組込み型ソフトウェア開発における生産性・信頼性向上のための設計方法論に関する考察

6R－9

久保田優二†　有馬康彦†　島村昭範†　藤本洋‡
†富士通北海道通信システム㈱　‡日本大学

1.はじめに

近年、組込み型ソフトウェア(以下、組込みソフトと略す)に対する要求は多様化しており、機能も複雑化している。こうした組込みソフトを、高品質にかつ短期間で開発するには、組込みソフトの特性に即した開発手法の確立と CASE ツールによる支援環境が重要である。

我々は、組込みソフトの設計作業における段階的詳細化過程を支援する設計支援方式 MRV(Multi-layered-Refinement by Views)[1]に基づいて、設計支援システム Designpartner を開発し適用を進めてきた。

本稿では、設計作業におけるバグ混入の防止方法として設計書の書式に対する提案と、検出可能なバグの種類について考察を述べる。

2.設計作業における問題点

ソフトウェアの設計作業は、タスク構成の定義とかタスク間イベント受け渡し関係の定義など、複数の設計観点毎に設計書を作成し、設計書間の矛盾を取り除く必要があると考える。矛盾のある設計情報に従ってプログラムを作成すると、明らかにバグを混入することになる。従って、設計書作成時により多くの矛盾を取り除くことが、生産性・信頼性向上のために必要である。

その為には、各設計書に対して記述されるべき設計要素と設計要素間の関連を明確にし、矛盾が検出可能な書式を規定することが大切である。

3.設計書の書式定義

組込みソフトは、特定の入力イベントに対して定められた機能を実行すると言う特性より、タスクの入出力イベント決定、入出力イベントに対する機能割付け及び機能を実現する関数の仕様定義が設計作業の中で重要である。本稿では、①タスク間シーケンス、②タスク関連図、③状態遷移表、④関数仕様の書式を定義し、効率的にバグを取り除くことが可能であることを示す。

なお、書式定義に当たっては、各設計書の作成目的を箇条書きで示す。また、設計書を構成する設計要素と各々の関連をOMT[2]のオブジェクトモデル記法を用いて示す。

(1)タスク間シーケンスの書式

目的	タスク間のイベント受渡し関係定義
	あるタスクの入出力イベント対応関係定義
	共通リソースのアクセスタイミング定義

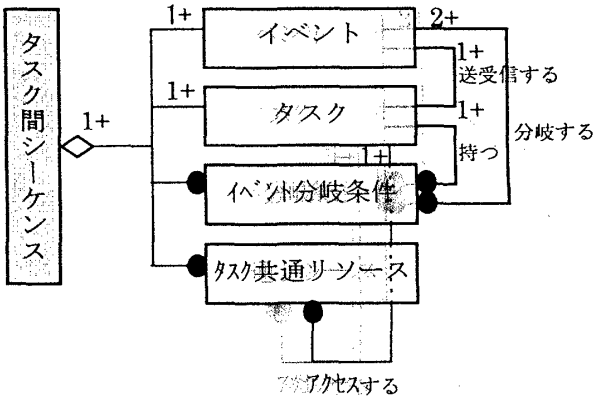

図 1：タスク間シーケンスの設計要素

(2)タスク関連図の書式

目的	タスク間インタフェースの通信手段定義
	共通リソースのアクセス有無定義

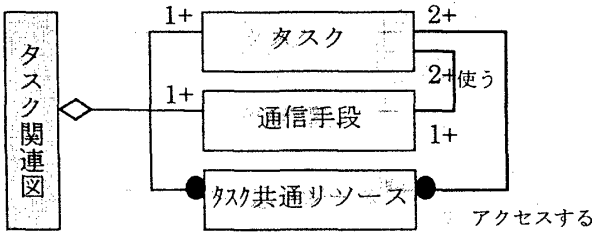

図 2：タスク関連図の設計要素

A Study of Design Method for Embedded Software with Productivity and Reliability

Yuji KUBOTA, Yasuhiko ARIMA, Akinori SHIMAMURA†,
Hiroshi FUJIMOTO‡.
†Fujitsu Hokkaido Communication Systems Ltd.
‡Nihon University

