Reflection :— Accessing type-metadata dynamically from any assembly with/without reference added to our application.

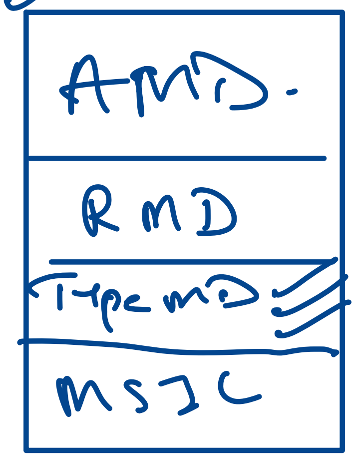
tool/tech.

[Reading type-metadata at Runtime]

```
class CMath
int Add(x,y)
int Sub(x,y)
```

myMath.dll

"D:\\ .... \.dll"



[Serializable]

```
Assembly asm = Assembly.LoadFrom( Pathof dll)
Type[] types = asm.GetTypes();
    Type type = types[0];
        (-mymath.CMath)
MethodInfo[] methods = type.GetMethods()
    { Add, sub}
    MethodInfo method1 = methods[0]
        Add.
  MethodInfo meth2 = methods[1]
        Sub

ParameterInfo[] params = method1.GetParameters();
MemberInfo [] members = type.GetMembers();
PropertyInfo [] properties = type.GetProperties();
ConstructorInfo [] ctors = type.GetConstructors();
```

```
Assembly asm = Assembly.LoadFrom("  ");
Type[] types = asm.GetTypes();
Type type = types[0]

type.isClass = true;
type.isPublic = true;
type.isSealed = false;
type.isPrivate = false;
    .isProtected = false;
    .isInherited = false;
    .isStatic = false;
    .isInterface = false;
    .isSerializable = true;
MethodInfo[] mths = type.GetMethods()
MethodInfo method = methods[0];
method.Name = "Add"
method.ReturnType = "System.Int32";
```

ParameterInfo [] allparams = method.GetParameters();

x, y

ParameterInfo para1 = allparameters [0];
(x)

para1. ParameterType = "System.Int32";

para1. isRef = false.

para1. isValue = true;

para1. Name = "x"

TypeName = "CMath"

MethodName = "Add"

Dynamically obj = obj. Add( )