DevLab Proposal for Code Contract Tools

# Description

Contracts are method pre-conditions, method post-conditions, and object invariants. The CLR 4.0 is shipping a contract library for encoding contracts in a language-agnostic way for all .NET languages. This enables programmers to author contracts in any language compiled on top of the CLR. Contracts get compiled by the standard compilers (be it csc for C#, vbc for VB, or the F# compiler) into a language-agnostic, machine readable format.

Research has designed the contract library along with tools to take advantage of these machine readable contracts at the MSIL level.

* **The Contract Rewriter**: The contract rewriter (ccrewrite) instruments an assembly as a post build step such that the resulting assembly performs runtime checking of contracts. The main actions performed during this rewriting are
  + moving post-conditions into the appropriate place on method exits
  + generating code to store away Old values (values from the pre-state)
  + generating code to access the method result
  + inheriting contracts from parent methods
  + instrumenting the object invariant check at exits of public methods
  + checking the well-formedness of contracts
  + adding a string representation of the contract condition being checked for runtime diagnostics
* **The Static Contract Checker**: The static contract checker (cccheck) is an fxcop-like static analyzer performing deep semantic analysis of the code to validate explicit contracts in the code and on called methods. It is based on static analysis techniques developed in Research based on numerical abstract interpretation, an efficient heap abstraction, and a uniform treatment of contracts at the MSIL level.
* **The Contract Reference Assembly Generator**: The contract reference assembly generator (ccrefgen) produces reference assemblies (similar to CLR reference assemblies) that contain only public method signatures and their contracts, but no method bodies. The tool is required for performing proper modular cross-assembly analysis, as well as for shipping libraries in the form of a release library and a separate contract library.
* **Documentation Generation**: We currently have no explicit tool to generate API documentation from assemblies that supports contracts explicitly. We are in discussion with the Sandcastle team as they provide tools for API documentation generation.

The packaging of these tools takes the form of a number of command line utilities that can be run from the command line, as well as a VS2008/VS2010 plugin and MSBuild support. The VS plug-in adds an contract property pane to the property panes of C# and VB projects. When enabling runtime or static checking on this pane, the normal VS build based on msbuild performs extra steps, such as rewriting the assembly for runtime checking, building contract reference assemblies for dependee projects, and running the static checker.

The tools work on the CLR 4.0, CLR 3.5, and CLR 2.0. For pre CLR 4.0 platforms, we provide an external contract library. The Visual Studio integration works for VS2005, VS2008, and Dev10. Additionally, the tools work directly on the command line, as well as in conjunction with command line invocation of msbuild.

# Timeframe

We have a beta available of the tools described above that we could make available as a devlab tool in the next 1-2 months.

# Investment

# Why ship as a DevLab tool?

A DevLab tool provides the best way to evaluate the customer desire to use contracts and the corresponding tools. PDC audience was very excited and rated this tool (along with Pex) the best session of PDC. There is ample history for the desire to have a uniform contract representation and a suite of tools. From a competitive point of view, the Java community has JML as a specification language along with tools such as ESC/Java2. Our advantage is that we managed to integrate this in a language-agnostic way, so as to not require special compilers, whereas the JVM community is dependent on research compilers.

A devlab tool also provides us with the most flexible upgrade path and eventual inclusion as a Visual Studio tool.