高信頼組込みソフトウェア構築技術の統合

細合 晋太郎[†1] 菅谷 みどり[†2] 鵜川 始陽[†3]
北陸先端科学技術大学院大学[†1] 早稲田大学[†2] 京都大学[†3]

1. はじめに

組込みソフトウェアは,様々な工業製品の心臓部に組込まれ,その機能や品質はこれらの製品の価値を決める重要な要素である.特に組込みソフトウェアにおいての不具合は,その多くが致命的な製品の不具合となる.このため高信頼性は組込みソフトウェアに対する最も重要かつ緊急な要求である.

近年,組込みソフトウェアは大規模・複雑化してきており,従来の開発方法が通用しなくなってきている.また大規模化に伴いOS,ミドルウェア,アプリケーションといった様々なレイヤにおいて,それぞれ高信頼性を保たなくてはならない.

本プロジェクトではこれらの問題に対し,最新のソフトウェア開発技法を組込みソフトウェア開発に投入することで解決を試みる.

現在までに,高信頼組込みソフトウェア構築の問題を(1)構築環境,(2)実行環境,(3)実行基盤の3つの視点から統合的に研究開発を行ってきた.

本稿ではそれぞれのプロジェクトの成果物を組合せることにより,さらに信頼性の高い組込みソフトウェア構築技術を提案する.

2. 高信頼組込みソフトウェア構築技法プロジェクトの概要

2.1. 組込み用オブジェクト指向分析設計技術(北陸先端科学技術大学院大学)

ソフトウェア開発上の問題の多くが分析・設計などの上流工程で作りこまれることはよく知られている.本プロジェクトでは,上流工程段階から正しさを確認・検証しながら組込みソフトウェアを構築できる方法論とそれを支える環境に関する研究開発を行ってきた.

設計をUMLで記述し,モデル検査や定理証明技術により上流工程での検証を行う.実装としてUMLよりモデル検査を行うことのできるUMLcheckerがある.

2.2. 組込み用実時間Java技術(京都大学)

通常のJava処理系の場合は,アプリケーションを一時的に中断しごみ集めの処理を行う.この方法では,中断時間の予測が難しくアプリケーションの実時間性を保証できない.そこで,スナップショットごみ集め[1]およびリターン・バリア[2,3]を用いて,ごみ集めの実時間化を行った.これらの技術を用いることで,ごみ集めを小さな単位に分割し,アプリケーションの実行中に少しずつ進めることができる.

本プロジェクトでは,実時間ごみ集めJava処理系であるKVM[4]に対し実装を行った.

2.3. 組込みシステム向け基盤ソフトウェア(早稲田大学)

本プロジェクトでは,オペレーティングシステムによるCPUの使用を制限する資源保護の仕組みを提案した.

資源保護を実現するためのアカウンティングシステム(CABI)[5]では,プロセスごとのCPUの実行時間を監視し,その使用率を制限する手段を提供する.アカウンティングシステムを用いる事で,CPU資源の占有の防止によるシステムの信頼性向上が得られる.又,プロセスの実行時間の細粒度化による実時間アプリケーションの応答性の向上[6],SMPシステム[7]への拡張の実装を行った.

3. 高信頼組込みソフトウェア構築技法プロジェクトの統合

3.1. 統合システム

現在まで3つ方向から推し進めてきたプロジェクトの統合を行った.これにより信頼性の高いソフトウェア構築することができる.次に統合システムの貢献例を示す.

3.2. 実時間性の保証

実時間アプリケーションを実行するためのシステムは,システム全体としての実時間性が保証されていなければならない.OS,ミドルウェア,アプリケーション,ハードウェア,ライブラリなど,システムの実時間性に関わるすべての要素において実時間性を保証する必要がある.

我々のシステムでは,組込みソフトウェアにおいて大きな割合を占める,OSとミドルウェアについて実時間性を保証する.

- OSレベル

  CABIにより,実時間アプリケーションのプロセスに一定のCPU資源が与えられる.

Integration of High Reliability Embedded Software Construction Technology.

[†1] Shintaro Hosoai [†2] Midori Sugaya [†3] Tomoharu Ugawa
[†1] Japan Advanced Institute of Science and Technology
[†2] Waseda University [†3] Kyoto University

・ ミドルウェア(VM)レベル
実時間ごみ集めにより,ごみ集めによる実時間性の阻害を回避することができる.

