Design by Contract

Wouter Swierstra, Hans Philippi

October 4, 2019

MSO so far

- Recap Abstract Classes
- UP & Requirements
- Analysis & UML
- OO & GRASP principles
- Design Patterns
  (Facade, Adapter, Strategy, Bridge, Abstract Factory)
- Midterm about these topics

Today's lecture

And now for something completely different...

Today's lecture

How to design *correct* software?

A philosophical question. . .

What does it mean for software to be correct?

A philosophical question. . .

What does it mean for software to be correct?

- Free of bugs?
- Will not loop?
- Never throws an exception?
- Does what it's supposed to?

Specifications

- The main issue of software correctness is that it *meets* its specifications.

Specifications

- The main issue of software correctness is that it *meets* its specifications.
- But who writes specifications?

Specifications

- The main issue of software correctness is that it *meets* its specifications.
- But who writes specifications?
- And who says that the specifications do not contain bugs?

An example: absolute value

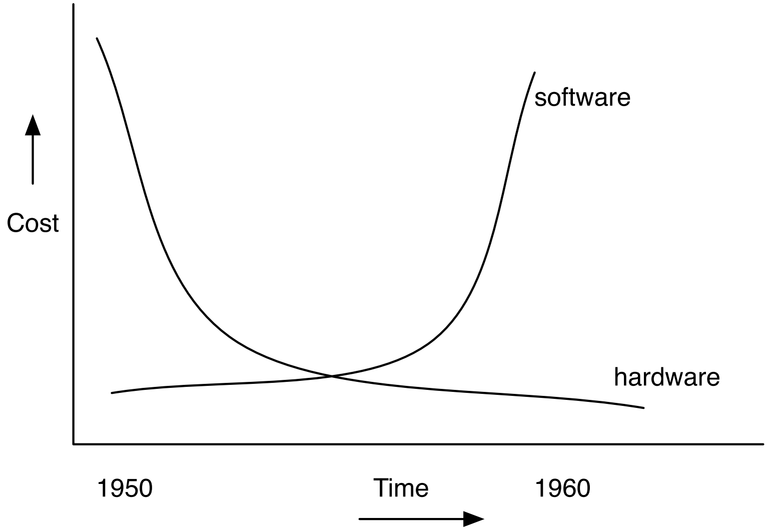
```
public int AbsoluteValue(int i)
{
   if (i > 0)
     return i;
   else
     return -i;
}
```

Does `AbsoluteValue(x)` return a number greater than or equal to 0 for all x?
How could we prove that this is the case?
Or how could we disprove it?

A bit of history. . .

- In the early days of computing, *hardware* was expensive and tricky to develop
- Midway through the last century, something started to change..

The Software Crisis



The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

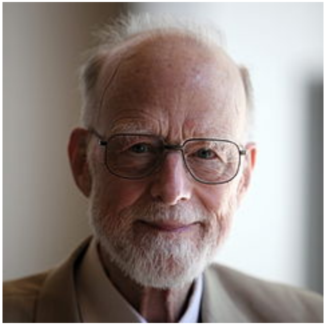
Edsger Dijkstra, *The Humble Programmer*, Turing Award Acceptance Speach, 1972

What can we do?

- Software is suddenly becoming a lot more complex
- How can we cope with this complexity?
- Note that these issues were already identified before the era of OO programming

Foundations of structured programming

Edsger Dijkstra
*1930, †2002


Tony Hoare
*1934

Hoare logic

$$\{P\}\ C\ \{Q\}$$

Read as:
Provided the *precondition* P holds at the state before the execution of program C, then the *postcondition* Q will hold afterwards, or C does not terminate.

Hoare logic - examples

- {x > 3} x := x+1; {x > 4}
- {x > 3} x := x+1; {x > 40}
- {x > 3} y := 4; x := y {x = 4}
- {x > 3} while true (x := 3) {x = 3}

Question: Which of these statements are valid?

Designing a logic. . .

