組込み RTOS 向けアプリケーションツール

TraceLogVisualizer（TLV）

フェーズ 2 プロジェクト計画書

目次

改訂履歴

版番	日付	更新内容	更新者
1.0.0	08/09/17	初版	後藤隼弐

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」として開発するソフトウェアの開発プロジェクト（以下、本プロジェクト）の計画を記述することである。本書では特にフェーズ1の評価結果と反省点、またフェーズ2の実施概要について述べる。

1.2. 本書の適用範囲

本書では、フェーズ1の評価結果、設計および実装における反省点について、また、フェーズ2の実施概要および目標について述べる。

1.3. 概要

本プロジェクトは3フェーズから構成されており、2008 年 5 月から 8 月にかけてフェーズ1を実施してきた。フェーズ1は TLV の UI 評価や要求抽出、設計手法の探索を目的として行われ、開発関係者4名および RTOS 開発者・利用者１１名により評価が行われた。2008 年 9 月から 11 月にかけて実施するフェーズ2では、フェーズ1での評価結果をもとに要求の再定義、設計方針の決定を行う。

フェーズ1で評価を行った結果、いくつかの不具合と新たな要求、UI性能の対するフィードバックが得られた。評価の結果得られた新たな要求は、フェーズ2で追加する要求とは独立に記述する。フェーズ2で追加する要求についてはフェーズ2要求仕様書を参照すること。フェーズ1の実施は、設計手法の探索も目的として含まれているため、フェーズ1での設計段階における反省点、および実装段階における反省点についてもまとめることとする。最後にフェーズ1の評価結果を踏まえたフェーズ2の実施目標を述べる。

2. フェーズ1の評価

2.1. 評価方法

フェーズ1の評価は、RTOS の開発者および利用者を対象に行った。評価人数は15名であり、評価期間は 1 週間である。評価は、評価に必要なファイルをパッケージ化したものを配布し自由に試してもらった後、UI 操作性能や足りない機能、欲しい機能、実装機能に対する改善案、不具合などを自由回答形式で収集した。なお、評価の前には使用方法の説明およびフェーズ1で提供している機能の説明を行った。

2.2. 評価結果

2.2.1. 不具合

評価の結果得られた不具合は以下の2つである。

- ログを読み込んでいる状態で新たにログを読み込もうとすると挙動がおかしくなる
- タスクの強制待ち状態に関する表示が意図したものと異なる

前者の不具合については設計の段階で考慮しなかった操作のため発生したものである。後者の不具合は特定条件化で起こる現象のため、テストパターンに抜けがあったことがわかった。

これら不具合の収集から、フェーズ1での設計の抜け、テストの抜けが明らかとなった。

2.2.2. UI 操作性能

UI 操作性能に対するフィードバックとしては以下のような意見が得られた。

- マウスの各ボタンへの拡大縮小移動などの対応の好みが人それぞれ異なる
- サブウィンドウの操作が思い通りにいかない
  - 異なるウィンドウに移動したつもりが移動せずにタブ切替え表示となってしまう
- 目盛りサイズや行サイズの指定が思い通りにいかない
  - 大雑把な指定は出来るが細かい指定を「+」「-」ボタンで調整するのは大変
  - 値を直接入力する形式の提案
- マーカー、テンポラリマーカー、カーソルマーカーなどの区別がよくわからない
- マーカー間の時間差表示が邪魔になり時刻表示が見えない
- マーカーが増えると表示の妨げとなる
- マーカーあるいはカーソルを移動する際にタスクの状態変化など、イベントに吸着すると便利
- ウィンドウの位置、サイズ、各サブウィンドウの配置などが記憶されず毎回指定するのが面倒
- 表示メニューにおいて、表示しているウィンドウと表示していないウィンドウの差異がなくわかりにくい
- 可視化ウィンドウの情報表示エリアにおける列の入れ替えが思い通りに行かない

事前の操作説明により基本操作については評価者全員が短時間で扱えるようになったため、操作の難易度は高くないといえる。しかし、マーカーの扱いについてはネガティブな意見が多く得られたため、フェーズ2において改善を行いたい。

