南山大学 数理情報学部 情報通信学科 2007 年度卒業論文要旨集

車載ネットワークのフレームワーク設計方法の提案

2004MT008 江口 悠樹　2004MT064 水谷 拓人

指導教員 青山 幹雄

1. はじめに

ソフトウェア開発において，汎用機能を提供するものとしてフレームワークが利用されている．しかし，開発者がフレームワークを設計する際に，規模の増大による設計の複雑さが問題とされている．本研究では，この問題を解決するフレームワークの設計方法を提案する．

2. フレームワーク設計の問題と解決策

2.1. フレームワークとは

フレームワークとは，開発者がカスタマイズできるアプリケーションの骨組みである[1]．フレームワークの中でカスタマイズ可能な部分をホットスポットと呼ぶ．また，今後の変更が起こらないと思われる部分をフローズンスポットと呼ぶ．

2.2. フレームワーク設計の問題点と解決策

フレームワークを設計する際，フレームワーク規模の増大による複雑さが問題とされている．この問題を解決するために本研究では，フレームレットとデザインパターンを用いた設計方法を提案する．

3. 提案する設計方法

3.1. フレームレット分割

フレームレットとは，フレームワークの機能を複数のまとまりのある機能に分割したものである．フレームワーク規模の増大問題を解決するため，フレームワークを複数のフレームレットに分割して設計する．各フレームレットは互いに疎結合であるため，それぞれ設計解を導き，その後統合するプロセスをとることが可能である．

3.2. デザインパターンの適用

デザインパターンとは，ソフトウェア開発において繰り返し現れる設計課題とその解決方法を整理したものである[2]．例として，ステートパターンを説明する．ステートパターンとは，複数の状態を持つプログラムを作成する際，「状態」をクラスで表現するデザインパターンである．そして Context インタフェースを状態管理するクラスとして置いている．

4. CAN フレームワーク設計への適用

4.1. CAN とは

CAN(Controller Area Network)とは，車載ネットワークの標準通信プロトコルである[3]．CAN プロトコルは OSI 参照モデルのデータリンク層と物理層を規定している．通常，CAN のプロトコルは CAN コントローラというハードウェアで実装されている．

4.2. CAN フレームワークの設計プロセス

CAN フレームワークの設計に提案設計方法を適用する．図 1 に設計プロセスを示す．提案設計方法が反映されている部分は，「フレームワーク概念定義」フェーズでフレームレットに分割する部分．そして，「フレームレット概念定義」フェーズで各フレームレットのホットスポットを特定し，ステートパターンを適用してクラス図で表現する部分である．その後，フレームレットを統合する設計を「フレームレットアーキテクチャ設計」で考え，統合を行う．

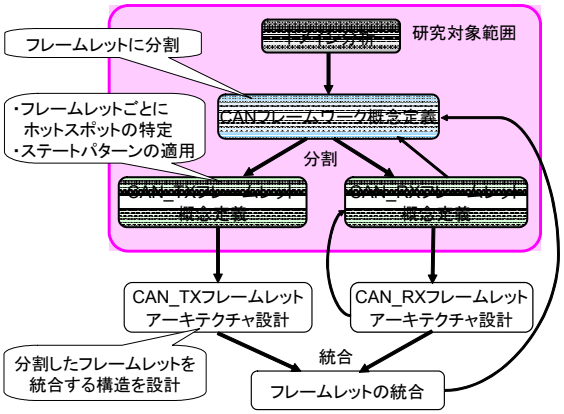

図 1　CAN フレームワークの設計プロセス

4.3. CAN フレームワーク設計における前提条件

図 2 は CAN コントローラの構成と送受信処理の動作を示す．CAN フレームワークの対象を CAN コントローラ中の CAN core とし，設計する上で次の前提条件を設定する．

(1) フレームワークの適用範囲を，CAN core と CAN バス間のフレーム送受信処理に限定
(2) 本研究で扱う設計問題は図 2 の CAN core に含まれる 7 つのユニットのうち CAN_RX，CAN_TX に限定
(3) 他 5 つのユニットから提供される同期機能やフレーム作成機能は利用できるものとする
(4) 設計プロセスの「ドメイン分析」から「フレームレット概念定義」までを対象とする

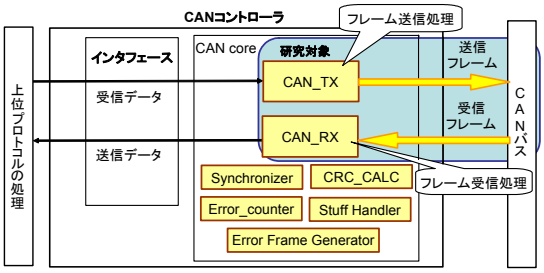

図 2 CAN **コントローラの構成と処理**

4.4. ドメイン分析

対象ドメインは, CAN バスに対して CAN core が行う処理である. 前節の前提条件(2)より, 設計対象はフレーム受信処理を行うCAN_RXとフレーム送信処理を行うCAN_TXである. また CAN_RX と CAN_TX は標準フレーム規格と拡張フレーム規格で, 処理の違いを考慮する必要がある. 標準, 拡張フレーム規格を考慮した4種類のフレームを図3に示し, 各フレームの説明を表 1 に示す. リモートフレームのフィールド構成はデータフィールドの有無以外, データフレームと同じである.

