Refactoring to an Immutable Architecture

Vladimir Khorikov
PROGRAMMER

@vkhorikov www.enterprisecraftsmanship.com

Vocabulary Used

Immutability

Inability to change data

State

Data that changes over time

Side effect

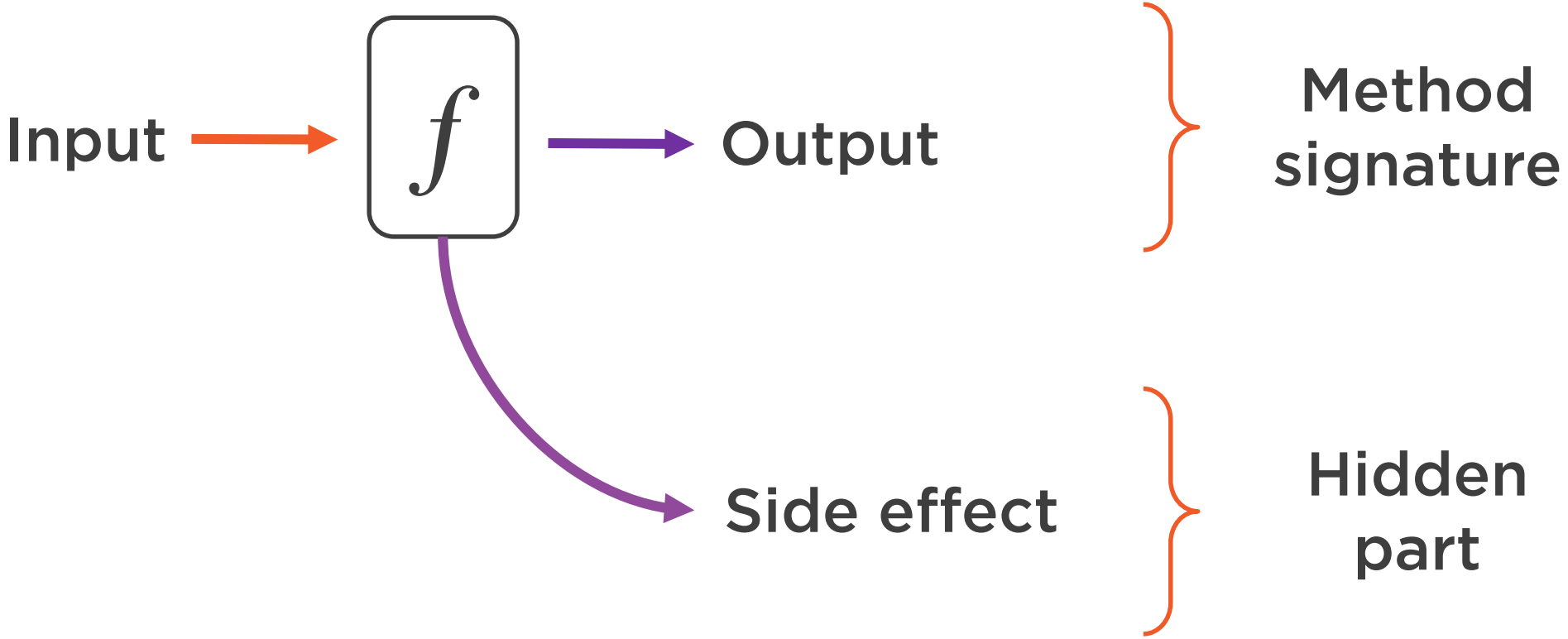
A change that is made to some state

```
public class UserProfile {
    private User _user;
    private string _address;

    public void UpdateUser(int userId, string name) {
        _user = new User(userId, name);
    }
}

public class User {
    public int Id { get; }
    public string Name { get; }

    public User(int id, string name) {
        Id = id;
        Name = name;
    }
}
```

