


Konstruksi Perangkat Lunak

PERTEMUAN 7 **Runtime Configuration**




Ekspektasi Luaran

Mampu menerapkan teknologi runtime configuration dalam konstruksi perangkat lunak.



Outline

- Motivasi
 - Compile time vs Runtime Binding
 - Contoh code
 - Final takes
- 

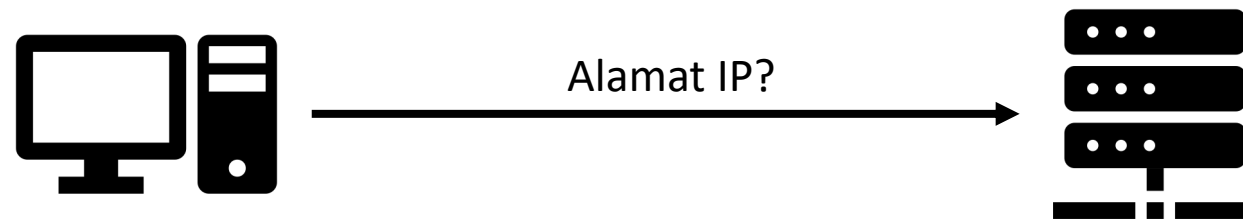
Runtime Configuration

Motivasi



Apa yang harus dilakukan?

Sebuah sistem software akan dibuat menggunakan arsitektur client-server. IP dari server baru dapat ditentukan pada saat instalasi di lapangan.




Solusi “Hard-code”

Solusi paling sederhana, buat constant untuk diedit nantinya:

```
const String ipAddress = "123.123.123.123";
```

Apa masalahnya?

- Source code harus tersedia saat instalasi.
 - Waktu instalasi menjadi lebih lama karena perlu melakukan kompilasi source code terlebih dahulu.
 - Jika IP server akan berubah, maka perlu dilakukan kompilasi ulang lagi.
- 

Solusinya

Membuat variable yang nilai awalnya ditentukan pada saat program berjalan:

runtime configuration

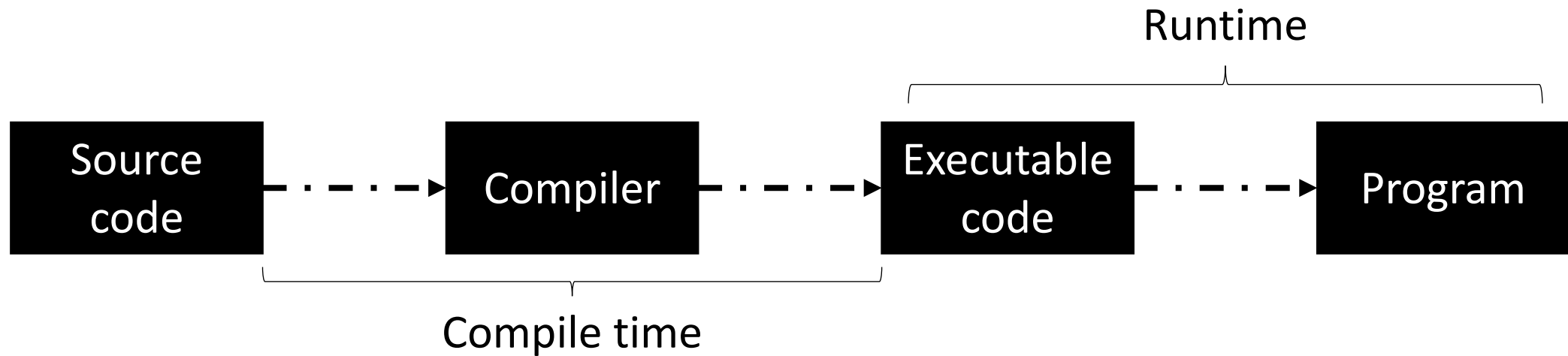


Runtime Configuration

Compile time vs Runtime Binding



Compile time dan Runtime



Binding time

“the time at which the variable and its value are bound together (Thimbleby 1988).”

```
const String ipAddress = "123.123.123.123";
```



