組込み RTOS 向けアプリケーション開発支援ツール TLV（トレース ログ ヴィジュアライザー） フェーズ 5 リファクタリング仕様書

2010 年 4 月 27 日

改訂履歴

版番	日付	更新内容	更新者
1.0	10/4/23	新規作成	市原大輔

目次

1 はじめに

1.1 本書の目的

本書の目的は、文部科学省先導的 IT スペシャリスト育成推進プログラム「OJL による最先端技術適応能力を持つ IT 人材育成拠点の形成」プロジェクトにおける、OJL 科目ソフトウェア工学実践研究の研究テーマである「組込み RTOS 向けアプリケーション開発支援ツールの開発」に対して、その開発するソフトウェアに対する設計を記述することである。

本書は特に、フェーズ 5 におけるリファクタリング作業に関する記述を行う。

1.2 本書の適用範囲

本書は、組込み MPRTOS 向けアプリケーション開発支援ツールの開発プロジェクト（以下本プロジェクト）のフェーズ 5 におけるリファクタリング作業に関する記述を行う。

1.3 用語の定義/略語の説明

表 1　用語定義

用語・略語	定義・説明
TLV	Trace Log Visualizer
MPRTOS	マルチプロセッサ対応リアルタイムオペレーティングシステム
トレースログファイル	RTOS のトレースログ機能を用いて出力したトレースログや、シミュレータなどが出力するトレースログをファイルにしたもの
標準形式トレースログファイル	本ソフトウェアが扱うことの出来る形式をもつトレースログファイル。各種トレースログファイルは、この共通形式トレースログファイルに変換することにより本ソフトウェアで扱うことが出来るようになる。
変換ルール	トレースログファイルを標準形式トレースログファイルに変換する際に用いられるルール。
可視化ルール	標準形式トレースログファイルを可視化する際に用いられるルール。
TLV ファイル	本ソフトウェアが中間形式として用いるファイル。前述の標準形式トレースログファイルは、この TLV ファイルの一部である。
EBNF	Extended Backus?Naur Form の略。

1.4 概要

本書では、組込み MPRTOS 向けアプリケーション開発支援ツールのソフトウェアの仕様を記述する。本書は特に、フェーズ 5 におけるリファクタリング作業に関する記述を行う。

2 概要説明

2.1 リファクタリングを実施する理由

現状の TLV では、標準形式トレースログのパースにおいて、1 のような複雑な正規表現を計 8 回適用している。このような複雑な正規表現は、可読性が低く保守が難しいため、リファクタリングを実施した。

リファクタリング後の TLV は、標準形式トレースログのパースを専用のパーサに任せる。パーサは、TLV 変換ルール・可視化ルールマニュアル (rule-maual.pdf) の「2.1.2 標準形式トレースログの定義」にある EBNF のように記述することで、構文を直感的に理解できる。また、重複したコードおよび生成規則変更時の変更箇所も減らすことができるため、保守性も向上する。

Listing 1　リファクタリング前のパースコード例

```
m = Regex.Match(_log, @"^(\[[^\]]+\])?(?<objectType>[^\[\]\(\)\.]+)\([^\)]+\)(\.[^\s]+)?$");
        if (m.Success)
                ObjectType = m.Groups["objectType"].Value;
        HasObjectType = m.Success;
```

Listing 2　リファクタリング後のパースコード例

```
public ITraceLogParser OBject()
{
        Begin();

        var object_ =
                ObjectTypeName().Char('(').AttributeCondition().Char(')')
                .OR().
                ObjectName();

        object_.ObjectValue = Result();

        object_.HasObjectTypeValue = false;
        return (ITraceLogParser)object_.End();
}
```

