組込みソフトウェア開発における 振る舞いとデータに基づくトップダウンなモデリング手法

岡田康治† 松浦佐江子†

組込みソフトウェア開発ではハードウェアの仕様がシステムの動作に大きく影響することから，ハードウェア側からのボトムアップ開発が多い．しかし，ボトムアップ開発ではソフトウェア設計が不自然になりやすく，ソフトウェアの再利用性が低下しがちである．本稿では，再利用性の向上を目的として，システムに要求される振る舞いのトップダウンな分析と，ハードウェア仕様の調査結果の設計情報としてのモデル化を組み合わせる開発方法を提案する．また，本手法を ET ロボコンへ適用した結果から，利点と一般の組み込みソフトウェア開発への適用可能性を議論する．

1. はじめに

組込みソフトウェア開発では，ハードウェアの仕様や誤差といった実際に動作させなければ確認出来ない特徴がシステムの動作に大きく影響する．実際の開発では，これらの特徴を把握するために開発初期に簡単なプログラムを作成して調査することが一般的である．しかし，多くの場合その調査結果はメモやドキュメント，計測データとして残されるだけであり，ソフトウェアの要求が変化した時にその調査結果からシステムへの影響やシステム設計の変更方針を決定することは難しく，ソフトウェアの再利用による効率的な開発が困難になる要因の一つである．また，ハードウェアからのボトムアップ開発ではシステム全体への要求を満たすために，ソフトウェアに不自然な設計が生じることがあるため,システムへの要求をトップダウンに分析，設計することが望まれている．しかし，システムの動作はハードウェアに起因する特性に依存するため，トップダウンな分析・設計のみではじめから再利用性の高い設計を行うことは困難である．

本稿では,図 1 に示すように,ユースケース分析により，システムに要求されている振る舞いをトップダウンに分析するとともに，部分的な実装によるハードウェアの振る舞いの調査結果をシステム設計の設計情報としてモデル化し，この設計情報を基に再利用性の向上を図るラウンドトリップな開発方法を提案する．また，ET ロボコン[1]への適用結果に基づき，一般の組み込みソフトウェア開発への適用可能性を議論する．

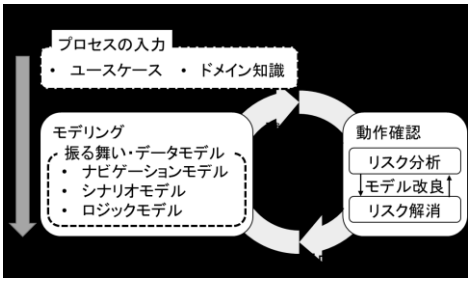

図 1 モデリングプロセス

†芝浦工業大学
† Shibaura Institute of Technology

2. 提案手法

2.1 モデリング

UML のアクティビティ図はシステムに要求されている振る舞いをアクション系列とその制御構造としてモデル化できる．そこで，本手法ではユースケースとドメイン知識を基に，システムへの要求をその想定できる振舞い系列としてアクティビティ図により，トップダウンに決定する．ユースケースはシステムが外部に提供する機能であり，それぞれの達成目標をもち，システムへの要求はユースケースの組み合わせで達成する．アクティビティ図はサブアクティビティという要素で構造化できるため，各ユースケースをサブアクティビティとしてその順序関係を定義することでシステムへの要求を定義したアクティビティ図を「ナビゲーションモデル」と呼ぶ．各ユースケースはその目標を達成するための具体的な振舞い系列となるため，これを定義したアクティビティ図を「シナリオモデル」と呼ぶ．さらに，シナリオを実現するための具体的な振舞い系列を「ロジックモデル」と呼び，サブアクティビティとして定義する．ロジックモデルは期待される振舞いを実現するアルゴリズムとデータを定義するものである．このデータは例えばハードウェアから取得するデータや制御理論に基づいて計算するデータ，ドメイン知識に基づく重さ，長さ，時間制限のようなドメイン固有の条件判定に用いられる定数がある．本手法ではこれらをオブジェクトノードで記述する．このデータは例に挙げたように振る舞いにおける意味が異なる．その為，その違いによって表 1 のように分類し，分類毎にデータを理解・実装する上で最低限必要な情報を設計情報として記述する．

表 1 データの分類と振る舞い上の意味

2.2 試験実装

モデリングで作成したモデルはその動作を確認するために

順次試験実装を行う．ただし，ロジックモデルを実装する場合は同時に動作する必要があるロジックや参照データの参照先のロジックモデルを作成している必要がある．

実装は，モデルの各アクションを関数に，各データを変数・定数に置き換え，さらにモデルのフローに従ってアクションの関数を呼び出す関数を定義する．このフローに対応する関数はモデルのサブアクティビティの構造に従って他モデルの関数を呼び出す．この置き換えによりモデルとソースコードが一対一に対応するため，関数の実装や変数・定数の値は対応するデータの設計情報を基に容易に定義できる．

