<table>
<thead>
<tr>
<th>functional component</th>
<th>reflection</th>
<th>persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration</td>
<td>observation</td>
<td></td>
</tr>
</tbody>
</table>
CoFRA

C1

CoCo

C2

CoCo

C3

host 1

application level

component configurators

application configuration

configurations

description

application assemblies

CoFRA

binary components

host 2

local assembly repository

CoFRA
Adaptive.Net

- Dynamic Reconfiguration
  - Object Updates
  - Object Migration
  - Addition of Components
  - Removal of Components
  - Change of Component Attributes
public class Counter : MarshalByRefObject
{
    [Property] // Configuration Hook
    private long step;
    private long count = 0; // State

    public void Reset()
    {
        count = 0;
    }

    public void Increase()
    {
        count += step;
    }

    public long GetCount()
    {
        return count;
    }
}

public class CounterClient
{
    [Connection]
    private Counter counter = null;
    public void Main()
    {
        while(true)
        {
            Transaction.Begin("counter");
            counter.Increase();
            Console.WriteLine("Counter Value :" + counter.GetCount());
            Transaction.End("counter");
            Thread.Sleep(1000);
        }
    }
}
Architectural Patterns for Adaptive Applications

- Voter Pattern
  - Compare output of 3 objects supporting the same interface
- Safe update pattern
  - Update a running object to a new version
- Load balancer pattern
- Migration pattern
  - Migrate an object from one host to another
- Filter, Compression, Encryption Pattern
- Simplex Pattern
  - Used in Foucault’s Pendulum