2.2 リファクタリングを実施する対象

リファクタリング対象は、`TraceLog` クラス、特にそのコンストラクタである。

標準形式トレースログをパースするときにパーサを用いるように、`TraceLog` コンストラクタを変更する。

3　変更内容

3.1　クラス構成

TraceLog クラスがパースを委譲する標準形式トレースログ用のパーサを追加する。今回の変更に伴う影響範囲は、TraceLog クラス以外に存在しない。

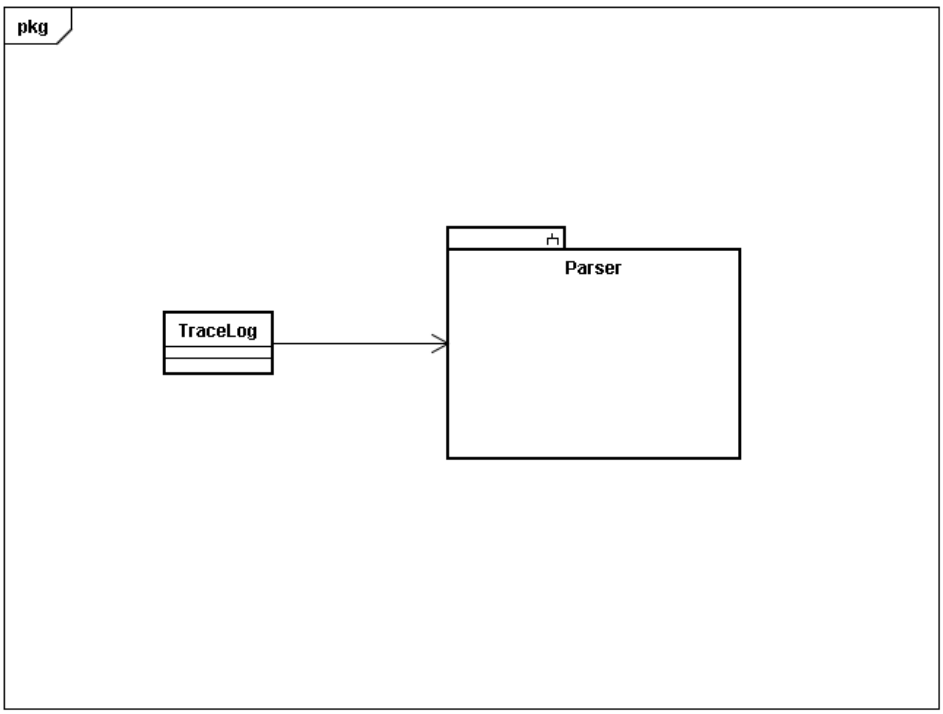

図 1　リファクタリング後のクラス構成

3.2　処理の流れ

リファクタリング前の処理の流れは、図 2 のように、TraceLog コンストラクタにて正規表現を用いたパースを行っている。

リファクタリング後の処理の流れは、図 3 のように、TraceLog コンストラクタでパーサクラスのメソッドを呼び、パーサクラスにて標準形式トレースログのパースを行う。パースが終了したら、パーサクラスから結果を受け取り、各値を設定する。

