Interception

Jim Fawcett

CSE681 – Software Modeling and Analysis

Fall 2007

References

- Essential .Net, Volume 1, The Common Lanaguage Runtime, Don Box with Chris Sells, Addison-Wesley, 2003
- Aspect-Oriented Programming, Shukla, Fell, Sells, MSDN, March 2002
- Advanced .Net Remoting, Ingo Rammer, Apress, 2002
- Microsoft .Net Remoting, Scott McLean, James Naftel, Kim Williams, Microsoft Press, 2003

What is Interception?

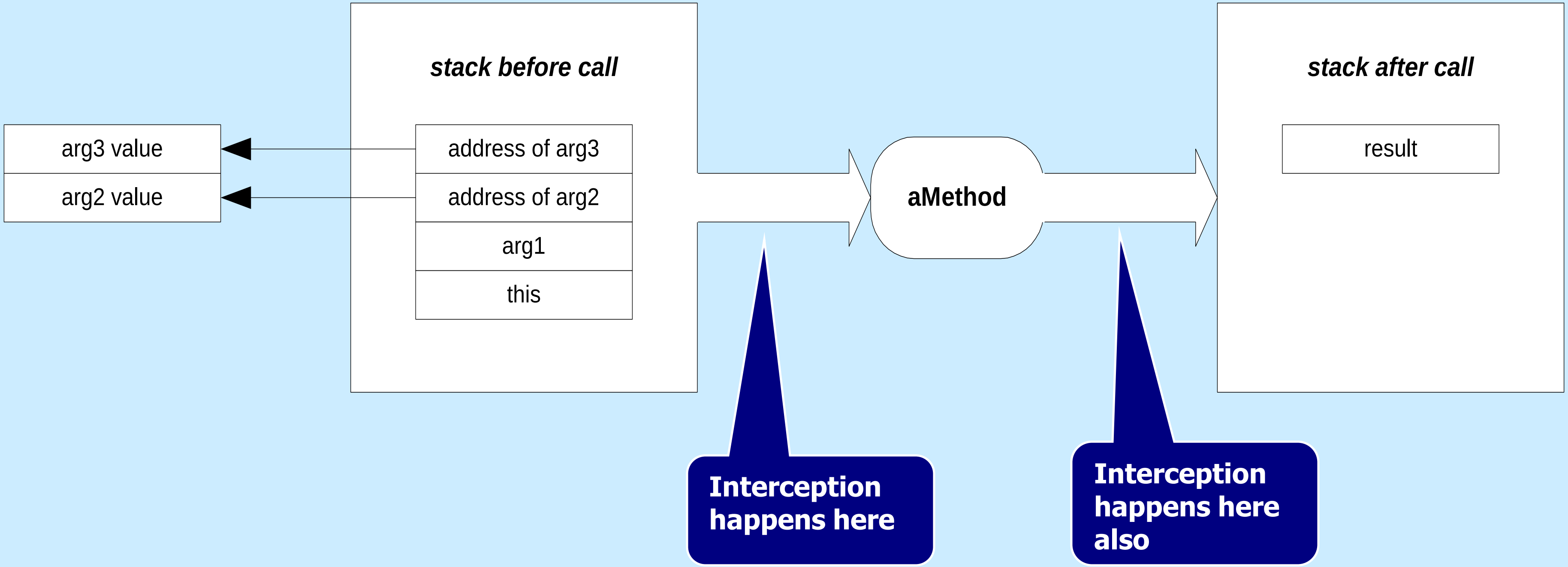
- Interception is the process of inserting processing:
  - after a client call, but before the method executes
  - after method execution, but before the thread of exectution returns to the client

- This processing, in .Net, is usually specified by an attribute:
  - [Serializable]
  - [OneWay]

- One use of interception is to attempt to separate solution domain processing from problem domain processing.

