The component oriented software technique in .NET and applications

Nguyen Lan Oanh, Duong Thi Quy, Tran Hai Thanh, Nguyen Thu Phuong

TNU - University of Information and Communication Technology, Thai Nguyen, Viet Nam

ABSTRACT

Component-oriented programming technique is a new theory developed from object-oriented programming. Component-oriented programming provides a more reusable structure and better modularity, with greater flexibility than object-oriented programming and library-based approaches. The article aims to study the main characteristics of component-oriented programming methods and software construction applications to help monitor and manage computer use via the Internet.

Keyword: *Component programming, Components, Components in Java, Components in .NET, Component oriented software*

1. INTRODUCTION

The phrase "component" has existed in the computer industry for a long time. As a matter of fact, the concept of components existed even before computers were invented. A builder uses ingredients from other industries to build a house. The most common real-world example is that car manufacturers use many components from other industries to produce a car. In the computer hardware industry, design engineers don't need to produce basic hardware components from scratch for each product. Processor chips, memory chips, circuit boards, and network cards are available to build a powerful and powerful mainframe system.

Component Oriented Programming (COP) allows programs to be built from existing software structures, by reusing, self-healing blocks of computer code. Components are subject to certain standards, including interfaces, connections, development, and deployment of components in a variety of shapes and sizes, from small-sized application components that can be delivered direct intermediate translation, to huge components containing extended functions. In principle, each component can be reused independently of the context, in other words can be used anytime, anywhere [1].

2. RELATED WORKS

In 1975 Freed Brooks, an IBM project manager, wrote The Mythical Man-month. Brooks wrote a chapter titled "No Silver Bullet" explaining that software systems are complex. He predicts there will be no single technique - no silver bullet - that can improve the productivity of the requirements list for every system. Brooks presents two possible methods to reduce software complexity: "Buy before Build" and "Reuse before Buy". Key concepts are introduced to help reduce costs in software development technology.

IBM pioneered the study of the System Object Model in the early 1990s. Some of the contributions applied in component software are OLE and COM. The software component model continues to achieve remarkable results.

According to Paul Allen, at present, more than 70% of new software systems are developed on the basis of components. Components are usually developed object-oriented and written in different languages, run in different environments, can be distributed everywhere, and new software developers are not provided with the source code.

In fact, the Component-Based Software Development method has reduced the cost of software development projects. Compared with standard traditional technology, component-based software engineering is more concerned with how to build software. Through the reuse of components, the software development life cycle is shortened, while increasing flexibility when using and maintaining the software. Furthermore, software development has the potential to increase software quality[1]
Component discussions can also be found in many conferences and publications [Koza 1999; Parrish 1999; Wang 2000; Yacoub 1999; Fischer 2002; Fukazawa 2002]

3. PROPOSED METHOD

The main problem of software engineering is how to create effective quality software. Components are seen by many software engineers as an important technology for solving the software crisis. The software industry revolution will happen through component-based software development technology [1] [2].
There are several reasons why COP is important. COP provides a higher level of abstraction. There is an increasing number of reusable component libraries that support application development for different domains [2] [3].
The most popular reusable component deployment is to deploy a component with a strong name, to register it with GAC. A shared component with strong name can make itself unique by public/private key pair. A shared component registered with GAC can make itself to be shared anywhere. The steps needed to create a shared .NET component are as follows: [1]
Step 1: Create a pair of public key/private key by sn.exe utility

```
>sn -k mykey.snk
```

The public key is for verification against private key, which is signed as a cryptographic signature stored in component assembly. The public key is stored in a manifest of the assembly. When a client references the component, the public key token is stored in the client's assembly.
Step 2: Embed the following lines into the source code of component:
Using System.Reflection:

```
[assembly:AssemblyKeyFile ("mykey.snk")]
[assembly:AssemblyDelaySign (false)]
[assembly:AssemblyVersion ("1,2,3,4")]
```

The following command will sign the signature with the component immediately without a delay:

```
>csc /t:library mycomponents.cs
```

The next command line will store a public key token in the client component.

```
>csc /r:mycomponent.dll /out:myapplication.exe myapplication.cs
```

