BASTA!

.NET, WINDOWS, VISUAL STUDIO

Besseres C#

Workshop

BASTA! Spring 2014

Code Contracts

Workshop

Rainer Stropek

software architects gmbh

Web http://www.timecockpit.com
Mail rainer@timecockpit.com
Twitter @rstropek

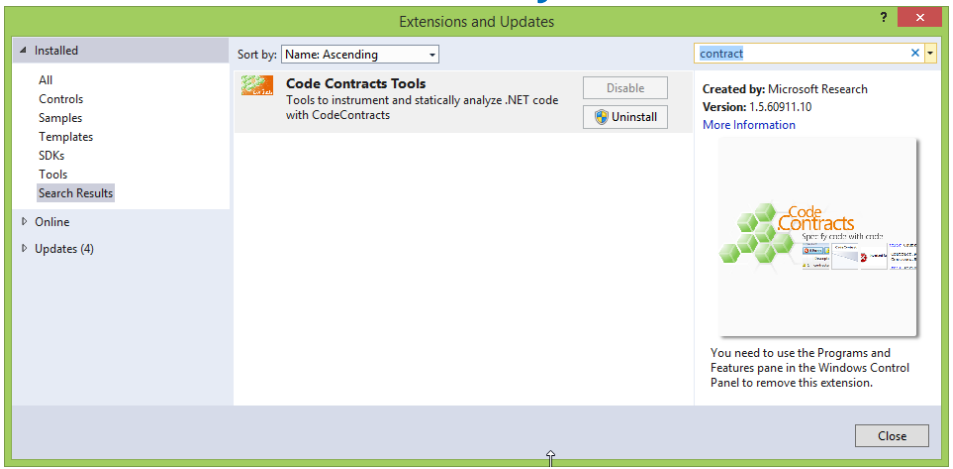
time cockpit
Saves the day.

What are Code Contracts

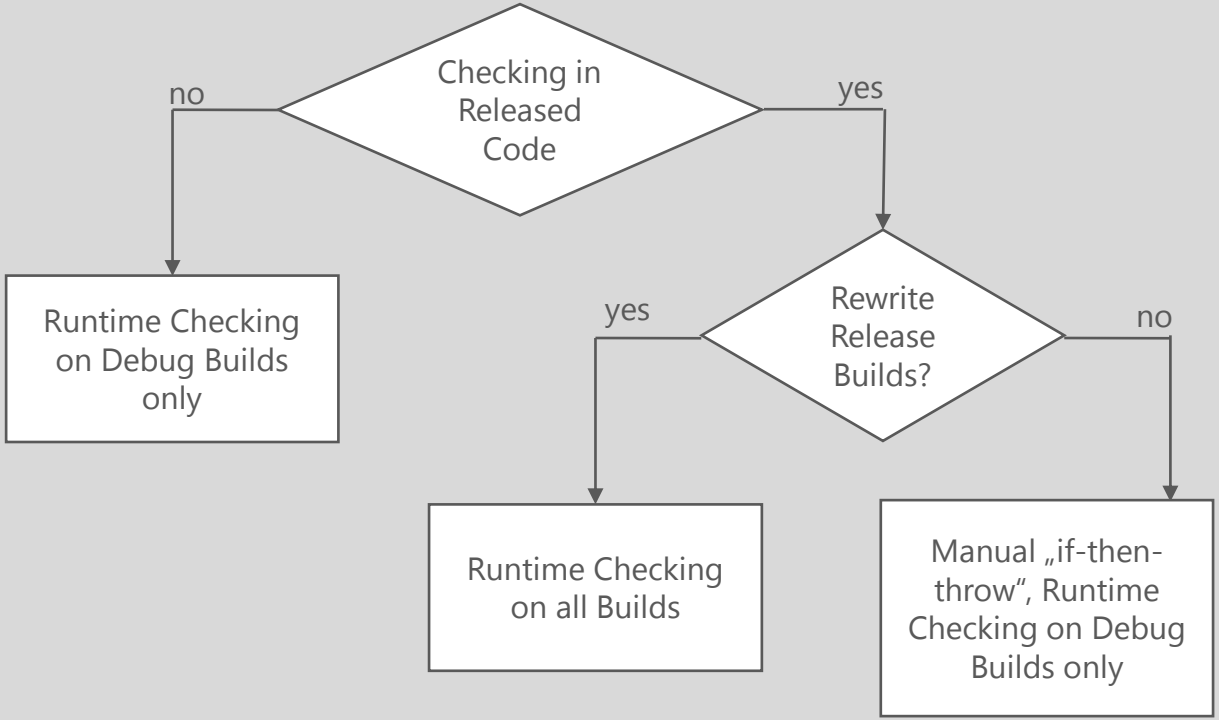
- Conditions
  - Preconditions
  - Postconditions
  - Object invariants
- Runtime Checking
  - Rewrites IL Code
  - Conditional
- Static Code Analysis
- Documentation
  - E.g. with Sandcastle