Invoking a Method

int aMethod(int arg1, ref int arg2, out int arg3)



stack before call

address of arg3

address of arg2

arg1

this

arg3 value

arg2 value

aMethod

stack after call

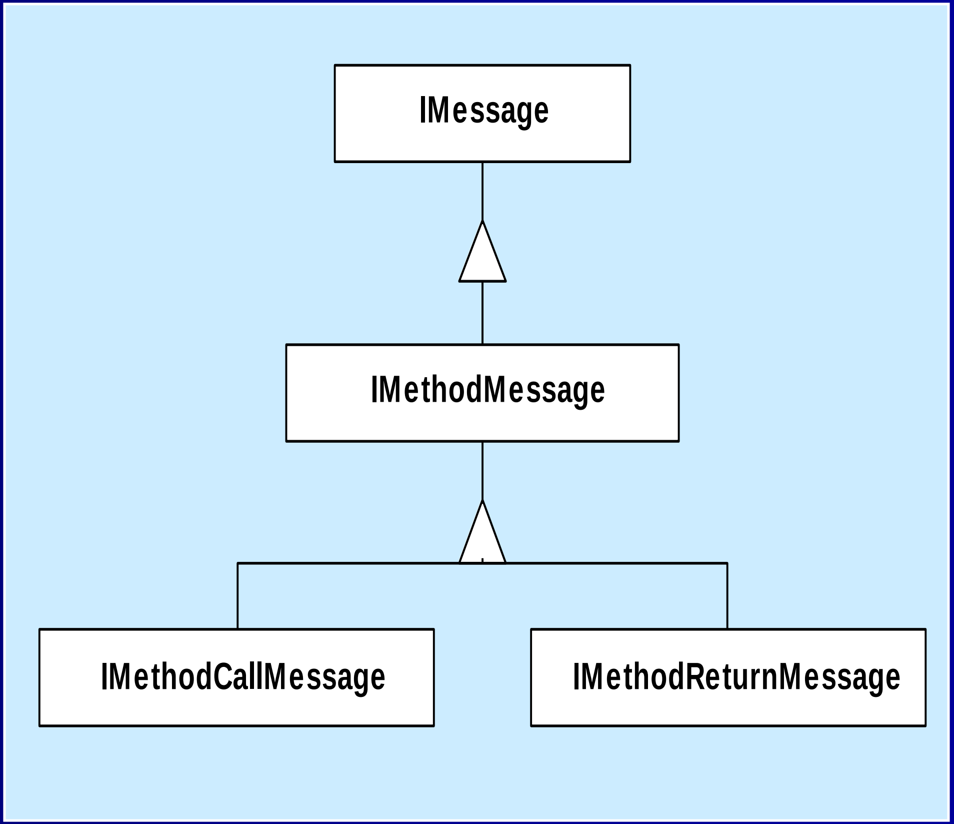
result

Interception happens here

Interception happens here also

Invocation Message Model

- The CLR makes method call-stack transformation accessible via the IMessage interface.
- IMethodMessage provides access to method arguments, return value, and to the metadata for the method via a MethodBase property.
- This provides access to stack frame contents without requiring knowledge of the stack layout.