Binding time

Code-writing time


```
private String ipAddress = "123.123.123.123";
```

Compile time

```
const String ipAddress = "123.123.123.123";
```

Runtime

```
public String ipAddress { get; set; }  
  
public ServerConfig()  
{  
    ipAddress = readIPServerConfig();  
}
```



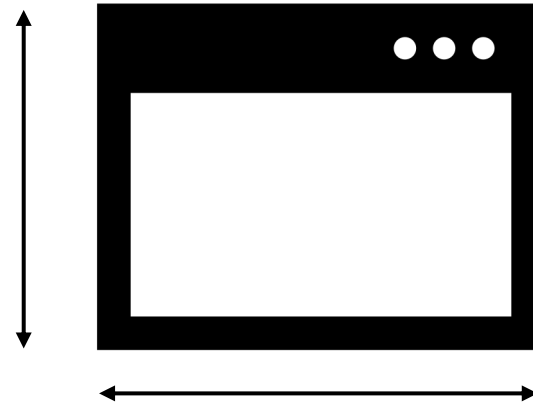
Runtime Configuration

Contoh Kode

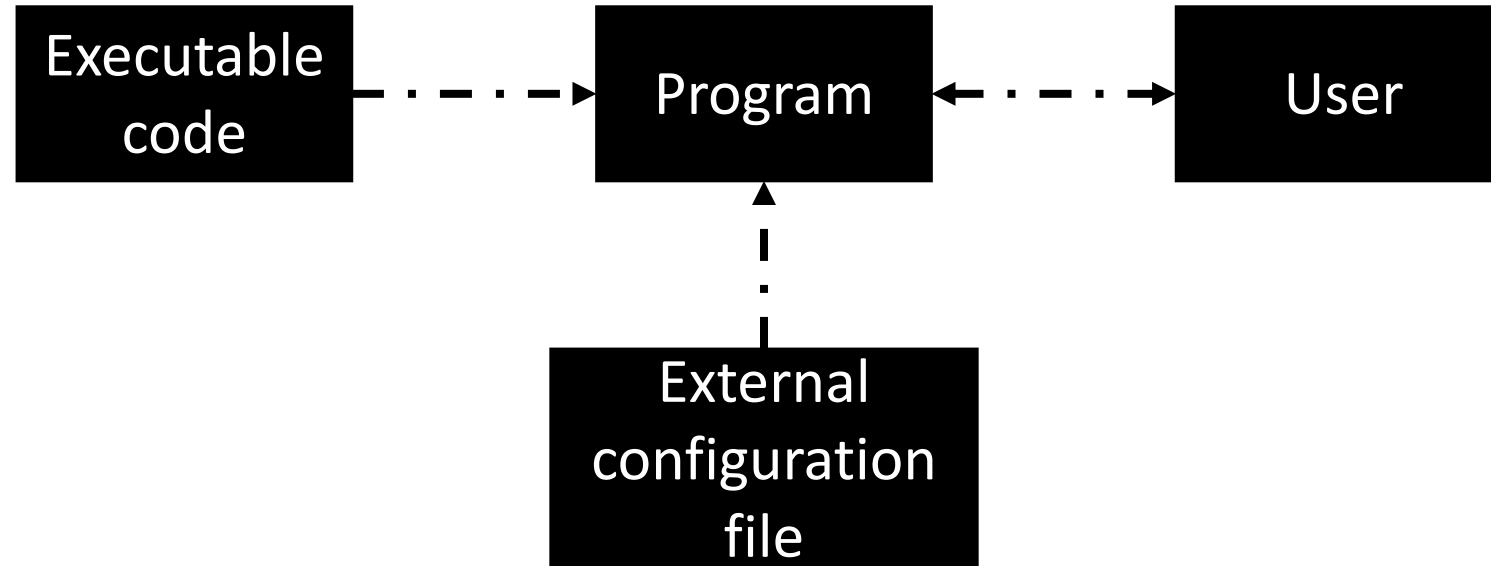


Contoh Kasus

Runtime configuration untuk mengatur dimensi dari layar aplikasi.



