状態遷移表による設計手法(2):

なぜ状態遷移表を使うと、品質の良い開発ができるのか (1/2)

2012年05月23日 10時30分 公開

[塚田 雄一 キャッツ, @IT MONOist]

はじめに

組み込みソフトウェアが抱える一番の課題は「**設計品質の向上**」です。そして、この設計品質の向上にはモデルベース設計が有効であり、数あるモデルの中でも“**状態遷移系モデル**”が最も多く使われています。このあたりの詳細については、前回お伝えした通りです。

本連載の主役である「**状態遷移表**」は、“イベント”と“状態”を全て網羅的に表現できるため、設計の「モレ」「ヌケ」の発見・防止に大きな効果があり、設計品質の向上が期待できます。

第2回では「**なぜ状態遷移表を使うと、品質の良い開発ができるのか**」をテーマに、その詳細を説明していきます。

なお、本連載では以下の6つのテーマを順番にお届けしていきます。

1. **(前回):状態遷移表設計手法の概要**
2. **なぜ状態遷移表を使うと、品質の良い開発ができるのか**
3. **状態遷移表を使用した要求分析モデル**
4. **状態遷移表を使用した設計モデル(拡張階層化状態遷移表)**
5. **状態遷移表からの実装**
6. **状態遷移表を使用したテスト手法**

今回紹介する組み込みシステム環境について

リアルタイム性を強く要求される組み込みシステムの場合、しばしば「**リアルタイムOS**」が使用されます。

リアルタイムOSは、実行コンテキストを保持するオブジェクト(タスク、もしくはスレッド、以下“タスク”で統一)単位で動作します。そして、確実に実行要求を受け取るハンドラを起動して、実行要求を取りこぼすことがないようにスケジューリングされた動作を行います。そのため、高優先度のタスクを確実に実行しなければならないシステムには、リアルタイムOSがよく利用されています。

本稿では、“**リアルタイムOSで動作するタスクの振る舞い設計**”を題材に、今回のテーマを詳しく掘り下げていくことにします。

今回説明に使用するタスク構造

まず、今回説明に使用するタスク構造を以下に記します。

- **Aデバイスからデータを受信した際は、Aデバイスコントロールタスクへ要求イベントが渡され、メイン管理タスクへイベントが送られる**
- **Bデバイスからデータを受信した際は、Bデバイスコントロールタスクへ要求イベントが渡され、メイン管理タスクへイベントが送られる**
- **メイン管理タスクでは、それらの要求イベントを管理し、要求に応じてCデバイスコントロールタスク、もしくはDデバイスコントロールタスクへ送信要求イベントを送信する**
- **Cデバイスコントロールタスクは、メイン管理タスクから送信要求イベントを受け取ると、Cデバイスドライバへデータを送信する**
- **Dデバイスコントロールタスクも、メイン管理タスクから送信要求イベントを受け取ると、Dデバイスドライバへデータを送信する**

そして、**図1**は上記のタスク構造を図で示したものです。

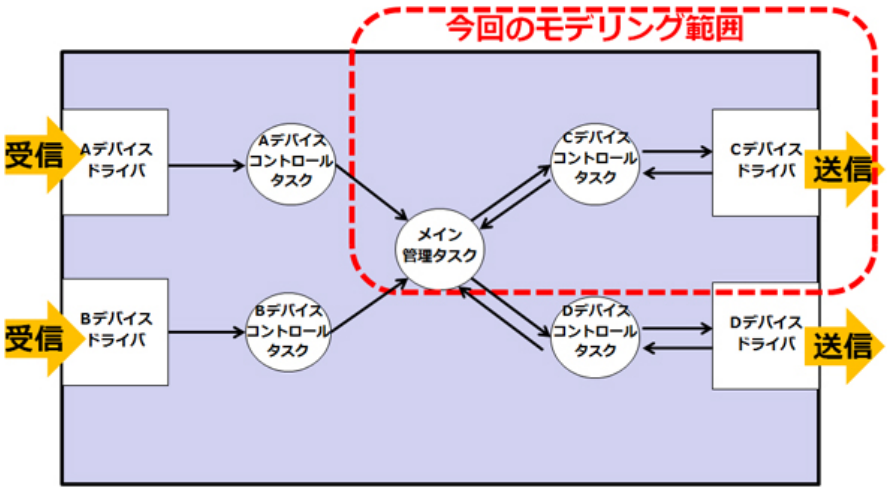

図1　タスク構造について

今回説明に使用するタスクの要求仕様書

図1の例では、「Aデバイスコントロールタスク」「Bデバイスコントロールタスク」「Cデバイスコントロールタスク」「Dデバイスコントロールタスク」と、4つのデバイスコントロールタスクが存在しています。