If the signature delay is needed, we can sign the signature late by

```
>sn -R mycomponent.dll mykey.sn
```

The signature is verified when the component is registered with GAC in step #3 to ensure that the component is not altered since the assembly was built.
At run time, the public key token in the client manifest is verified against the public key that is part of component identity. If they match, then it indicates that this is the right component wanted.
Fig -1 shows the private and public key pair in manifests of the component and its client component.
Step 3: Register the shared component in GAC.

```
>gacutil /i mycomponent.dll
```

Step 4: Using shared component. The client must make a reference to the shared component to be reused.

```
>csc /t:exe /r:mycomponent.dll /out:myapp.exe myapp.cs
```

The following steps show how to build a shared component:

```
>sn -k originator.key
>csc /t:library /out:TempConv.dll TempConvComp.cs
>gacutil /i TempConv.dll
>csc /r:TempConv.dll /t:exe /out:TempConvCSClient.exe
TempConvCSClient.cs
```

In order to reuse the shared component, the client source code must "use namespaces" where namespace is available in the assembly

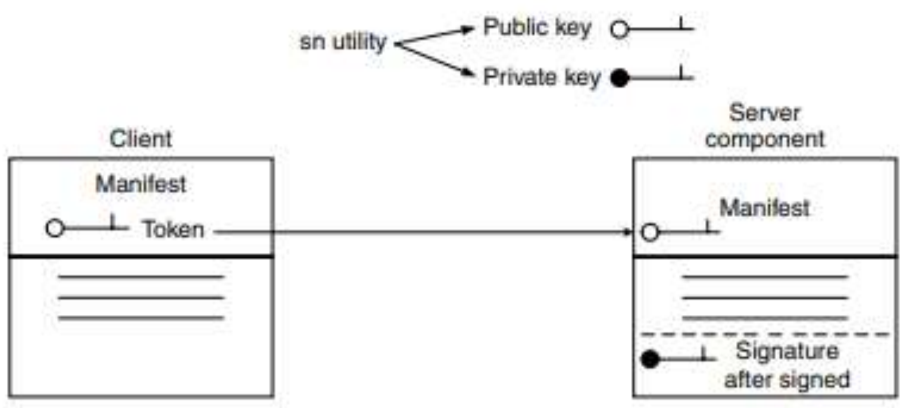


Fig -1: Public key pair in .NET component

4. PROPOSED METHOD

4.1 Building ideas

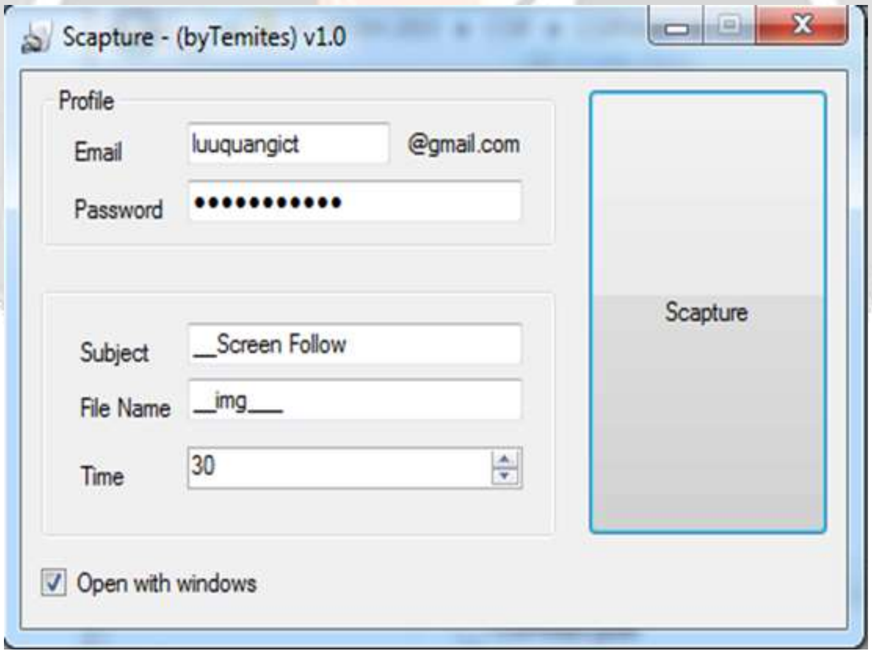
- We build an application to create components and assemble components into complete software:
+ Build a component that automatically takes pictures of the working computer screen interface
+ Build a component that automatically attaches screenshots to registered emails.
+ Build Configuration component: allows to configure screen capture time, number of images sent to email in one time, choose to start with the operating system, register email will receive screenshots**.**