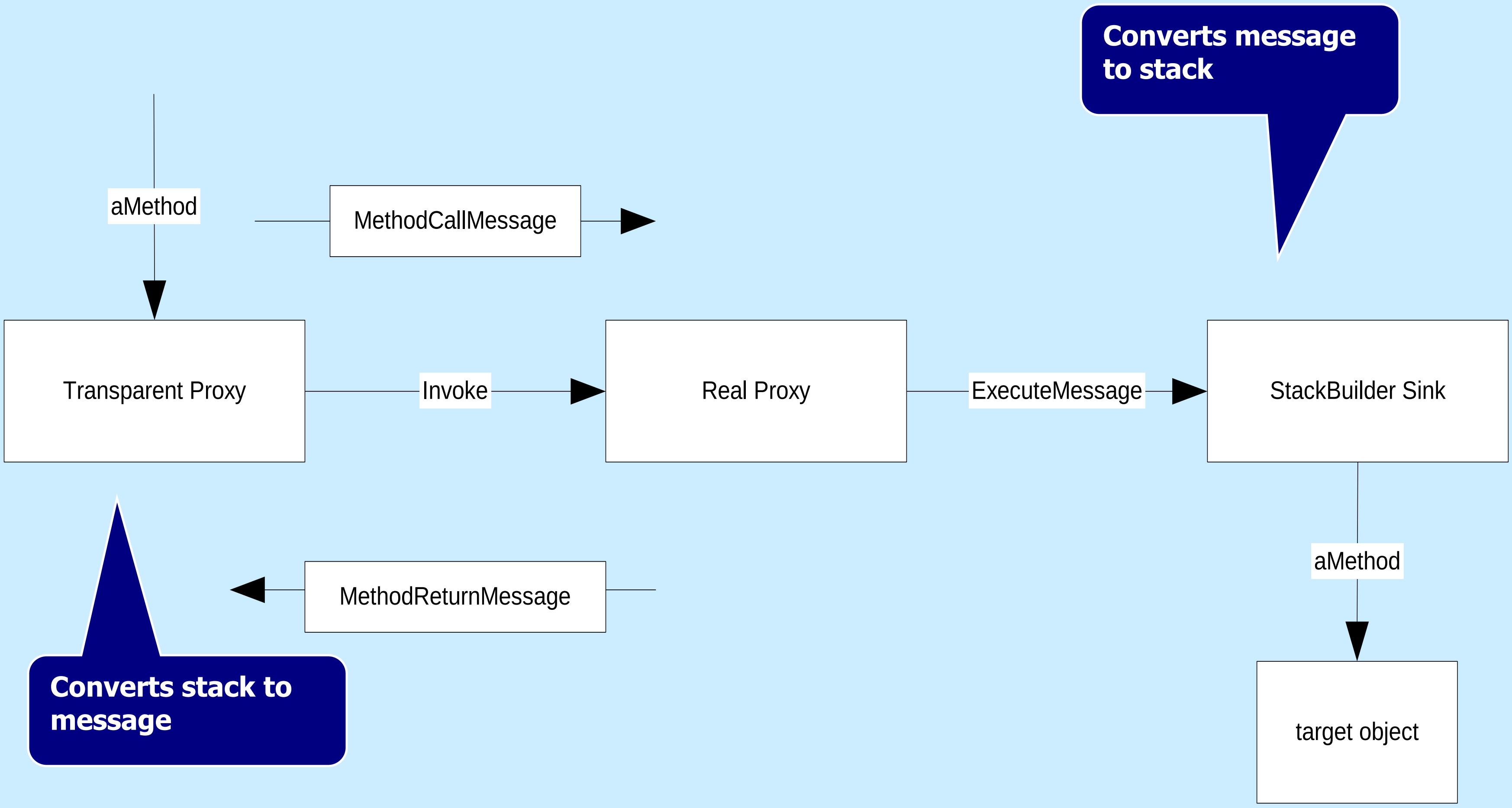


Creation of Messages

- A transparent proxy, created by the CLR, is used to translate method calls into messages.
- The transparent proxy is always associated with a real proxy, responsible for transforming a MethodCallMessage into a MethodReturnMessage.

  The transparent proxy then uses the MethodReturnMessage to transform the call stack into the result stack configuration.