Tony Hoare (inspired by Floyd and Dijkstra) designed a series of rules to reason about programs

Skip

The simplest rule is for the `skip` command, that does nothing

$$\{P\}\ \text{skip}\ \{P\}$$

For example, $\{x > 3\}$ skip $\{x > 3\}$ is valid

Assignment

$$\{P[e/x]\}\ x := e\ \{P\}$$

The *precondition* for

$$x := e\ \{P\}$$

is uniquely determined by

$$\{P[e/x]\}$$

i.e. the predicate P where each x is substituted by e

Assignment

$$\{P[e/x]\}\ x := e\ \{P\}$$

For example:

$$\{x + 1 \leq N\}\ x := x + 1\ \{x \leq N\}$$

is valid

Composition

Suppose we know that

$$\{P\}\ c_1\ \{Q\} \text{ and } \{Q\}\ c_2\ \{R\}$$

What can we deduce about $c_1; c_2$?

Composition

Suppose we know that

$$\{P\}\ c_1\ \{Q\} \text{ and } \{Q\}\ c_2\ \{R\}$$

We can deduce

$$\{P\}\ c_1; c_2\ \{R\}$$

Composition

For example, if we have

$$\{ x + 1 = 43 \}\ y := x + 1\ \{ y = 43 \}$$

and

$$\{ y = 43 \}\ z := y\ \{ z = 43 \}$$

we can deduce

$$\{ x + 1 = 43 \}\ y := x + 1;\ z := y\ \{ z = 43 \}$$

Hoare logic

- You can add more and more rules - some of which can be quite tricky (like how to deal with loops, recursive functions, pointers, . . .)
- Using these rules you suddenly have the ability to *prove* statements about your program.
- This was a major breakthrough!

Back to analysis and design. . .

- These notions of *precondition* and *postcondition* are still applied throughout computer science today
- In particular, we are going to look at *Design by Contract* or *Code Contracts* – design and programming methodology to write robust software
- These ideas influence your entire design: from use case and requirements, through the UML, down to the code.
- ... and don't forget: *in most cases,* **finding** *the bug requires much more effort than repairing the bug*

Bertrand Meyer and Eiffel

Bertrand Meyer (*1950) introduced *Design by Contract* in 1986.
His language *Eiffel* has special support for contracts.

Contracts

In real life, we make *contracts* all the time.
Think of the Post.nl parcel delivery service:

	Obligations	**Benefits**
Customer	Provide a letter or parcel no heavier than 2 kg; Pay 6.50 euros	Guaranteed delivery within two days
Post.nl	Deliver parcel to correct address within two working days	No need to deal with unpaid deliveries, parcels that are too heavy, big, etc.

Contracts

Contracts establish some agreement between two parties:

- Each party expects certain benefits, but is willing to incur obligations to obtain them
- These benefits and obligations are documented in a contract document

This contract document protects both parties!

Assertions in code

C# provides methods to help write pre- and postconditions in your code Example:

```
public float reciprocal (int x) {
    return 1/x;
}
```

Clearly this code can fail – what happens when we call it with 0?

Assertions in code

```
public float reciprocal (int x) {
    Debug.Assert(x != 0);
    return 1/x;
}
```

- If another method calls `reciprocal(0)`, the code will throw an exception and show the call stack
- Variants exist that will allow you to display custom error messages, etc.

Assertions in code

Besides checking preconditions, we can also use assertions to check that our code produces the correct result:

```
public int[] sort (int[] xs) {
    int[] result = ...
    Debug.Assert(isSorted(result));
    return result;
}
```

Now if anyone changes the method body, introducing a bug, our program will fail when run

Contracts in software design

The same ideas can be applied to *software design*

- Decorate methods with assertions, stating which contractual obligations:
  - it *requires* of the caller
  - it *ensures* for the caller

These assertions form an important piece of *documentation*, specifying what a method should do

Case study: customer relation management

Suppose we need to write software for managing customers relations. Typical use cases may include:

- adding a new customer
- billing customers
- making appointments with customers
- ...

Use case example: adding a new customer

1. An account manager fills in a customer's name and address information in the system
2. The system adds the customer to the customer database; it returns a new, unique customer ID
3. The customer is sent a welcome email
4. The marketing department is notified to pursue potential business leads

Use case example: adding a new customer

Preconditions: What preconditions would you associate with this use case?
Postconditions: What postconditions would you associate with this use case?