Runtime Binding Architecture



Step 1. Membuat Class Penampung Konfigurasi

```
class Config
{
    //Attribute untuk diserialisasi
    public int Height { get; set; }
    public int Width { get; set; }

    //Constructor kosong untuk deserialisasi
    public Config() { }

    public Config(int width, int height)
    {
        Height = height;
        Width = width;
    }
}
```

Step 2. Membuat Class Untuk Membaca dan Menulis File Konfigurasi

```
class UIConfig
{
    public Config config;

    public const String filePath = @"config.json";

    public UIConfig(){ ... }

    private Config ReadConfigFile(){ ... }

    private void SetDefault(){ ... }

    private void WriteNewConfigFile(){ ... }
}
```


Step 2.1. Membuat method untuk membaca file konfigurasi

```
private Config ReadConfigFile()
{
    String configJsonData = File.ReadAllText(filePath);
    config = JsonSerializer.Deserialize<Config>(configJsonData);
    return config;
}
```

Step 2.2. Membuat method untuk menulis file konfigurasi

```
private void WriteNewConfigFile()
{
    JsonSerializerOptions options = new JsonSerializerOptions()
    {
        WriteIndented = true
    };

    String jsonString = JsonSerializer.Serialize(config, options);
    File.WriteAllText(filePath, jsonString);
}
```

Step 2.3. Membuat method untuk membaca file dan menulis file baru jika belum ada

```
public UIConfig()
{
    try
    {
        ReadConfigFile();
    }
    catch (Exception)
    {
        SetDefault();
        WriteNewConfigFile();
    }
}
```























Step 3. Memanggil class-class yang sudah dibuat untuk memberikan data konfigurasi

Pada class Form UI

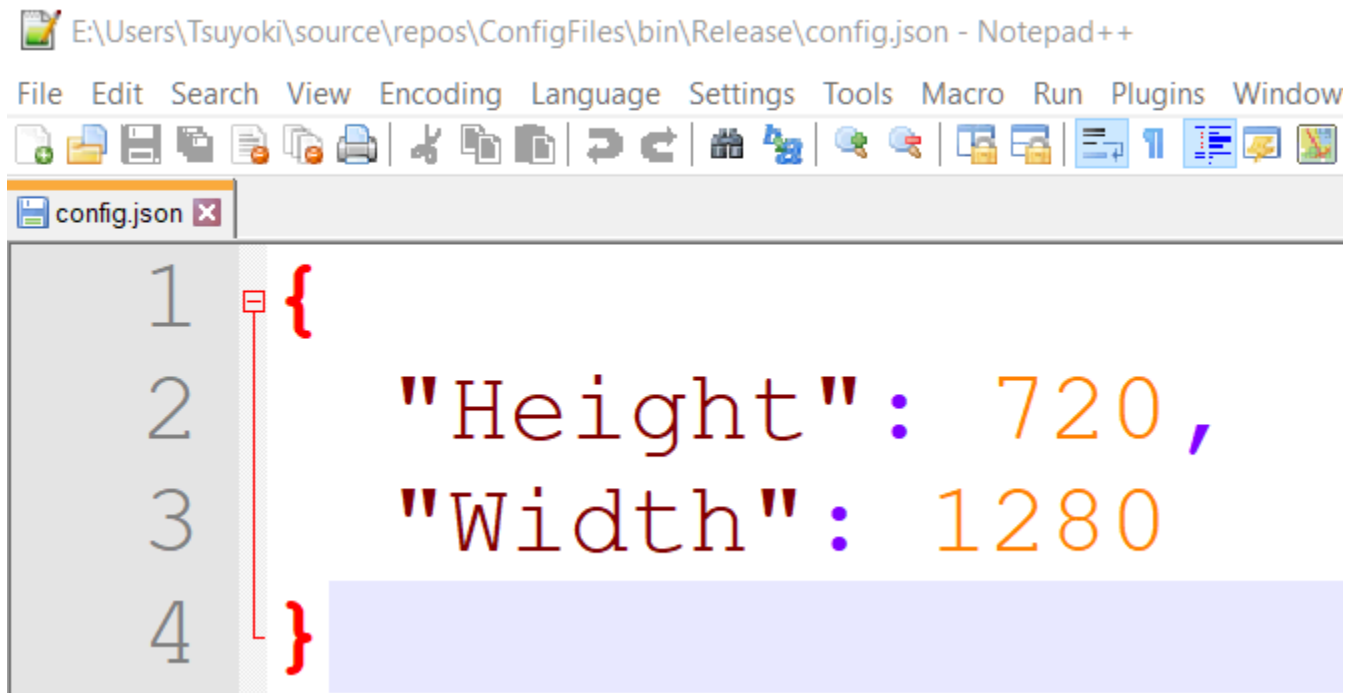
```
private void Form1_Load(object sender, EventArgs e)
{
    //Membuat object konfigurasi
    UIConfig uIConfig = new UIConfig();

    //Membaca data konfigurasi untuk digunakan untuk setting
    Height = uIConfig.config.Height;
    Width = uIConfig.config.Width;
}
```

