よいと思うところ

- FCL が意外と使いやすい。Java のクラスライブラリに比べると、はるかにネーミングやカテゴリ別けが分かりやすい。適当にドキュメントを眺めているだけで、おやっと思って使ってみる、といったクラスがしばしばある。
- for のカウンタ変数や foreach の要素変数がローカルにできる。
- Generic。使いまくり。
- リテラル文字列がある。これがなくてやってられるか。

微妙なところ

- デリゲート。分かりにくくはないと思うが。
- SDK に最新のコンパイラをつけて欲しい。まあ、現実に .NET 3.5 のプラットホームってほとんどないから、まだ C# 2.0 でもいいけど。
- 何かと言うと Visual Studio が出てくる。RAD ツール使うと、余計な決まりごとを覚えなくてはならないし、そんなに生産性上がるか？特に GUI なぞまったく作る気がないユーザは。それに、フレームワークも見えにくくなって理解が進まない。とにかく GUI なんぞ余計なものは絶対に作りたくない。 => 最近はオライリーの本が増えてよい。VS なんて使いたくないし覚えたくない。
- 意外と速いけど微妙な遅さ。

閑話休題：
今まで漠然と、CLR は JVM みたようなもの、と思っていたが、よくよく読んでみると違うらしい。

MSIL は CLR によって逐次実行されることは決して無い。
MSIL は必ず JIT コンパイルされ、コンパイルされたネイティブコードが実行される（それを管理するのが CLR）。
ということは、実行時に、IL を実行しなければならないコストはゼロのはずだ。なぜなら IL はそのままでは決して実行されないから。実行開始時にコンパイルされるだけ。
で、ネイティブコードが実行されるなら、なぜ .NET アセンブリの実行はこんなにも遅いのか。

- CLR による実行管理のオーバーヘッド。ガーベッジコレクションとかアプリケーションドメインの管理やリフレクション情報の管理など、普通の C/C++ アプリケーションが行っていないことを行わなければならない。
- FCL の実行が遅い。

といったことが考えられる。
MD5 計算では Perl にもひけを取らないのに、テキスト処理になるととたんに遅くなったりするのは後者の関係なんだろうな。。。
現在では、JRE とは JVM + JIT コンパイラのことのようなので、CLR とほぼ同じようなものであろう。Java バイトコードが逐次実行されることは現代ではもはやない、ということなんじゃないだろうか。