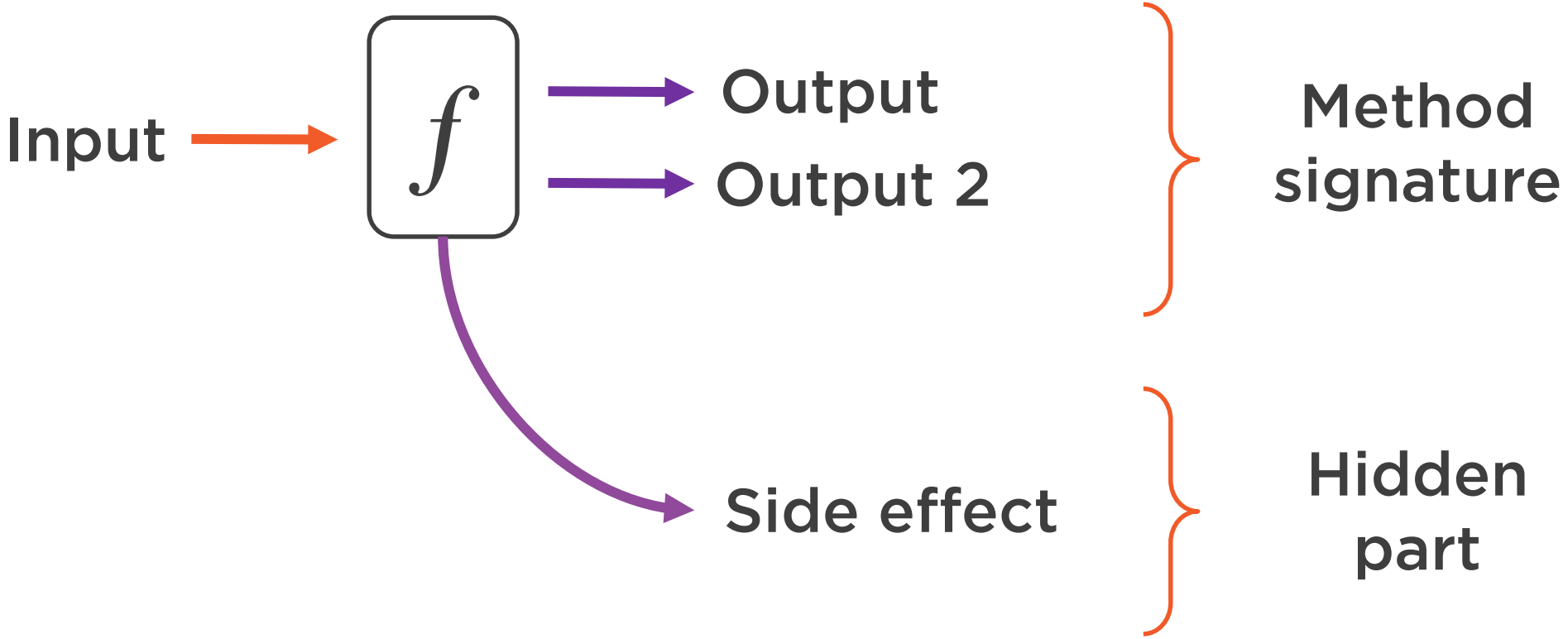
Why Does Immutability Matter?

Mutable operations = **Dishonest code**

Why Does Immutability Matter?



Why Does Immutability Matter?



```
public class UserProfile {
    private readonly User _user;
    private readonly string _address;

    public UserProfile(User user, string address) {
        _user = user;
        _address = address;
    }

    public UserProfile UpdateUser(int userId, string name) {
        var newUser = new User(userId, name);
        return new UserProfile(newUser, _address);
    }
}

public class User {
    public int Id { get; }
    public string Name { get; }

    public User(int id, string name) {
        Id = id;
        Name = name;
    }
}
```

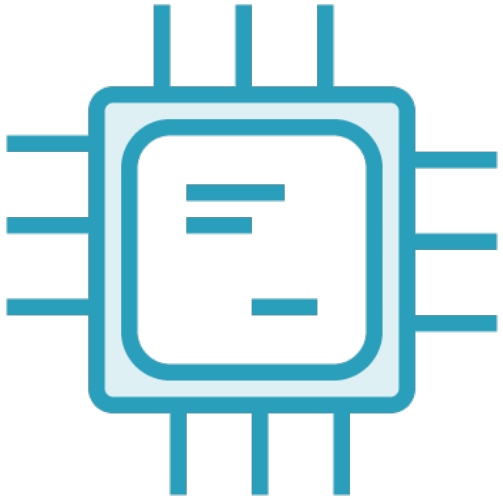
Why Does Immutability Matter?

Increased readability

A single place for validating invariants

Automatic thread safety

Immutability Limitations



CPU Usage

Memory Usage

Immutability Limitations

```
ImmutableList<string> list = ImmutableList.Create<string>();
ImmutableList<string> list2 = list.Add("New item");
```

