ソフトウェアレビューの観点を導出するためのメタモデル構築

元山　厚[†1,†2]　中谷　多哉子[†1]

開発の現場では，ソフトウェアレビューによって設計の品質を高める工夫がなされている．それにもかかわらず，現実にはテスト工程で，設計仕様の誤りや記述漏れが指摘されている．特に基本設計仕様書から詳細設計仕様書間の仕様の引き継ぎ漏れや誤りに起因する欠陥が，設計仕様自体の誤りよりも多いことを複数の事例で確認した．本稿では，設計仕様書のレビューの精度を向上させるために，設計仕様として定義すべき事項および構造を示す設計仕様のメタモデルを示し，その設計仕様のメタモデルをソフトウェアレビューの観点として提案する．さらに，提案方法を適用したソフトウェアレビュー実験を実施し，設計仕様のメタモデルの有効性を確認した結果を示す．

The metamodel development to derive the perspective for software review

Atsushi Motoyama[†1,†2] and Takako Nakatani[†1]

Software review is one of the effective measures for improving software quality. Errors and omissions of design specifications are detected in the test process. In examples,we confirmed that there were more defects due to omissions and errors of transfer from external design specifications to internal design specifications than errors of design specifications in themselves. In this paper,we propose the metamodel which shows the components and structure which should be defined as design specifications,and the software review method which made the perspective of the review the metamodel of design specifications. Furthermore,we evaluated the effectiveness of the metamodel of design specifications in the experiment. We discuss the result.

†1 筑波大学大学院ビジネス科学研究科 Graduate School of Business Sciences,University of Tsukuba
†2 株式会社日立製作所 Hitachi,Ltd.

1. はじめに

1.1 研究の背景と目的

ソフトウェア開発において，ソフトウェアの品質と開発の効率を向上させるためには，設計工程で設計上の欠陥を発見し，テスト工程まで設計上の欠陥が発見できない事態を避ける必要がある．近年，情報システムが社会や企業とって不可欠なものとなり，ソフトウェアの品質が企業活動に与える影響は益々大きくなっている．ビジネスを優位に進めるために，ソフトウェア開発に掛けられる期間やコストは圧縮される傾向がある．ソフトウェアの欠陥を修正するコストは，開発サイクルの後半になるほど上昇する．そのコストの差は小規模で重要性の低いシステムでさえ 5 倍，一般的な規模のシステムでは 100 倍に達する[1)] と言われている．ソフトウェア開発プロセスの前半，即ち設計工程で高い品質を確保することが，ソフトウェアの欠陥修正の期間とコストを削減するために効果的である．

開発の現場では，ソフトウェアレビューによって設計の品質を高める工夫がなされている．しかし，現実にはソフトウェア開発のテスト工程で，設計仕様やプログラミングを検証する際に，設計仕様の誤りや記述漏れが多く指摘されている．我々がこれまで開発にかかわってきたプロジェクトでは，欠陥が混入された原因として最も多かったのが，設計仕様書の誤りであることを，複数の事例で確認した．

設計仕様書の誤りの中で，設計仕様自体の誤りより，設計仕様書間での仕様の引き継ぎ漏れや誤りが，多い傾向があった．この引き継ぎ漏れや誤りの原因は，設計仕様書における設計要素間の関連の不整合となっていた．その不整合は，要求仕様に基づきソフトウェアの機能仕様が記される基本設計仕様から，基本設計仕様に基づいて作成されソフトウェアの構造の仕様が記される，詳細設計仕様を経る過程で発生していた．例えば，基本設計仕様書に含まれる画面設計仕様書と，詳細設計仕様書に含まれるメソッド仕様書の間で，画面で入力されるデータの属性とメソッドの引数の属性の対応がとれない，設計仕様書間の不整合が指摘されることがある．

複数の設計仕様書を比較したソフトウェアレビューも実施されているが，担当者の記憶や直観に依存した方法となっており，改善の余地がある．設計の品質を向上させるために，基本設計仕様から詳細設計仕様に至る仕様のトレーサビリティを確保し，設計仕様書のソフトウェアレビューの精度を向上させなければならない．

