複数言語対応のソースコード処理ツールのフレームワークとその利用例

１．背景

近年、様々なプログラミング言語（以降、言語）が開発され、その多様化が進んでいる。一方、言語によって記述されたソースコードを処理するツール（以降、処理ツール）が存在する。処理ツールの処理内容は、ソースコードの解析と変形に分けられる。ソースコードの解析が主なツールとして、ソフトウェアメトリクス測定ツールやバグパターン検出ツールが、ソースコードの変形が主なツールとして、ソースコード整形ツールやアスペクト指向プログラミング（Aspect Oriented Programming；AOP）処理系が挙げられる。これらの処理ツールは、ソフトウェアの品質や開発効率を向上させるツールとして注目を浴びている。

現在、多くの処理ツールが言語毎に開発されている。例えば、バグパターン検出ツールの FindBugs [1] は Java 言語のみに、JSLint [2] は JavaScript 言語のみに対応している。また、AOP 処理系の AspectJ [3] は Java 言語のみに、AOJS [4] は JavaScript 言語のみに対応している。このように、言語と処理ツールの間には多対多の関係がある。例えば、各処理ツールが 1 つの言語のみに対応していて、全ての処理ツールと言語の組み合わせを考える場合、**エラー! 参照元が見つかりません。**の左部のように「言語の種類数」×「処理ツールの種類数」通りの実装が必要になる。

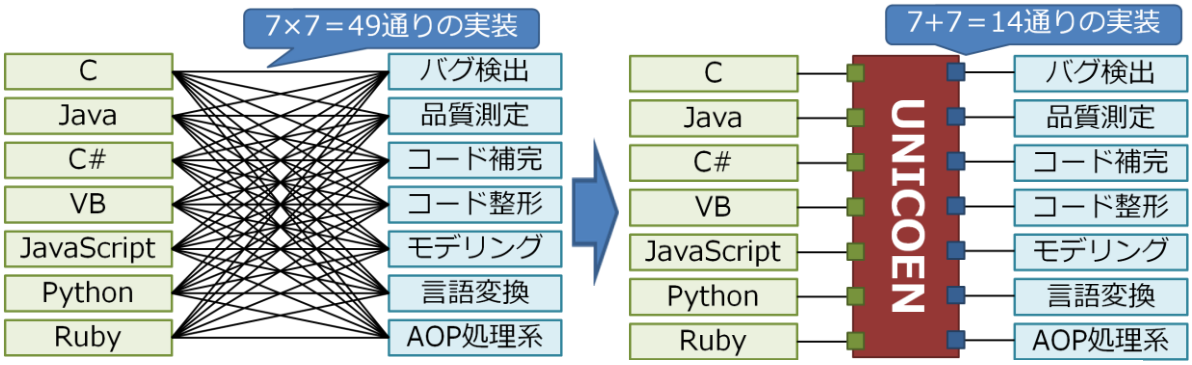

図 1. UNICOEN による言語と処理ツール間の関係の簡略化

２．目的

１章で述べた多対多の関係は、以下の 2 つの問題を引き起こす。***問題 1*：全ての処理ツールが全ての言語に対応するために莫大な開発コストが必要である点**と***問題 2*：対応言語が異なる同じ種類の処理ツール間に差異が生じる点**である。

問題 1 が原因で、いずれの処理ツールにも対応されない言語が存在する。そのため、プログラマが処理ツールによるソフトウェアの品質向上や開発効率向上の

恩恵を受けられない場合がある。例えば、Ruby 言語に対応したバグパターン検出ツールは存在しないため、Ruby プログラマはバグパターン検出ツールを利用してバグの検出が行えない。また、***問題2*** が原因で、異なる言語に対応した同じ種類の処理ツールを組み合わせた場合、処理ツール間の差異のために、期待する結果が得られなかったり、組み合わせるために追加コストが必要になったりする。そのため、複数の言語を利用するプロジェクトで、両方の言語のソースコードを対象とするために、複数の処理ツールを組み合わせて利用できない場合がある。