Hasil kompilasi program

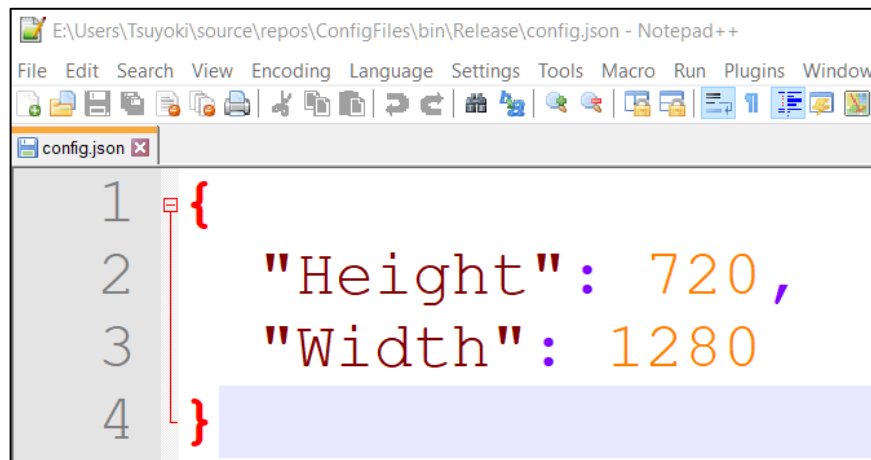
 config.json	3/26/2021 1:29 PM	JSON File	1 KB
 ConfigFiles.exe	3/28/2021 9:35 PM	Application	10 KB
 ConfigFiles.exe.config	3/26/2021 12:52 PM	XML Configuration...	1 KB
 ConfigFiles.pdb	3/28/2021 9:35 PM	Program Debug D...	36 KB
 Microsoft.Bcl.AsyncInterfaces.dll	10/19/2020 6:40 PM	Application extens...	21 KB
 Microsoft.Bcl.AsyncInterfaces.xml	10/19/2020 6:40 PM	XML Document	18 KB
 System Buffers.dll	2/19/2020 10:05 A...	Application extens...	21 KB
 System Buffers.xml	2/19/2020 10:05 A...	XML Document	4 KB
 System.Memory.dll	2/19/2020 10:05 A...	Application extens...	138 KB
 System.Memory.xml	2/19/2020 10:05 A...	XML Document	14 KB
 System.Numerics.Vectors.dll	5/15/2018 1:29 PM	Application extens...	114 KB
 System.Numerics.Vectors.xml	5/15/2018 1:29 PM	XML Document	180 KB
 System.Runtime.CompilerServices.Unsafe....	10/19/2020 6:46 PM	Application extens...	17 KB
 System.Runtime.CompilerServices.Unsafe....	10/9/2020 8:10 PM	XML Document	18 KB
 System.Text.Encodings.Web.dll	10/19/2020 6:40 PM	Application extens...	65 KB
 System.Text.Encodings.Web.xml	10/9/2020 8:10 PM	XML Document	62 KB
 System.Text.Json.dll	12/12/2020 2:42 A...	Application extens...	348 KB
 System.Text.Json.xml	10/9/2020 8:10 PM	XML Document	239 KB
 System.Threading.Tasks.Extensions.dll	2/19/2020 10:05 A...	Application extens...	26 KB
 System.Threading.Tasks.Extensions.xml	2/19/2020 10:05 A...	XML Document	10 KB
 System.ValueTuple.dll	5/15/2018 1:29 PM	Application extens...	25 KB
 System.ValueTuple.xml	5/15/2018 1:29 PM	XML Document	1 KB

