<2023-02-10 fre> Example: SMSprucerUpper

Mikael Svahnberg*

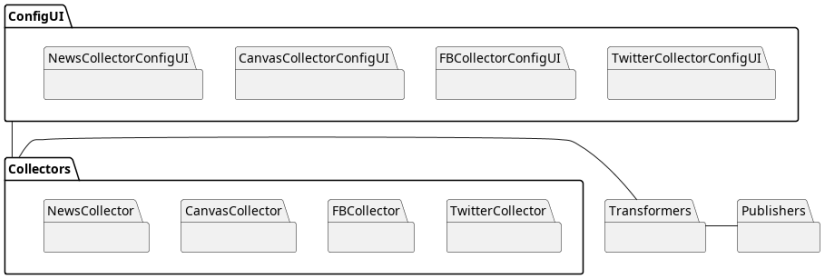
2023-02-01

Contents

1 Conceptual Model

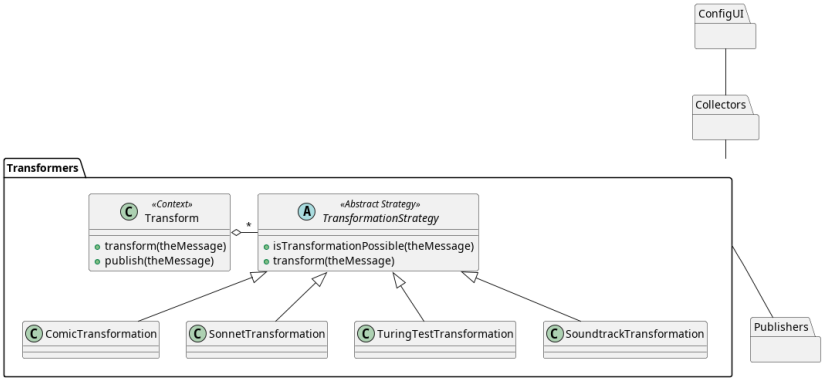
*Mikael.Svahnberg@bth.se

2 Package Model



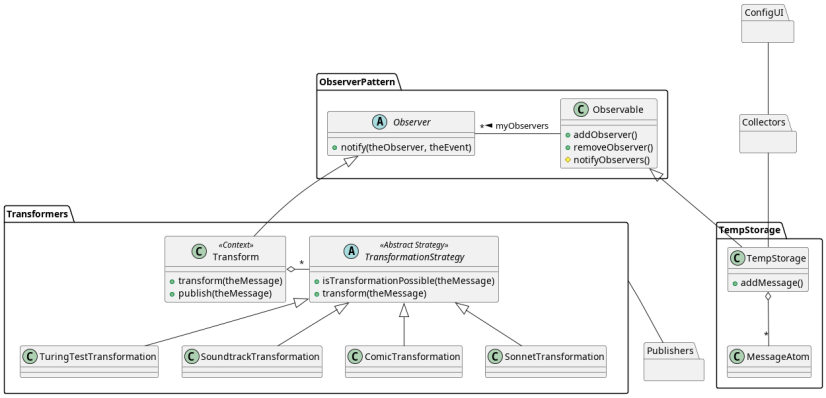
3 The Transformers Package

Implementing the Transformers using a `Strategy Pattern`.



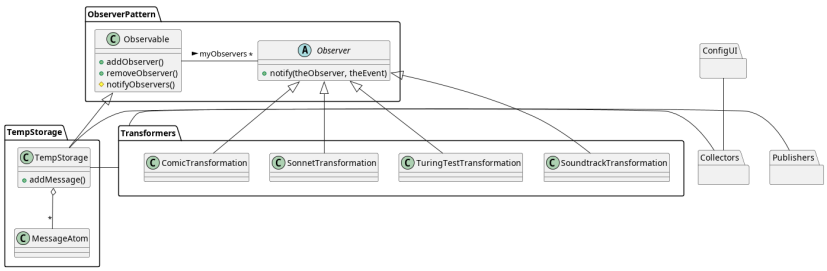
4 Fixing the Collectors → Transformers dependency

1. Make Collectors write to a `TempStorage`.
2. Impose an `Observer Pattern` to allow TempStorage to notify when new data is available.



5 Cleaning up Design Patterns

Since Observer is essentially a Strategy Pattern, we can clean up and merge our Strategy Pattern with our Observrer. This means that each Transformer now implements a `notify()` method.