3.3. 安全性の保証

・アプリケーション
デッドロックなどの防止は OS や VM では回避することは難しい. UMLchecker で扱うことのできるモデル検査技術では,アプリケーションを全網羅探査することで,起こりうる障害を予め検出することが可能である.

・OS
事前にチェックしえないアプリケーションについては,安全性を確保することは難しい.
CABI を用いることにより未検証アプリケーションに対して,占有を防ぐ機能安全を実現することが可能である.

4. 実装

4.1. 動作環境

・OS : VMWare 2.0.2 上の Vine Linux 4.1
・Kernel : linux-kernel2.6.18.1 + CABI_x86_v1.2patch 適用したもの
・JavaVM : KVM 1.1 に実時間ごみ集めを実装したもの

4.2. 想定するアプリケーション

2 つのタスク(A,B)が 2 つの共有資源(Printer, Scanner)を取り合いながら動作する Java アプリケーションで,A は一定周期で動作するリアルタイム処理が必要なタスク,B は多くのメモリを扱うタスクを想定している.

4.3. 適用手順

まず開発環境である Eclipse 上でアプリケーションの UML モデルを作成し,そのモデルを元に UMLchecker でモデル検査を行った.

次にモデルより Java コードを作成し,KVM 上にて実行を行った. 今回は実時間ゴミ集めを適用した KVM と通常の KVM の 2 種類の JavaVM で動作を比べた.さらに未検証のアプリケーションによる CPU の占有が発生する状況を想定し,高負荷プログラムと同時に動作させた, またこのプログラムを CABI の制御下に置き効果を確認した.

4.4. 動作結果

以下に実時間性に関する計測を行った結果を示す. 表内の値は, タスク A 内で時刻をカウントし, 遅れた時間を表している.

	単一アプリケーション	未検証アプリによる占有	CABI による占有回避
KVM	371ms	4445ms	383ms
実時間 KVM	0ms	4244ms	3ms

表 1 : 実時間 KVM と CABI による実時間性への影響

5. まとめ

本稿では,これまで行ってきたプロジェクトを統合することにより,以下の二つの性質について信頼性を高めることができることを確認できた.

安全性の保証 : UMLchecker により,アプリケーションでデッドロックが発生しないか検査を行うことができた. また CABI により未検証のアプリケーションによる占有を防ぐことができた.

実時間性 : 実時間 KVM にて VM レベルの実時間性を, CABI にて OS レベルの実時間性を保証し, 組合せることによりシステムとしての実時間性を高めることができた.

今後の課題として,現在 VMWare 上での動作を行っているものを,実際の組込みボードに対して実装を行い,実機上にて動作の確認を行うことを計画している.

参考文献

[1] T. Yuasa: Real-time garbage collection on general-purpose machines, The Journal of Systems and Software, Vol.11, No.3, pp.181-198, 1990.

[2] 湯淺太一, 中川雄一郎, 小宮常康, 八杉昌宏: リターン・バリア, 情報処理学会論文誌, 41 巻 SIG9 (PRO8) 号, pp.1-13,2000.

[3] T. Yuasa, Y. Nakagawa, T. Komiya, M.Yasugi: Return Barrier, Proceedings International Lisp Conference, San Francisco,2001.

[4] Sun Microsystems, Inc.: The K Virtual Machine -- White Paper,http://java.sun.com/products/kvm/wp

[5] Midori Sugaya, Shuichi Oikawa, Tatsuo Nakajima; Design and Implementation of Accounting System for Embedded Information Appliances, EUC2005, December 2005. pp.110-120.

[6] Midori Sugaya and Yuki Kinebuchi, Shuichi Oikawa, Tatsuo Nakajima, VPE: VirtualPeriodic Execution for Embedded System, RTAS2007, pp 56-59.

[7] 菅谷みどり,湯浅陽一,中島達夫,SMP 環境における資源管理手法の提案, 組み込みシステム研究会, 情報処理学会, 2007, 12.研究報告, No.2007-EMB-6.

[8] Tomoji Kishi,Toshiaki Aoki,Shin Nakajima, Natsuko Noda and Takuya Katayama: Project Report: High Reliable Object-Oriented Embedded Software Design,The 2nd IEEE Workshop on Software Technology for Embedded and Ubiquitous Computing Systems (WSTFEUS'04, pp144-148, 2004.