Configuration Files yang Dihasilkan



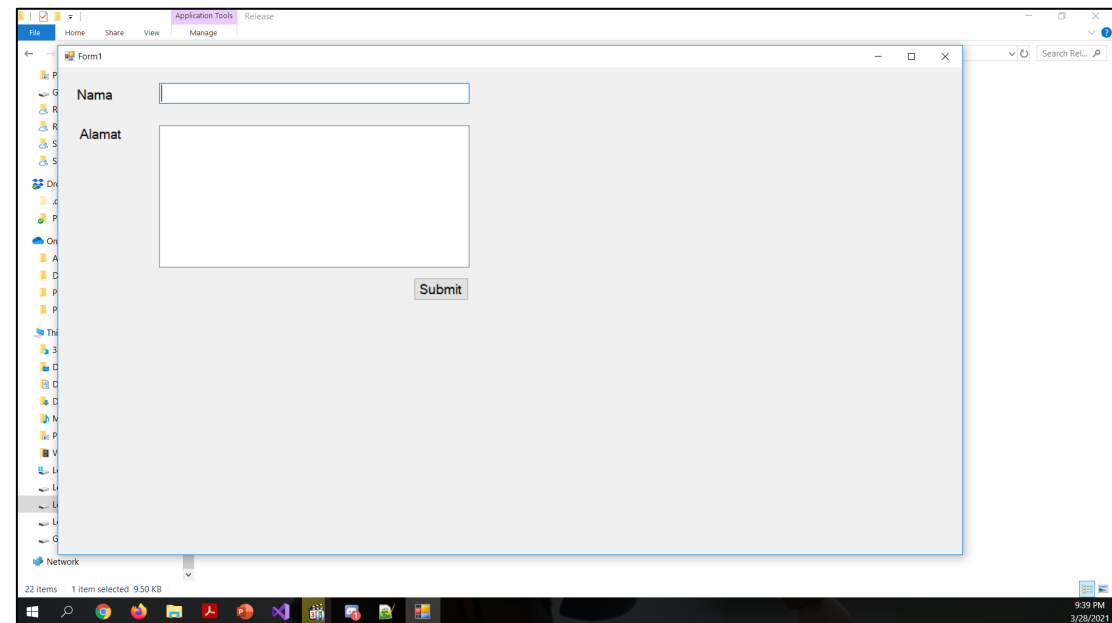
```
E:\Users\Tsuyoki\source\repos\ConfigFiles\bin\Release\config.json - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window  
config.json  
1 {  
2     "Height": 720,  
3     "Width": 1280  
4 }
```