以上から、***問題1*** と ***問題2*** によって、優れた言語や処理ツールが存在するにもかかわらず、使用する言語によってはその恩恵が十分に得られない。

我々は、上述した問題点を解決するために、ソースコード処理フレームワーク UNICOEN（UNIfied source COde ENgineering framework）を開発した。UNICOEN は言語と処理ツール間の多対多の関係を簡略化して、**エラー! 参照元が見つかりません。** の右部のように言語と UNICOEN 間、処理ツールと UNICOEN 間の多対一の関係に簡略化する。

３．開発の内容

本プロジェクトでは、C# 4.0 で UNICOEN を開発した。そのため、UNICOEN は.NET Framework を動作環境として必要とする。なお、.NET Framework と互換性のある Mono の両方で動作確認を行っている。我々は UNICOEN の有用性を確認するため、C、Java、C#、Visual Basic、JavaScript、Python、Ruby の 7 種類の言語を UNICOEN に対応させた。また、UNICOEN を利用して、ソフトウェアメトリクス測定ツール UniMetrics と CodeCity [5] とのマッシュアップツール、AOP 処理系 UniAspect の 3 種類の処理ツールを開発した。その上で、UNICOEN を利用して開発した処理ツールと既存の処理ツールについて、対応する言語数と開発コストの観点から比較した。その結果、２章で述べた問題点を解決できたことを確認した。

UNICOEN は***改善1***：**言語非依存な統合コードモデルを提供**して、その上で、***改善2***:**統合コードモデル上の汎用的な共通処理をコールドスポットとして提供**する。統合コードモデルは言語非依存な共通の抽象構文木（Abstract Syntax Tree; AST）の仕様を与える。ソースコードから得られる言語非依存な共通の抽象構文木のインスタンスを統合コードオブジェクトと呼ぶ。図 2 のように、汎用的な共通処理は、対応言語拡張者向け API、ツール開発者向け汎用 API、ツール開発者向け特化 API の 3 種類に別れ、3 層のレイヤーアーキテクチャを形成する。図中の青い四角が処理ツール、緑色の四角が言語の構文と統合コードモデル間のマッピング処理を示す。

なお、UNICOEN には、処理ツール開発者と対応言語拡張者の 2 種類のユーザが存在する。処理ツール開発者は、統合コードモデル上でソースコード処理を記述することで、対応する言語を意識せずに処理ツールを開発できる。UNICOEN を利用して開発した処理ツールは、UNICOEN が対応する全ての言語に対応できる。一方、対応言語拡張者はマッピング処理を開発することで、処理ツールを意識せずに対応

言語を追加できる。UNICOEN が対応する言語について、UNICOEN を用いて開発された全ての処理ツールが対応できる。

このようにして、UNICOEN は、**エラー! 参照元が見つかりません。**の対応言語と処理ツール間の多対多の関係を対応言語と UNICOEN 間および処理ツールと UNICOEN 間の多対一の関係に簡略化する。さらに、処理ツール開発者および対応言語拡張者は、汎用的な共通処理を利用することで、低い開発コストで処理ツールとマッピング処理を実装できる。