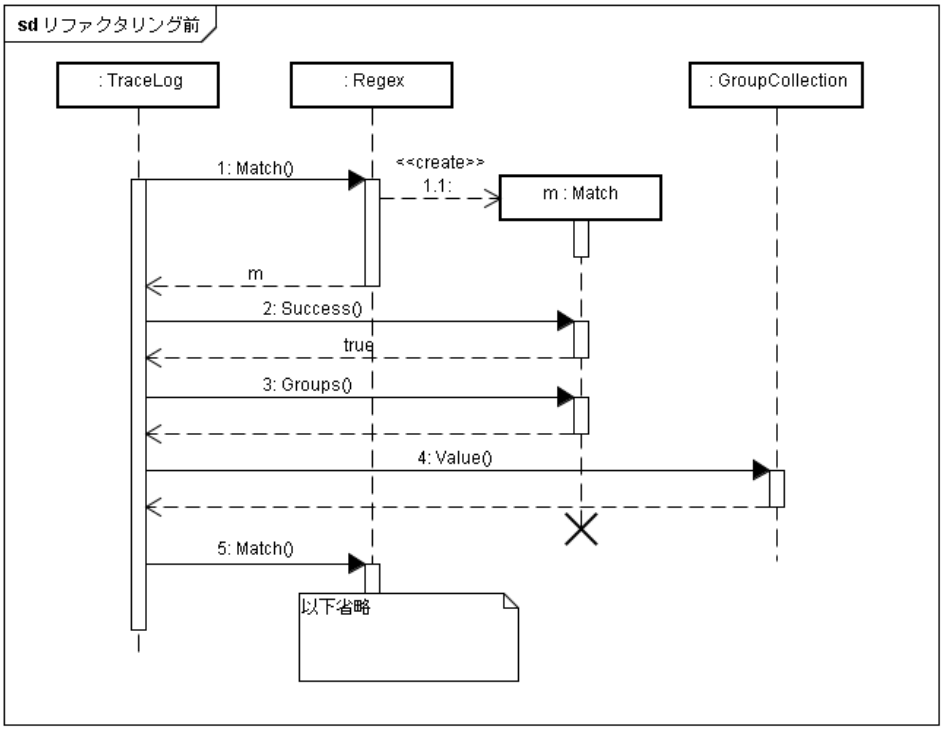

図２　リファクタリング前の標準ログ変換処理

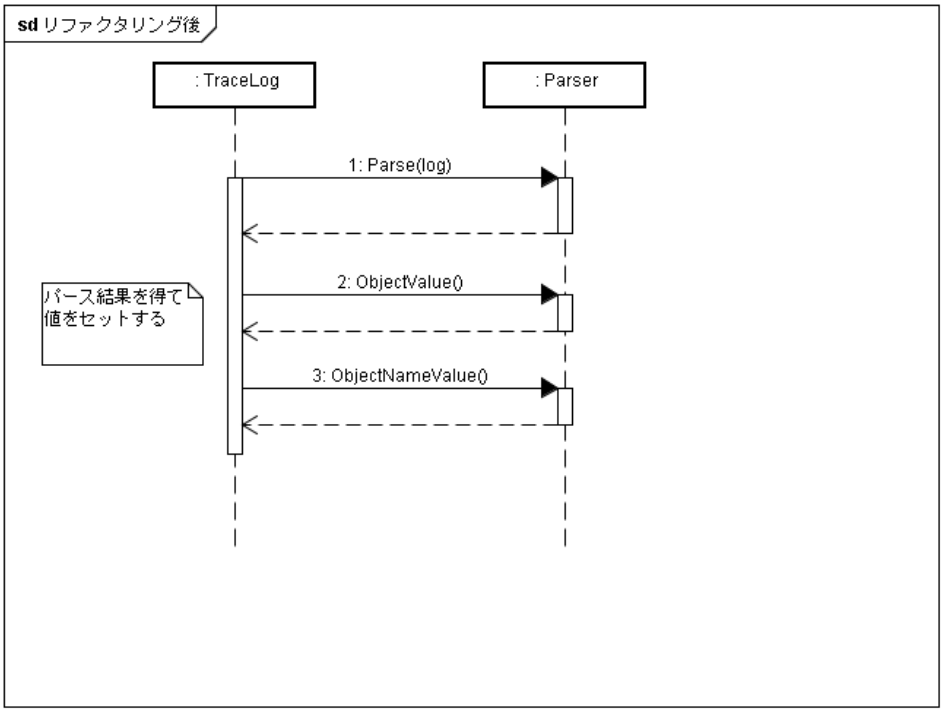

図３　リファクタリング後の標準ログ変換処理

3.3 パーサの構成

パーサは、コードで EBNF を表現し、直感的に理解できるようにする。例として、Haskell のパーサコンビネータライブラリ Parsec を参考にした次の URL にあるものが挙げられる。

LukeH's WebLog : Monadic Parser Combinators using C# 3.0
http://blogs.msdn.com/lukeh/archive/2007/08/19/monadic-parser-combinators-using-c-3-0.aspx