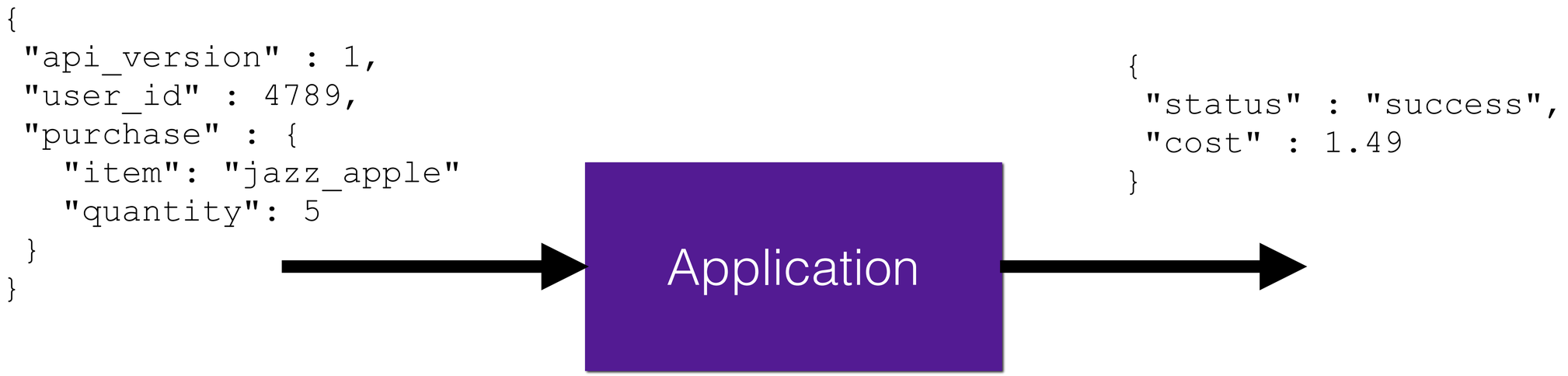
Middleware

Designing and Maintaining Software (DAMS)