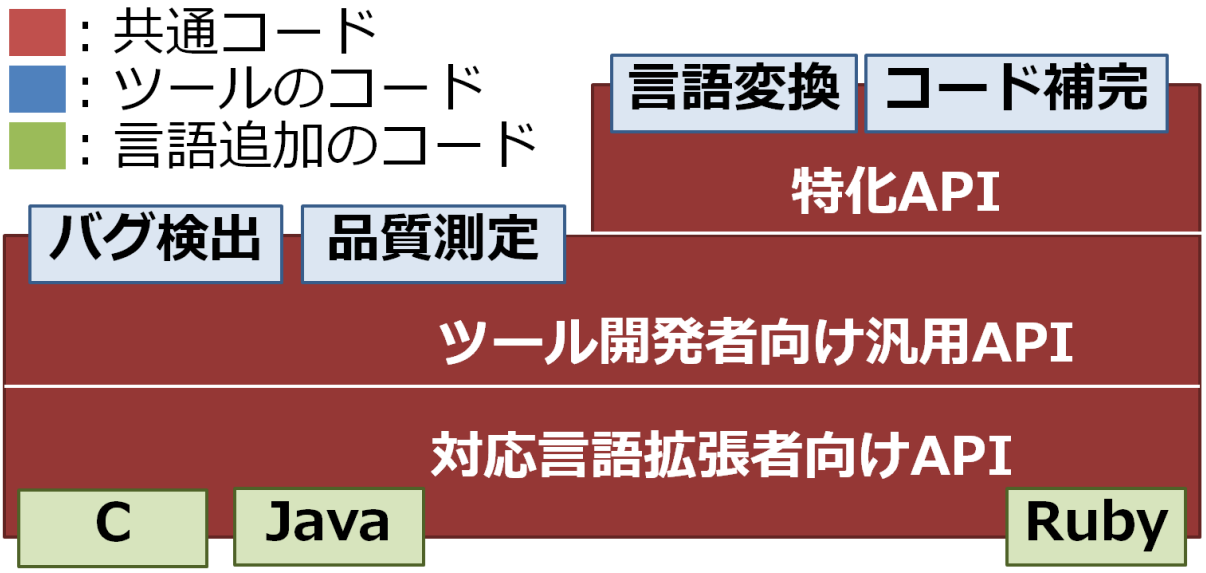

図 2. UNICOEN のアーキテクチャ

４．従来の技術との相違

言語やソースコードを対象とするツールにおける既存の基盤ソフトウェアとして、GCC [6] や LLVM [7] などのコンパイラフレームワークや Java VM や.NET Framework などの中間言語の実行環境が挙げられる。これらの既存ソフトウェアは、各言語のソースコードのセマンティクスを完全に解析して、中間言語に変換する。この中間言語は UNICOEN の統合コードモデルに該当するため、中間言語を介することで、言語を意識せずに処理ツールを開発できる。

しかし、既存ソフトウェアはコンパイラや言語処理系など、セマンティクスを完全に解釈した上でソースコードを処理するツールを対象としている。既存ソフトウェアの中間言語にマッピングする際、完全に意味解析を行う必要があり、対応する言語を追加するコストが非常に大きい。実際、これらの既存ソフトウェアは LLVM を除いて 10 年以上開発されているにもかかわらず、本プロジェクトで UNICOEN に対応させた言語全てに対応するものは存在しない。さらに、既存ソフトウェアでは、ソースコードの意味を厳密に表現できるように中間言語にマッピングする。しかし、言語は多様である。様々な言語のセマンティクスを失わずにマッピングできるような共通の中間言語は、マシン語に近い非常に抽象度の低い仕様となる。その

ため、既存ソフトウェアを利用して処理ツールを開発する際、厳密なセマンティクスの情報を得られるものの、ソースコードの抽象度は失われており、シンタックスの情報は得られない。このことは、既存ソフトウェアでは、コード整形ツールなどのシンタックスの情報を必要とする処理ツールを開発できないことを示す。

一方、UNICOEN は、意味解析を完全に行わないことで、対応する言語の追加および、処理ツールの開発コストを削減する。UNICOEN は完全な意味解析を必要としないため、構文解析の処理を実装するだけで、対応する言語を追加できる。また、統合コードモデルは構文解析の結果に基づいてソースコードを構造化するため、シンタックスの情報に基づく処理を実装しやすい。そのため、コード整形などの変形処理を行うツールは UNICOEN を利用することで容易に実装できる。このような**シンタックスに着目する処理ツールの開発を支援する基盤ソフトウェアは存在しておらず、この相違点こそが UNICOEN が革新的でソフトウェアである所以である。**