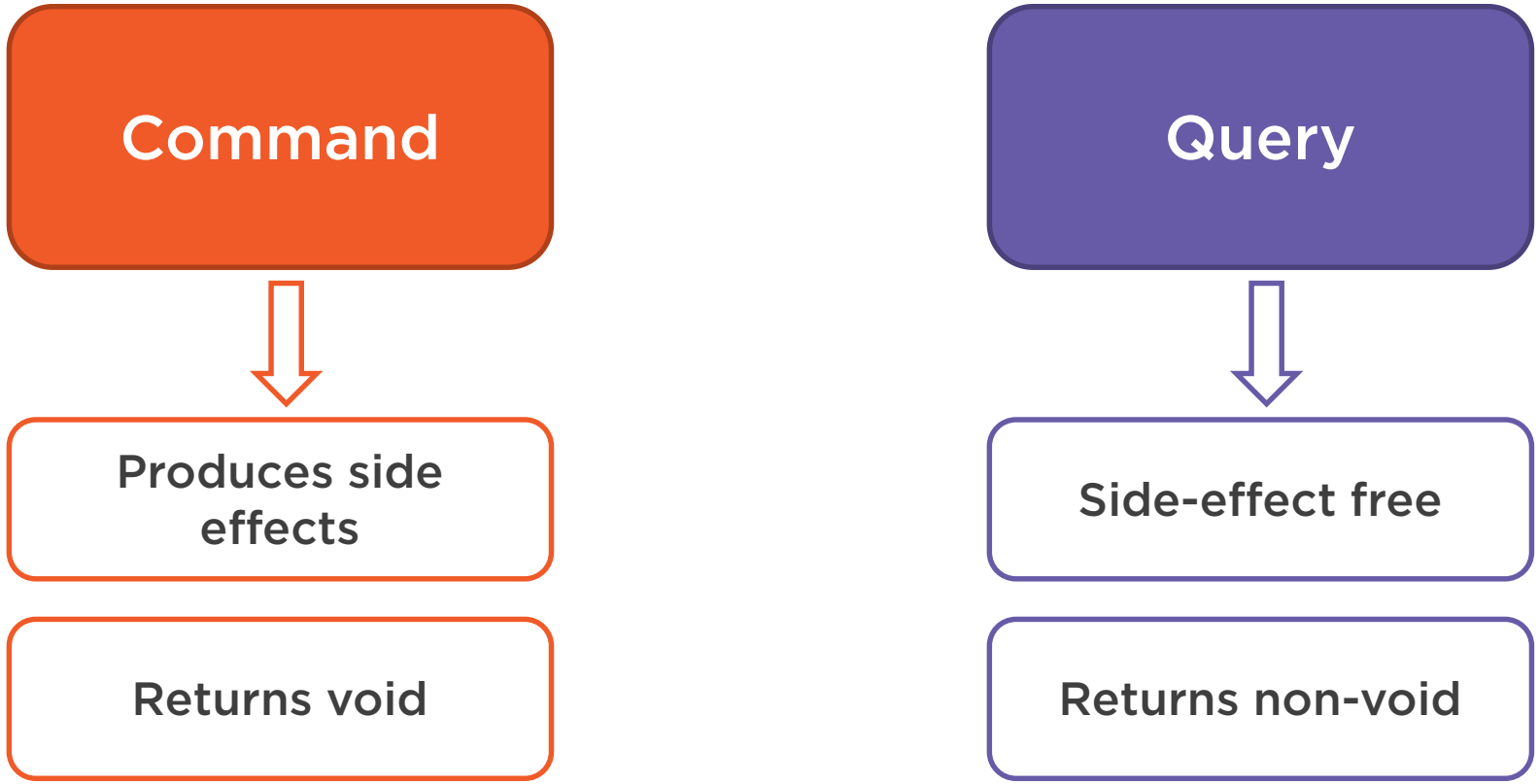
```
ImmutableList<string>.Builder builder = ImmutableList.CreateBuilder<string>();
builder.Add("Line 1");
builder.Add("Line 2");
builder.Add("Line 3");

ImmutableList<string> immutableList = builder.ToImmutable();
```

How to Deal with Side Effects

```
public class CustomerService {
    public void Process(string customerName, string addressString) {
        Address address = CreateAddress(addressString);
        Customer customer = CreateCustomer(customerName, address);
        SaveCustomer(customer);
    }

    private Address CreateAddress(string addressString) {
        return new Address(addressString);
    }

    private Customer CreateCustomer(string name, Address address) {
        return new Customer(name, address);
    }

    private void SaveCustomer(Customer customer) {
        var repository = new Repository();
        repository.Save(customer);
    }
}
```

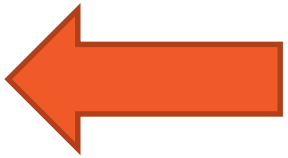
How to Deal with Side Effects

Command–query separation principle

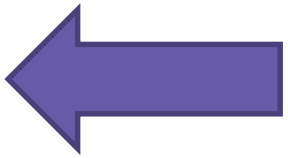


How to Deal with Side Effects

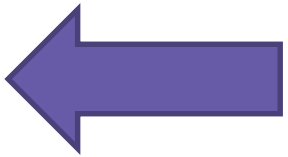
```
public class CustomerService {
    public void Process(string customerName, string addressString) {
        Address address = CreateAddress(addressString);
        Customer customer = CreateCustomer(customerName, address);
        SaveCustomer(customer);
    }

    private Address CreateAddress(string addressString) {
        return new Address(addressString);
    }

    private Customer CreateCustomer(string name, Address address) {
        return new Customer(name, address);
    }

    private void SaveCustomer(Customer customer) {
        var repository = new Repository();
        repository.Save(customer);
    }
}
```