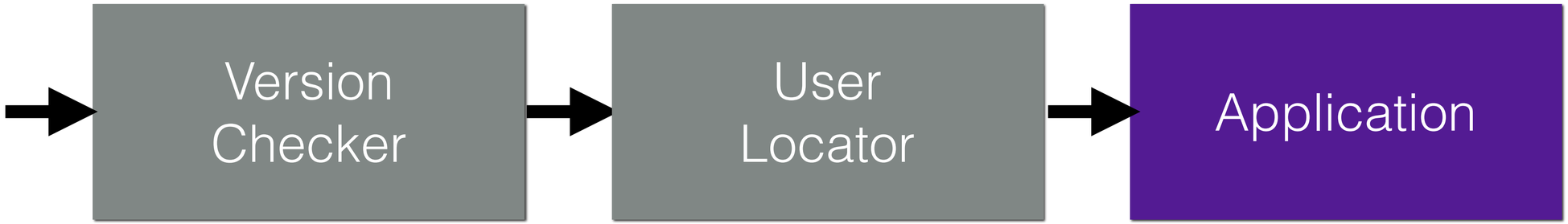
Louis Rose


PROFIT
FUNDING
CUSTOMER
BUSINESS
YOU
COMMUNITY
ENVIRONMENT
SUCCESS
INNOVATION

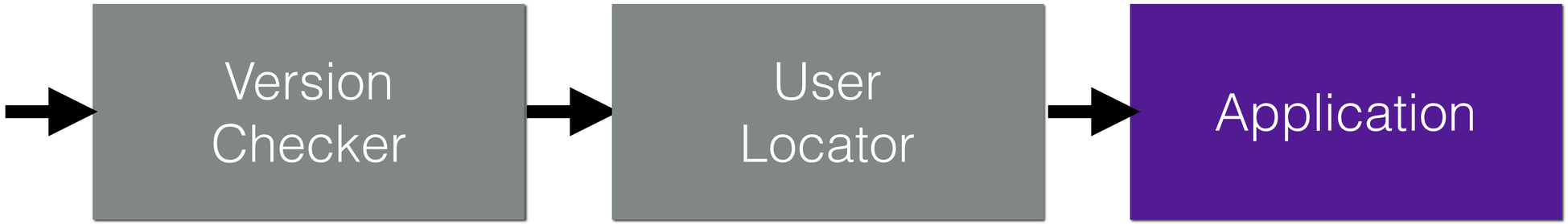
Message Processors



Middleware ~= Filters



Middleware ~= Filters

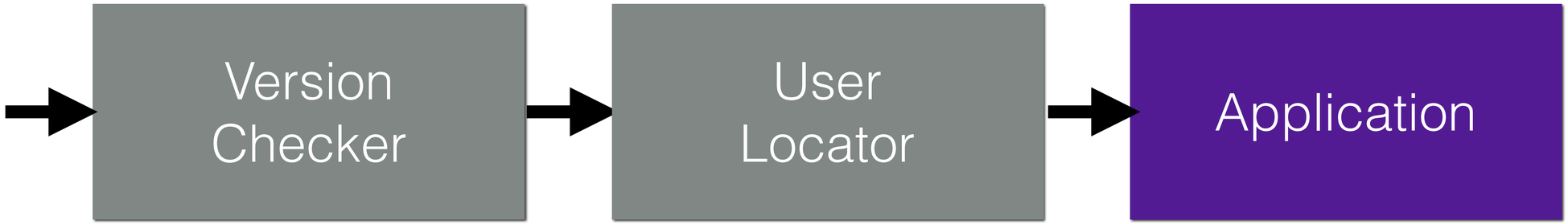


Middleware ~= Filters

```
{
 "api_version" : 1,
 "user_id" : 4789,
 "purchase" : {
   "item": "jazz_apple"
   "quantity": 5
 }
}
```

```
{
 "api_version" : 1,
 "user_id" : 4789,
 "purchase" : {
   "item": "jazz_apple"
   "quantity": 5
 }
}
```

```
{
 "api_version" : 1,
 "user" : {
   "name" : "Joe Bloggs"
 },
 "purchase" : {
   "item": "jazz_apple"
   "quantity": 5
 }
}
```



Accepting Middleware

```
module Marketplace
  class Middleware
    attr_accessor :successor

    def call(context)
      fail "All middleware must implement call(context)"
    end
  end

  class Application < Middleware
    def run(**context)
    end

    def add_middleware(middleware)
    end
  end
end
```

Implementing Middleware

```
require "marketplace"

class VersionChecker < Marketplace::Middleware
  def call(context)
    successor.call(context) if context[:api_version] == 1
  end
end

app = Marketplace::Application.new
app.add_middleware(VersionChecker.new)
app.run(api_version: 1, user_id: 4789, purchase: {...})
```

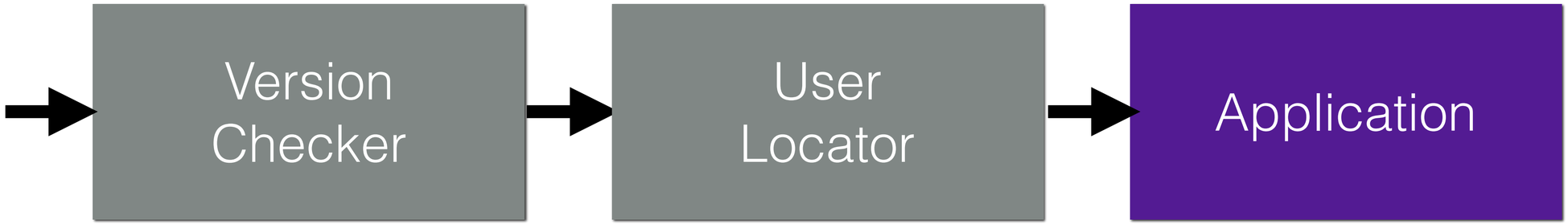
Benefits

Middleware implementations can be defined in a different component to the application

