RiseFlash

Mike Barnett, Manuel Fahndrich, and Francesco Logozzo

<http://visualstudiogallery.msdn.microsoft.com/1ec7db13-3363-46c9-851f-1ce455f66970>

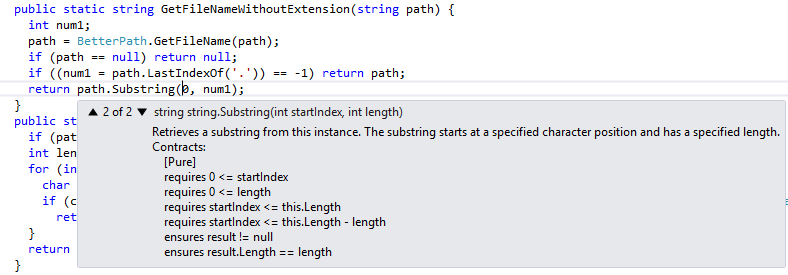
**CodeContracts Background**

CodeContracts is a standardized convention for authoring design choices and program invariants as preconditions, postconditions, and object invariants in .NET C# code. CodeContracts are intended to be both human and machine readable and are part of your code. The authoring capability is present in .NET 4.0 and beyond so that any code base can contain contracts without impacting the build or the shipped bits. Without tools though, CodeContracts are simply comments, helpful if someone reads the code. Our team in RiSE produces the CodeContract tools, available from the VS gallery. The tools consist of five main components:

* ccdoc: augments documentation XML with contract tags for other tools to generate API documentation (e.g., with Sandcastle).
* ccrewrite: instruments .NET assemblies with runtime contract checks, by inserting pre-conditions, post-conditions, and object invariant checks in all the necessary places. Instrumentation takes care of inheriting contracts from interfaces and virtual methods so that e.g., a single contract authored on an interface is checked in all implementations. Full control over the runtime failure behavior of contracts is available.
* ccrefgen: produces contract reference assemblies, i.e., assemblies containing just the metadata and contracts, but no method bodies. These contract reference assemblies are useful for separate compilation, so that each assembly can be instrumented/analyzed while knowing only the contract reference assembly of dependencies. These can also be shipped alongside release bits, so that other developers can get the benefit of coding against contracts, even if the release bits don’t contain them.
* cccheck: is our static checker, where most of our research is targeted. It attempts to verify that each method’s code conforms to the contracts for that method and all the contracts of called methods. It can be used as a bug finding tool by filtering all but the most likely bugs.
* VisualStudio integration: to control the various tools, C# projects are equipped with an extra CodeContracts property pane where the runtime checking, static checking, documentation generation, etc. is enabled per project and per build configuration.

**Tools Status**

The tools recently moved from their original home on Devlabs, where we had over 73K downloads to their new home on the [VS gallery](http://visualstudiogallery.msdn.microsoft.com/1ec7db13-3363-46c9-851f-1ce455f66970). Since the move, we have had over 5000 new downloads. Additionally, we have released new [VisualStudio plugins](http://visualstudiogallery.msdn.microsoft.com/02de7066-b6ca-42b3-8b3c-2562c7fa024f) on the VS gallery that display contracts when hovering over methods, on method signatures, and on metadata displays. The new plugins are less invasive and much more stable.



**Recent Results**

To make deep static analysis scale while keeping it sufficiently precise requires substantial amounts of code annotations in the form of contracts. In order to help the developer discover and author these necessary contracts, we have improved the static analysis with a number of novel contract inference steps, as well as suggestions on fixing warnings. We demoed these improvements along with a Roslyn IDE integration at [TechFest 2013](http://techfest/details/20130238).

[1] Francesco Logozzo and Tom Ball, [Modular and Verified Automatic Program Repair](http://research.microsoft.com/apps/pubs/default.aspx?id=170385), in *Proceedings of the 27th ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'12)*, ACM SIGPLAN, 23 October 2012

[2] Patrick Cousot, Radhia Cousot, Manuel Fahndrich, and Francesco Logozzo, [Automatic Inference of Necessary Preconditions](http://research.microsoft.com/apps/pubs/default.aspx?id=174239), in *in Proceedings of the 14th Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'13)*, Springer Verlag, January 2013

[2] Patrick Cousot, Radhia Cousot, Francesco Logozzo, and Mike Barnett, [An Abstract Interpretation Framework for Refactoring with Application to Extract Methods with Contracts](http://research.microsoft.com/apps/pubs/default.aspx?id=170382), in *Proceedings of the 27th ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'12)*, ACM SIGPLAN, 23 October 2012

**Static Analysis for Code Reviewing**

Can we use the contract static analysis tool for code bases that are not annotated and don’t necessarily use contracts? We are working on what we term *semantic baselineing*¸ a technique to consider the previous version of a program bug-free, and use all the assumptions made in that previous version of the program to check for new problems in the current version of the program. Any new assumptions that are not verifiable in the new version will thus be flagged. We think this technique has great potential for identifying new problems introduced in changelists, and thus beneficial during code reviewing. Contact us for more info or to collaborate on this idea.

**Engagements**

We have worked closely with the [CloudDev](http://clouddev/) team over the last few months to integrate the CodeContract tools into their continuous integration (CI) build. Both rewriting and static checking (on low) are enabled. Static analysis is cached on a team SQL server so that developers can quickly re-analyze code without the cost of a full analysis. As a result of this engagement, we have improved the support for async post-conditions, and fixed pdb issues with async and iterator methods. A number of bugs have been identified by our tools.

**Future Plans**

We want to move the static analysis into a cloud service to take advantage of incremental shared analysis and parallelization. At the same time, we also want to allow crowd-sourcing of contracts for third-party assemblies.