ここでは、それらの中から“Cデバイスコントロールタスクの要求仕様書”について考えてみましょう。

図2に示す要求仕様書では、Cデバイスドライバが正常に動作したケースと、正常に送信が完了しなかった異常ケースについて記しています。

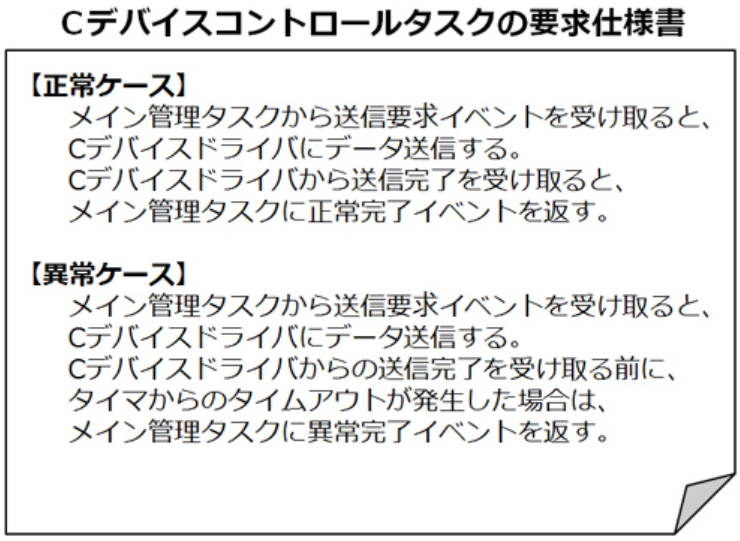

図2　Cデバイスコントロールタスクの要求仕様書

シーケンス図で表現してみよう

図2の要求仕様書はご覧の通り、日本語で記述されています。まずは、タスク間におけるイベントの関連をイメージしやすくするために、これを「**シーケンス図**」で表現してみます。

図3のシーケンス図は、「メイン管理タスク」「Cデバイスコントロールタスク」「Cデバイスドライバ」「タイマ」におけるイベントの関連を表現したものです。また、正常ケース(**図3左**)と異常ケース(**図3右**)の動作を分割して表しています。

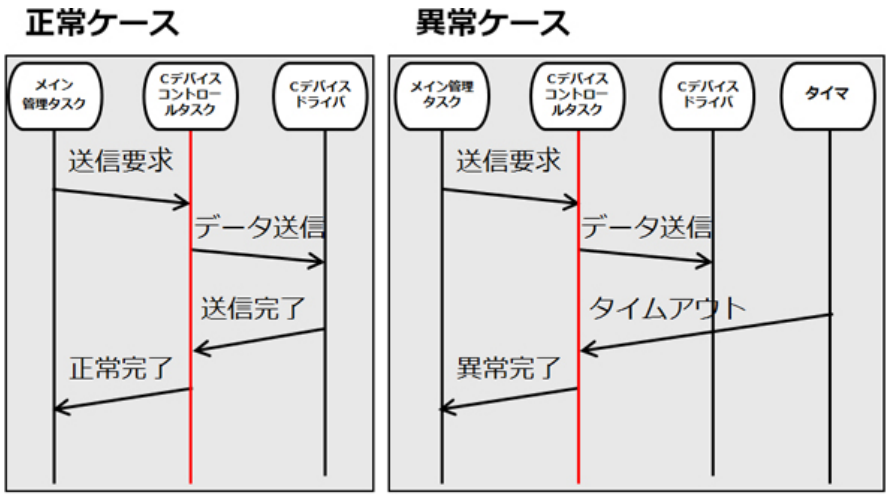

図3　図2の要求仕様書をシーケンス図で表現したもの

フローチャートで表現してみよう

次に、状態遷移表設計手法を使用せずに、シーケンス図からフローチャートを作成した場合を考えてみます。

シーケンス図のうち、Cデバイスコントロールタスクに向けて内側に矢印「→」が向いているイベントが、Cデバイスコントロールタスクへの“入力イベント”になります。該当するのは、「メイン管理タスクからの送信要求」「Cデバイスドライバからの送信完了」「タイマからのタイムアウト」の3つです(**図4**)。