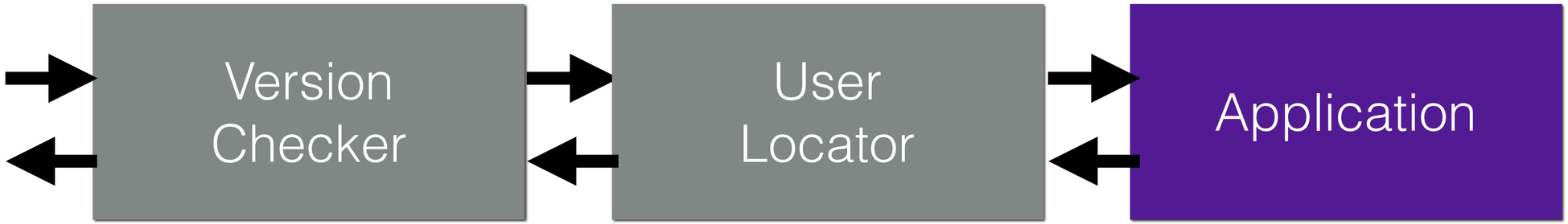
Middleware implementations can be re-used over different projects (e.g., authentication)

Middleware and application can be scaled independently: your ops team will be happy

Simplex Middleware

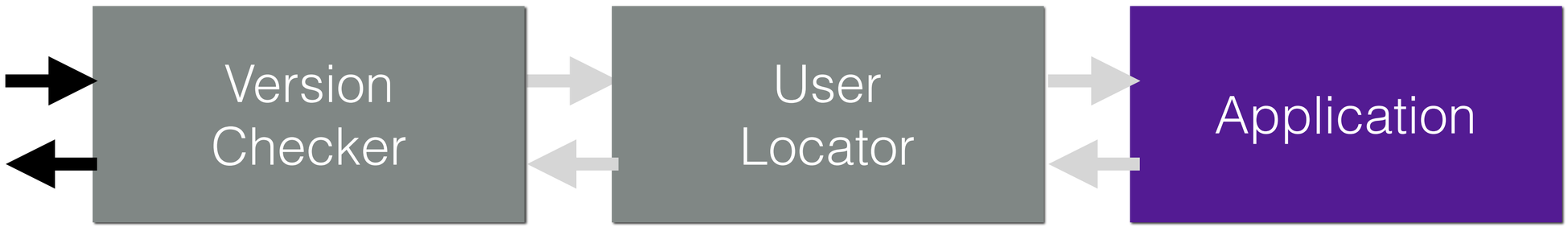


Duplex Middleware



Duplex Middleware

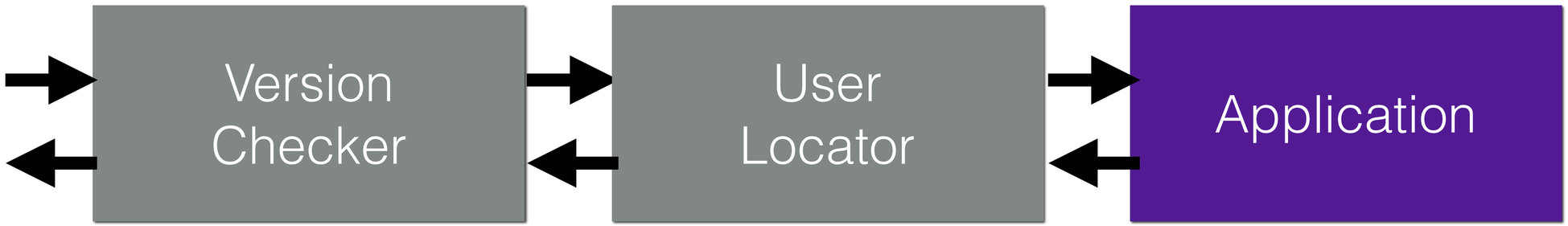
```
{
 "api_version" : 2,
 "user_id" : 4789,
 "purchase" : {
   "item": "jazz_apple"
   "quantity": 5
 }
}
```




```
{
 "error" : {
   "message" : "Marketplace API version 2 is
                not supported by this server."
 }
}
```

Duplex Middleware

```
{
 "api_version" : 1,
 "user_id" : 4789,
 "purchase" : {
   "item": "jazz_apple"
   "quantity": 5
 }
}
```




```
{
 "status" : "success",
 "cost" : 1.49,
 "remaining" : 68.51
}
```

```
{
 "status" : "success",
 "cost" : 1.49
}
```

Summary

The middleware pattern is widely used to extend message-oriented applications

Middleware implementations are independent of (and reusable between) applications

Duplex middleware is often better for request-response messaging (e.g., web applications)