Code Contracts in .NET and VS

- System.Diagnostics.Contracts namespace
- Visual Studio Gallery



Runtime Checking



```
Contract.Requires(x != null);


// Use the following syntax if you need to throw a specific
// exception. This is only available if you run code contracts
// rewriter on all builds.
Contract.Requires<ArgumentNullException>(x != null, "x");


// „Legacy requires“ – necessary if you want runtime checking
// even on release builds without running code contracts
// rewriter
if ( x == null )
{
  // Single throw statement, nothing else!
  throw new …Exception(…);
}
// No „else“ clause
…
// Necessary after all “legacy requires”
Contract.EndContractBlock();
```

Preconditions

Contract.Requires

Note that members mentioned in preconditions must have proper visibility

- Important so that caller could check precondition prior calling
- See also ContractPublicPropertyName

```
Contract.Ensures(this.X > 0);


// Postcondition in case of an exception
Contract.EnsuresOnThrow<Exception>(this.X == 0);


// Postcondition on result
Contract.Ensures(Contract.Result<int>() > 0);


// Accessing old values in ensures
Contract.Ensures(Contract.OldValue(this.X) < this.X);


// Postcondition on out variables
Contract.Ensures(Contract.ValueAtReturn(out x) > 0);
```

Postconditions

Contract.Ensures

Note that all postconditions for async methods are checked when the method returns the task

Exception: Ensures that contain *Contract.Result<…>().Result*

```
[ContractInvariantMethod]
private void ObjectInvariant()
{
  // Nothing else than calls to Contract.Invariant
  Contract.Invariant(this.x >= 0);
  Contract.Invariant(this.x > this.y);
  ...
}


// Use invariants on automatic properties instead of pre- and
// postconditions
public int X { get; private set; }

[ContractInvariantMethod]
private void ObjectInvariant()
{
  // Will become Contract.Ensures in property getter and
  // Contract.Requires in property setter
  Contract.Invariant(this.X >= 0);
}
```

Invariants

ContractInvariantMethod

Conditions indicating whether an object is in a valid state

Checked at the end of each public method

```
// Checked during runtime and static code analysis
Contract.Assert(this.x > 0);

// Checked during runtime but not proven through static code
// analysis
Contract.Assume(this.x > 0);
```

Assert and *Assume*

```
Contract.Requires(Contract.ForAll(
  listOfValues, // IEnumerable
  value => value > 0));


// Indexed-based syntax
Contract.Ensures(Contract.ForAll(
  0,
  Contract.Result<int[]>().Length,
  index => Contract.Result<int[]>()[index] > 0))
```

Quantifications

Alternatives

Contract.Exists
Enumerable.All
Enumerable.Any

```
static class ContractHelper
{
  // Helper methods for „legacy requires“
  [ContractArgumentValidator]
  public static void NotNull(object arg, string paramName)
  {
    if (arg == null)
    {
      throw new ArgumentNullException(paramName, …);
    }

    Contract.EndContractBlock();
  }
}


ContractHelper.NotNull(x, "x");
```

Contract Helpers

Use *ContractArgumentValidator* for legacy requires

Use *ContractAbbreviator* for set of contracts

Contract Inheritance

- Contracts are inherited
- Preconditions must be defined at the root of the inheritance chain
  Ancestors cannot have weaker or stronger preconditions
- Add stronger postconditions or invariants in ancestor types if needed

