Task 2: Souce code complexity

CookPit

Halstead’s program volume

The Halstead’s program volume is a metric to calculate the complexity of a programm without running it directly

You can calculate:
- Difficulty
- Effort (time)
- Implementation time
- Volume
- Length
- Vocabular

What you need:

-All Operands (Identifier, Typename, Typespec and Constants)

- All Operators (+,-,:,*,==,=>,...etc.)

Parameters needed	Description
n1	Number of distinct operators
n2	Number of distinct operands
N1	Total number of operators
N2	Total number of operands

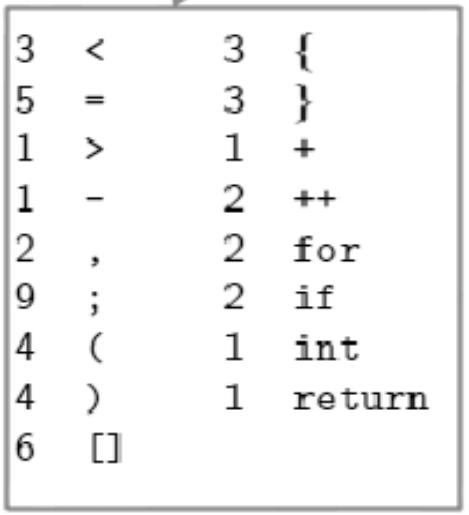
Value	Formula
Halstead Difficulty (D)	D = (n1 / 2) * (N2 / n2)
Halstead Length (N)	N = N1 + N2
Halstead CalculatedLength (Nx)	Nx = n1 * log2(n1) + n2 * log2(n2)
Halstead Volume (V)	V = N * log2(n)
Halstead Effort (E)	E = V * D
Halstead Vocabulary (n)	n = n1 + n2

```
void sort ( int *a, int n ) {
int i, j, t;

  if ( n < 2 ) return;
  for ( i=0 ; i < n-1; i++ )  {
      for ( j=i+1 ; j < n ; j++ ) {
          if ( a[i] > a[j] ) {
              t = a[i];
              a[i] = a[j];
              a[j] = t;
          }
      }
  }
}
```

$V = 80 \log_2(24) \approx 392$

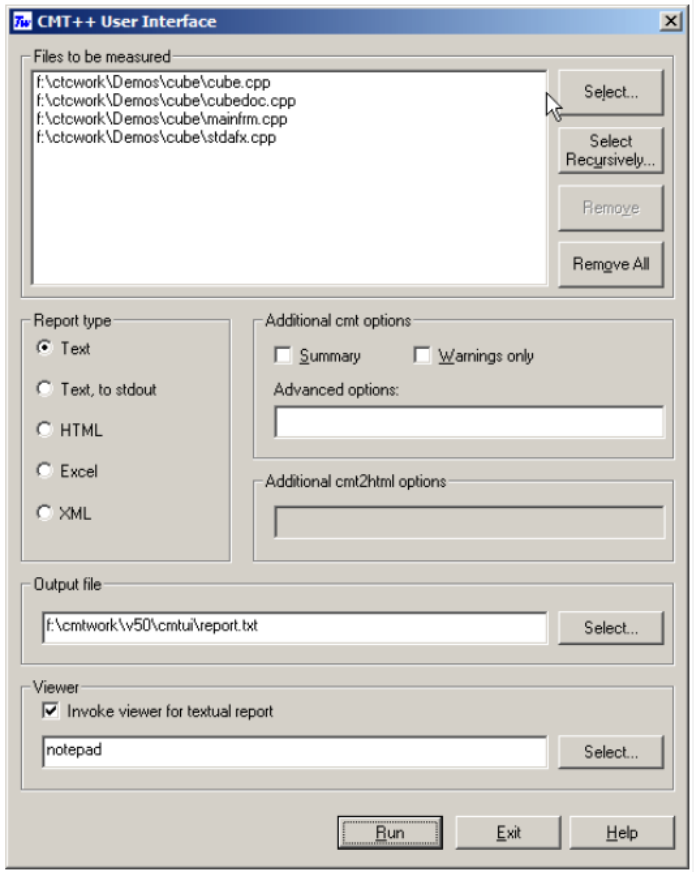
- Ignore the function definition
- Count operators and operands



3	<	3	{
5	=	3	}
1	>	1	+
1	-	2	++
2	,	2	for
9	;	2	if
4	(	1	int
4	)	1	return
6	[]		

