1	1	0	0	1	0	1	1
0	1	1	0	0	1	1	0	1
0	1	1	0	0	1	1	1	1
0	1	1	0	1	0	0	0	1
0	1	1	0	1	0	0	1	1
0	1	1	0	1	0	1	0	1
0	1	1	0	1	0	1	1	1
0	1	1	0	1	1	0	0	1
0	1	1	0	1	1	0	1	1

0	1	1	0	1	1	1	0	1
0	1	1	0	1	1	1	1	1
0	1	1	1	0	0	0	0	1
0	1	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0	1
0	1	1	1	0	0	1	1	1
0	1	1	1	0	1	0	0	1
0	1	1	1	0	1	0	1	1
0	1	1	1	0	1	1	0	1
0	1	1	1	0	1	1	1	1
0	1	1	1	1	0	0	0	1
0	1	1	1	1	0	0	1	1
0	1	1	1	1	0	1	0	1
0	1	1	1	1	0	1	1	1
0	1	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	1
1	0	0	0	0	0	1	0	1
1	0	0	0	0	0	1	1	1
1	0	0	0	0	1	0	0	1
1	0	0	0	0	1	0	1	1
1	0	0	0	0	1	1	0	1
1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	1	1
1	0	0	0	1	0	1	0	1
1	0	0	0	1	0	1	1	1
1	0	0	0	1	1	0	0	1
1	0	0	0	1	1	0	1	1
1	0	0	0	1	1	1	0	1
1	0	0	0	1	1	1	1	1
1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	1	1
1	0	0	1	0	0	1	0	1

1	0	0	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	0	1	0	1	0	1	1
1	0	0	1	0	1	1	0	1
1	0	0	1	0	1	1	1	1
1	0	0	1	1	0	0	0	1
1	0	0	1	1	0	0	1	1
1	0	0	1	1	0	1	0	1
1	0	0	1	1	0	1	1	1
1	0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	1	1
1	0	0	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1
1	0	1	0	0	0	0	0	1
1	0	1	0	0	0	0	1	1
1	0	1	0	0	0	1	0	1
1	0	1	0	0	0	1	1	1
1	0	1	0	0	1	0	0	1
1	0	1	0	0	1	0	1	1
1	0	1	0	0	1	1	0	1
1	0	1	0	0	1	1	1	1
1	0	1	0	1	0	0	0	1
1	0	1	0	1	0	0	1	1
1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	1	1
1	0	1	0	1	1	0	0	1
1	0	1	0	1	1	0	1	1
1	0	1	0	1	1	1	0	1
1	0	1	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	0	1	1	0	0	0	1	1
1	0	1	1	0	0	1	0	1
1	0	1	1	0	0	1	1	1
1	0	1	1	0	1	0	0	1
1	0	1	1	0	1	0	1	1
1	0	1	1	0	1	1	0	1
1	0	1	1	0	1	1	1	1

1	0	1	1	1	0	0	0	1
1	0	1	1	1	0	0	1	1
1	0	1	1	1	0	1	0	1
1	0	1	1	1	0	1	1	1
1	0	1	1	1	1	0	0	1
1	0	1	1	1	1	0	1	1
1	0	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	1
1	1	0	0	0	0	0	1	1
1	1	0	0	0	0	1	0	1
1	1	0	0	0	0	1	1	1
1	1	0	0	0	1	0	0	1
1	1	0	0	0	1	0	1	1
1	1	0	0	0	1	1	0	1
1	1	0	0	0	1	1	1	1
1	1	0	0	1	0	0	0	1
1	1	0	0	1	0	0	1	1
1	1	0	0	1	0	1	0	1
1	1	0	0	1	0	1	1	1
1	1	0	0	1	1	0	0	1
1	1	0	0	1	1	0	1	1
1	1	0	0	1	1	1	0	1
1	1	0	0	1	1	1	1	1
1	1	0	1	0	0	0	0	1
1	1	0	1	0	0	0	1	1
1	1	0	1	0	0	1	0	1
1	1	0	1	0	0	1	1	1
1	1	0	1	0	1	0	0	1
1	1	0	1	0	1	0	1	1
1	1	0	1	0	1	1	0	1
1	1	0	1	0	1	1	1	1
1	1	0	1	1	0	0	0	1
1	1	0	1	1	0	0	1	1
1	1	0	1	1	0	1	0	1
1	1	0	1	1	0	1	1	1
1	1	0	1	1	1	0	0	1

1	1	0	1	1	1	0	1	1
1	1	0	1	1	1	1	0	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	1
1	1	1	0	0	0	0	1	1
1	1	1	0	0	0	1	0	1
1	1	1	0	0	0	1	1	1
1	1	1	0	0	1	0	0	1
1	1	1	0	0	1	0	1	1
1	1	1	0	0	1	1	0	1
1	1	1	0	0	1	1	1	1
1	1	1	0	1	0	0	0	1
1	1	1	0	1	0	0	1	1
1	1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	1	1
1	1	1	0	1	1	0	0	1
1	1	1	0	1	1	0	1	1
1	1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	1	1
1	1	1	1	0	0	1	0	1
1	1	1	1	0	0	1	1	1
1	1	1	1	0	1	0	0	1
1	1	1	1	0	1	0	1	1
1	1	1	1	0	1	1	0	1
1	1	1	1	0	1	1	1	1
1	1	1	1	1	0	0	0	1
1	1	1	1	1	0	0	1	1
1	1	1	1	1	0	1	0	1
1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1

En segundo lugar, hemos designado como variables A y B a las variables que representan los condicionales en los métodos que obtienen la Mantenibilidad y la Adecuación Funcional

De las 4 filas existentes, solo son correctas en las que una de las variables esté a Verdadero y la otra a Falso, o ambas en falso ya que no se pueden cumplir todas las condiciones a la vez.

A	B	A o B
0	0	0
0	1	1
1	0	1
1	1	1

10.- Cobertura Modificada de Condiciones /Decisiones

Teniendo en cuenta la primera tabla y sabiendo que hemos designado como variables A, B, C, D, E, F, G y H como las variables que representan los condicionales en los métodos que obtienen las variables que deciden la Mantenibilidad y la Adecuación Funcional

De las 256 filas existentes, solo existe dos tipos posibles de caso que pueda adecuarse a esta sección. La primera es la primera fila, que es la única que satisface cuando cada posible valor de una condición determina la salida de la decisión al menos una vez, siendo en este caso todas las variables dominantes.

El segundo caso, en el que cada variable que de un valor positivo (1) o verdadero y todas las demás sean 0 o negativo serán situaciones en la que la susodicha variable determinará dada su condición, la decisión final.