表明とタイプ置換原理のモジュラー推論を利用したミッションクリティカルシステムのための開発アプローチの提案

橋本　隆成[†]

The Proposal of Software Development Approach for Mission Critical Systems Using the Assertions and Type Substitution Principle and Modular Reasonings

HASHIMOTO Takanari

ねらい　表明を定義しタイプ置換原理のモジュラー推論で演繹推論を行うミッションクリティカルシステムのための開発アプローチを提案する。また，本アプローチを支援する演繹推論と表明コードの生成を自動で行うモデリングツールを紹介する

キーワード　オブジェクト指向開発，コンポーネントベース開発，モジュラー推論，不変条件，再利用,自動化，契約による設計 ™，タイプ置換原理,

Target: This paper presents new software development approach which uses assertions, calculating type substitution principle and modular reasonings for mission critical systems. And, this paper introduces a modeling tool for the approach. The tool generates assertions, calculates modular reasonings, and also generates code of asserts automatically.

Keywords: Object-Oriented Development, Component-Based Development, Modular Reasoning, Invariant, Re-use, Automatic, DesignByContracts™, Type Conformance Rules, Type Substitute Principle

１．想定する読者・聴衆

本発表は,高い信頼性が求められるミッションクリティカルシステムの設計,実装を行う開発者,品質保証の担当者および管理者を対象としている。特にモデルドリブン開発(MDD：Model Driven Develop)の開発者や実施を検討している開発者を焦点に置いている。

２．背景

多くの開発現場では高品質と高生産性を両立させることが厳しい状況にある。品質と生産性の向上には形式的アプローチが有用であることが知られている。形式的アプローチは各種提案され,それぞれ専用の支援ツールも存在するものが多い。しかし、まだ開発現場で広く利用されている状況とまで至っていない。

一方､近年開発現場では，オブジェクト指向開発やコンポーネントベース開発が浸透し,SysML／UML によるモデルドリブン開発が積極的に実施されている。しかし,数多く提唱されているオブジェクト指向開発やコンポーネントベース開発方法論や開発プロセスには,形式的アプローチで扱う工学的な理論・技法が取り入れられていないものが殆どである。

また，開発現場で広く使われている商用の SysML／UML 用のモデリングツールにも工学的な理論・手法をサポートしたものがほとんど存在していないのが現状である。

３．課題

開発現場ではオブジェクト指向やコンポーネントベース開発により品質と生産性の向上を期待して,モデルやソフトウエア部品の再利用,あるいはコード自動生成技術の利用も行われている。しかし,設計,インスペクション(レビュー)および試験ケースのいずれにおいても,開発者・品質保証担当者の恣意や経験に強く依存しており工学的な理論を活用した客観的作業が実施されていない。つまり多くの開発プロジェクトでは,要件や設計の欠陥や不備を確実に防ぐ方法が十分に採用されておらず,結果が頻発し,大きな手戻り工数が発生している。生産性と品質の向上には，工学的理論を利用した作業を実施し,上位工程から欠陥混入を確実に防ぎ,手戻りの発生を最小限にすることが不可欠である。再利用についてもモデル,コンポーネントあるいはコードの再利用が期待どおりに達成されていないばかりでなく,工学的な方法で再利用が実施されていない危険な状況も散見される。

４．提案

4.1. 提案する開発アプローチの概要と特徴

本発表では開発現場で浸透しているオブジェクト指

[†]HASHIMOTO SOFTWARE CONSULTING
HASHIMOTO SOFTWARE CONSULTING
Sagamigaoka 5-4-308, Zama-shi, Kanagawa, 252-0001 Japan
E-mail: hashimoto@hsc-i.com

向開発やコンポーネントベース開発に,形式的アプローチで取り扱う工学的な考え方を取り入れた開発アプローチを提案する。提案するアプローチは,SysML／UMLによるオブジェクト指向開発やコンポーネントベース開発のメリットと形式的アプローチの工学を融合させた相乗効果を狙っている。

この開発アプローチは,詳細な作業行程,作業手順,成果物の種類およびフォーマットを厳密に規定しているものでない。工学的な理論,原理および技法を SysML／UML によるオブジェクト指向開発やコンポーネントベース開発と併用し，高品質と高生産性の実現を達成する効果的な作業の進め方ついに言及している。そのため各開発現場で使用している開発アプローチに取り入れることが可能となっている。

提案するアプローチは開発上位行程から下位行程にかけて段階的に分析・設計作業を実施していくことで高い品質と生産性を実現することを意図している。成果物であるモデルを,各作業行程を経て詳細化されていく作業の中で,適切かつ効果的に工学的な理論,原理および技法を利用して正当性や妥当性を担保していく。