1. An account manager fills in a customer's name and address information in the system
2. The system adds the customer to the customer database. It returns a new, unique customer ID
3. The customer is sent a welcome email
4. The marketing department is notified to pursue potential business leads

Use case example: adding a new customer

Preconditions:
The customer is not yet entered in the database
There is a live connection to the database and servers
The account manager is logged in and authorized to add new customers

. . .

Postconditions:
The customer is now stored in the database
The customer is assigned a unique ID

. . . .

Why?

Why write such pre- and postconditions in use cases?

- It can help developers figure out *what* is really going on, and under which conditions they can abort
- It's a first step towards (automated) testing
- It provides information for writing *contracts*
- It forces *you* to think about what is really going on

Defensive programming

Taking these ideas seriously leads to *defensive programming*, where you assume that everyone is out to crash your code

- Never trust input – if you assume the argument is greater than 0, add an assertion to check this
- Fail early and openly – check any preconditions before entering the method body
- Document your assumptions – explicitly state the preconditions, postconditions, and invariants upon which your code relies

Contracts in code

I want to discuss one implementation of these ideas, Microsoft's *Code Contracts* library - available for you guys in C# **now**.

Lots more info on the Code Contracts website:

`http://research.microsoft.com/en-us/projects/contracts/`

Play online at:

`http://rise4fun.com/CodeContracts`

Contract

You can think of Code Contracts as establishing a contract between the *users* of a class or method and the *developers* of a class or method

Contract

You can think of Code Contracts as establishing a contract between the *users* of a class or method and the *developers* of a class or method

More concrete:

- You and one of your team members (*”what the hell did he do here?”*)

Contract

You can think of Code Contracts as establishing a contract between the *users* of a class or method and the *developers* of a class or method

More concrete:

- You and one of your team members (*"what the hell did he do here?"*)
- You and an external person responsible for parts of the software (*"what the hell did she do here?"*)

Contract

You can think of Code Contracts as establishing a contract between the *users* of a class or method and the *developers* of a class or method

More concrete:

- You and one of your team members (”*what the hell did he do here?*”)
- You and an external person responsible for parts of the software (”*what the hell did she do here?*”)
- You and yourself (”*what the hell did I do here?*”)

Simple (Pseudocode) example

```
public int addCustomer(String name)
{
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

Simple example

```
public int addCustomer(String name)
{
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

What are the preconditions?

Simple example

```
public int addCustomer(String name)
{
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

What are the preconditions?

- essential customer data should not be null
- this customer does not yet exist in our database

Adding preconditions

```
public int addCustomer(String name)
{
    Contract.Requires(name != null);
    Contract.Requires(!customers.Contains(name));
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

- Add preconditions using Contract.Requires
- These are checked before entering the method body

Postconditions?

```
public int addCustomer(String name)
{
    Contract.Requires(customers != null);
    Contract.Requires(!customer.Contains(name));
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

What about postconditions?

- Guarantee that the name now occurs in the customers dictionary

Adding postconditions

```
public int addCustomer(String name)
{
    Contract.Requires(customers != null);
    Contract.Requires(!customer.Contains(name));
    Contract.Ensures(
      customers[Contract.Result<int>()]
      == name);
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

- Add postconditions using Contract.Ensures
- Use Contract.Result to refer to the result of a method
- Postconditions are checked just before the method returns

Establishing blame

One important reason for using contracts is to establish *blame*; suppose A uses a method created by B

- who is at fault when a precondition for the method does not hold?
- who is at fault when a postcondition for the method fails to hold?