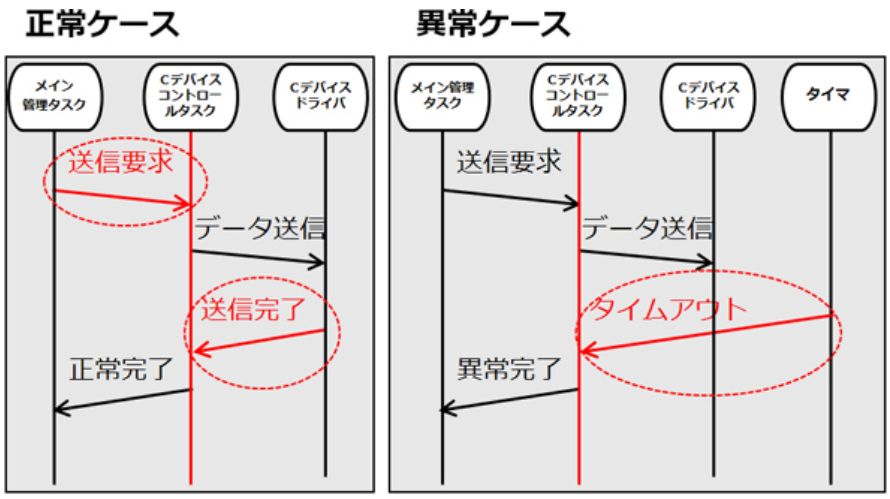

図4　Cデバイスコントロールタスクの入力イベント

また、Cデバイスコントロールタスクから外側に矢印「→」が向いているイベントが、Cデバイスコントロールタスクからの“出力イベント”になります。該当するのは、「Cデバイスドライバへのデータ送信」「メイン管理タスクへの送信完了」「メイン管理タスクへの異常完了」の3つです(**図5**)。

化、島津製作所と富士通が共同開発
ネクスティが販売パートナーになった QNX、マイコンレベルのプロセッサもカバー
日本初のAIプロダクト品質保証ガイドライン、QA4AIコンソーシアムが発行へ

よく読まれている編集記者コラム


どうして地下鉄のホームにLiDARがあるのか、東京メトロに聞いてきた


自分にとっての「ぴったり」とは? 個人のモノづくりの価値をもう一度問う


「LOVOT」のプロダクトデザインから学んだ“仕事の流儀”

» 編集後記一覧

人気記事ランキング - PR -

提供 AUTOMOTIVE JOBS


【動画で解説】2019年度の採用開始!今年の転職トレンドは?


＜高齢者事故対策＞ペダル踏み間違い防止や逆走防止など様々なアプローチが登場


45%が「勝手にブレーキをかけてくれる」と認識 自動ブレーキにまつわる誤解とリスク


【ホンダ】HEV/PHEV用で熱効率40%超を達成したエンジン戦略


男性がつけていたら恥ずかしい「図柄入りご当地ナンバー」ランキング

» 他の記事を見る

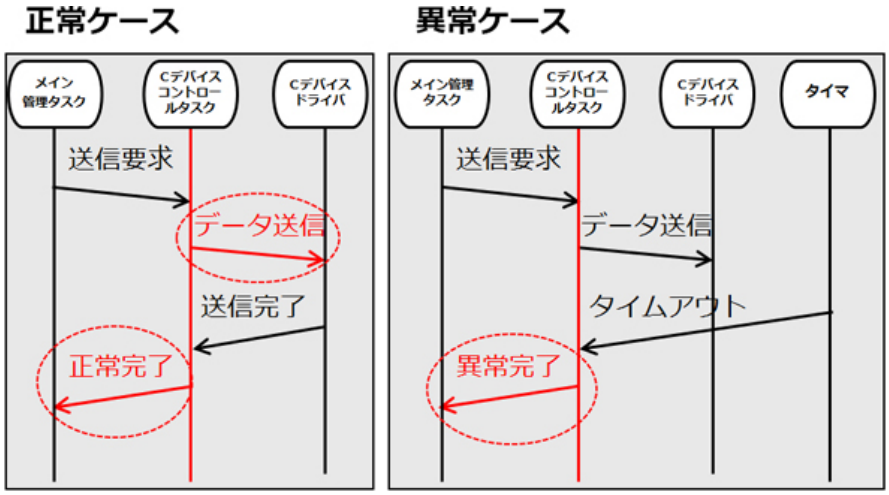

図5　Cデバイスコントロールタスクからの出力イベント

通常は、**図4**と**図5**の情報を参考にして実装を行いますが、今回はさらにフローチャートで表現してみたいと思います。

「メイン管理タスクからの送信要求」「Cデバイスドライバからの送信完了」「タイマからのタイムアウト」などの入力イベントと、「Cデバイスドライバへのデータ送信」「メイン管理タスクへの送信完了」「メイン管理タスクへの異常完了」などの出力イベントを記述していくと、**図6**のようなフローチャートになります。

