在庫管理プログラムの設計に対する JML 記述と ESC/Java2 を用いた検証の事例報告

尾鷲　方志† 　　岡野　浩三†（正員）
楠本　真二†（正員）

Design of Warehouse Management Program in JML and Its Verification with ESC/Java2

Masayuki OWASHI†, *Nonmember*, Kozo OKANO†, and Shinji KUSUMOTO†, *Members*

† 大阪大学大学院情報科学研究科，豊中市
Graduate School of Information Science and Technology, Osaka University, Toyonaka-shi, 560–8531 Japan

あらまし　本論文では，契約に基づく設計が行える JML を用いて在庫管理問題の仕様作成を行い，ESC/Java2 を用いて検証した．この結果や考察について述べる．

キーワード　ESC/Java2，JML，在庫管理問題，設計検証，契約による設計

1. まえがき

Design by Contract（契約に基づく設計，以下 DbC とする）[1] に基づいた開発では，表明により契約を示すことで詳述段階におけるデバッグが効率化され，品質が向上する．DbC を実現するための言語である JML は，Java プログラムに対しコード実体と結び付いた契約を記述することができ，ESC/Java または ESC/Java2 を用いてコードが契約を侵すおそれがないかを検証することができる．

本論文では，プログラム設計の共通問題である在庫管理問題 [3] について，JML を用いて仕様を記述した．また，実際に Java を用いてプログラムを記述した後，ESC/Java2 を用いて検証を行い，設計と実装がともに ESC/Java2 において妥当であることを 1 分程度で示すことができた．更に，例外処理に関する仕様，実装の正当性に関しても十分な記述をし，ESC/Java2 において妥当であるとの出力を得られた．また，文献 [2] で議論されていた自明な例外を明示することの妥当性ついて，検証時間の観点から妥当性を示した．

2. 在庫管理プログラムの仕様

2.1 概　　要

在庫管理問題に対するクラス図は図 1 に表すように，“受付係”，“倉庫係”，“注文”，“顧客”，“品目”，“コンテナ” の 6 のクラスで表現できる．

受付係は ReceiptionDesk クラスで表され，外部からの要求を受け取り，倉庫係に在庫が要求を満たすか確認させ，可能ならその要求された商品の出庫を倉庫係に依頼し，不可能ならば未処理要求リストとして保存する．同様に，外部からコンテナを受け付け，倉庫係に搬入させる．このとき，未処理の要求があればその要求の処理をする．外部から渡されたデータを直接処理できるのは受付係のみである．

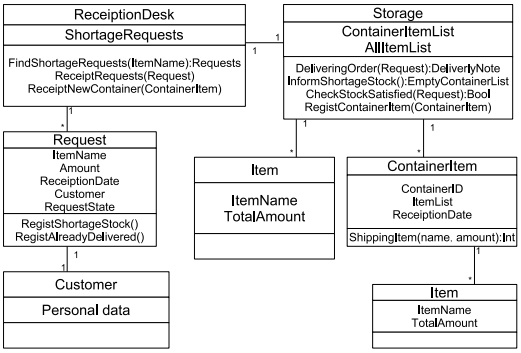

図 1　在庫管理プログラムのクラス構成
Fig. 1　Class diagram of warehouse management program.

倉庫係は Storage クラスで表され，搬入したコンテナのリストをもつ．受付から送られてきた要求に対し，在庫が要求を満たしているかどうかを返す．また，出庫指示に対して，どのコンテナからどれだけ出庫したかを示す出庫指示書を受付に返す．全品目の集合の各要素は一意であり，また，コンテナの集合に対し，各コンテナのコンテナ番号は一意である．品物の搬入はコンテナ単位で行われ，品目単位での出庫が行われる．

注文は Request クラスで表され，要素として品名と数量，要求者（顧客），更に要求が不足している等の状態を示す値をもつ．一度に要求できる品目は 1 種類のみである．

2.2 実装上の工夫

倉庫に対しての商品の搬入はコンテナ単位で行われるが，注文は品目単位で行われるため，コンテナの集合だけでなく，その中身をまとめた全品目の集合も倉庫係に保持させた．全品目の集合とコンテナの集合中の内蔵品の合計は互いに矛盾しない．