Table -1: Methods and properties of the application

	Method/Properties	**Functions**
	Md5_Encrypt ()	Encrypt the data from plaintext to ciphertext by use the encryption key.
COPSecurity	Md5_Decrypt ()	Decryption
	Method/Properties	**Functions**
COPUser32	HideProccess	Hide program process
	FindTaskManager()	Find Task Manager program
	Method/Properties	**Functions**
	CreateFolder()	Creat a folder with pathfolder given
COPFolder	DeleteFolder()	Delete a folder with pathfolder given
	Method/Properties	**Functions**
	setRegistry()	Set in regestry to start with windows, the program has path is excutablePath with the name is value.
COPRegistry	setRegistry()	Create a registry with a subkey
	Method/Properties	**Functions**
	getData(string pathFile)	Read file has path is pathFile and return full contens in the file.
	getArrayData()	Read data from file has path is pathFile
COPData	getArrayData()	Similar to the function above, but cut the readable content without decoding.
	writeDataToFile()	Add one data row inputWrite in the end of file has path is pathFile.

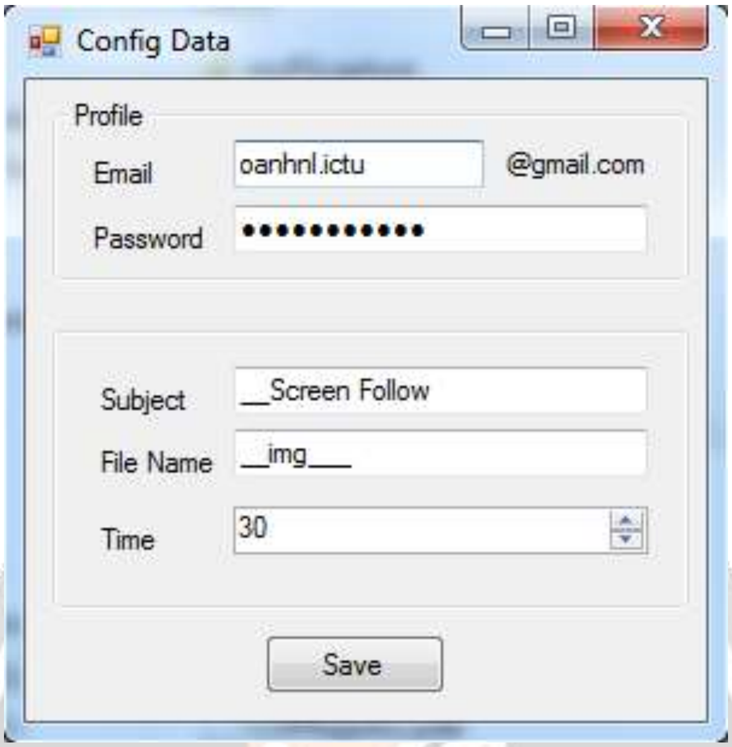
	writeDataToFile()	Encrypt the input string with the key and then overwrite to the file has path is pathFile.
	clearFileDatas()	Clear all contens in the file has path is pathFile.
	Method/Properties	**Functions**
	string[] listFile	The array that stores the file path will be attached to the email to be sent
	getFile()	
	sendImg	Send the images in the listFile to the email address
COPSendmail	sendErorMessage	Send faulty content to email address
	Method/Properties	**Functions**
	getImg( )	Take a screenshot and return an image.
COPCaptureScreen	scapture()	Save the captured image into a folder with path is pathFolder , and write the captured file name in the file pathListFile (here is the scListImg.dll file)

4.2 Experimental results



Email luuquangict @gmail.com
Password ••••••••••••
Subject __Screen Follow
File Name __img___
Time 30
[✓] Open with windows