Additional Topics

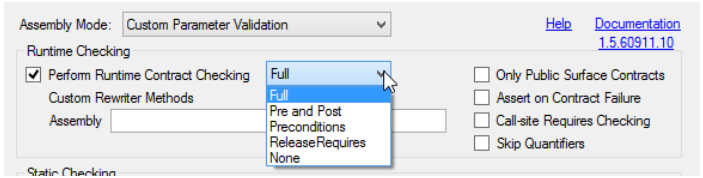
- Contracts for interfaces and abstracts
  See *ContractClass* and *ContractClassFor*
- Use *Pure* attribute to mark side-effect free methods
  Such methods can appear in contracts
- Explicitly control runtime checking and inheritance
  ContractRuntimeIgnored
  ContractOption

Tips, Tricks, and Best Practices

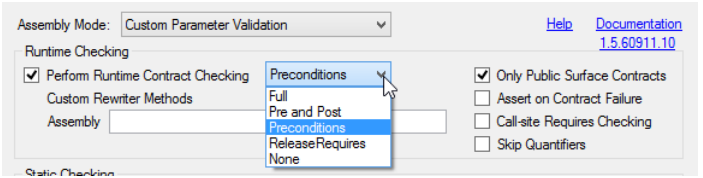
- Do you want to have runtime checking on release builds?
  If yes, decide whether you want to run code contract rewriter on release builds

- Example

Debug



Release



Checking Level	Enabled Runtime Checks						
	Legacy	Requires⟨E⟩	Requires	Ensures	Invariants	Asserts	Assumes
Full	X	X	X	X	X	X	X
Pre and Post	X	X	X	X			
Preconditions	X	X	X				
ReleaseRequires	X	X					
None							

Source: Code Contracts User Docs

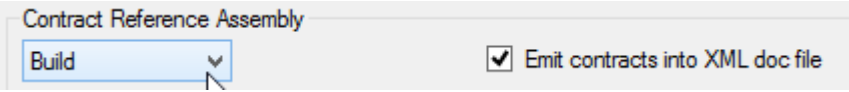
```
[TestClass]
public class InitializeTests
{
  [AssemblyInitialize]
  public static void AssemblyInitialize(TestContext tc)
  {
    Contract.ContractFailed += (sender, e) =>
    {
      e.SetUnwind(); // cause code to abort after event
      Assert.Fail(e.FailureKind.ToString() + ":" + e.Message);
    };
  }
}
```

Tips, Tricks, and Best Practices

- Make sure that contract failures are not hidden with try-catch during unit tests
- Don't write unit tests to intentionally contract failures
  - Exception: Legacy requires

Tips, Tricks, and Best Practices

- If necessary, you can alert other developers to use code contract tools for building
- Code contract tools install handy code snippets
- Deliver a Contract Reference Assembly
  Make contracts available to referencing assemblies



- Add contract information in XML doc file
  Sandcastle understands them

For details see Code Contracts User Docs

BASTA! Spring 2014

Q&A

Thank your for coming!

Rainer Stropek

software architects gmbh

Mail rainer@timecockpit.com
Web http://www.timecockpit.com
Twitter @rstropek

time cockpit
Saves the day.

time cockpit is the leading time tracking solution for knowledge workers. Graphical time tracking calendar, automatic tracking of your work using signal trackers, high level of extensibility and customizability, full support to work offline, and SaaS deployment model make it the optimal choice especially in the IT consulting business.

Try time cockpit for free and without any risk. You can get your trial account at http://www.timecockpit.com. After the trial period you can use time cockpit for only 0,20€ per user and day without a minimal subscription time and without a minimal number of users.

time cockpit

time cockpit ist die führende Projektzeiterfassung für Knowledge Worker. Grafischer Zeitbuchungskalender, automatische Tätigkeitsaufzeichnung über Signal Tracker, umfassende Erweiterbarkeit und Anpassbarkeit, volle Offlinefähigkeit und einfachste Verwendung durch SaaS machen es zur Optimalen Lösung auch speziell im IT-Umfeld.

Probieren Sie time cockpit kostenlos und ohne Risiko einfach aus. Einen Testzugang erhalten Sie unter http://www.timecockpit.com. Danach nutzen Sie time cockpit um nur 0,20€ pro Benutzer und Tag ohne Mindestdauer und ohne Mindestbenutzeranzahl.