機能モデルを用いたソフトウェア変更支援

1R-7

山田宏之† 手塚慶一††
†愛媛大学 ††大阪大学

1．はじめに

ソフトウェアの一部を変更すると，その変更が他の部分に影響（相互作用）を与えるために，変更は容易でない．我々は，変更を支援するために，相互作用に関する情報を陽に記述し，利用することにより，相互作用を指摘できるシステムについて考察している[1]．本システムでは，相互作用がモジュール間の機能的な関係により発生すると考え，相互作用に関する情報として，モジュール間の機能的な関係を陽に表現することができる枠組み（モデル表現）を用いる．本稿では，本モデルを用いた変更支援について述べる．

2． 機能モデル

オブジェクト指向プログラミングスタイルで記述されたソフトウェアでは，複数のオブジェクトが互いに関連することにより，所望の機能が実現される．したがって，ある機能が変更されると，それに関連したオブジェクトに相互作用が発生する可能性がある．そこで，相互作用に関する情報として，オブジェクト間の機能的な関係を抽象化し，ソフトウェアをモデル化したもの（機能モデル）を用いる．本機能モデルは，オブジェクトを表すノード，オブジェクト間で実現される機能を表すアークから構成されるネットワークで表現する．また，機能的な関係に重点をおいてソフトウェアを抽象化するために，機能モデル内のノードとソフトウェア内で実現されるクラスとが必ずしも１対１対応ではない．

特に，与えられた問題を解くための基本的機能を実現しているオブジェクト（基本オブジェクト）から構成される機能モデルを基本機能モデルと呼ぶ．また，問題に対応した機能は基本的機能を実現している基本オブジェクトの下位オブジェクトとして定義される．したがって，基本機能モデルの各ノードは下位ノードをもち，機能的関係をもつ下位ノード間に機能リンクが張られ，個々のソフトウェアに対応した機能モデルが形成される．

さらに，基本機能モデルは，相互作用の有無を検出するだけでなく，変更作業の抜けが検出できる．たとえば，基本機能モデルで記述されている機能リンクが下位ノード間にない場合には，変更作業が完全になされていない可能性があり，変更作業の抜けが検出される．

3．フェリーシミュレーションの機能モデル

ここでは，文献［２］に現れるフェリーシミュレーションを例に挙げる．これは，本島と島との間をフェリーが往復して，一方から他方へ車を運ぶシミュレーションである．このシミュレーションソフトウェアの機能モデルを図１に示す．ただし，この図は４．２で述べる変更がなされている．

この図において，四角形は基本オブジェクト，楕円はシミュレーションモデルに応じて定義されるオブジェクトを表現するノードである．実線はオブジェクト間の機能的関係，点線はノード間の上位・下位関係を表現する．

たとえば，ノードSimulationObjectとノードSimulationとの間には到着定義リンクがある．このリンクは，ノードSimulationObjectあるいはその下位ノードに対応するオブジェクトの到着定義がノードSimulationあるいはその下位ノードでなされることを示している．したがって，このモデルよりクラスSimulationObjectの下位クラスが定義されたときに，クラスSimulationの下位クラスを修正する必要があることが分かる．具体的な，修正法に関する情報は，各ノードに蓄えられており，必要に応じて参照される．

4． 機能モデルを用いた変更支援

4．1 相互作用の処理

ソフトウェアが変更されるとき，まず，変更の対象ノードからモデル内に記述された上位・下位関係にしたがって，変更に関連する情報が検索される．もし，情報があれば，ユーザからの要求に応じてそれをユーザに示し，変更作業の手助けをする．変更が実施された後には，その変更により，実現あるいは変更された機能に関連する機能リンクをたどり，相互作用が予想されるオブジェクトを探索し，相互作用の有無を調べる．

相互作用が予想されるオブジェクトを変更する場合にも，その上位・下位関係をたどり，変更に関連する情報を調べる．さらに，相互作用が予想されたオブジェクトが変更された場合には，その変更された機能に関連する機能リンクをたどり新たな相互作用の有無を調べる．

以後，この過程を繰り返し，モデル全体を探索し，かつ，ユーザが新たな変更要求を示さなくなるとき，システムはその変更作業を終了する．

4．2 変更例

フェリーシミュレーションのモデルに新しくトラックを追加することによる変更について検討する．新しく追加されるトラックは，本島と島との間をフェリーを用いて往復し，本島から島へ荷物を運ぶ．さらに，トラックの動作とフェリーの動作とは同期しており，トラックがフェリーに乗船しない限りフェリーは出港しない．

このシミュレーションモデルの変更に関係する対象ソフトウェア上の主要な変更は，新しいクラス Truckの追加とクラスFerryのタスクの変更である．

4．3 変更支援例

まず，ユーザがクラスTruckの付加を宣言する．クラス

Assistance for modifying software using a functional model
Hiroyuki YAMADA† and Yoshikazu TEZUKA††
† Ehime Univ., †† Osaka Univ.