4.3.提案開発アプローチの理論・手法

本発表では提案する開発アプローチの中から幾つかの重要な理論や原理に焦点を当てる。具体的にはクラスやコンポーネント[9]の表明(「不変条件(Invariant)」「操作の事前条件(Pre-Condition)」「操作の事後条件(Post-Condition)」「操作の例外(Exception)」)、これは「契約による設計 ™(Design By Contracts™)」」[3] [4]として知られている。「開放閉鎖原則(Open-Closed Principle)」[4],「タイプ置換原理(type substitution principle)」,「モジュラー推論(modular reasoning)」である。上記の理論や原理を,オブジェクト指向開発やコンポーネントベース開発に適用する基本事項と応用例を紹介する。

例えば,契約による設計 ™は,主として詳細設計と実装に焦点が置かれており,開発ライフサイクル全体を通じて適用できる実践的な手法・方法論としての性質は弱い上に効果的な支援ツールや SysML／UML ベースのツールで契約による設計 ™をサポートしたのも存在しない。提案する開発アプローチでは，契約による設計 ™を SysML／UML によるオブジェクト指向開発やコンポーネントベース開発のアプローチで利用する効果的な方法を紹介している。

4.4.提案開発アプローチの支援ツール

厳しい開発期間と開発予算が存在する開発現場では高品質と高生産性を両立させるには,効果的な開発プロセスとともに開発支援ツールの利用が不可欠である。

そこで,提案するアプローチで利用する工学的な理論,原理および技法をサポートした SysML／UML ベースのモデリングツールを開発した(図 1)。

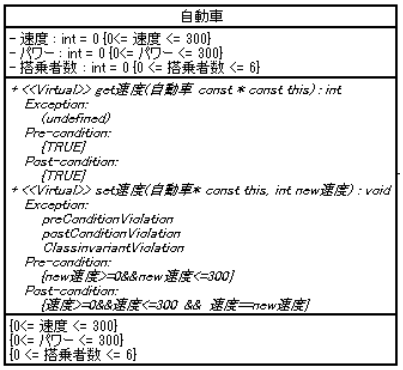

図 1 本発表のツールで表明を定義したクラスの例

Fig. 1. An example of assertions of a class defined by the tool

4.5. 軽量推論エンジンによる表明自動生成と推論機能

機能拡張や再利用のための優れた設計原理に開放閉鎖原理がある。この原理はデザインパターンやプロダクトラインエンジニアリングアプローチで積極的に利用されている。この開放閉鎖原理は継承による多相(Polymorphism)を利用するため，タイプ置換原理のモジュラー推論を実施し,「振る舞いレベル(Behavior Level)」の型安全(Type Safe)を担保する必要がある。この推論が「真(true)」となるクラス,コンポ―ネントは「振る舞いサブタイプ(Behavior Subtype)」[2]と呼ばれる。

タイプ置換原理のモジュラー推論は「Liskov 置換原理[1] [2]」が有名であるが,多数のタイプ置換原理の演繹推論式が提案されている[6] [7]。

本発表のツールでは振る舞いレベルの型安全な機能拡張と再利用を保証する 6 個の演繹推論規則をサポートするので設計や再利用の方針に従い最適な規則を選択することができる(図 2)。[manual]以外を選択した場合は,スーパータイプであるクラスの表明情報から継承関係にあるサブクラスが振る舞いサブタイプになるように表明を自動生成する。舞いサブタイプになるように表明を生成させる技法として「パーコレーション(Percolation)」[5]が知られている。パーコレーションは単純であるが,表明の特に事前条件と事後条件が極めて冗長になる欠点がある。本支援ツールでは，設計者がスーパータイプの表明を予めツールで設定すれば,パーコレーションのこの欠点を排除し,最適化されたサブタイプの表明の条件式を生成する。

これにより深い継承階層でも首尾一貫した方法で振る舞いの意味的な正当性が保証できる。例えば,図 3 のように「Satisfies」を選択した場合は,Liskov 置換原理に対応するモジュラー推論式で判定を行う。「Satisfies」は保守的な安全性の高いモジュラー推論式のため,「Satisfies」で≪LSP satisfied≫と判定されると継承階層内の任意の位置のクラスに修正を加えても,あるいは継承階層内の任意の位置にクラスを新規に追加あるいは削除しても振る舞いの意味的な正当性が継承階層全体