５．期待される効果

我々は、UNICOEN を利用して開発した 3 種類の処理ツールと既存の処理ツールについて、対応言語数と開発コストの観点から比較した結果、２章で述べた問題点を解決できたことを確認した。ソフトウェアメトリクス測定ツール UniMetrics と、Ruby 言語のみに対応した既存ツールの Saikuro [8] との比較結果を**エラー! 参照元が見つかりません。**に示す。また、ソフトウェア品質の可視化ツール CodeCity と UNICOEN のマッシュアップツールと、C#言語のみに対応した CodeCity 専用ツール [9] の比較結果を表 2 に示す。

表 1. ソフトウェアメトリクス(McCabe の複雑度)の測定プログラムのコード行数

	対応言語	測定部分の行数
既存のRubyツール	Ruby	340
UNICOEN利用ツール	C, C#, Java, VB, Ruby JavaScript, Python	**1**

表 2. CodeCity による可視化のための専用ツールのコード行数

ツール名	対応言語	測定部分のコード行数
既存の処理ツール	C#	2968
UNICOEN利用ツール	C, C#, Java, VB, Ruby JavaScript, Python	**370**

UNICOEN を利用することで、1 つのソフトウェアで複数の言語に対応して、かつ、それぞれに同じ機能を提供できた。このことは、***問題 2*** を解決する。また、複数の言語に対応するのにも関わらず、実装に必要なコード行数を減らすことができた。このことは、***問題 1*** を解決する。特に、**エラー! 参照元が見つかりません。**では 340 倍もの削減に成功した。

Java と JavaScript 言語で開発された JsUnit [10] について、UNICOEN を利用して開発したツールで得た測定値を CodeCity で可視化した結果が**エラー! 参照元が見つかりません。**である。CodeCity は、ソフトウェアメトリクスの測定結果を読み込み、その結果を街に見立てて可視化するツールである。街全体が 1 つのソフトウェア全体を、建物 1 つ 1 つがソフトウェアを構成するクラスを表す。**エラー! 参照元が見つかりません。**は、Java と JavaScript 言語で記述したクラスをそれぞれ青と緑に、建物の高さをステートメント数に、胴回りの太さをメソッド数に対応付けて可視化した。既存ツールでは、複数の言語を 1 つの街で表現できなかったが、**エラー! 参照元が見つかりません。**のように、UNICOEN を利用することで、言語によって建物の色付けを変更して、それらを 1 つの街として表示できるようになった。

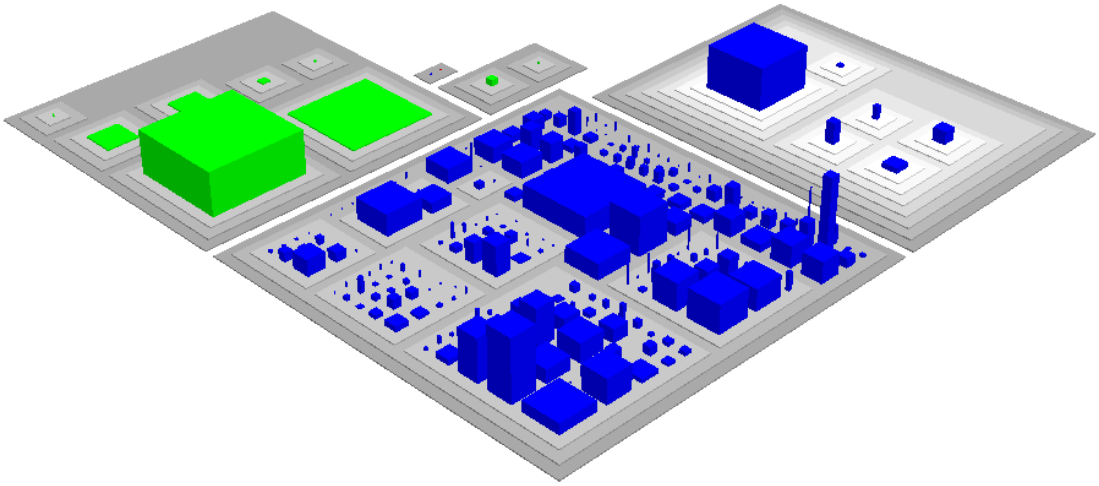

図 3. UNICOEN と CodeCity による Java と JavaScript 言語で開発された JsUnit の品質の可視化

図 3 では JavaScript 言語で構成される街の一部に、1 つだけ突出した建物が見られる。これは、クラスによってモジュール化せずに、1 つのファイルに詰め込んでしまうという、JavaScript 言語で記述されたプログラムの傾向を表している。このように、CodeCity を用いて可視化した結果に、言語による実装の特色が現れることが判明した。このことは、**UNICOEN がソフトウェア品質の可視化ツールの新たな可能性を開いたことを示唆する。**