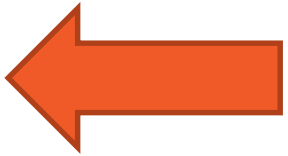
Command



Query



Query

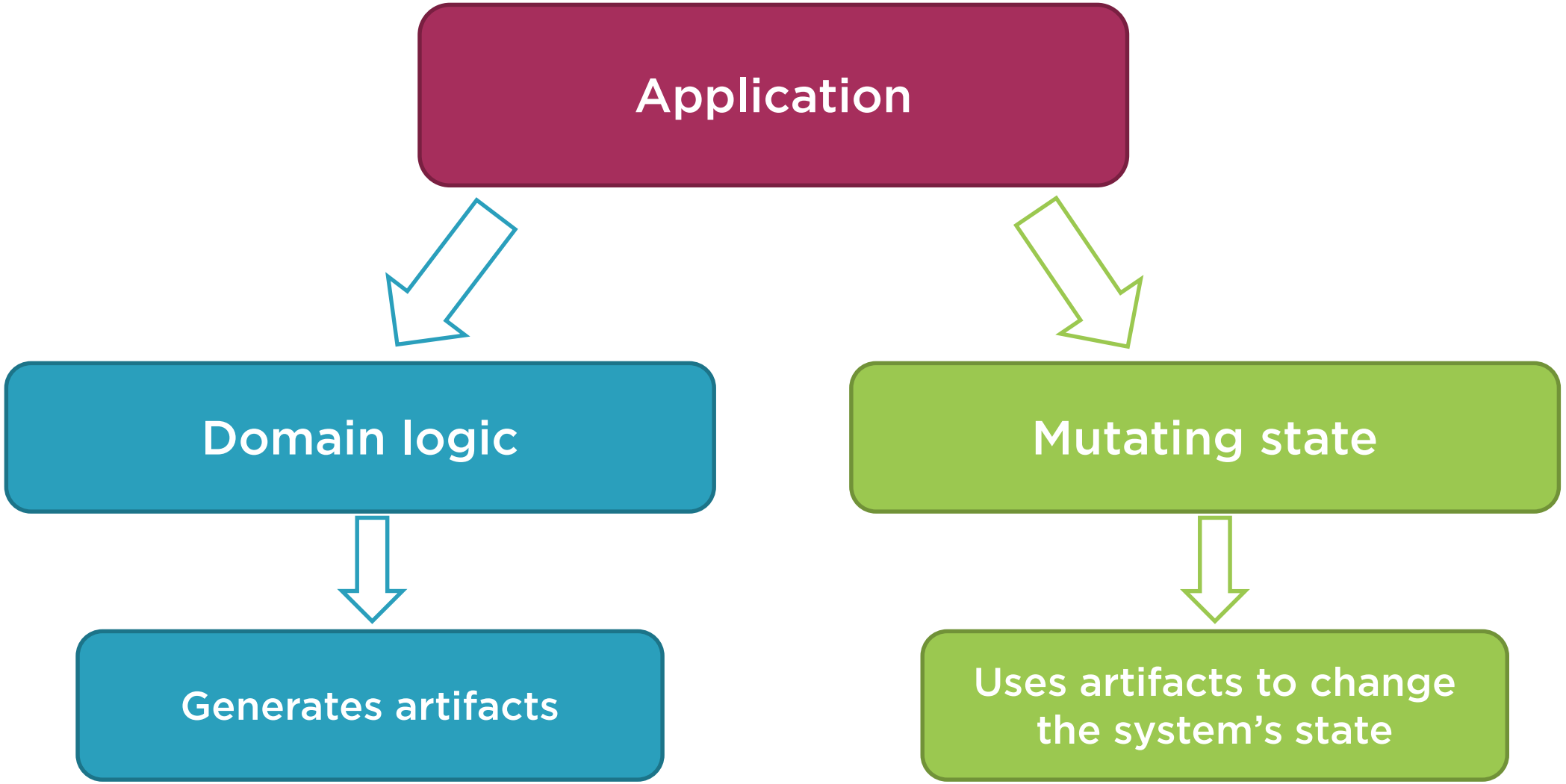


Command

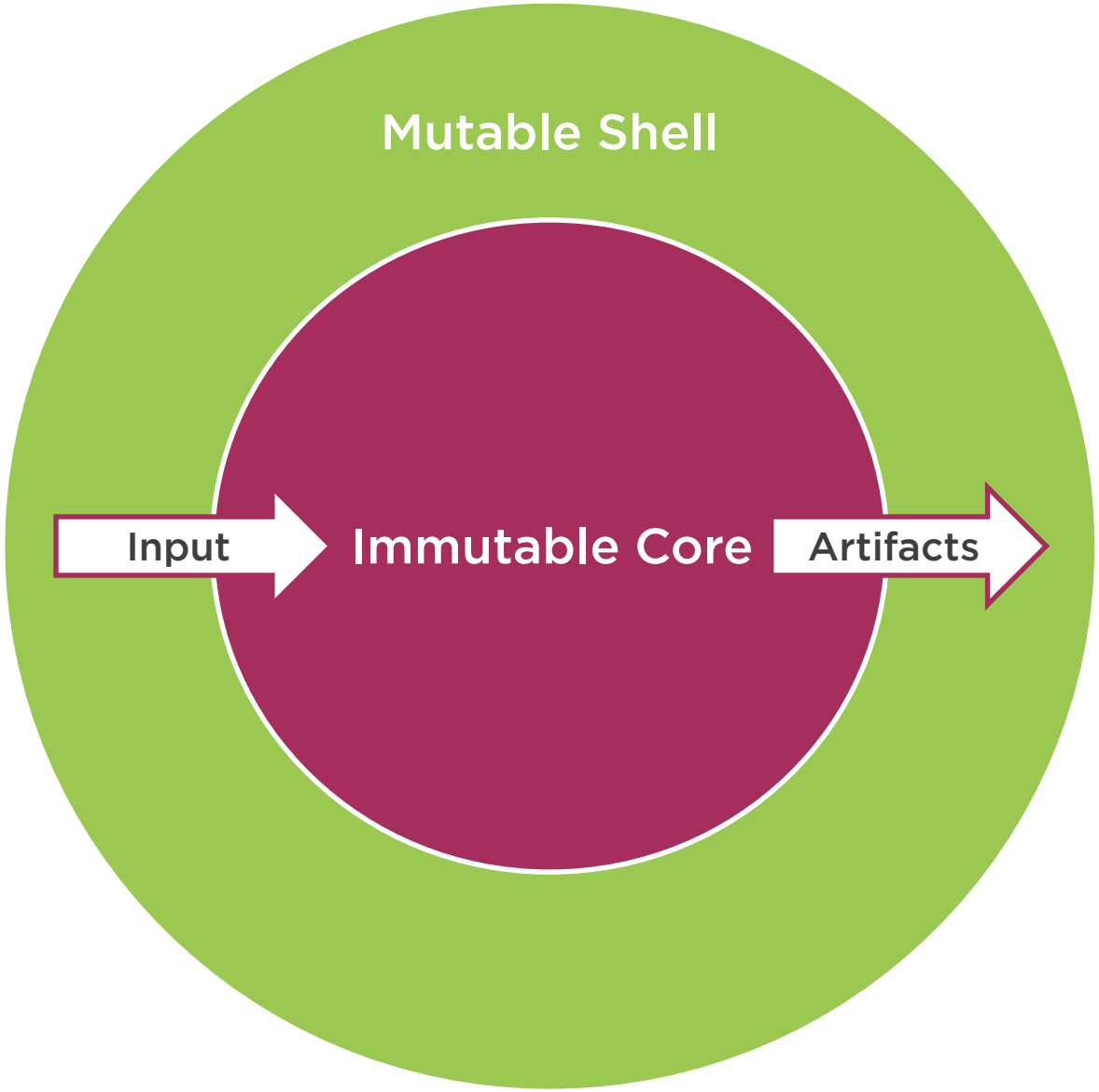
How to Deal with Side Effects

```
var stack = new Stack<string>();
stack.Push("value");              // Command
string value = stack.Pop();       // Both query and command