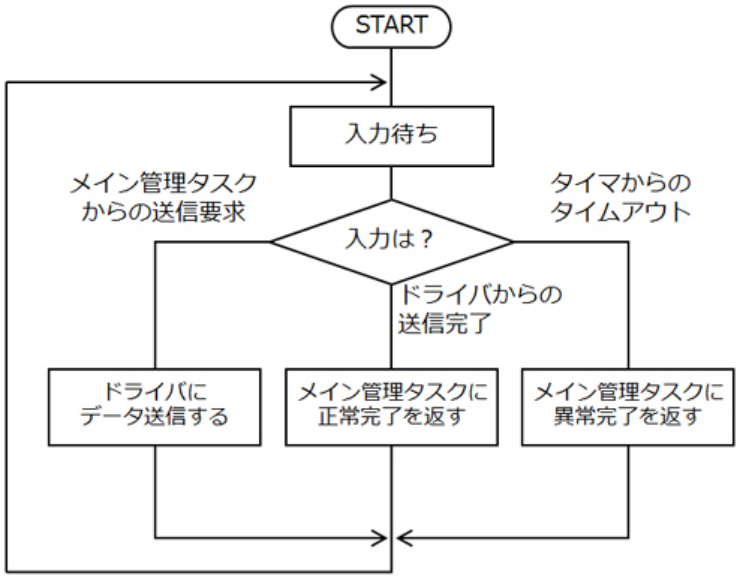

図6　フローチャートで表現

現場で発生する不具合の例

それでは、**図6**のフローチャートを基に“不具合の可能性”を考えてみましょう。

組み込みシステムの開発現場で実機試験を行っていると、「通常は、ドライバから正常にデータが送信されているが、何かのタイミングで正常にデータが送信されず、ハードウェアがハングアップしてしまう」などの不具合が発生することがよくあります。

実際に、こうした不具合の原因を調べてみると、「メイン管理タスクから送信要求イベントを受け、デバイスドライバに対してデータを送信している最中に、連続してメイン管理タスクから送信要求イベントが発生するといったケースの対応(処理)が設計から抜けてしまっていた……」などということがあり得ます(**図7**)。

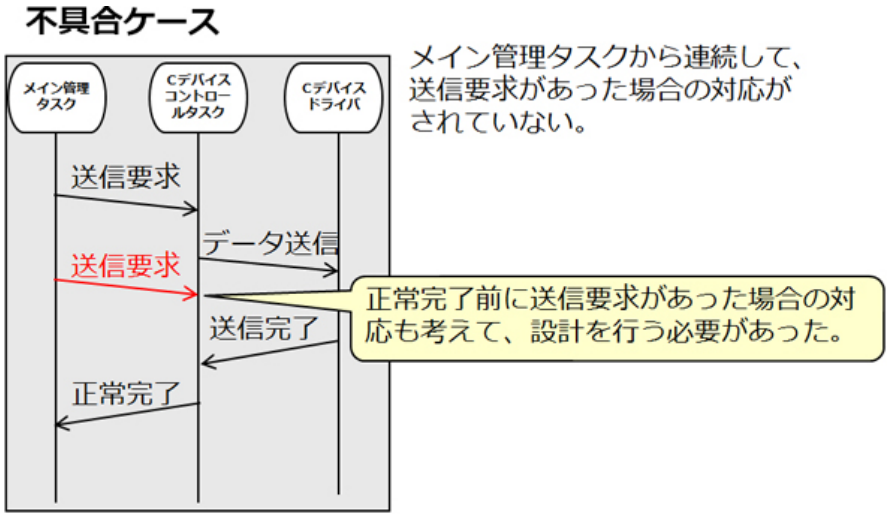

図7　不具合の可能性

ソースコードは、フラグの嵐になってしまう

先の不具合対応として、「送信中」と「送信中以外」を判断するための“フラグ”を使用します。具体的には、送信中にメイン管理タスクから送信要求があった場合に、「メイン管理タスクにビジーを返す」などの処理を追加するのです(**図8**)。