6 Some Code for the Transformers::notify() method

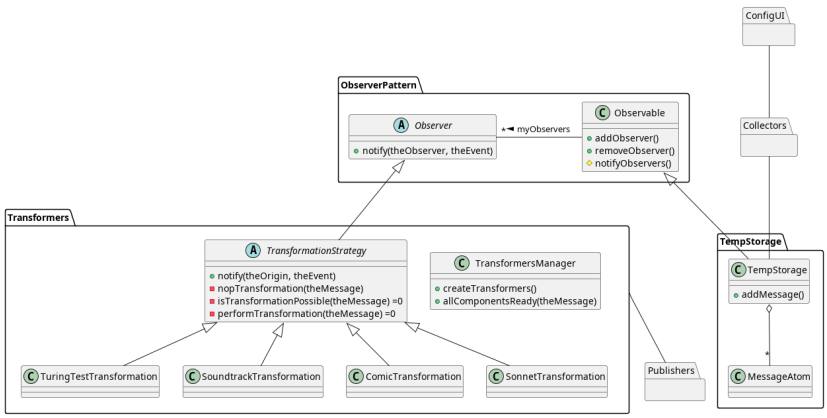
```
ComicTransformation::notify(theOrigin, theEvent) {
  if(isTransformationPossible(theEvent.message)) {
    Publishers::publish( performTransformation(theEvent.message) );
  } else {
    // Do nothing
  }
}
```

7 New TransformationStrategy class

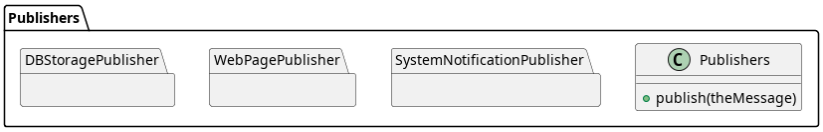
The problem is now that each Transformer implements the same method in exactly the same way.

Solution: Re-introduce the TransformationStrategy class, but for the purpose of gathering behaviour common for all transformers.

Each Transformation *is a* TransformationStrategy, which *is an* Observer, so by inference all Transformations are also Observers.



8 The Publishers Package



Some code for the `publish()` method.

```
(defun publish (theMessage)
  (add-component! theMessage)
  (when (all-components-ready? theMessage)
    (publish-dwim! theMessage)))

(defun all-components-ready? (theMessage)
  (Transformers::allComponentsReady theMessage))
```

A problem now is that the `publish()` method has to go back to the Transformers package to figure out if everything us ready. This means the connection between the two packages is tighter than it needs to be. So we create a new class in the Transformers package, a `MessageMerger` class.

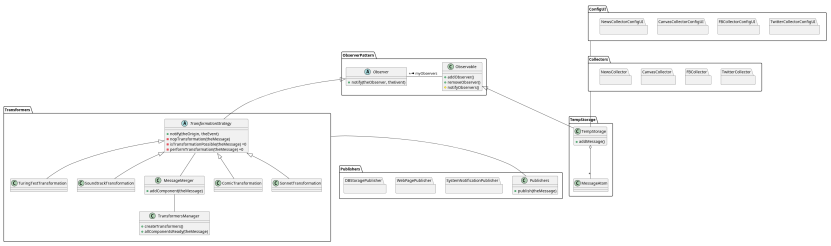
9 Modified Transformers Package

All in one go. Re-including everything which I have hidden along the way so this is the final diagram with everything in it:

```
package ConfigUI {
 package TwitterCollectorConfigUI {}
 package FBCollectorConfigUI {}
 package CanvasCollectorConfigUI {}
 package NewsCollectorConfigUI {}
}

package Collectors {
 package TwitterCollector {}
 package FBCollector {}
 package CanvasCollector {}
 package NewsCollector {}
}

package ObserverPattern {

class Observable {
+addObserver()
+removeObserver()
#notifyObservers()
}
```

```
abstract class Observer {
+notify(theObserver, theEvent)
}

Observable - "*" Observer : myObservers >

}


package Transformers {

class TransformersManager {
+createTransformers()
+allComponentsReady(theMessage)
}

'class Transform <<Context>> {
'+transform(theMessage)
'+publish(theMessage)
'}

abstract class TransformationStrategy {
+notify(theOrigin, theEvent)
-nopTransformation(theMessage)
-isTransformationPossible(theMessage) =0
-performTransformation(theMessage) =0
}

Observer <|-- TransformationStrategy

TransformationStrategy <|-- ComicTransformation
TransformationStrategy <|-- SonnetTransformation
TransformationStrategy <|-- TuringTestTransformation
TransformationStrategy <|-- SoundtrackTransformation

class MessageMerger {
+addComponent(theMessage)
}

MessageMerger -- TransformersManager

TransformationStrategy -- MessageMerger

}


package Publishers {
class Publishers {
+publish(theMessage)
}
```

```
package SystemNotificationPublisher {}
package WebPagePublisher {}
package DBStoragePublisher {}
}

package TempStorage {

class TempStorage {
+addMessage()
}

Observable <|-- TempStorage

class MessageAtom
TempStorage o-- "*" MessageAtom

}

ConfigUI -- Collectors
Collectors -- TempStorage
Transformers -- Publishers
```



10 Modified Code for the Transformers::notify() method

```
TransformationStrategy::notify(theOrigin, theEvent) {
  if(isTransformationPossible(theEvent.message)) {
    myMessageMerger.addComponent( performTransformation(theEvent.message) );
  } else {
    myMessageMerger.addComponent( nopTransformation(theEvent.message) );
  }
}
```