Contract-based Software Development

Rasmus Guldborg Pedersen

January 2015

Overview

1. The 6 Principles
2. Interface Specification

Design of interfaces as contracts

Present a number of principles for designing interfaces as contracts. You may give examples in order to clarify the principles. How can we specify interfaces? You may include Code Contracts.

Principle 1

Separate queries from commands.

Example: *Pop* for a stack.

Principle 1

Separate queries from commands.

Example: *Pop* for a stack.
Introduce *Peek* and *Remove*.

Principle 2

Separate basic queries from derived queries.

Example: *Peek* is derived from *ElementAt*.

Principle 3

For each derived query, write a postcondition that specifies what result will be returned in terms of one or more basic queries.

Assume: *Count*

Principle 3

For each derived query, write a postcondition that specifies what result will be returned in terms of one or more basic queries.

Assume: *Count*

$Peek \rightarrow Post : Result = ElementaAt(Count)$

Principle 4

For each command, write a postcondition that specifies the value of every basic query.

$Remove \rightarrow Post : Count = \textbf{old} Count - 1$

Principle 5

For every query and command, decide on a suitable precondition.

$Peek \rightarrow Pre : Count > 0$

Principle 6

Write invariants to define unchanging properties of objects.

$Invariant : Count \geq 0$

Interface Specification

Interface specifications using Code Contracts.

Interface

```
public interface ISimpleQueue {
    void Enqueue(object item);
    object Dequeue();
    object ElementAt(int index);
    int Count();
}
```

Contract

```
abstract class ISimpleQueueContract {
    public void Enqueue(object item) {
        Contract.Requires(item != null);
        Contract.Ensures(Count ==
            Contract.OldValue(Count()) + 1);
        Contract.Ensures(ElementAt(Count()) == item);
        // ...
    }
    // ...
}
```

Associating Interface with Contract

```
[ContractClass(typeof(ISimpleQueueContract))]
public interface ISimpleQueue { /* ... */ }

[ContractClassFor(typeof(ISimpleQueue))]
abstract class ISimpleQueueContract { /* ... */ }
```

The End

"Testing shows the presence, not the absence of bugs."
— Edsger W. Dijkstra