こうした対応を繰り返していくうちに、フラグが増えていき……。結果的にソースコードは**“フラグの嵐”**になってしまいます。

※補足:“フラグの嵐”とは、いろいろなフラグが存在しており、適切な箇所でフラグ操作を行うための管理ができない状況を意味する。

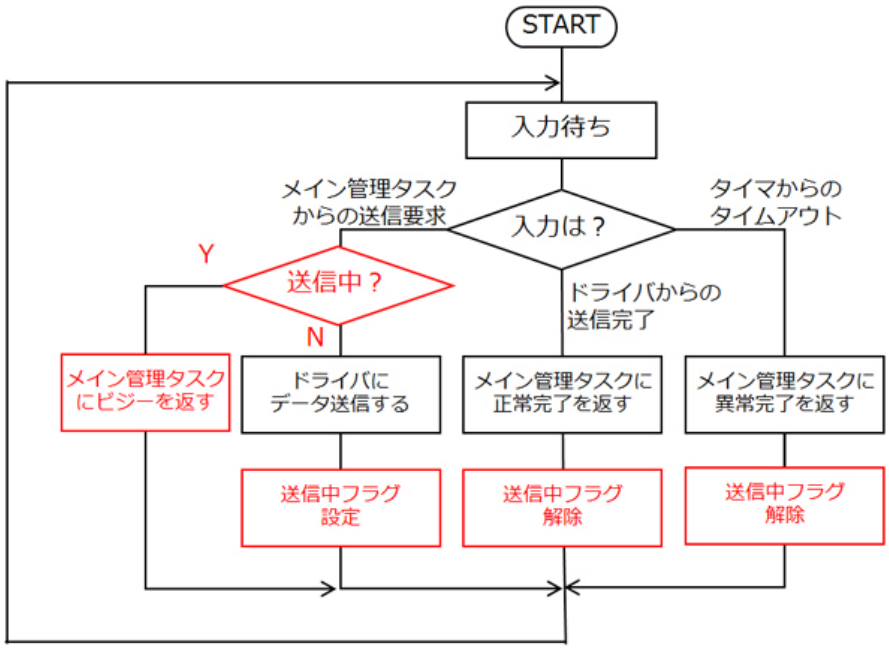

図8　ソースコードが“フラグの嵐”

関連キーワード

ソフトウェア | モデルベース開発 | 設計 | 組み込み | 組み込みソフトウェア | UML | 組み込みシステム | モデリング

→ 次ページ　**状態遷移表設計手法**

状態遷移表による設計手法(2):

なぜ状態遷移表を使うと、品質の良い開発ができるのか (2/2)

2012年05月23日 10時30分 公開

[塚田 雄一 キャッツ, @IT MONOist]

前のページへ 1 2

状態遷移表設計手法

このように、“不具合が発見されてから、それを修正する”ということを繰り返すのではなく、こうした不具合を設計段階で早期に発見し、対応できる手法はないのでしょうか――。

そうです。実はこうした要求に応えられるのが、状態遷移表設計手法なのです。

それでは、状態遷移表設計手法について解説していきます。状態遷移表設計手法を行うには、まず、適切なイベントと状態を抽出することが重要です。イベントの抽出については、先ほど説明をしていますので、以降では、状態を抽出する方法を説明していきます。

状態の抽出方法

シーケンス図を参照すると、Cデバイスコントロールタスクに対する入力イベントは、「メイン管理タスクからの送信要求」「Cデバイスドライバからの送信完了」「タイマからのタイムアウト」の3つであることが分かります。そして、そこには、メイン管理タスクからの送信要求が来る前の「送信要求待ち」と、送信要求を受け、Cデバイスドライバからの送信完了イベントを待っている「送信中」という2つの状態が存在しています(**図9**)。

このように、同じ“イベント待ち状態”であっても、メイン管理タスクから送信要求イベントを待っている「送信要求待ち」状態と、Cデバイスドライバから送信完了イベントを待っている「送信中」状態は、別の状態であることが分かります。

