1C-6

アプリケーション競合管理方式

奥山 玄† 才田 好則† 臼井 和敏†
†日本電気 (株) システムプラットフォーム研究所

1 はじめに

携帯端末の高機能化に伴い, 携帯端末上ではさまざまなアプリケーションが動作するようになった. 携帯電話を例にすると, 通話機能だけでなく, メーラやブラウザ, Java などのアプリケーションが動作している. これらのアプリケーションのなかには, サウンドや通信デバイスなどの端末内のリソースを使用するアプリケーションがある. 携帯電話はリソースが限られているため, あるリソースが使用されていた場合, 同じリソースを必要とするアプリケーションは起動できないといった状態に陥ることがある. また, 着信の際には電話アプリケーションを起動させるなど, 優先的に行う処理がある. この場合, 実行中のアプリケーションは中断するなど, ある決められた状態へと遷移する [1]. このように, アプリケーションの起動が阻害されたり, 他アプリケーションの起動によって状態が遷移する状況をアプリケーション競合という.

本論文では, 携帯電話のようにリソースが限られている携帯端末をターゲットとし, 上記のアプリケーション競合について現状の問題点の分析を行い, それを解決する方法について述べる.

2 携帯端末の競合処理

本章では, 携帯端末のアプリケーション競合に関して, 携帯電話を例として競合処理の概要および問題点について述べる.

2.1 PC と携帯電話の競合処理の違い

PC(パーソナルコンピュータ) は, リソースが枯渇した場合, その旨をユーザに通知する. この際, ユーザは, アプリケーションを終了するなどの処理を行う. このように, リソースの枯渇による PC の競合処理はユーザに任されているため, アプリケーション開発者は競合処理を考慮する必要がない. これに対し携帯電話は, 競合処理をすべて端末内のみで解決する. すなわち, リソースの枯渇を事前に防ぐため, アプリケーションの起動や状態遷移の要求があった際に, その要求の可否を判断する. このため, アプリケーション開発者は, 常に端末内のリソース残量を考慮して開発を行う必要がある.

2.2 携帯電話の競合処理

携帯電話の競合処理の例を図 1 に示す. これは, アプリケーション 1 が起動中にアプリケーション 2 が起動しようとする際の例である. この場合, 競合マネージャは, 競合判定部が競合テーブルを参照し, アプリケーション 2 に競合結果を返す. 競合テーブルとは, アプリケーション間の競合判定が記述されているテーブルであり, 起動や状態遷移の許可/不許可はこのテーブルを基に決定される [2].

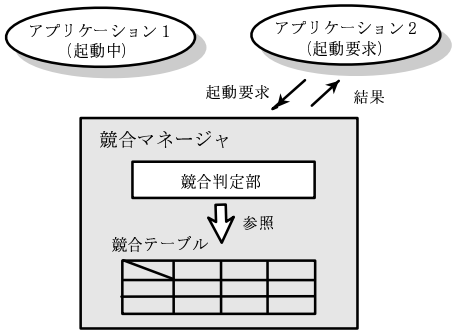

図 1: 携帯電話の競合処理の例

2.3 現状の問題点

現状の競合処理には, 以下の問題点がある.

1. 競合テーブルの巨大化・複雑化
2. アプリケーション追加時の開発工数の増大

1. は, アプリケーションの状態毎に競合判定を行うため, 競合テーブルはアプリケーション数が増加するとその状態分大きくなることを指す. すなわち, 携帯端末の高機能化が進むにつれ, 競合テーブルが巨大化することを指す.

2. は, 追加するアプリケーションに対する競合判定を既存の全アプリケーションに追加する必要があるため, 競合テーブルの作成, 検証の工数が増大することを指す. これにより, 不具合の発生率の増加が危惧される.

3 アプリケーション競合管理方式

本章では, 我々が提案するアプリケーション競合管理方式について述べる. 本方式は, 競合管理を簡易化することで以下を達成することを目的とする.