Configuration Files yang Dihasilkan

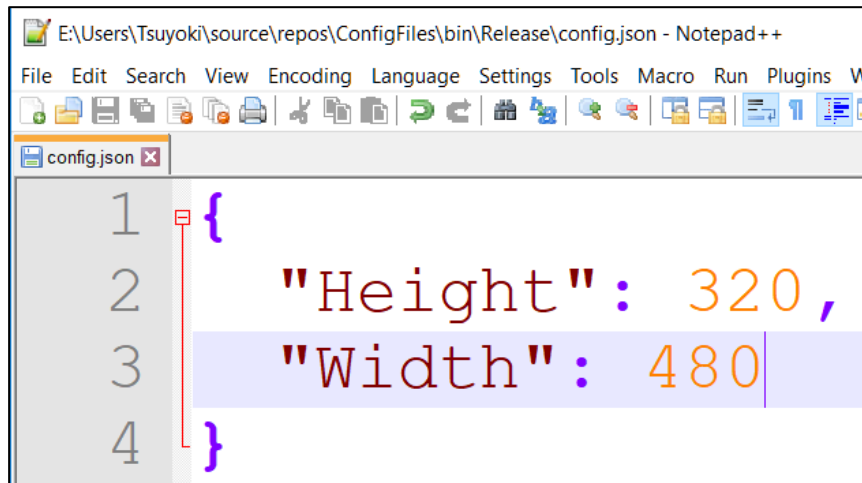


The screenshot shows a Notepad++ window with the file path `E:\Users\TsuYoki\source\repos\ConfigFiles\bin\Release\config.json`. The file content is a JSON object with two properties: `"Height": 720` and `"Width": 1280`. The text is color-coded: opening and closing curly braces are red, quotes are dark red, colons are purple, and the values are orange. Line numbers 1 through 4 are visible on the left margin.

```
1 {  
2   "Height": 720,  
3   "Width": 1280  
4 }
```

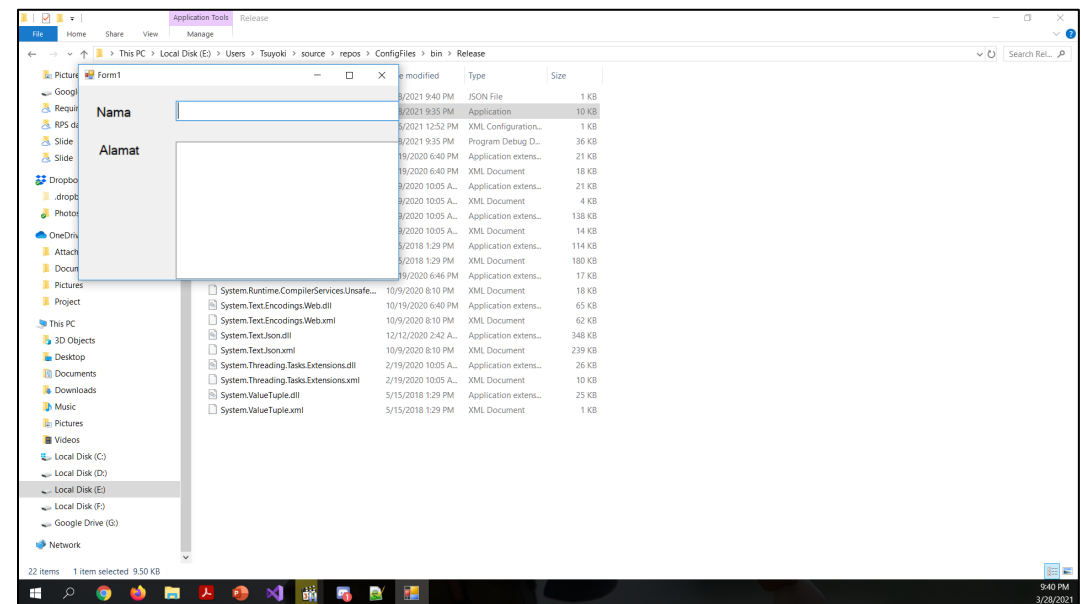


Configuration Files yang Dihasilkan



The screenshot shows a Notepad++ window with the file path `E:\Users\Tsuynoki\source\repos\ConfigFiles\bin\Release\config.json`. The file content is a JSON object with two properties: `"Height": 320` and `"Width": 480`. The text is color-coded: opening and closing curly braces are purple, quotes are red, colons are blue, and the values are orange. A vertical line is positioned at the end of the second line.

```
1 {  
2   "Height": 320,  
3   "Width": 480  
4 }
```




Runtime Configuration

Final Takes



Final Takes

- Gunakan runtime configuration untuk memungkinkan fleksibilitas dari perangkat lunak yang dibangun.
 - Ada baiknya konsep defensive programming diterapkan juga untuk *safety* ketika membaca file konfigurasi yang mungkin diubah dengan tidak benar.
- 

Q&A

Any questions or comments?



Further reading

- SWEBOK V3.0
 - S. McConnell, Code Complete 2nd Edition, Microsoft Press, 1993.
Chapter 10.6
- 