本稿では，設計仕様書のソフトウェアレビューの精度を向上させるために，次の 3 点の課題を解決する．第 1 の課題は，設計仕様として記述すべき事項や，それぞれの設計仕様の構

造を管理することである．第 2 の課題は，基本設計仕様と詳細設計仕様を構成する要素間の関連でのトレーサビリティを確保することである．第 3 の課題は，設計仕様の構造や，設計仕様を構成する要素間の関連に基づいたソフトウェアレビュー方法を提案することである．

第 1 および第 2 の課題に対処するために，設計仕様として定義すべき事項および構造を示す設計仕様のメタモデルを提案する．本稿での設計仕様のメタモデルとは，ソフトウェアまたはシステム仕様のモデルである設計仕様書に対して，その構造をモデル化したものである．設計仕様のメタモデルにより，設計仕様として設計すべき内容と，設計要素間の関連の明示が可能となる．設計仕様書の誤りの原因である，設計仕様書における設計要素間の関連の不整合は，設計仕様のメタモデルと設計仕様書を対応付けることにより，設計仕様の構成要素間の関連が明確になり，設計要素間の矛盾や抜け漏れを発見することが可能になる．

第 3 の課題に対処するために設計仕様のメタモデルと，基本設計仕様と詳細設計仕様の設計仕様のメタモデル間の関連を示す関連モデルをソフトウェアレビューの観点（perspective）[2] とすることを提案する．設計仕様のメタモデルの構成要素と，設計仕様のメタモデル間の関連をソフトウェアレビューの観点とすれば，設計仕様の構造と設計項目間の関連に基づいたソフトウェアレビューが，効果的，網羅的に実施できる．ここで，関連モデルとは，設計仕様のメタモデルの構成要素間の関連を，ソフトウェアレビューを実施すべき箇所としてモデル化したものである．また，観点とは，ソフトウェアレビューを実施すべき箇所である．本稿では，設計仕様のメタモデルと関連モデルが，ソフトウェアレビューの精度を向上することを確認する．そのために，ソフトウェアレビュー実験を実施し，設計仕様のメタモデルと関連モデルの有効性を評価する．

1.2　研究のドメイン

設計仕様のメタモデルを構築する際の問題として，一つのメタモデルで全てのソフトウェアの設計仕様の意味を定義することは困難である点が挙げられる．ソフトウェアやシステムが実現する目的は多様であり，設計仕様のメタモデルはドメイン知識へ強く依存するためである．本稿では，企業で行われる業務のシステム化を目的にした業務アプリケーションソフトウェア/システムにドメインを限定し，設計仕様のメタモデルを開発する．設計仕様として設計すべき内容に共通部分が多く，基本設計仕様書や詳細設計仕様書のモデルをメタモデルとしての定義が可能になると考える．

販売管理や，会計管理などの企業の業務に向けた業務アプリケーションソフトウェアは，目的が多様であるが企業の業務で発生した情報をソフトウェアの情報入力画面から入力し，データベースに登録してそれを参照するという，ソフトウェアの利用者からの処理の流れは共通している．現在，業務アプリケーションソフトウェアの開発では，ソフトウェアの画面遷移，データベースへのアクセス，プログラム構成をフレームワークとして，共通の仕組みを実装することが多い．そのフレームワークと合わせて，設計仕様書の標準体系を規定している．筆者らが従事した複数の業務アプリケーションソフトウェアの開発プロジェクトでは，様々な顧客向けの業務アプリケーションソフトウエアが，標準として規定したフレームワークと設計仕様書の標準体系を使用して開発されていた．標準の規定であるフレームワークと設計仕様書の体系のモデルをメタモデルとして定義すれば，ソフトウェアの設計仕様の意味の多くの部分を網羅することが可能になる．したがって，標準の規定である設計仕様書の体系を前提とした業務アプリケーションソフトウェアの開発プロジェクトでは，提案する設計仕様のメタモデルが再利用可能となり，設計仕様のメタモデルの作成工数を軽減できる．