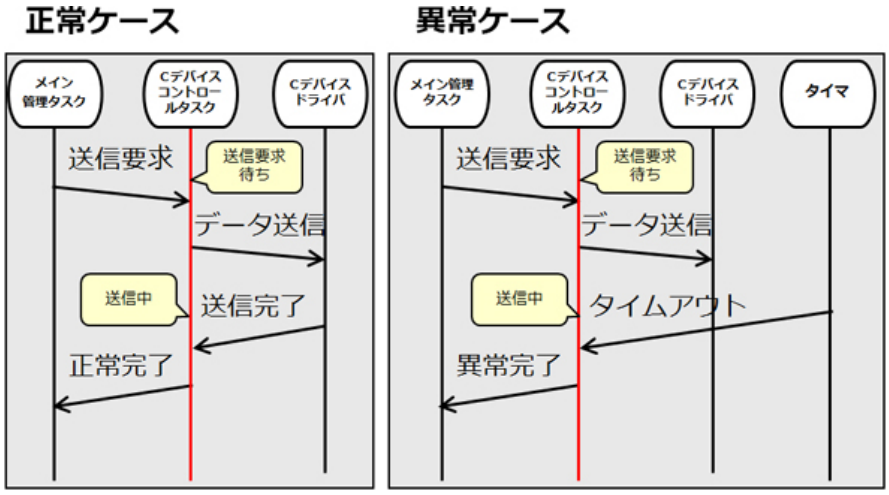

図9　Cデバイスコントロールタスクの状態について

状態遷移図で表現した場合

それでは状態の抽出ができましたので、「**状態遷移図**」を作成してみましょう。

まず、「送信要求待ち」と「送信中」の2つ状態を記します。そして、送信要求待ち状態で、メイン管理タスクからの送信要求イベントが発生した場合は、ドライバにデータを送信し、送信中状態に遷移します。送信中状態で、ドライバからの送信完了イベントが発生した場合は、メイン管理タスクに正常完了を返し、送信要求待ち状態に遷移します。また、送信中状態で、タイマからタイムアウトイベントが発生した場合は、メイン管理タスクに異常完了を返し、送信要求待ち状態に遷移します(**図10**)。

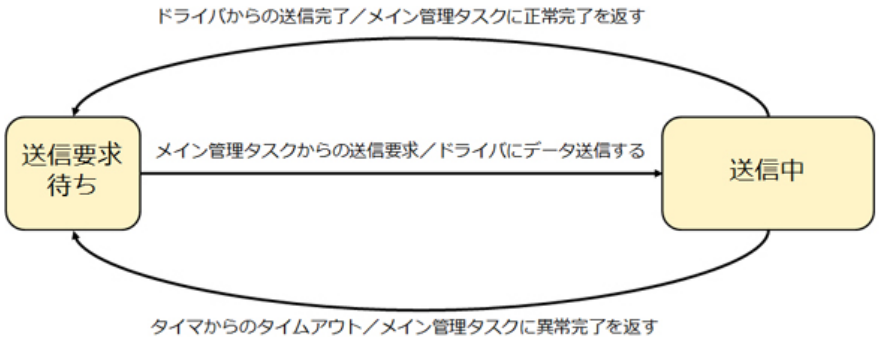

図10　状態遷移図

状態遷移表で表現した場合

次に、「**状態遷移表**」で表現してみましょう。

表上部の状態部分に「送信要求待ち」と「送信中」の2つを記述します。また、表左部にメイン管理タスクからの「送信要求イベント」、ドライバからの「送信完了イベント」、タイマからの「タイムアウトイベント」の3つを記述します。

そして、送信要求待ち状態で、メイン管理タスクからの送信要求イベントが発生した場合は、「ドライバにデータを送信し、送信中状態に遷移する」と記述します。送信中状態で、ドライバからの送信完了イベントが発生した場合は、「メイン管理タスクに正常完了を返し、送信要求待ち状態に遷移する」と記述します。また、送信中状態で、タイマからのタイムアウトイベントが発生した場合は、「メイン管理タスクに異常完了を返し、送信要求待ち状態に遷移する」と記述します(**図11**)。

状態遷移表は、状態とイベントの全ての組み合わせを表現します。これを踏まえ、**図11**をご覧ください。いかがでしょうか。送信中状態にメイン管理タスクからの送信要求イベントが発生した場

合、そして、送信要求待ち状態にドライバからの送信完了イベントが発生した場合、同じく送信要求待ち状態にタイマからのタイムアウトイベントが発生した場合の3つの処理を考えずに設計していることが明確になります。