```

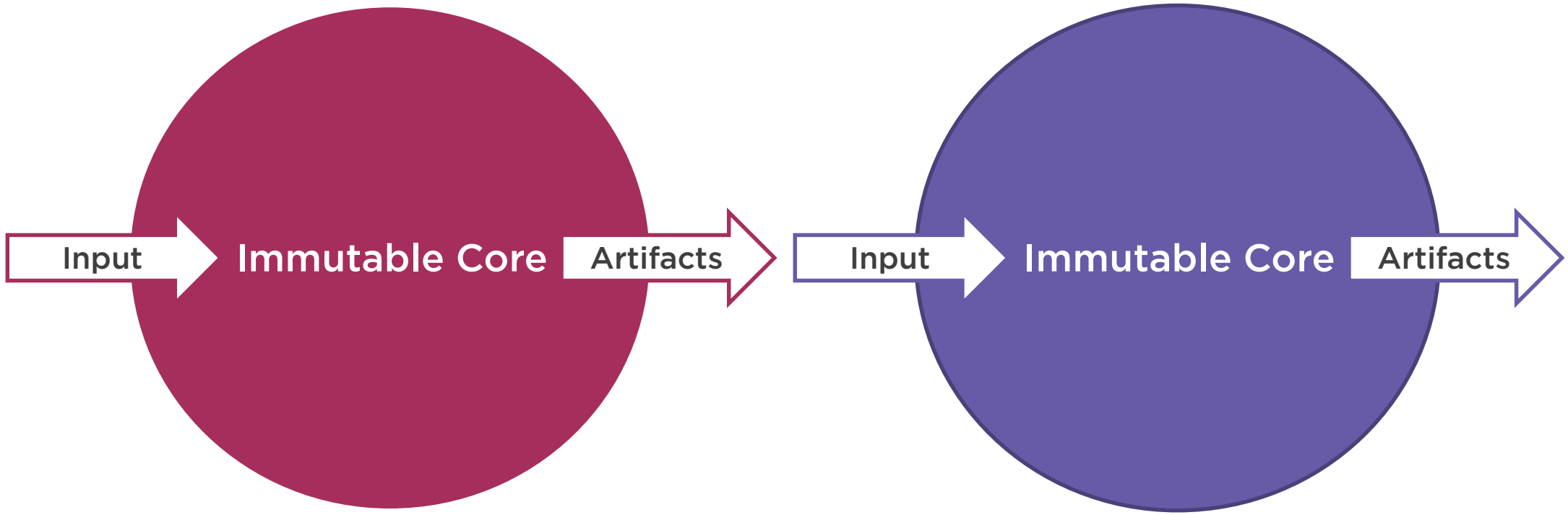
How to Deal with Side Effects



How to Deal with Side Effects



How to Deal with Side Effects



Example Introduction

Audit manager

```
1; Peter Peterson; 2016-04-06T16:30:00
2; Jane Smith;     2016-04-06T16:40:00
3; Jack Rich;      2016-04-06T17:00:00
```