本稿の構成は以下の通りである。2 章では，関連研究を紹介する．3 章では，設計仕様の構造と構成要素間の関連を調査・解析し，基本および詳細設計仕様のメタモデルと，メタモデル間の関連モデルを示す．4 章では，設計仕様のメタモデルと，設計仕様のメタモデル間の関連モデルに基づくソフトウェアレビュー方法を提案し，評価実験により，提案する設計仕様のメタモデルの有効性の評価を行った結果を示す．5 章では，評価を行った結果に対する考察を述べ，最後に纏める．

2. 関 連 研 究

設計仕様のメタモデルに関する研究として，Ramesh ら[3] は，トレーサビリティを構成する要素の型，およびこれらの要素間に存在する関係の型をトレーサビリティ情報として表現するために，メタモデルを参照モデルとして使用することを提案している．また，大橋ら[4] は，ビジネスアプリケーション開発における要求定義工程と基本設計工程を対象に，モデルを利用したトレーサビリティの構築と，それを利用した完全性検証をおこなうための取り組みを示し，その有効性を評価した．

これらの研究は，設計仕様のメタモデルを定義し，トレーサビリティの構築・管理方法は示しているが，設計仕様のメタモデルを使用し，欠陥を検出して設計の品質を向上する方法は示していない．また，基本設計仕様と詳細設計仕様の間のトレーサビリティについては，言及していない．そのため，ソフトウェアレビューの観点（perspective）を導出することができない．本稿では，ソフトウェアレビューの観点を導出するために，設計仕様のメタモデルを示し，ソフトウェアレビューの精度を向上させることを目指す．さらに，設計仕様のメタモデルからソフトウェアレビュー観点を導出する方法と，その観点に基づくソフトウェ

アレビュー方法を提案する．

設計仕様のソフトウェアレビューでは，系統立てた方法を適用するのが効果的である．ソフトウェアレビューで使用される設計仕様書を調べる方法には，様々なものが提案されており，その有効性が研究されている[5]．例えば，チェックリストに記載された質問を順番に参照する方法，ユースケース記述など利用者とシステムとの対話手順に基づいてレビューを行う方法[6]，アクターの観点（利用者，設計者など）や品質特性の観点でレビューする方法[2]がある．また，Travassos ら[7] は，オブジェクト設計での要求定義書と設計仕様書の関連，及び異なる種類の設計仕様書の関連に基づくレビュー方法（Traceability-Based Reading (TBR)）を提案している．

ここまでに示した設計仕様書のレビュー方法には，設計仕様書間や設計仕様を構成する要素間の関連を検証するレビュー観点がない．そのため，設計仕様書間や設計仕様を構成する要素間の関連の不整合を検出することに対して，効果的な方法の提案にはなっていない．設計仕様のメタモデルの構成要素をレビューの観点とし，構成要素が持つ他の設計仕様の構成要素への関連を確認するレビューであれば，設計仕様を構成する要素間の不整合を検出し，設計仕様の妥当性と設計仕様を構成する要素間の関連の妥当性を検証することが可能となる．

なお，設計仕様書の単位での参照関係を把握したレビューでは，設計仕様書間の不整合に起因した欠陥を検出する手法として不十分である．設計仕様を構成する要素間の関連を踏まえていないため，不整合をチェックする箇所がレビューア任せになってしまい，不整合の欠陥全てを網羅して検出することはできない．さらに，設計仕様を構成する要素間の関連の不整合を確認するだけのレビューでは，設計仕様としての記述が誤っている欠陥や，関連がそのものが誤っている欠陥を検出することはできない．この理由は，設計仕様の内容の妥当性や，設計仕様を構成する要素間の関連の妥当性を検証するレビューの視点がないためである．設計仕様のメタモデルの構成要素を，レビューの観点とし，構成要素が持つ他の設計仕様の構成要素への関連を確認するレビューであれば，設計仕様を構成する要素間の不整合を検出し，設計仕様の妥当性と設計仕様を構成する要素間の関連の妥当性を検証することが可能となる．本稿では，設計仕様のメタモデルの構成要素を，レビューの観点としたソフトウェアレビュー方法を提案する．