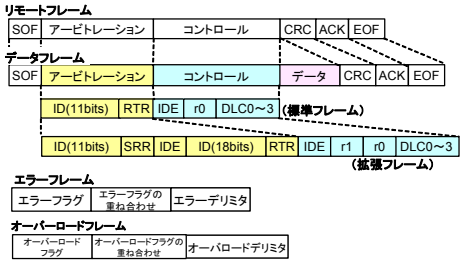

図 3 4 種類のフレームと各フィールド構成

表 1 フレーム名と説明

フレーム名	説明	対応
データフレーム	データ送信	TX
リモートフレーム	受信側が送信要求	TX
エラーフレーム	エラー通知	TX
オーバーロードフレーム	受信準備未完了を通知	RX

表1に示すように, CAN_RX, CAN_TX で扱うフレームは決まっている. CAN_RX が扱うフレームは, 受信未完了を伝えるオーバーロードフレーム. CAN_TX が扱うフレームは, フレーム送信するCANコントローラ側のCAN_TXが送信するデータフレームとエラーフレーム. そして, フレーム受信する CAN コントローラ側の CAN_TX が送信するリモートフレームである.

4.5. フレームワーク概念定義

4.5.1. フレームレット分割

CAN core に含まれる 7 つのユニット同士は独立している. 従って, 7 つのユニットをフレームレット分割の基準とする. 各フレーム名と機能を表 2 に示す.

表 2 分割したフレームレットとその機能

フレームレット名	機能
Synchronizer	CAN コントローラを同期
CRC_CALC	伝送誤りがないか判断
Error Frame Generator	エラーフレームを作成
Stuff Handler	周期的に再同期をさせる
Error_counter	エラーカウントし一定数以上でバスオフ
CAN_TX	フレーム送信処理
CAN_RX	フレーム受信処理

4.6. CAN_RX フレームレット概念定義

4.6.1. CAN_RX フレームレットのホットスポット特定

CAN コントローラがフレームを受信する状態遷移を図 4 に示す. CAN プロトコルの規格の違いによる CAN_RX, CAN_TX の処理を考慮している. 図 4 より, 同期してから CTRL でデータ長を測る処理は標準, 拡張フレーム規格で異なる. 従ってここをホットスポットとし, IDLE から EOF までを RX ホットスポット部分と呼ぶことにする.

RX ホットスポット部分に対し, オーバーロードフレームのフィールド構成は 1 つである. 従って, 処理は共通となるのでフローズンスポットとなる. Intermission から End of Intermission を RX フローズンスポット部分と呼ぶことにする.

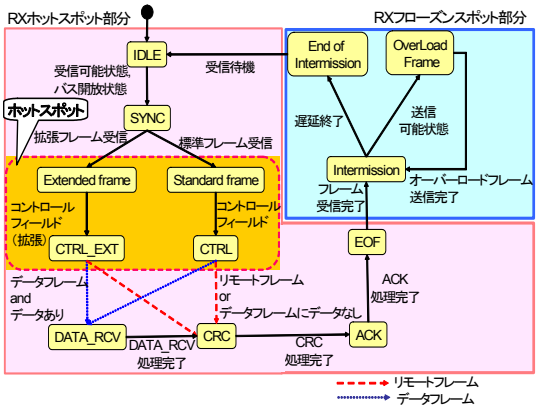

図 4 フレーム受信側の状態遷移図

4.6.2. CAN_RX 処理にステートパターン適用

RX ホットスポット部分は標準フレーム規格と拡張フレーム規格の処理を 2 つの「状態」に分けて処理するため, ステートパターンを適用する. そして, RX フローズンスポット部分はオーバーロードへの遷移があるかないかを 2 つの「状態」に分けて処理するためステートパターンを適用する.

RX ホットスポット部分と RX フローズンスポット部分にステートパターンを適用したクラス図を図 5 に示す.

1) RX ホットスポット部分の設計

状態管理をさせるために Rx_Context インタフェースを置き, 実装を Rx_Frame クラスで行う. doIDfilter メソッドで, フレームの ID から受信するフレーム規格を判断する. 判断結果から, changeState メソッドで各規格の処理へ遷移する.

