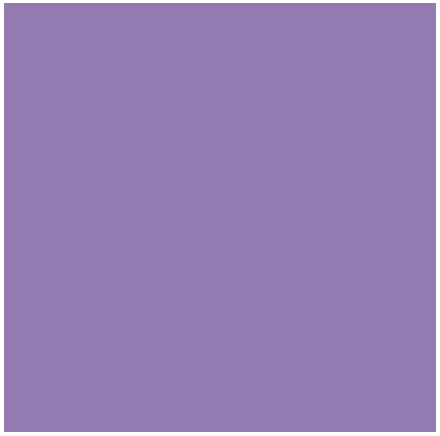
Software Quality Evaluation SQLite C#

André Costa
Manuel Sousa
Ricardo Alves

Summary

- Introduction
- C# SQLite Quality Evaluation
- Unit Testing and Code Contracts
- Final Remarks
- Future work

C# SQLite



- SQLite is a software library that implements a self-contained transactional SQL database engine
- C#-SQLite is an independent reimplementation of the SQLite software library version 3.6.23
- http://code.google.com/p/csharp-sqlite/

Code Quality Evaluation

C# SQLite

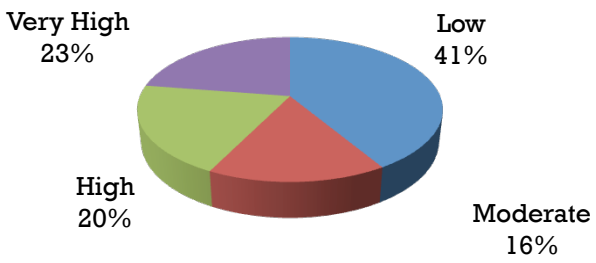
- SQLite divided in 6 different projects
  - Server
  - Client
  - Shell
  - TCL
  - Benchmark
  - Silverlight*
- Evaluation of each project done separately
- SIG maintainability measuring model

Code Quality Evaluation

C# SQLite Server results

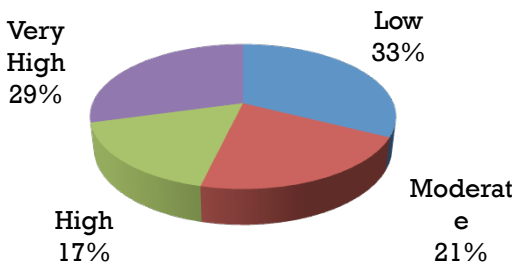
Properties	values	Rank
Volume	25420 code lines	++
Duplication	1246 code lines (4.72%)	++
Unit Testing	0%	--

Complexity Risk - SQLite



Rank: --

Unit Size Risk - SQLite



Rank: --

Final Result

★★

	Volume	Complexity per unit	Duplication	Unite size	Unit testing	
	++	--	++	--	--	
analyzability	X		X	X	X	o
changeability		X	X			o
stability					X	--
testability		X		X	X	--

Evaluation Results

SQLite server	★★
SQLite Client	★★
SQLite shell	★★
SQLite TCL	★★
SQLite Benchmark	★★

Software Improvement Group

Statistics

- C# SQLite is 5 different projects
- Total Volume: ~67000 line of code
- Functions with >300 lines common (biggest has >1000 lines)

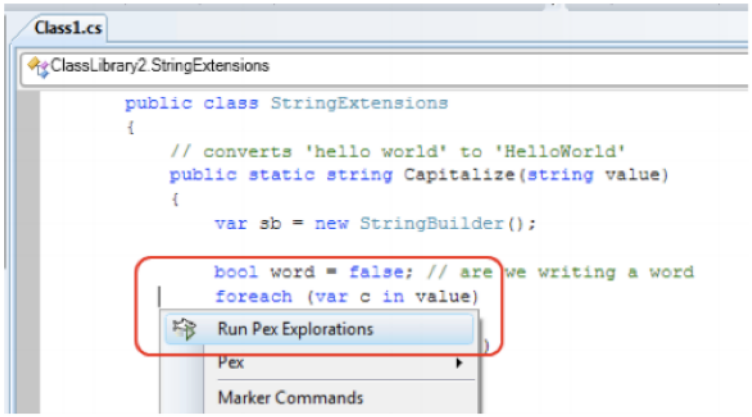
- Complex project!
  - "Manual" approach might not be the best solution
  - Tools?