Establishing blame

One important reason for using contracts is to establish *blame*

- who is at fault when a precondition does not hold?
  - The code that *called* the method defining the failing precondition: A is to blame
- who is at fault when a postcondition fails to hold?
  - The method that defines the postcondition: B is to blame

Example

```
public int addCustomer(String name)
{
    Contract.Requires(name != null);
    Contract.Requires(!customer.Contains(name));
    Contract.Ensures(
      customers[Contract.Result<int>()]
      = name);
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

- If the precondition fails, for example name is null, the code calling this method is at fault; it should have checked the name

Example

```
public int addCustomer(String name)
{
    Contract.Requires(customers != null);
    Contract.Requires(!customer.Contains(name));
    Contract.Ensures(
      customers[Contract.Result<int>()]
      = name);
    int id = customers.freshId();
    customers.Add(id,name);
    return id;
}
```

- If the postcondition fails, the error is in this method (or the method it calls)

Strong or weak?

- A very *strong* precondition places a lot of responsibility on the code *calling* this function
- A very *strong* postcondition places a lot of responsibility on the method ensuring that this postcondition holds

Why use Code Contracts?

- Generate *dynamic* checks, as the code runs
- Also *statically checks* contracts!
- Generate *documentation*!

Demo

`http://rise4fun.com/CodeContracts`

Invariants

- Besides pre- and postconditions, Floyd-Hoare-Dijkstra identified another important notion: *invariants*
- An invariant is a property that holds continuously throughout execution

Invariants: example

```
int x := 0;
while (x < 10)
{
  x := x + 1;
}
```

What *invariant* can we identify in the body of this loop?

Invariants: example

```
int x := 0;
while (x < 10)
{
  x := x + 1;
}
```

What *invariant* can we identify in the body of this loop? Choices may include:

- x is in the range {0 .. 10}
- x <= 10
- x is an integer value
- ...

Invariants in design-by-contract

- In the same spirit, the Design by Contract lets you identify invariants that should hold on your objects:
- Question: what invariants should our CRM software satisfy?

Invariants in design-by-contract

- In the same spirit, the Design by Contract lets you identify invariants that should hold on your objects:
- What invariants should our CRM software satisfy?
- We have address information about all our customers;
- We don't have duplicate customers in our database;
- ...

Specifying invariants

```
class Rational
{
  int numerator; int denominator;

  ....

  [ContractInvariantMethod]
    void ObjectInvariant ()
    {
      Contract.Invariant(denominator != 0);
      Contract.Invariant(numerator < denominator);
    }
}
```

Checking invariants

- When should invariants be checked?
- This is not an easy question

```
class Rational
{
  int numerator; int denominator;

  void normalize()
  {
    int x = gcd(numerator, denominator);
    denominator = denominator/x;
    numerator = numerator/x;
  }
}
```

This may temporarily break the invariant...

Invariant checking

- Microsoft's Code Contracts library chooses to check object *invariants on every* exit from a *public* method.
- Other design choices exist. . .

Contracts and inheritance

There is tricky interaction between contracts and inheritance

- Suppose a class Collection has a contract associated with the `sort()` method.
- Now you define a subclass of the Collection class, `ShapeCollection` that overrides the `sort()` method.
- What contract should the new `sort()` method satisfy?

Contracts and inheritance

This is not an easy question.

- If `ShapeCollection` requires a stronger precondition, calls to `sort` on a `Collection` may trigger failure
- If `ShapeCollection` ensures a weaker postcondition, calls to `sort` may not get the guarantees they are expecting

Both the opposite changes are allowed, that is, subclasses are allowed to weaken preconditions (but not strengthen them) and strengthen postconditions and invariants (but not weaken them)

Interaction with other features

- You can also add contracts to abstract methods or interfaces
- There is lots more to read about Code Contracts – check the website!

Why?

We talk about design by contract for various reasons:

- It makes design a more precise process – the more exact input you can give developers about your design, the better software they will write
- It is a design philosophy that fits well with the rest of the course
- I hope it motivates you to think about software correctness, or how such a tool might work . . .

How?

How does Microsoft's Code Contract library work?

How?

How does Microsoft's Code Contract library work?

Magic.

Learning more

We teach a lot of courses that help you learn how such tools work, and how to write you own:

- Compiler Construction
- Automated Program Analysis
- Program verification
- ...

Material covered

Code Contracts:
http://research.microsoft.com/en-us/projects/contracts/

A first taste of design by contract:
http://www.pearsonhighered.com/samplechapter/0201634600.pdf

Design by contract:
http://en.wikipedia.org/wiki/Design_by_contract
Bertrand Meyer's *Applying Design by Contract*.