- 競合関連の不具合の発生率減少
- アプリケーションの新規/追加導入のしやすさの実現

Application Conflict Management System
† Gen Okuyama (g-okuyama@bk.jp.nec.com)
† Yoshinori Saida (y-saida@cp.jp.nec.com)
† Kazutoshi Usui (k-usui@cd.jp.nec.com)
System Platforms Research Laboratories, NEC Corporation (†)

3.1 概要

アプリケーション競合管理方式の概要図を図 2 に示す. 競合マネージャは, 競合データ生成部, 実行制御部およびデータベースを備えている. 競合データ生成部は, アプリケーションの競合に関する属性を持たせた記述ファイルを解析し, 競合に関する情報 (競合データ) をデータベースへと格納する. 実行制御部は, アプリケーション実行時にデータベースを参照することにより競合管理を行う.

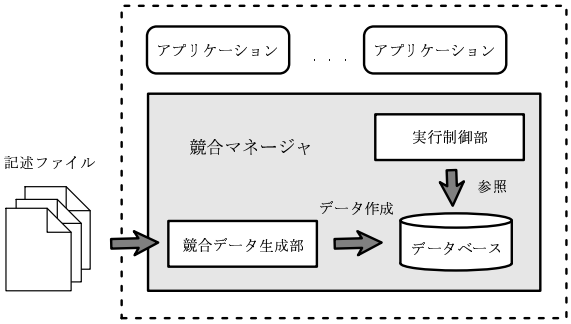

図 2: アプリケーション競合管理方式概要図

3.2 方法

本アプリケーション競合管理方式は, 以下の方法で競合管理を行う.

- リソースに着目した分類

  アプリケーションが使用するリソースに着目し, 競合管理を行う. リソースは, 携帯端末内に実在する実リソースおよび実在しない仮想リソース (後述) を含む.

- 仮想リソースの導入

  実リソースに基づかない競合処理を定義するために, 仮想リソースという概念を導入する. これにより, 例えば通話しているときはいかなる場合でも Java は起動不可とするというような, 実リソースの奪い合いではなくユーザの操作感を優先した競合動作を記述することが可能となる.

- 記述ファイルの提供

  アプリケーション毎に記述ファイルを用意し, そのファイル中にアプリケーションが要求するリソースを指定する.

- 競合データの生成

  記述ファイルを解析し, 競合データを生成する.

3.3 記述ファイル

前節で述べた記述ファイルには, 以下を指定する.

- 状態

  アプリケーションの状態名と優先度を, アプリケーションが取り得る状態分記述する.

- 起動条件

  アプリケーションが上記状態に遷移する際に必要なリソースと, そのリソースが確保できなかった場合の動作を指定する.

- 消費リソース

  アプリケーションが上記状態に遷移した後に必要とするリソースを指定する.

アプリケーション競合が発生した際, 起動中のアプリケーションの消費リソースと起動や状態遷移を要求しているアプリケーションの起動条件リソースとを比較する. リソースが一致した場合は起動不可となり, 一致しない場合は起動許可となる. 起動不可となった場合, 起動を要求したアプリケーションは, 起動条件に指定されている動作を行う. すなわち, 起動条件にアプリケーションの終了が指定されている場合は終了する.

起動条件に指定するリソースと消費リソースとを区別して指定可能とした理由は, アプリケーションの起動後及び状態遷移後は必ずしもそのリソースを占有しない, すなわち他のアプリケーションが対象リソースを利用するのを許可するというケースが存在するためである.

3.4 効果

本アプリケーション競合管理方式は, 前節までに述べたように, 記述ファイルから自動生成された競合データを基に競合管理を行う. 記述ファイルはアプリケーション毎に独立しているため, 他のアプリケーションを意識せずにアプリケーションを開発することが可能である. これにより, 携帯端末開発時の新規アプリケーション追加だけでなく, ダウンロードによる追加にも容易に対応できる競合処理を提供することが可能である.

4 まとめ

本論文では, 競合管理を簡易化するアプリケーション競合管理方式について述べた. 本方式は, アプリケーション毎に用意した記述ファイルを解析し, 競合管理に必要な競合データを自動生成する機能を提供する. これにより, 競合関連の不具合の発生率が減少し, また新規アプリケーション追加に対応した競合管理が可能となる.

参考文献

[1] NTT DoCoMo, Inc.:“ i モード [Q&A 集] ”, http://www.nttdocomo.co.jp/mc-user/i/java/qa_503i.html.

[2] 志多伯純:“ 携帯情報端末装置 ”, 特開 2003-177926(2003-06-27).