3 Tests in bestehendem Code hinzufügen

Neben neuen Features, sind Erweiterungen bzw. Wartung der Software die größten Herausforderungen eines Softwareentwicklers. Oft stehen wir vor Code der schlecht verständlich und ohne Tests ist. Dort größere Refaktorisierungen zu machen, ist mit dem Risiko verbunden bestehendes Verhalten zu verändern.
Daher ist unser erstes Ziel, Tests zu schreiben, die dieses Verhalten dokumentieren.

3.1 Wo soll ich meine Änderungen am Besten durchführen? (Änderungspunkte)

In den vorherigen Kapiteln haben wir gelernt, wie wir bestehenden Code besser verstehen können.
Die Änderungspunkte bestimmen wir mittels einer Effektskizze. Die Wurzel der Effektskizze sind unsere Stellen wo wir Änderungen vornehmen werden.

3.2 Wie soll ich die Änderungen testen? (Testbarkeit)

Testbarkeit wird oft durch Abhängigkeiten beeinträchtigt. Es gibt verschiedene Herangehensweisen um dennoch Tests dafür schreiben zu können.

3.2.1 Seam

Dazu müssen wir die Problemstellen finden und dort einen Austauschpunkt (Seam) erstellen.

Definition. Ein Seam ist eine Stelle wo man das Verhalten in einem Programm verändern kann, ohne an dieser Stelle den Quelltext ändern zu müssen.

- Objektseam mittels Schnittstelle (wenn mehrere Funktionen)
- Objektseam mittels virtueller Methode (bei Einzelfunktion)

3.2.2 Funktionalität extrahieren

Die zu testende Funktionalität wird in eine Funktion/Klasse extrahiert, die sich leichter testen lässt. Oft genutzt beim Refaktorisieren nach **IOSP** bzw. **PoMO**.

3.2.3 Compile and Try

Naiv einen Test zu schreiben und dann die Compile- und Laufzeitfehler nach und nach zu lösen, indem wir die notwendigen Werte/Abhängigkeiten richtig setzen.

3.3 Was soll ich testen? (Testfälle)

Um sinnvolle Tests zu schreiben muss ich wissen, welche Szenarios mit welchen Ergebnissen ich benötige.

3.3.1 Logikpfade verfolgen

Dazu muss ich alle möglichen Pfade durch meine Funktion durchwandern und für jeden einen Test schreiben.

3.3.2 Alle möglichen Eingaben

Hier schaue ich, welche Werte meine Parameter überhaupt annehmen können und schreibe für jeden Bereich einen Test.

3.4 Wo frage ich mein Ergebnis für den Test ab? (Testpunkte)

Irgendwo muss mein Test das Ergebnis abfragen können.
Diese Testpunkte finden wir anhand der Blätter der Effektskizze.

Refaktorisierungen für Tests

vorher | **nachher**

Extrahiere in virtuelle Methode

```
class LoadSave {
  bool LoadEffect() {
    ...
    if(failure) {
      ::ShowMessage(L"wrong effect");
      return false;
    }
  }
};
```

```
class LoadSave {
  bool LoadEffect() {
    ...
    if(failure) {
      this->ShowMessage(L"wrong effect");
      return false;
    }
  }
  virtual void ShowMessage(const wchar_t* msg) {
    ::ShowMessage(msg);
  }
};
```

Extrahiere Schnittstelle

```
class Persistence {
  bool Load(LargeProgramData& data,
            int version) {
    std::ofstream file("checkpoint.xml");
    if(version > 5) {
      int length = 0;
      file.read(&length, sizeof(length));
      if(length > 0) {
        data.title.resize(length);
        file.read(data.title,...);
      }
    }
    ...
  }
};
```

```
class FileReader {
  virtual bool ReadNumBytes(void*, size_t) = 0;
  virtual bool Open(const char* filename) = 0;
  virtual ~FileReader() = default;
};
//Implementation of FileReader skipped here

class Persistence {
  bool Load(LargeProgramData& data,
            int version) {
    reader->Open("checkpoint.xml");
    if(version > 5) {
      int length = 0;
      reader->ReadNumBytes(&length, sizeof(length));
      if(length > 0) {
        data.title.resize(length);
        reader->ReadNumBytes(data.title, ...);
      }
    }
    ...
  }
  FileReader* reader = nullptr;
};
```

Trenne nach IOSP und PoMO

```
bool WaveBuffer::SetSize(long newSize,
                         long flags) {
  if(newSize < 0)
    newSize = GetMaxBufferSize();
  else if(newSize < sizeof(WAVEFORMATEX))
    newSize = sizeof(WAVEFORMATEX);
  if(newSize > currentSize
  || flags & FLAG_RECREATE) {
    DeleteBuffer();
    buffer = new char[newSize];
    if(buffer == nullptr) {
      currentSize = 0;
      return false;
    }
    currentSize = newSize;
  }
  if(buffer != nullptr)
    memset(buffer, 0, currentSize);
  return true
}
```

```
bool WaveBuffer::SetSize(long newSize, long flags) {
  newSize=RectifyBufferSize(newSize,GetMaxBufferSize);
  if(IsResizeNecessary(newSize, flags))
    return ResizeBuffer(newSize);
  else
    return ResetBufferContent();
}

bool WaveBuffer::IsResizeNecessary(long size, long f) {
  return size > currentSize || f & FLAG_RECREATE;
}
...
```