2.3 動作確認

試験実装で作成したプログラムを実際に動かし，モデルの目標を達成しているか確認する．本手法では「モデルの目標を達成出来ない現象」をリスクと呼び，その原因推定と対策決定を行うリスク分析と，その対策をモデルとソースコードに反映して改良するリスク解消を繰り返す．

リスク分析では具体的な動作アルゴリズムを定義したロジックモデルからリスクの原因を調べる．初めに対策を施す必要があるリスクの発生源のモデルを特定する．特定にはモデリングで定義したロジックモデルの目標を利用する．実際に生じたリスクと各モデルの目標を比較し，現象が目標に反していればそのモデルをリスクの発生源と判断する．

リスクを起こすロジックモデルを特定したら，次にリスクの原因を分析する．ここでは，図 2 のようにデータに付加した設計情報を利用してデータの依存ツリーを作成し，出力に近いデータからログを取ってグラフ化してデータの値が想定外の異常な値を取っているか確認する．

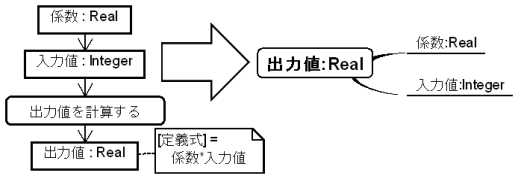

図 2 設計情報からのデータの依存ツリーの作成

異常なデータを見つけたら，モデルのフローや設計情報，依存ツリーを基に原因を推定し，実験データの具体値の変更や計算式の調整といった対策を決定する．

対策を決定したらリスク解消を行う．ここでは，対策に従って設計情報を含めてモデルを修正し，修正部分に対応したソースコードをモデルと一致するように変更して再び動作確認を行う．このロジックの動作確認を繰り返してもリスクを解消出来ない場合は上位のシナリオやナビゲーションモデルへと分析の範囲を広げ，新たなロジックやユースケースの追加といった対策も検討する．

動作確認を繰り返し，最終的にリスクが発生しなくなったらモデリングを再開する．新たなシナリオモデルを作成する際に，これまでに想定した振る舞いと同じ振る舞いが必要であれば，その振る舞いを実現するロジックモデルを再利用する．

3. 適用結果と考察

本モデリング手法を ET ロボコン大会用ソフトウェア開発に適用した結果，モデルの試験実装に用いたプログラムをそのまま本番で利用し，完走に成功した．また．開発初期で作成したライントレースや姿勢維持のロジックモデルをその後作成したシナリオでもそのまま利用することが出来たため，ロジック単位での再利用は本手法で可能である．

しかし一般への適用を考える場合，ET ロボコンでは規約により固定されていたハードウェアが変更されたり，または未定の場合もあり得る．ハードウェアを変更する場合はロジックモデルのハードウェアパーティションの内容を変更後のハードウェアに必要な性能の制約として，未定の場合はモデル上で先にハードウェアから得られる生データやそれを加工して生成するデータを定義し，それを採用するハードウェアの要求仕様として利用出来ると考えられる．

4. 関連研究

田村らは Simulink で設計した制御モデルをソフトウェア設計へ盛り込むために，制御モデル中の制御上重要な意味を持つデータを UML クラス図へ変換する手法を提案している[2]．制御モデルを直接ソフトウェア設計に持ち込むという点では有効であるが，制御を再利用する上では，その制御がどのような要求の下で必要なのかが明示されていないと議論が出来ない．その為，本手法のようにトップダウンに振る舞いの要求を定義する必要がある．

5. まとめと今後の課題

本稿ではシステムに要求されている振る舞いをトップダウンにモデル化しつつ，部分的に実装して調査したハードウェアの振る舞いをシステムの設計情報としてモデル化することで再利用性の向上を図るモデリング手法を提案し，ET ロボコンへの適用によってロジック単位での再利用が可能であることを確認した．

一般への適用可能性に関しては，モデル中のハードウェアに関する記述をハードウェアに対する制約として分析する手法を導入することで適用可能と考えられる．

また，再利用を更に効率化するために，新たなシナリオ下でロジックを再利用する際に確認すべき点の項目化を目指す．ロジックを再利用する際にはリスクの再発に注意する必要があるため，リスクの種類やその原因となるドメインの概念，対策を形式化することで，項目として利用できると考えられる．

参考文献

1) ET ロボコン: http://www.etrobo.jp，(2013/08/08 時点)
2) 田村雅成，神山達哉，添田隆弘，兪明連，横山孝典: Simulink モデルと UML モデルを用いた組み込み制御ソフトウェア開発のためのモデル変換環境，情報処理学会論文誌 Vol.53 No.12 pp.2660-2670(2012)