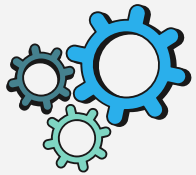
1	0
2	1
1	2
6	a
8	i
7	j
3	n
3	t

	Total	Unique
Operators	N1 = 50	n1 = 17
Operands	N2 = 30	n2 = 7



Software for automatic calculation of Halstead's program volume

Cyclomatic complexity



Cyclomatic complexity is used to calculate the quantitive complexity of a program

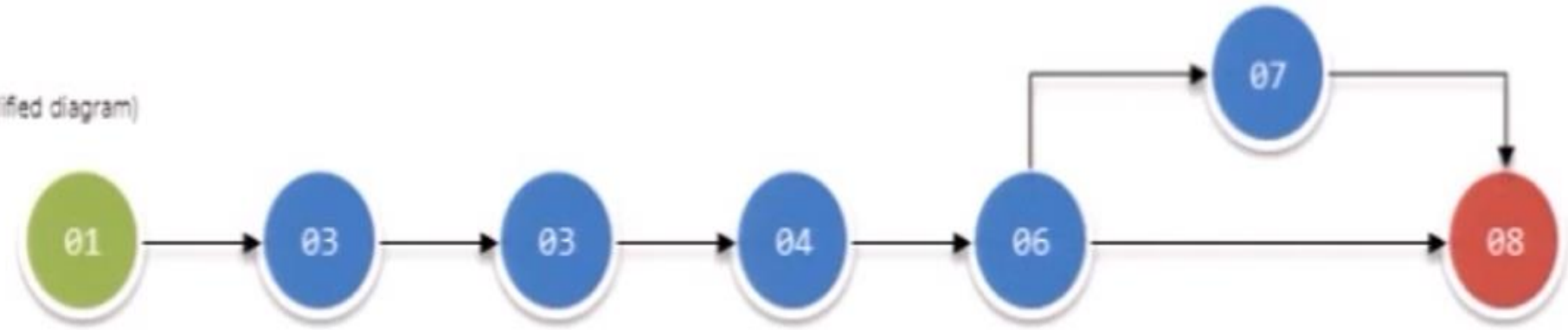
You can calculate by the amount of the atomic conditions or by calculating it with a control flowgraph using it nodes.

Z = c +1;
M = e – n + 2p;

Z, M	=	complexity
c	=	atomic conditions
e	=	number of edges
n	=	number of nodes
p	=	number of connected components

```
01 void Timer::restart()
02 {
03     const bool wasRunning(m_time->isRunning());
04     m_time->reset();
05
06     if (wasRunning)
07         m_time->start();
08 }
```

(simplified diagram)



```
01  void GraphicsItem::setScheme(ColorScheme * scheme)
02  {
03      m_scheme = scheme;
04
05      updateBrushes();
06      updateTooltips();
07  }
```

(simplified diagram)

```
1  int berechne(int x){
2
3      int ergebnis = 0;
4      if (x % 2 == 0) {
5          ergebnis = 2 * x;
6      } else {
7          ergebnis = 3 * x;
8      return ergebnis;
9  }
```

```
1  int berechne(int x){
2
3      int ergebnis = 0;
4      switch(x){
5          case 0: ergebnis = 1; break;
6          case 2: ergebnis = 17; break;
7          case 3: ergebnis = 10; break;
8          default: ergebnis = 3; break;
9      }
10     return ergebnis;
11 }
```

```
1  void regelBestimmen(int x){
2      if (x < 35)      {// Regel 1; return}
3      else if (x < 50) {// Regel 2; return}
4      else if (x < 100){// Regel 3; return}
5      else if (x > 200){// Sonderregel}
6      return;
7  }
```

```
1  int berechne(int x){
2
3      int ergebnis = 0;
4      if ((x % 2 == 0) && (x > 0)) {
5          ergebnis = 2 * x;
6      } else {
7          ergebnis = 3 * x;
8      return ergebnis;
9  }
```

Software for automatic calculation of Halstead's program volume

ndepend

&

Weighted methods per class (WMC)

The WMC metric is defined as the sum of complexities of all methods declared in a class. This metric is a good indicator how much effort will be necessary to maintain and develop a particular class.

There are three methods to calculate:
- McCabe's Cyclomatic Complexity [#mcccn]_
- Lines of Code
- 1 (Number Of Methods or Unweighted WMC)

A good value for the WMC calculation is about 35, bad values are about 36 and higher

Weighted Methods per Class - Example

Using McCabe's cyclomatic complexity.

Class: Vector

Method:	Insert	Remove	Sort	isEmpty	Clear	Find
Complexity:	3	2	4	2	1	3

Class complexity: WMC = 3 + 2 + 4 + 2 + 1 + 3 = 15