で完全に保証される。

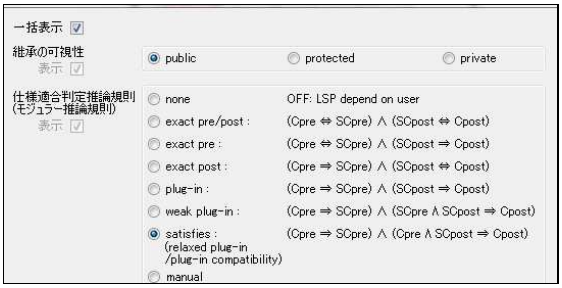

図 2 仕様適合判定規則の推論規則

Fig. 2.The deductive inference rules for Specification-Matches

再利用に「Satisfies」の推論を適用し「真」と判定されると≪LSP satisfied≫と表示される [2][6] [7]。

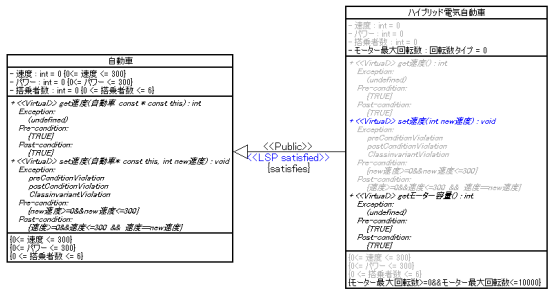

図 3 本発表のツールの演繹推論エンジンによる判定の例

Fig. 3. An example of the result of Deductive reasoning calculation engine by the tool

4.6 表明コードの自動生成機能

表明を直接サポートする Eiffel 言語[3]以外の言語では手動で表明であるクラス不変条件と各操作の事前条件,事後条件および例外定義を実装する必要がある。クラスが多数であるときや継承階層が深いときは,手動による表明の実装は間違いも起きやすく工数もかかり決して単純な作業では無い。本支援のツールは表明のコードを C,C++などの言語で自動生成可能であるため,正確に表明を実装すると共に作業工数削減も期待できる。表明は assertion コードとして生成されるが,試験終了後に assertion コードの無効化が可能であるためコードサイズと実行処理速度に影響を与えない。また,ツールの設定でコード生成の際に表明コードを生成させないことも可能である。

５．効果

ミッションクリティカルなシステムや組込み系システムでは,新規開発よりも保守・改修による機能の拡張や再利用ベースの開発が多い。本発表の開発アプローチは,機能の拡張や再利用に有利であるオブジェクト指向開発やコンポーネントベース開発のモデルドリブン開発を工学的な作業を取り入れることを提案した。これにより保守・改修および再利用の作業でも属人的な評価や恣意による作業から脱却し工学的な作業を実現する。またツールによる自動処理により多数のクラスやコンポーネントを扱う場合もミスを防ぎ作業を効率化できる。演繹推論などの数学や論理学に精通していない開発者や品質保証担当者でも本発表のツールを利用することで効果を享受できる。

６．まとめ(今後の課題)

本発表で提案した開発アプローチは並行・並列システムにも適用が可能である。現在並行・並列システムへ適用のために本発表の開発アプローチの拡張を進めている。表明を利用したオブジェクト指向開発やコンポーネントベース開発の考え方を並行・並列システムにも適用する場合は, オブジェクトやクラスの並行・並列の計算モデルに依存する。採用する計算モデルによって表明によるアプローチを並行・並列に拡張できる難易度が異なる。表明と継承には「継承異常(inheritance anomaly)」の問題が知られている。オブジェクト指向アプローチと表明によるアプローチを並行・並列に拡張し継承異常に対応したものに「SCOOP(Simple Concurrent Object-Objected Programming)」[4]がある。SCOOP は Eiffel 言語用の CSP ベースの並行・並列拡張ライブラリーである。また, 航空宇宙分野で利用されるプログラム言語 Ada83／Ada95 は通信メカニズムとして CSP ベースの同期通信メカニズムである「ランデブー(Rendezvous)」を備えている。Ada 言語に表明を利用し分散並行・並列に拡張する DARGOON が知られている。

７．文献

[1] B. Liskov, Data Abstraction and Hierarchy, *SIGPLAN Notices* 23(5), May 1988.

[2] B.Liskov and J.M.Wing. A Behavioral Notion of Subtyping . ACM T OPLAS,16(6):1811 1841, November 1994.

[3] Meyer, B., *Eiffel – the Language*, Prentice Hall, 1992

[4] Meyer, B., *Object oriented software construction*, 2nd Ed., Prentice Hall,1997.

[5] Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools, Addison-Wesley, 1999

[6] Y. Chen and B.H.C.Cheng, A Semantic Foundation for Specification Matching. In G.T. Leavens and M. Sitaraman (Eds.), Foundations of Component-Based Systems, Cambridge Univ. Press, 2000, 91-109.

[7] A.M. Zaremski/J.M. Wing, Specification Matching of Software Components, *ACM Transactions on Software Engineering and Methodology* **6**(4), 333-369, October 1997.

[8] C.A.R. Hoare, An axiomatic basis for computer programming, *Communications of the ACM*, Oct. 1969, 576-583.

[9] C. Szyperski, *Component Software. Beyond Object-Oriented Programming*, Addison-Wesley, 1998.