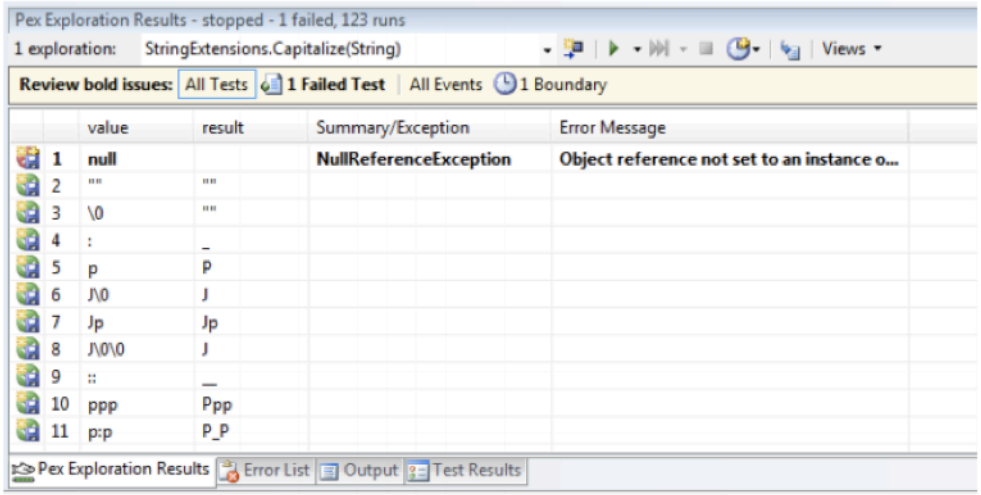
PEX and MOLES frameworks

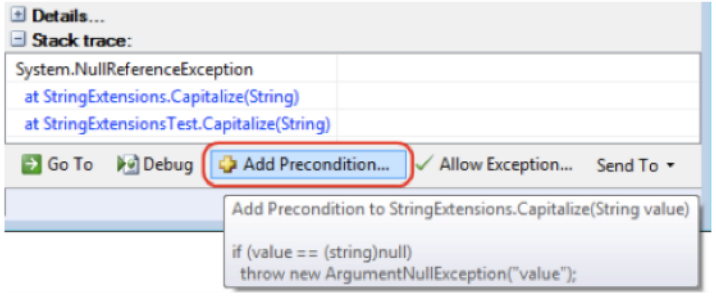
- PEX
  - Pex automatically generates test suites with high code coverage
- MOLES
  - Moles supports unit testing by providing isolation by way of detours and stubs.

Moles - for .NET

PEX


Class1.cs
ClassLibrary2.StringExtensions
public class StringExtensions
{
// converts 'hello world' to 'HelloWorld'
public static string Capitalize(string value)
{
var sb = new StringBuilder();
bool word = false; // are we writing a word
foreach (var c in value)
Run Pex Explorations
Pex
Marker Commands


Pex Exploration Results - stopped - 1 failed, 123 runs
1 exploration: StringExtensions.Capitalize(String)
Views
Review bold issues: All Tests 1 Failed Test All Events 1 Boundary
value result Summary/Exception Error Message
1 null NullReferenceException Object reference not set to an instance o...
2 "" ""
3 \0 ""
4 : _
5 p P
6 J\0 J
7 Jp Jp
8 J\0\0 J
9 :: __
10 ppp Ppp
11 p:p P_P
Pex Exploration Results
Error List
Output
Test Results


Details...
Stack trace:
System.NullReferenceException
at StringExtensions.Capitalize(String)
at StringExtensionsTest.Capitalize(String)
Go To
Debug
Add Precondition...
Allow Exception...
Send To
Add Precondition to StringExtensions.Capitalize(String value)
if (value == (string)null)
throw new ArgumentNullException("value");

Statistics

- Unit tests: 3321
- Assertions: 1037
- Contracts: 371

Unit Tests	Server	Client	TCL	Shell
Coverage	~22%	~36%	~86%	~77%
% Success	~43%	~88%	~62%	~80%

Final Remarks

- C# SQLite is not very well written
- PEX was a great help
- …but has a few problems
- White box testing
- Low coverage on complex projects

Bibliography

- [1] I. Heitlager, T. Kuipers, J. Visser. A Practical Model for Measuring Maintainability, October 2007.
- [2] Microsoft Research. Code Contracts. http://research.microsoft.com/en-us/projects/contracts/, February 2011
- [3] Microsoft Research. PEX. http://research.microsoft.com/en-us/projects/pex/, February 2011
- [4] N. Tillmann, P. Halleux. Code Digging with Pex. Microsoft Research, October 2008.