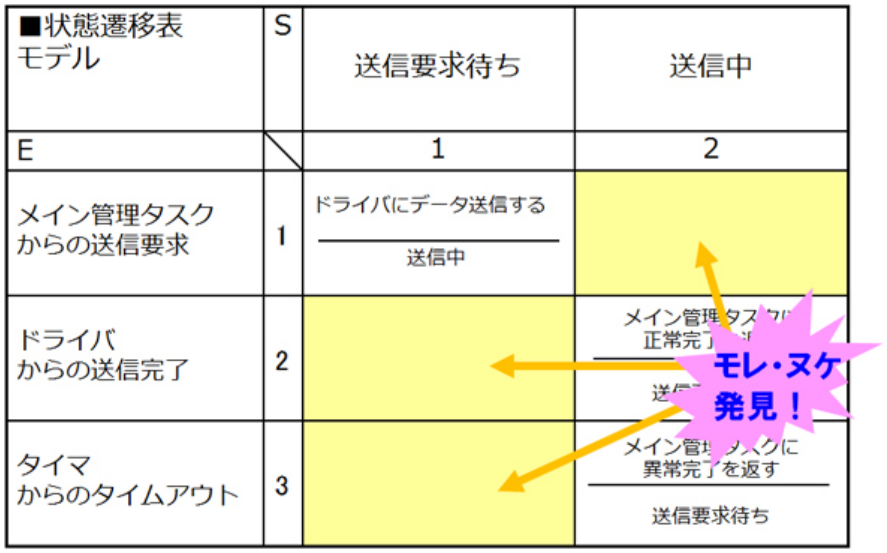

図11 状態遷移表

「モレ」「ヌケ」の発見による対応は、設計段階で行える

送信中状態にもかかわらず、再びメイン管理タスクから送信要求イベントが発生した場合は、「メイン管理タスクにビジーを返す」処理を追加できます。つまり、設計段階で連続して送信要求イベントが発生した際の対応を考えて設計することができるのです。

また、送信要求待ち状態で、ドライバからの送信完了が来た場合は「無視してよい」。そして、送信要求待ち状態で、タイマからのタイムアウトが発生するということはタイマ起動前ではあり得ないため「不可である」など、状態とイベントの全ての組み合わせを、「モレ」「ヌケ」なく設計できます(**図12**)。

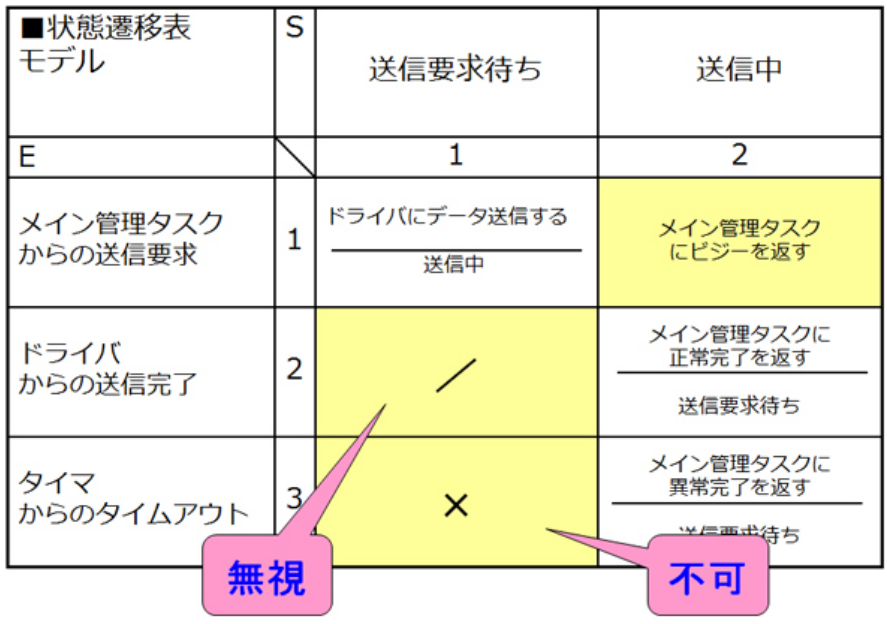

図12 「モレ」「ヌケ」の発見による対応は、設計段階で行える

開発コストにおける効果

化、島津製作所と富士通が共同開発
ネクスティが販売パートナーになった QNX、マイコンレベルのプロセッサもカバー
日本初のAIプロダクト品質保証ガイドライン、QA4AIコンソーシアムが発行へ