3. 記述概要

各クラスに対して invariant 節を用いた不変条件，また，各メソッドに対し requires 節，ensures 節を用いた事前，事後条件及び，signals 節を用いて例外が起こる条件，また確実に例外が起こり得ないことを明示した．各クラスに対しての詳細な記述は，文献 [4] を

```
/*@ public behavior
        requires r != null;
        requires checkStockSatisfied(r);
        assignable allitemlist, containerlist;
        ensures (\sum ContainerItem i; \result.contains(i);
                ((Item)i.getContainedItem().get(0)).getAmount())
                        == r.getAmount();
        ensures (\forall ContainerItem i; \result.contains(i);
                ((Item)i.getContainedItem().get(0)).getName()
                        .equals(r.getName()));
  @*/
```

図 2　倉庫の出庫処理の JML 記述（メソッド記述は略）
Fig. 2　JML description of “Delivering order” in Storage class. (without method body)

参照のこと．記述と検証作業に 2 か月ほどかかり，6 回のバージョン変更を行っている．

また，これらの JML 記述や Java プログラムの規模については約 1000 行（内 JML は 400 行）であった．

3.1　主要なメソッドに対する記述例

倉庫係における出庫処理の契約を図 2 に表す．r は受付係から送られてきた注文を表す．

まず，事前条件として注文 r が null でないこと，注文に対し在庫が十分あることが挙げられる．次に，assignable にてこのメソッドで更新され得る値を示す．ここではコンテナリスト，全品目リストが更新される．

最初の ensures 節で示される事後条件は，戻り値で表される出庫指示書で，どのコンテナから品目をどれだけ出荷したかの合計が注文された数量と等しくなることを示している．ここでは一つの注文で 1 品目しか注文できない仕様のため，出庫指示書で示されたコンテナから取り出す品目の先頭に出庫される品目が存在するために get で先頭の要素を参照する．

二つ目の ensures 節で示される事後条件は，出庫指示書に書かれたすべてのコンテナから取り出した品目が要求された品目と等しいことを示している．

4.　ESC/Java2 による検証結果と考察

文献 [4] の JML 記述に対し，ESC/Java2 を用いて一部警告はあるものの，ほぼすべてのクラスのメソッドに対し Valid であるとの結果を得ることができた．各クラスの検証にかかった時間を表 1 にまとめる．

上記の検証結果より，例外を明示することにより検証時間が目立って増加しているわけではないことがいえる．これより，文献 [2] においていわれた，例外が起こらない場合についてもそのことを明示すべきだ，という主張に対して，検証時間上のデメリットはないことを確認できた．

表 1　仕様記述に対する検証時間
Table 1　Verification time.

Class Name	例外記述あり (s)	例外記述なし (s)
ReceiptionDesk	7.265	7.172
Storage	9.094	9.328
ContainerItem	18.079	18.125
Request	4.797	4.297
Item	9.328	9.328
Customer	6.375	6.375
Total	54.938	54.575

また，仕様記述において，
\forall int i; 0 <= i && i < List.size();List.get(i)...
のように，配列を扱うようにオブジェクトを参照した方が ESC/Java2 を通した検証にかかる時間が少なかった．実際に ContainerItem クラスの不変条件において適用した結果，配列形式にした場合について 18.1 秒から 16.9 秒へと，1 秒程度の速度増加が得られた．他のクラスについて同様の傾向があった．

この理由として，Simplify の入力となる論理式では，オブジェクトのフィールドを配列として表現するため [2]，配列を直接扱った方が定理証明の時間が少なく済むと考えられる．一方，仕様記述の観点からは前者の記述方式の方が好ましいと考えられる．

5.　む　す　び

酒屋の在庫管理問題についてモデル化し，JML を用いて仕様を記述することができた．また，Java プログラムを記述し，ESC/Java2 を用いた検証にも 1 分程度で成功することが確認できた．

JML が比較的詳細な設計を記述するための言語であることから，より抽象的な設計記述を JML に変換することを研究課題として考えている．具体的には Alloy を用いた抽象段階での仕様の安全性が保証された設計から JML を自動生成することを考えている．

文　　献

[1] B. Meyer, Object-Oriented Software Construction, Second ed., Prentice Hall, 2000.
[2] D. Cok, “Reasoning with specifications containing method calls in JML and firstorder provers,” ECOOP Workshop FTfJP’2004 Formal Techniques for Java-like Programs, Oslo, Norway, June 2004. ICIS report NIII-R0426, pp.41–48, 2004.
[3] 二村良彦，雨宮真人，山崎利治，淵　一博，“新しいプログラミング・パラダイムによる共通問題の設計，” 情報処理，vol.26, no.5, pp.458–459, 1985.
[4] 尾鷲方志，岡野浩三，楠本真二，“JML を用いた在庫管理プログラムの設計と ESC/Java2 を用いた検証，” 信学技報，SS2007-22, 2007.

（平成 20 年 5 月 27 日受付）