Example Introduction


```
1; Peter Peterson; 2016-04-06T16:30:00
2; Jane Smith;     2016-04-06T16:40:00
3; Jack Rich;      2016-04-06T17:00:00
4; New Person;     Time of visit
```

Example Introduction


```
1; Peter Peterson; 2016-04-06T16:30:00
2; Jane Smith;     2016-04-06T16:40:00
3; Jack Rich;      2016-04-06T17:00:00
4; New Person;     Time of visit
```

Example Introduction

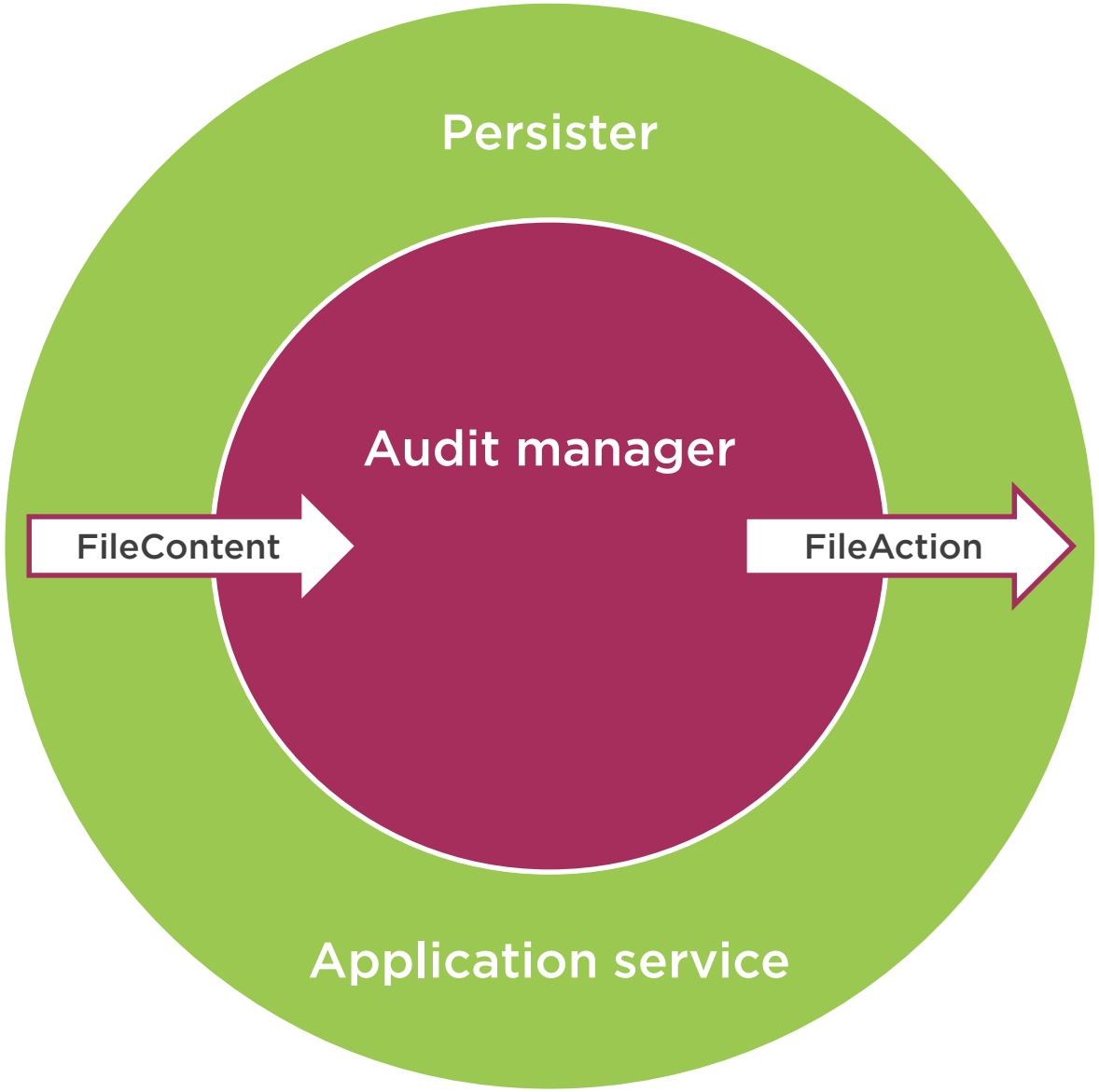
Audit manager

Log file processing

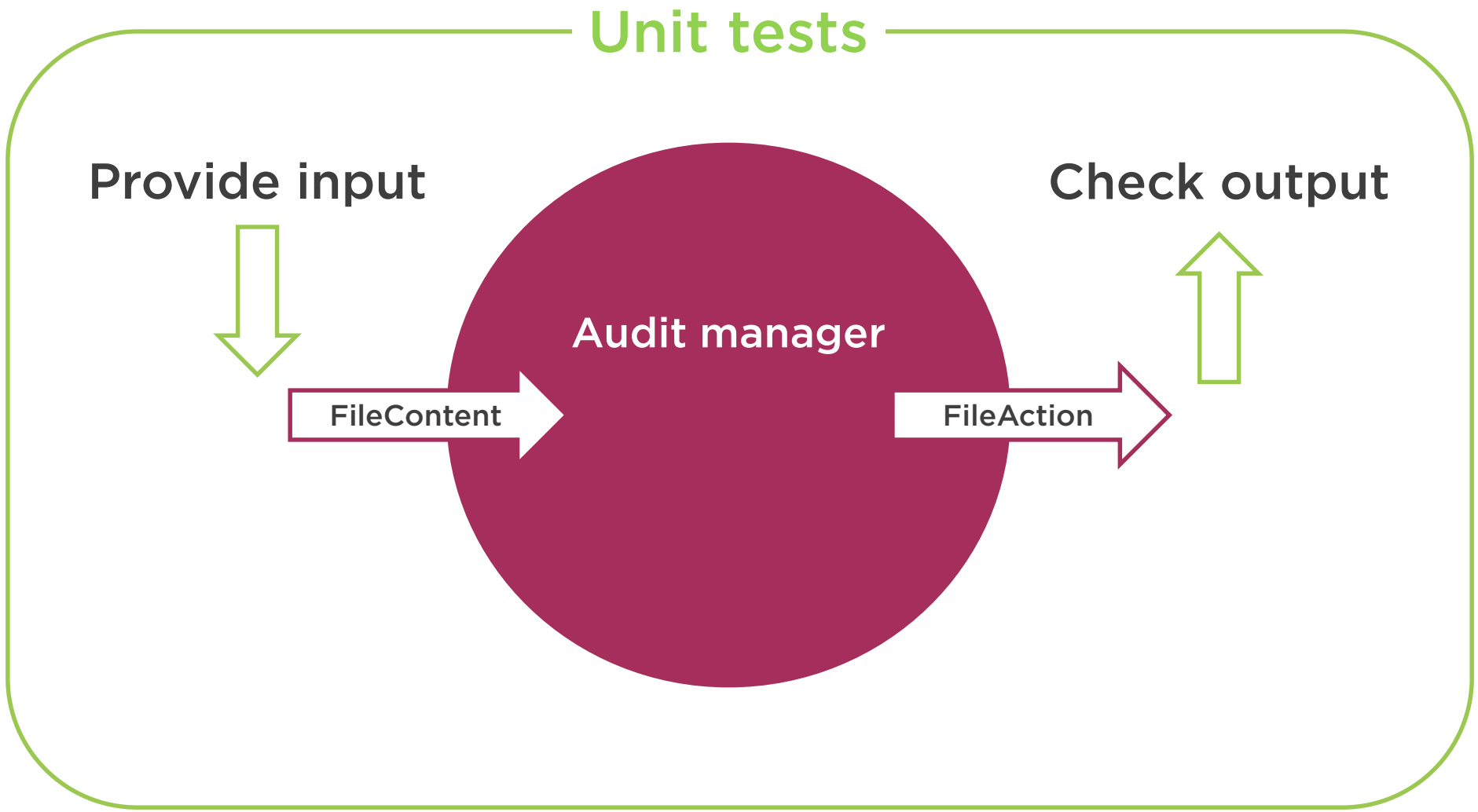
Operating files on the disk

Example Introduction

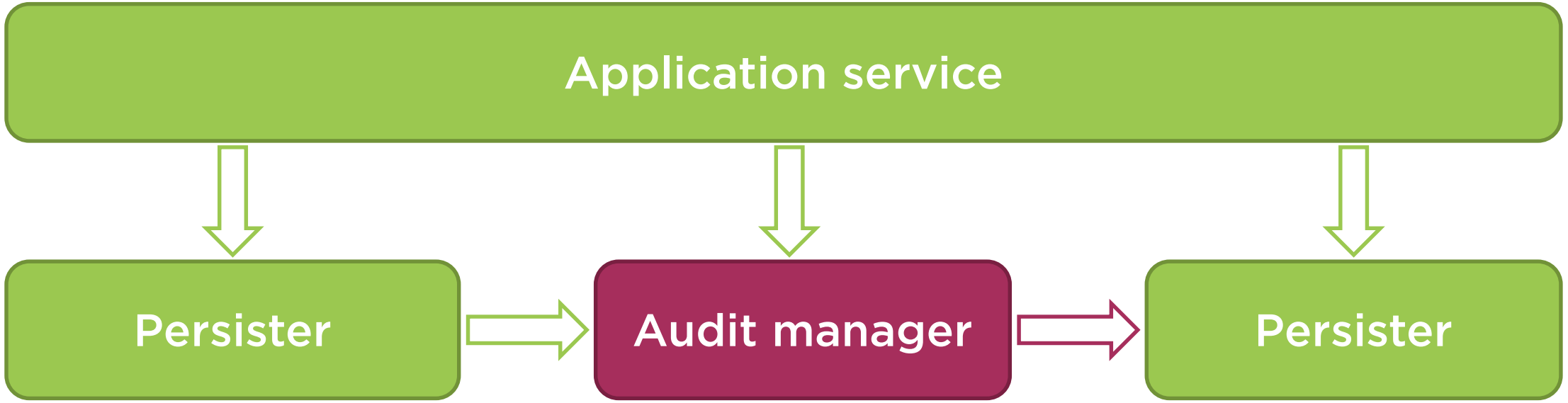
Recap: Refactoring to an Immutable Architecture