よく読まれている編集記者コラム

どうして地下鉄のホームにLiDARがあるのか、東京メトロに聞いてきた
自分にとっての「ぴったり」とは? 個人のモノづくりの価値をもう一度問う
「LOVOT」のプロダクトデザインから学んだ“仕事の流儀”

» 編集後記一覧

人気記事ランキング - PR -

提供 オートモーティブ・ジョブズ AUTOMOTIVE JOBS

【動画で解説】2019年度の採用開始!今年の転職トレンドは?
<高齢者事故対策>ペダル踏み間違い防止や逆走防止など様々なアプローチが登場
45%が「勝手にブレーキをかけてくれる」と認識 自動ブレーキにまつわる誤解とリスク
【ホンダ】HEV/PHEV用で熱効率40%超を達成したエンジン戦略
男性がつけていたら恥ずかしい「図柄入りご当地ナンバー」ランキング

» 他の記事を見る

最終的な実機試験において、“不具合が発見されてから、それを修正する”という行為を繰り返す手戻りは大きなコストとして表れてきます。

また、**図13**を参照して頂ければ明確ですが、修正後は再び、実装(プログラミング)、テストなどを繰り返し実施する必要があります。そのため、不具合が発生してから修正するのではなく、状態遷移表設計手法を使用して、不具合が発生しない設計を行うことが大切です。

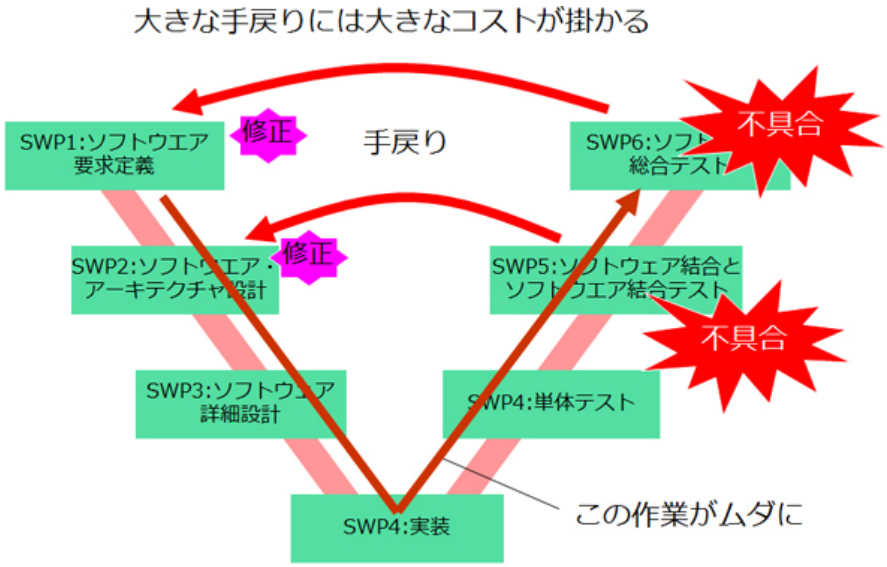

図13　不具合による手戻りは大きなコストとなる

今回のまとめ

きちんと設計を行わず、“不具合が発見されてから、それを修正する”という開発を繰り返す手戻りは大きなコストとなります。

状態遷移表は、イベントと状態の全ての組み合わせを考えて設計するため、処理を整理して、設計することができます。そして、設計時に「モレ」「ヌケ」を発見できるため、品質の良い開発が行えます。また、不具合発生による手戻りも削減できるため、開発効率も向上します。こうしたことから、組み込みソフトウェアの開発では、長年、状態遷移系モデルで設計が行われています。

さて、今回は“なぜ状態遷移表を使うと、品質の良い開発ができるのか”を紹介しました。次回は「**状態遷移表を使用した要求分析モデル**」をテーマに、実際に要求仕様から状態遷移表を作成するプロセスを見ていきたいと思います。お楽しみに!(次回に続く)

「状態遷移表による設計手法」バックナンバー

状態遷移表を使用したテスト手法【後編】
状態遷移表を使用したテスト手法【前編】
状態遷移表からの実装
状態遷移表を使用した設計モデル(拡張階層化状態遷移表)
状態遷移表を使用した要求分析モデル
なぜ状態遷移表を使うと、品質の良い開発ができるのか