状態を表す抽象クラスとして Rx_Frame_State インタフェースを置く. 状態を実装する具象クラスが Rx_Standard_State クラスと Rx_Extended_State クラスである. 具象クラス内の各メソッドが処理を実装する.

2) RX フローズンスポット部分の設計

Rx_Context インタフェースは状態管理を行うインタフェースを示し, Rx_Frame クラスで実装を行う. doIntermission は休止状態の管理を行うメソッドである.

Overload_State は, オーバーロードへの状態遷移を示している. doTrans_Overload は, オーバーロードフレームを送信処理するメソッドである.

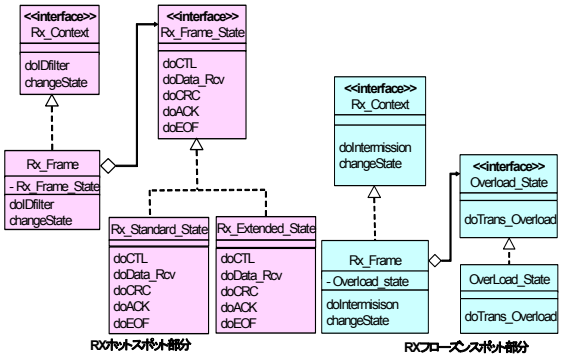

図 5 RX ホットスポット部分とフローズンスポット部分

4.6.3. CAN_RX フレームレット

図 5 の 2 つのクラス図は, 図 4 の状態遷移図からステートパターンを用いて別々のクラス図で表現している. 従って, CAN_RX フレームレットは図 5 のクラス図を組み合わせて表現できる. CAN_RX フレームレットのクラス図を図 6 に示す. 各クラス間の関係は次のようである.

状態管理を行う Rx_Context インタフェースの実装を Rx_Frame クラスが行う. ホットスポット部分を抽象クラス Rx_Frame_State と具象クラス Rx_Standard_State, Rx_Extended_State で表現する. フローズンスポット部分を Overload_State クラスで表現する.

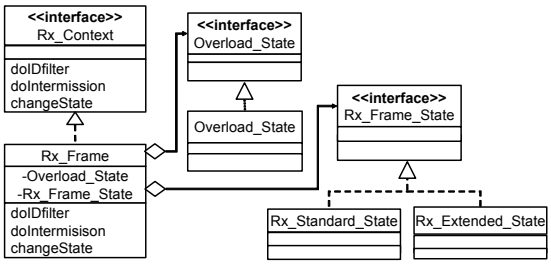

図 6 CAN_RX フレームレット

4.7. CAN_TX フレームレット概念定義

4.7.1. CAN_TX フレームレットのホットスポット特定

CAN コントローラが送信状態の状態遷移を図 7 に示す. SOF から TRANSMITTING へ状態遷移する際に, データ, リモートフレームの作成が行われる. 標準フレーム規格と拡張フレーム規格で作成されるフレームではビット数が異なるので, フレーム作成処理が異なる. 従って, ここをホットスポットとし, SOF から TRANSMITTING を TX ホットスポット部分と呼ぶことにする.

エラー通知を受けた場合, エラーフレームの作成を行い TX_ERROR からエラーフレームの送信を行う. エラーフレームのフィールド構成をフローズンスポットとし, TRANSMITTING から TX_ERROR を TX フローズンスポット部分と呼ぶことにする. なお, エラーフレーム作成を行うのは Error Frame Generator フレームレットである.

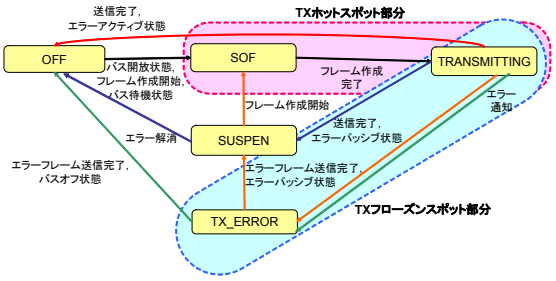

図 7 フレーム送信側の状態遷移図

4.7.2. CAN_TX 処理にステートパターン適用

CAN_RX と同様の理由から, TX ホットスポット部分と TX フローズンスポット部分にステートパターンを適用した. クラス図を図 8 に示す.

1) TX ホットスポット部分の設計

状態管理をする Tx_Context インタフェースを置き, 実装を Tx_Frame クラスで行う. doSOF は送信開始の処理を行うメソッドであり, doOFF はユニットを終了するためのメソッドである. また, doSUSPEN は送信の一時停止を行うメソッドである.