Recap: Refactoring to an Immutable Architecture

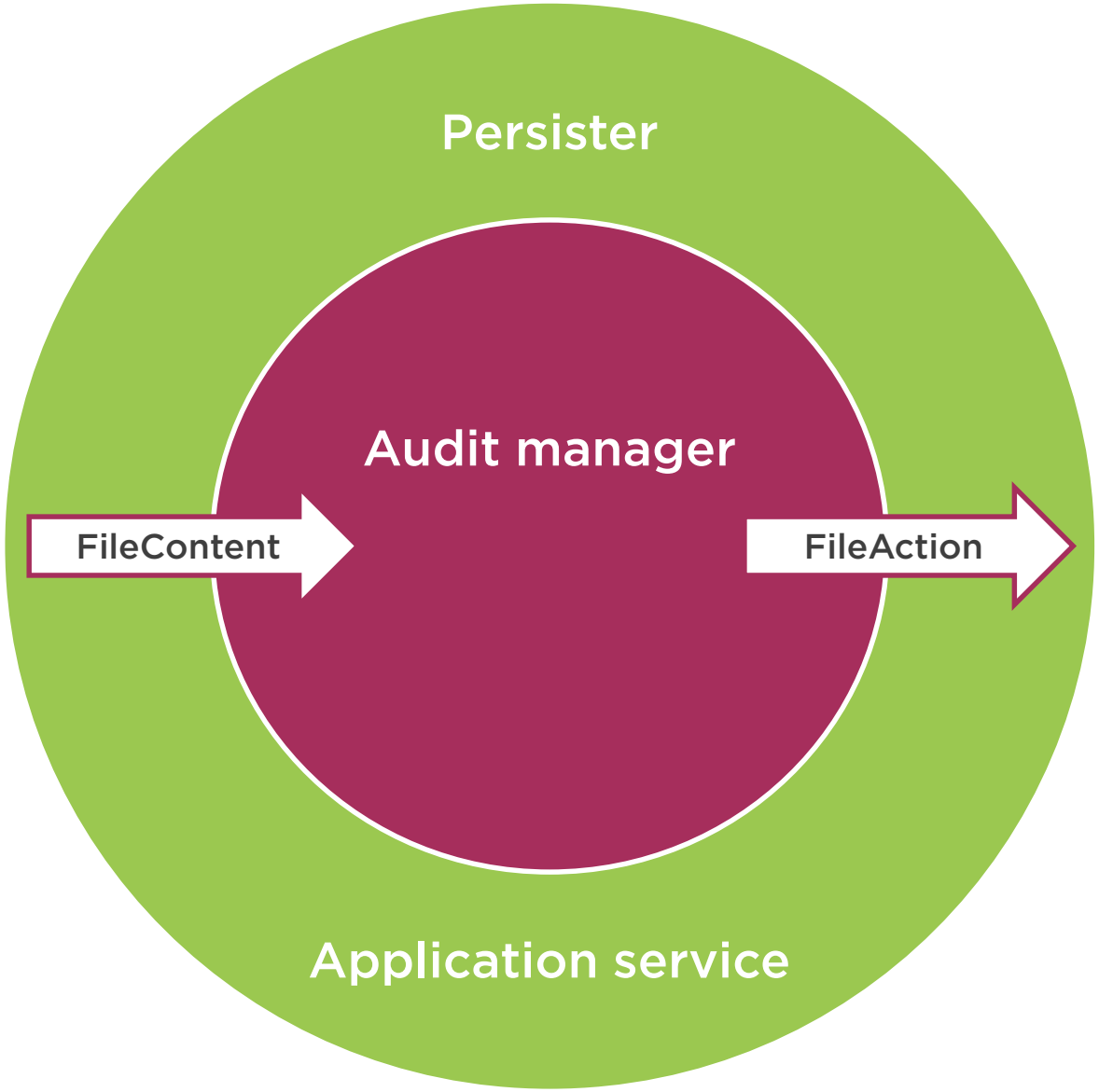


Recap: Refactoring to an Immutable Architecture



- Make the mutable shell as dumb as possible
- Apply side effect at the end of a business transaction

Recap: Refactoring to an Immutable Architecture



Recap: Refactoring to an Immutable Architecture

```
public FileAction AddRecord(
    FileContent currentFile, string visitorName, DateTime timeOfVisit)
{
    List<AuditEntry> entries = Parse(currentFile.Content);

    /* ... */
}
```

Summary

Using side effects makes your code dishonest

Dealing with side effects on the architectural level

- Immutable core
- Mutable shell

In the Next Module

Exceptions