得られた意見のうちいくつかは、実装コストや優先順位などで判断しフェーズ2の要求として追加した。

2.2.3. 機能に対する新たな要求

評価の結果、以下のような機能が求められた。プロジェクト開始時に設定した機能以外で求められた要求を示す。

- 表示エリアの特定箇所とテキストログとの間の双方向リンク
- 1クリックで最大拡大するボタン
- マーカーを挟んだマーカー間の時間差表示
- 可視化表示の凡例表示
- タスクの1つの状態の継続時間表示
- タスク内のシステムコール処理箇所の表示
- 統計情報の表示
  - 選択範囲内の各タスクの CPU 占有率など
- 動的情報の表示
  - ある時点でのリソースの状態表示
- 画面のキャプチャ
- 理解支援機能
  - 表示結果から推測されるタスク間の依存関係
  - ボトルネック箇所の表示
  - 起動タスクと被起動タスクの関連表示
  - 待ちタスクと待ち解除を指示するタスクの関連表示

これら得られた要求は実装コストや優先順位などで判断しフェーズ2の要求として追加した。

3. フェーズ1での設計・実装における反省

3.1. 設計

フェーズ1では TLV の GUI アーキテクチャとして PAC パターンを選択した。MVC パターンではなく PAC パターンを選択した理由は、実装言語の特徴や IDE の特徴などから適したアーキテクチャだと判断したからである。具体的には、Visual C#における GUI 開発クラスライブラリである Form クラスで MVC パターンを実装すると、V と C の結びつきが強くなってしまうということや、1つのフォームに対して Form クラスを継承したクラス1つが対応するため、IDE の支援どおりにプログラミングを行うと分割統治がうまくいかず、ソフトウェアの規模が大きくなるにつれ Form クラスを継承したクラスが何千行にも膨れ上がってしまうという問題がある。PAC パターンでは MVC パターンにおける V と C の役割を P が果たすため、結びつきの強さが問題にならない。またエージェント（1つの PAC のまとまり）同士の関係は C が管理し、下位層のエージェントが上位層のエージェントに対して依存性をもつというルールが存在するためエージェントを単位にシステムの分割管理が容易に行えるため開発が容易になる。

設計の反省点としては、PAC パターンの実現方法として、エージェント、P、A、C のそれぞれを個別のクラスとして設計してしまったことが挙げられる。これによりソースが肥大化してしまった。またエージェント間の通信コストも増えてしまった。エージェントを単位にクラスを設計し、P、A、C の役割をメソッドやイベント、デリゲートなどをもちいて実現する手法が有効ではないかと考えている。

3.2. 実装

実装段階での反省点としては、チーム開発がうまく進められなかったことである。

個々の開発者のリリース間隔が長く、マージ作業のコストが大きくなってしまった。また、ユニットテストの不在によりマージによる不具合の発生が把握できず、遅い段階での修正が頻発しデバッグに時間の多くを割いてしまった。

フェーズ2ではユニットテストを実施し、さらにリリース間隔を短く保ってマージコストを最小限に保つよう努力したい。

また、ソースコード内のコメントが少ないことも反省点として挙げられる。開発保守の継続のためにも、フェーズ2では積極的にコメントを残すようにしたい。

4. フェーズ2の実施概要

4.1. 概要

フェーズ2は、フェーズ１の成果を元に要求の再定義、設計手法の見直しを行い実施する。機能面では、共通形式への変換や可視化ルールのプラグイン化など、柔軟性の支柱となる機能を実装する。フェーズ2での要求仕様は別紙にて記述する。また、テストを実施しソフトウェアの信頼性を確保する。

4.2. 目標

フェーズ2の目標として以下を定めた。

- 共通形式変換、可視化ルールプラグイン化の実装
- アジャイルメソッドを基本としたチーム開発の実践
  - ユニットテストによるテストファーストプログラミング
  - （可能なら）ペアプログラミング
  - リファクタリング
- ソースコード内コメント率 5%～20%