Fig -2: Main interface



Email: oanhnl.ictu @gmail.com
Password: ●●●●●●●●●●●●
Subject: __Screen Follow
File Name: __img___
Time: 30

Fig -3: Email address configuration interface

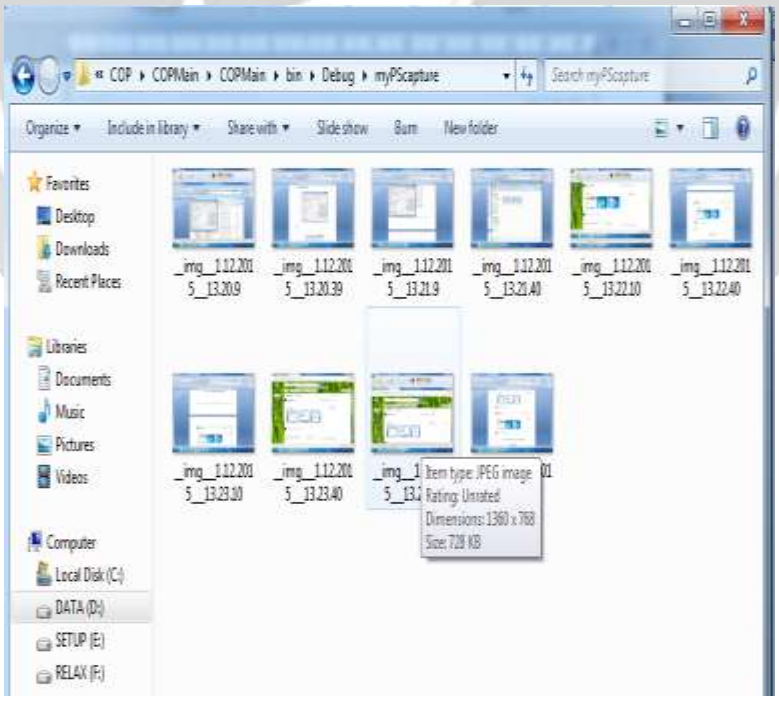


Fig -4: Screenshot saved in folder before sending mail

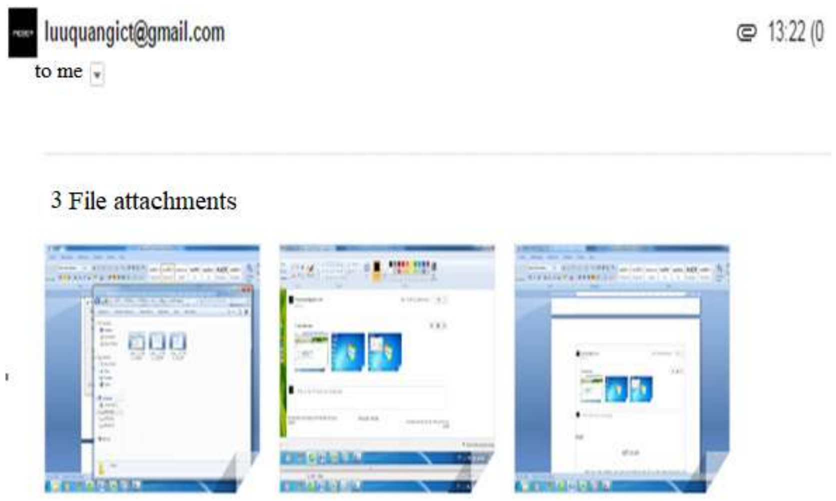


Fig -5: Mail received screenshot attached

5. CONCLUSIONS

We have proposed a application to help monitor and manage computer usage through the Internet. The theoretical results of the topic will be a reference for IT students in general, the experimental results are that the simulation software can be support to manage computer usage.

6. REFERENCES

[1]. Andy Ju An Wang, Kai Qian, *Component Oriented Programming*, Wiley, 2005
[2]. Deitel, *Java Web Services for Experienced Programmers*, Prentice Hall, 2003
[3]. Deitel, *Web Services, A Technical Introduction*, Prentice Hall, 2003
[4]. McGovern, James, *Java Web Services Architecture*, Morgan, 2003