オブジェクト指向設計における一貫性チェックについて

4C－3

原　勝彦[†]　宇都宮　公訓[††]

[†]筑波大学 第三学群 情報学類　[††]筑波大学 電子・情報工学系

1　はじめに

著者等は,OMT(Object Modeling Technique)[1]によるソフトウェア開発において使用する一貫性チェッカを検討しており,これまで,分析フェーズで行なう一貫性チェック[2],オブジェクト設計フェーズで行なう一貫性チェック[3]について発表してきた.今回,オブジェクト設計段階と実装モデル段階での一貫性条件についてより詳細に検討したので,報告する.

一貫性条件は実装環境に左右されないことが理想であるが,現実には,オペレーティングシステム,クラスライブラリ（フレームワーク）,言語処理系,データベース管理システム等の影響を受けてしまう[4].ここでは,Windows上でC++で実装することを想定して一貫性条件を検討した.

2　対象ドキュメント

設計は,分析フェーズで作られたドキュメントをもとに行なわれ,これらのドキュメントをより詳細化するとともに,新しいドキュメントを開発する.今回の一貫性条件の検討は,分析フェーズで得られる

- クラス図
- シナリオ（ユースケース）
- 事象トレース図
- 状態図
- データフロー図
- 操作仕様書

と,設計フェーズで得られる

- クラス図
- メッセージ階層図
- 事象トレース図
- オブジェクトインタラクション図
- オブジェクト表現
- 操作仕様書

等を対象にして行なった.メッセージ階層図,事象トレース図,オブジェクトインタラクション図はそれぞれに特長を持ち,使用される局面も異なるが,重複する記述を多く含むので,一貫性条件も多く重複する.

ここでの一貫性チェックには明らかなエラーだけでなく,確認を求めるものも含んでいる.以下,3～7に一貫性条件を列挙する.

Consistency Check for Object-Oriented Design

Katsuhiko HARA[†], Kiminori UTSUNOMIYA[††]

[†]College of Information Science,Univ. of Tsukuba

[††]Institute of Information Sciences and Electronics, Univ. of Tsukuba

3　メッセージ階層図において

1. クラスAがクラスBのメソッドを呼び出しているとき,クラス図中でAとBの間に関連が設定されていなければならない.
2. クラスAがクラスBのメソッドCを呼び出しているとき,Cは'public'か,Cが'protected'でかつAがBの子孫クラスのいずれかでなければならない.
3. メッセージ階層図中でのメッセージ送受信（メソッドの呼び出し）順序は事象トレース図中での事象の送受信順序と矛盾してはならない.
4. メッセージ階層図中の各クラスの活動はそれぞれのクラスの状態図でもトレースできなければならない.
5. クラスAがクラスBのメソッドを呼び出していれば,クラス図でクラスAからクラスBに向かう関連が存在しなければならない.
6. クラスAで内部インスタンスBを呼び出していれば,オブジェクト表現でBがAの内部インスタンスであると記述されていなければならない.
7. クラスAで内部インスタンスBが呼び出されており,Bのクラスがクラス図に記述されているときは,AはBの'集約'になっていなければならない.
8. 多相性の記述を確認する.
9. メソッドの呼び出しで渡される引数はデータフロー図のプリミティブプロセスでの記述と矛盾してはならない.
10. クラスAにおけるクラスBのメソッドCの呼び出しで,インスタンスに対するFOR指定があるときは,AからBに向かう関連の先の多重度は'多'でなければならない.
11. メッセージ受渡しで想定するクラスが設計フェーズの事象トレース図での記述と一致していなければならない.
12. メッセージ受渡しで想定するクラスが設計フェーズのオブジェクトインタラクション図での記述と一致していなければならない.
13. 同一クラスに対するメソッドの呼び出しで字下げが行なわれていれば,字下げされているメソッドは'private'指定されていることが多い.

4 事象トレース図において

1. メソッドの呼び出し（メッセージの受渡し）順序がメッセージ階層図でのメソッド呼び出しと矛盾してはならない.
2. メソッドの呼び出しで返される値が操作仕様書での記述と矛盾してはならない.
3. オブジェクトの生成と消去に関する記述が状態図の記述と矛盾してはならない.
4. 3メッセージ階層図においての1～8.

5 オブジェクトインタラクション図において

1. インスタンス間の（関連の）構造がクラス図の（関連の）構造と矛盾してはならない.
2. インタラクションの実行順序がメッセージ階層図中のメッセージ交換の順序と矛盾してはならない.
3. インタラクションの実行順序が事象トレース図中のメッセージ交換の順序と矛盾してはならない.
4. 引数によるクライアント - サプライア関係がクラス図の関連の向きと一致していなければならない.
5. インタラクションのメソッド名と引数名が操作仕様書の記述と一致していなければならない.
6. インタラクションの戻し値が操作仕様書中の仕様と一致していなければならない.
7. 3メッセージ階層図においての1～9.

6 実装モデルのクラス図において

1. どの関連にも向きが指定されていなければならない.
2. 分析フェーズでのクラス図にある関連が除かれている場合は, その旨を警告する.
3. 分析フェーズでは集約になっていたクラスが取り除かれていれば, 内部インスタンスに変わっているかどうか確認する.
4. 多重度が分析フェーズの多重度と矛盾してはならない.
5. 分析フェーズになく新しく追加されているクラスは確認を求める.

7 オブジェクト表現において

1. 操作や属性の継承はクラス図での記述と一致していなければならない.
2. 使用している内部インスタンスは実装モデルのクラス図での記述と一致していなければならない.
3. 多重継承は, 独立した汎化階層からの継承か, 同じ汎化階層からの互いに素でない汎化クラスからの継承でなければならない.
4. 制限による汎化では, R属性を更新するメソッドの継承を禁止することが明記されていなければならない.
5. 抽象操作は必ずサブクラスで再定義されていなければならない.
6. クラスライブラリ中のクラスを用いる場合は, 継承するメソッドを確認する.

8 おわりに

今後はUML[5, 6, 7]を採用し, さらに精度の高いチェックを行なうために, クラス図, 状態図, ユースケース, 事象トレース図, メッセージ階層図, オブジェクトインタラクション図, 操作仕様書, データ辞書等を形式化する予定である. そのため, Z++[8]を導入することにしており, すでに別のグループがZ++による状態図の形式化を行なっている. また, デザインパターン[9, 10]を用いて設計した場合の一貫性チェックも計画している.

参考文献

[1] Rumbaugh,J. et al.: *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

[2] 波潟由布子, 合原正男, 宇都宮公訓他: OMTにおけるモデルの一貫性について, 情報処理学会第52回全国大会講演論文集, 1996.

[3] 金田忠士, 松浦江里, 宇都宮公訓他: OMTにおけるオブジェクト設計の一貫性チェックについて, 情報処理学会第54回 全国大会講演論文集, 1997.

[4] Kurt,K.D.: *Applying OMT*, SIGS Books,1995.

[5] Rational Software Co.: *UML Semantics*, ver.1.1,1997. (http://www.rational.com/uml)

[6] Rational Software Co.: *UML Notation Guide*, ver.1.1,1997. (http://www.rational.com/uml)

[7] Fowler,M. et al.: *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley,1997.

[8] Lano,K.: *Formal Object-Oriented Development*, Springer,1995.

[9] Gamma,E. et al.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley,1994.

[10] Pree,W.: *Design Patterns for Object-Oriented Software Development*, Addison-Wesley,1994.