メッセージの送信処理をする Tx_Frame_State インタフェースを置いた. そして, 標準フレーム規格と拡張フレーム規

格の処理を 2 つの「状態」として処理するために，具象クラス Tx_Standard_State クラスと Tx_Extended_State クラスで実装した． CAN の送信でアービトレーションを行うため，doArbitration メソッドでアービトレーション機能を提供する．doTRANSMITTING メソッドでメッセージのフレーム化やフレーム送信を行う機能を提供する．

2) TX フローズンスポット部分の設計

E_Freme_State インタフェースはエラーフレームを送信する状態を示すクラスであり，具象クラス Error_State で実装する．doError_TRANS メソッドはエラーメッセージ送信を行う．

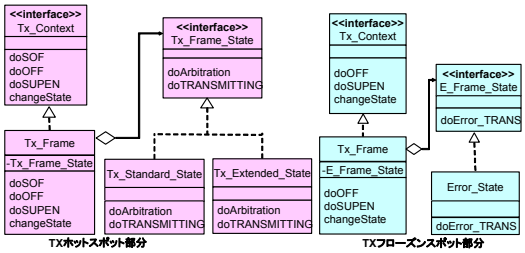

図 8 TX ホットスポット部分とフローズンスポット部分

4.7.3. CAN_TX フレームレット

CAN_RX フレームレットと同様，CAN_TX も状態管理に Tx_Context インタフェースを置いている．従って図 8 のクラスを組み合わせることが可能である．TX フレームレットのクラス図を図 9 に示す．各クラス間の関係は次のようである．

状態管理を行う Tx_Context インタフェースの実装を Tx_Frame クラスが行う．ホットスポット部分は Tx_Frame_State インタフェースを置き，Tx_Standard_State と Tx_Extended_State が実装する．フローズンスポット部分は，E_Frame_State インタフェースを Error_State が実装する．

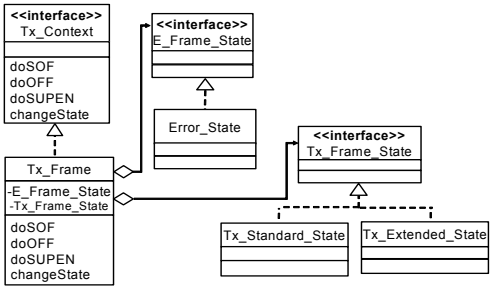

図 9 CAN_TX フレームレット

5. フレームレット評価

フレームレットの評価基準として，ホットスポットを変更してフレームレットが再利用できるか，またフローズンスポットが変更されない部分として適切かについて評価する．

(1) フレームレットのホットスポットの評価

CAN_RX，CAN_TX 共に標準フレームと拡張フレーム規格の状態遷移の違いをホットスポットとした．CAN_RX，CAN_TX 共に異なる規格をホットスポットにすることでホットスポット部分を変更が可能であり，適切といえる．また，標準と拡張フレームの規格に応じて実装でき，ホットスポット部分の実装を変更することで CAN_RX，CAN_TX のフレームレットを再利用可能である．

(2) フレームレットのフローズンスポットの評価

ホットスポットと別の状態のクラスで表すことでフローズンスポットの振る舞いがホットスポットより分離される．ホットスポットが変更されてもフローズンスポット部分の実装は影響されない．

6. 考察

(1) **フレームレット分割**

CAN フレームワークの設計にフレームレット分割を取り入れたことでまとまりのある複数の機能に分割することができた．

(2) **ホットスポット，フローズンスポットの特定**

ホットスポットを特定することによって変更，追加が可能な部分が特定できる．従って，柔軟なフレームレットが設計できる．また，フローズンスポットを特定することで今後変更されないと考えられる部分の特定ができる．従って，フレームレットの変更されない部分が明確にして設計できる．

7. 今後の課題

CAN_RX，CAN_TX フレームレットのクラス図から実装できるかどうか検証する必要がある．また，CAN_TX，CAN_RX フレームレット以外の 5 つのフレームレットを設計し，設計後に統合する必要がある．

8. まとめ

本研究では，フレームワークの規模増大による設計の複雑さを問題とした．解決方法として，フレームレットとデザインパターンを利用した設計方法を提案した．提案方法を用いて，CAN フレームワーク設計を行った．評価では，提案設計方法の妥当性を確認した．

参考文献

[1] R. Johnson ほか，パターンとフレームワーク，共立出版，1999.

[2] A. Pasetti, Software Frameworks and Embedded Control Systems, Springer, 2002 [佐藤 啓太，宇佐美 雅紀(訳)，フレームレット，翔泳社， 2005].

[3] 佐藤 道夫，車載ネットワーク・システム，CQ 出版社，2005.

[4] Q. Zhao, Integrated CAN Controller Design for the Imsys IM 31XX Microcontroller, http://web.it.kth.se/~srs/MSc_theses/Qiang_Zhao_MSc-thesis.pdf.