(3)状態遷移表の書式

目的	入力イベントとアクションの対応付け定義
	状態制御の定義
	タスクの入出力イベント対応関係定義

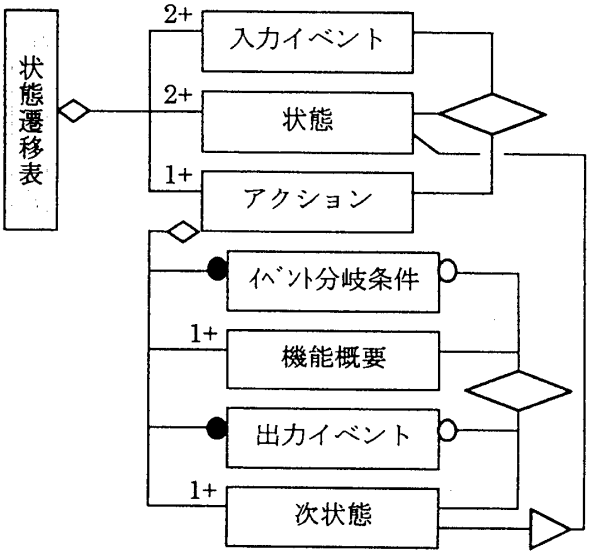

図 3：状態遷移表の設計要素

(4)関数仕様書の書式

目的	関数インタフェースの定義
	機能の詳細化
	共通リソースのアクセスタイミング定義

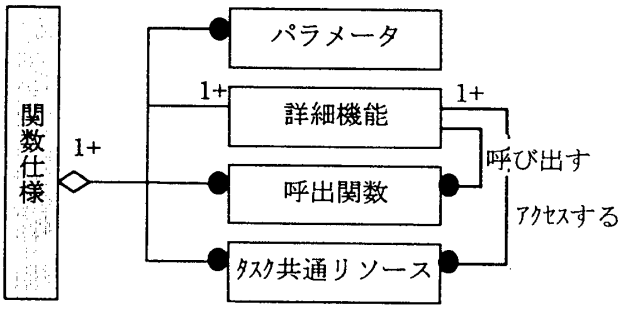

図 4：関数仕様の設計要素

各書式に対して、設計書間で共通な設計要素と設計書の関連をまとめると、図５のようになる。(多重度は省略)各々の設計観点で作成された設計書に対して、共通な設計要素の関連に対して矛盾を取り除く必要がある。

4.考察

本稿で提案する書式に基づいて、設計書を記述可能な CASE ツールを実現することによって、以下の不具合を防止することができる。

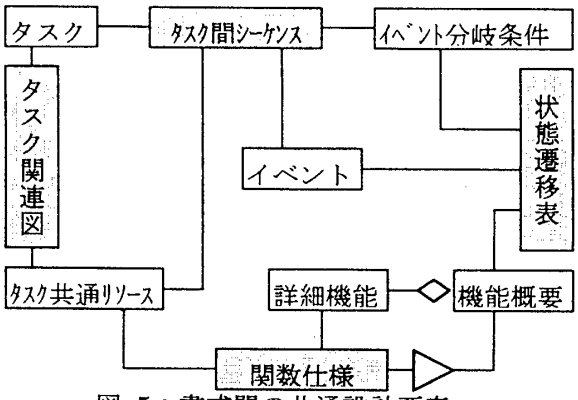

図 5：書式間の共通設計要素

①同一設計要素の設計書間不整合
複数の設計書間に存在する同一の設計要素(タスク、イベント等)に対して、一貫性を保証する。例えば、タスク間シーケンスに定義された入力イベントに対して状態遷移表の入力イベントと一貫性をチェックすることによって保証する。

②設計要素間の関連不整合
タスク間シーケンスにおけるタスク間のイベント送受信有無とタスク関連図におけるタスク間の通信手段有無に対して一貫性を保証する。

③設計要素の多重度矛盾
設計要素の多重度に対して矛盾を防止する。例えば、タスク関連図のタスク共通リソースに対する関連"アクセスする"が、１以下の場合を検出することにより、タスク共通リソースを設計するための条件が不確定であることを検出する。

④共通リソースのアクセスタイミング不整合
タスク共通リソースに対して、タスク間シーケンスで定義されているアクセスタイミングと関数仕様で定義されているアクセスタイミングの一貫性を保証する。

5.まとめ

本稿では、設計作業時のバグ混入を防止するための設計書に対する書式定義と、検出可能なバグの種類について考察を述べた。今後は、CASE ツールを実プロジェクトへ適用し効果を検証する予定である。

参考文献

[1] 深尾至、西山好雄、豊島康文、藤本洋："通信ソフトウェア向き設計支援環境"、情報処理学会論文誌、Vol.31 No.7 pp.1113-1122(1990年7月)

[2] J.Rumbaugh 他："ｵﾌﾞｼﾞｪｸﾄ指向方法論OMT"、ﾄｯﾊﾟﾝ、1992