Truck を定義することは，追加されるクラスに対応するノードを機能モデルに追加することになる．

ここでは，新しいクラスがクラスEventMonitorの下位クラスとして宣言されるので，機能モデル内のノードEventMonitorに下位ノードが付加されることになる．ノードが新たに付加されたので，付加されたノード（EventMonitor）から相互作用の有無を調べるために，ノードに接続されている機能リンクが調べられる．ノードEventMonitorには機能リンクがないので，ノードの上位・下位関係より，ノードSimulationObjectに接続されている４つの機能リンク（到着定義リンク，リソース生成リンク，リソース獲得リンク，リソース解放リンク）が発見される．ここで，到着定義リンクは新しくオブジェクト（ノード）がモデル内に付加されたために，また，リソース生成リンクはクラス Truckの定義中でリソースが生成されるために，それぞれのリンクはたどられるが，残りの２つのリンクはたどられない．

到着定義リンクはノードSimulationに接続されており，ノードSimulation内に記述されている情報の中から，到着定義に関連するものが取り出される．この場合には，クラスSimulationの下位クラス（クラスFerrySimulation）内のメソッドdefineArrivalScheduleを変更する必要があることが指摘される．このメソッドの修正後に，ノードTruckとノードFerrySimulationとは到着定義リンクで接続される．さらに，メソッド修正に伴う相互作用を調べるために，修正に関連するリンクを探すが，存在しないので，到着定義リンクに関連する処理は終了する．

リソース生成リンクはノードResourceに接続されている．そのノードに記述されている情報の中から，リソース生成に関するものが取り出される．しかしながら，該当するリソースを表現するノードが無いので，リソース定義リンクがたどられる．このリンクは，ノードSimulationに接続されており，リソース定義に関する情報から，クラスSimulationの下位クラス内のメソッド defineResourcesを変更する必要があることが指摘される．このメソッドの修正後，新しく定義されたリソースを表現するノードTruckCrossing がノードCoordinateResourceの下位ノードとして付加される．さらに，ノード FerrySimulationとノード TruckCrossingとの間にリソース定義リンクが張られ，ノード Truckとノード TruckCrossingとの間にリソース生成リンクが張られる．

つぎに，新しく付加されたノードがもつべきリンクはリソース獲得リンクとリソース解放リンクであることがノードResourceから示される．これらのリンクが接続されているノードSimulationObjectに記述されている相互作用に関する情報より，ノードSimulationObjectの下位ノードにリンクが張られるべきであることが分かる．リンクが接続されるべきノードが発見されると，相互作用に対する処理がなされる．この場合には，ノード Ferryが対象となり，クラス Ferryに対する変更はこのノードに記述されている情報を参照して行われ，最終的にノード TruckCrossingとノード Ferryとの間にリソース獲得リンクとリソース解放リンクが張られ修正が終了する．

5．おわりに

本稿では，ソフトウェア変更支援システムで，相互作用の影響箇所を検出するために，相互作用に関する情報が記述された機能モデルを用いた変更支援について述べた．本モデルは，ソフトウェアに現れるオブジェクト間の機能的な関係に着目して，ソフトウェアをモデル化したものである．

今後の課題としては，ソフトウェアから機能モデルを生成する過程を支援する機構の考察が挙げられる．

［謝　　辞］

日頃，御激励頂く愛媛大学教授相原恒博先生および教授高松雄三先生に感謝の意を表します．

［参考文献］

[1] 山田，手塚：”機能モデルに基づくソフトウェア変更支援システム”，信学技報，人工知能と知識処理，Vol. 89 (1989)

[2] Goldberg, A. and Robinson, D. : "Smalltalk-80 The Language and Its Implementation", Addison Wesley (1983).

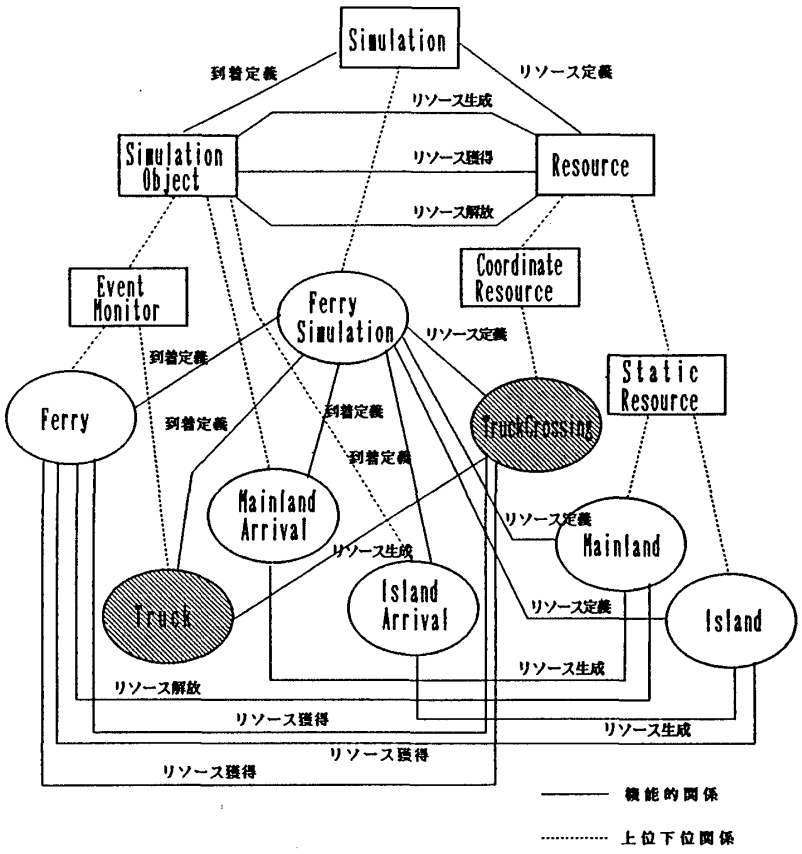

図１．ソフトウェアの機能モデル