以上から、UNICOEN は処理ツールの開発コストを低減して、かつ、複数の言語に対応したソフトウェアの開発を支援できる。例えば、処理ツールを 100 種類、言語を 25 種類考える。処理ツールを言語毎に開発して、1 つの処理ツールの開発に平均して 4000 行のコード行数が必要だとする。現状では、「言語の種類数」×「処理ツールの種類数」×「1 つの処理ツールあたりのコード行数」＝10,000,000 行ものコードが必要になる。その上、一般に処理ツールのコードは複雑であり、他のツールと比べて 1 行あたりに必要な工数が増大する傾向がある。一方、UNICOEN を利用して 1 つの処理ツールの開発に必要なコード行数を 200 行に削減したとすると、（「言語の種類数」＋「処理ツールの種類数」）×「1 つの処理ツールあたりのコード行数」＝25,000 行のコードだけで済む。これは実に 400 倍もの削減となる。

これによって、ソフトウェアエンジニアはどのような言語を使用しても、処理ツールの恩恵を受けられるようになる。このことは、我々の最終ビジョンである**「全てのソフトウェアエンジニアに笑顔を」**もたらす。

６．普及の見通し

UNICOEN はフレームワークであり、他の開発者に利用されて初めて価値が生まれる。我々は、処理ツール開発者および対応言語拡張者を呼び込むために、積極的に宣伝活動をする予定である。我々の目標値として、処理ツール開発者を 100 人、対応言語拡張者を 20 人、そして、UNICOEN そのものの開発者を 5 人呼びこむ予

定である。これらの開発者を呼び込むことで、2 年以内に UNICOEN を利用して開発した処理ツールが 100 種類、UNICOEN が対応する言語は 25 種類を超えるだろう。1 つの処理ツール毎に平均して 10,000 人の利用者が生まれると考えられる。すると、UNICOEN は処理ツールを介して間接的に 10,000,000 人もの利用者を生み出すことができる。以上から、UNICOEN は社会的に非常に大きな影響を与える。

７．クリエータ名（所属）

坂本　一憲　（早稲田大学）
太田　大地　（株式会社 Access）
大橋　昭　　（早稲田大学）
岩澤　宏希*（早稲田大学）*2011 年 3 月まで参加

関連 Web サイト

- UNICOEN 公式サイト
  http://www.unicoen.net/
- UNICOEN プロジェクトサイト（GitHub 内）
  https://github.com/UnicoenProject/UNICOEN

参考文献

[1] FindBugs TM - Find Bugs in Java Programs, http://findbugs.sourceforge.net/.
[2] JSLint,The JavaScript Code Quality Tool, http://www.jslint.com/.
[3] The AspectJ Project, http://www.eclipse.org/aspectj/.
[4] 大橋昭，久保淳人ほか: AOJS：JavaScript のためのアスペクト指向プログラミング・フレームワーク，コンピュータソフトウェア, Vol.28, No.3(2011), pp.114-131.
[5] CodeCity, http://www.inf.usi.ch/phd/wettel/codecity.html.
[6] GCC, the GNU Compiler Collection - GNU Project, http://gcc.gnu.org/.
[7] The LLVM Compiler Infrastructure Project, http://llvm.org/.
[8] Saikuro : A Cyclomatic Complexity Analyzer, http://saikuro.rubyforge.org/.
[9] Parsing and modelling C# systems, https://bitbucket.org/erikdoe/pmcs/.
[10] JsUnit, http://www.jsunit.net/.