Stack to Message to Stack

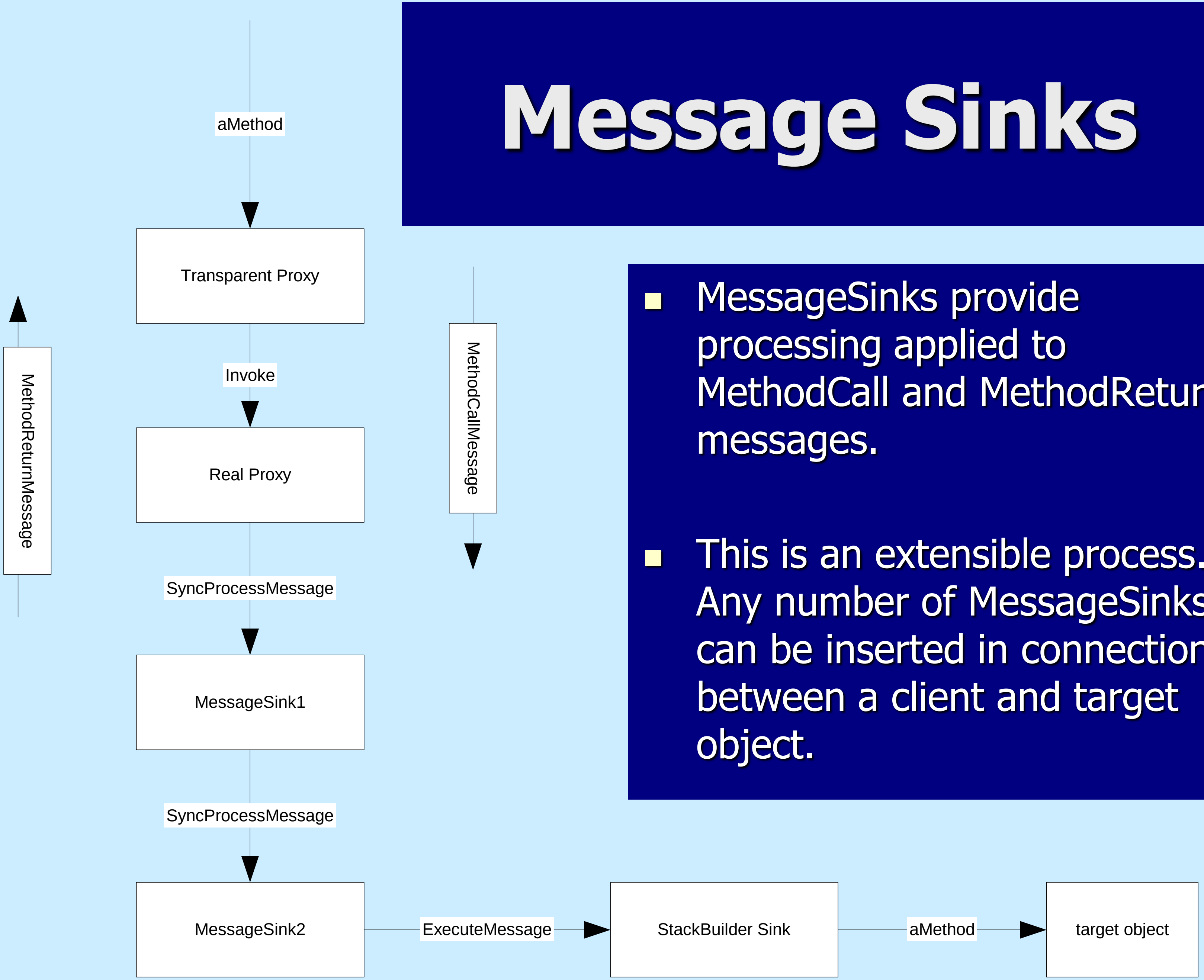


ContextBound Objects

- Deriving a class from System.ContextBoundObject ensures that every access to an object is through a transparent proxy.
- A context represents services required by the bound object.
- The whole purpose of interception is to automatically provide pre and post processing of method calls.
- This is done with MessageSinks.
- The context specifies what MessageSink process will be applied to a context bound object.

Message Sinks

- MessageSinks provide processing applied to MethodCall and MethodReturn messages.
- This is an extensible process. Any number of MessageSinks can be inserted in connection between a client and target object.



Installing Message Sinks

- The CLR gives context attribute objects the chance to install context properties as the context is being created.

- It also gives context property objects the opportunity to put MessageSinks between a proxy and ContextBound object when the proxy is created.

Afterword

- These notes summarize material provided in Chapter 7 of Don Box's "Essential .Net", Volume 1.
  - In that chapter the author provides a small example that shows code fragments illustrating how to build the interception apparatus.
- Ingo Rammer in his "Advanced .Net Remoting", provides examples of how channels work and how to build custom Message Sinks, in chapters 7, 8, and 9.
- Scott McLean, et. al., in ".Net Remoting", also provide examples of how to build interception in chapters 5, 6, and 7.

End of Presentation