3. 設計仕様のメタモデル

3.1 設計仕様書の構成の調査・解析

設計仕様書のメタモデルのベースとなる設計仕様書に記述されている情報を明らかにするため，業務アプリケーションソフトウェアの開発プロセスで使用されている設計仕様書を研究対象とし，設計仕様の構造と構成要素間の関連の調査と解析を行った．研究対象は，あるソフトウェアメーカーで使用されている，様々な顧客向けに業務アプリケーションソフトウェアを開発するための設計仕様書の体系を選択した．この設計仕様書の体系は，ソフトウェアメーカーが長年にわたり，様々な顧客向けに業務アプリケーションソフトウェアを開発してきた実績に基づいて作り上げた設計仕様書の標準である．基本設計，詳細設計の各工程で使用する複数の設計仕様書の形式を規定している．この設計仕様書の体系は，当該のソフトウェアメーカーでの，ここ 6 年間の業務アプリケーションソフトウェアの開発に適用が義務づけられている．

研究対象とした基本設計仕様書と詳細設計仕様書の構成要素を抽出し，構成要素間の関連を明らかにした．その結果に基づき，業務アプリケーションソフトウェアの基本設計仕様と詳細設計仕様の構成を定義した．

3.2 基本設計仕様のメタモデル

設計仕様に記述すべき事項や，その関連を明らかにするため，基本設計と詳細設計の各々のプロセス内での設計仕様の構造と構成要素間の関連を，設計仕様のメタモデルとして定義した．基本設計仕様のメタモデルを**図 1** に示す．

具体的な基本設計書の構造分析の結果に基づき，基本設計仕様のメタモデルは，業務要素，データ要素，データ操作要求，インタフェース，ロールの 5 個の構成要素群から構成されている．以下に，各構成要素群を解説する．

- 業務要素を中心にした構成要素群
  業務要素を中心にした構成要素群は，業務処理，業務プロセス，イベント，制約から構成されている．これらによって，ソフトウェアが実現する実世界の業務の構成が定義されている．業務処理は，業務の内容自体を示し，業務プロセスは業務の流れを示す．イベントはいつ業務が実施されるかを示す．制約は，業務の制約内容に応じて分類ができる．業務の制約は，外部制約と内部制約に分類される．外部制約とは，ソフトウェアの外部環境からソフトウェアに与えられる制約である．内部制約とは，外部制約以外の制約であり，ソフトウェアが業務を遂行する条件とソフトウェアの実行制約に分類される．

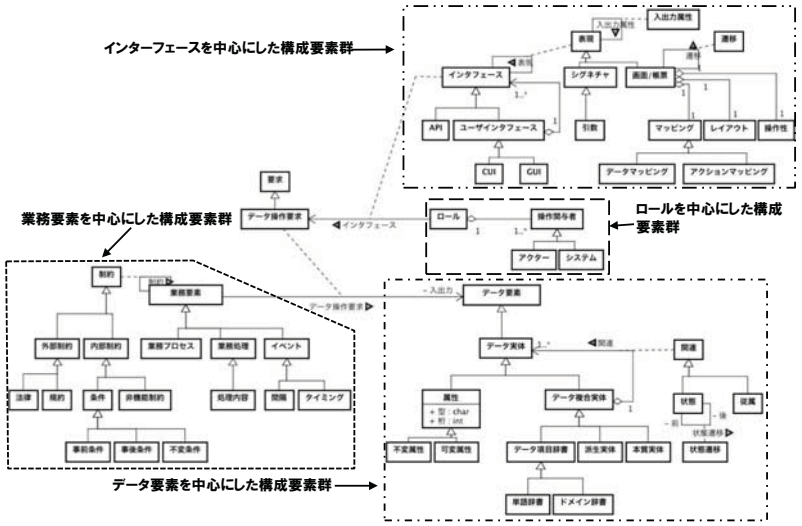

図 1　基本設計仕様のメタモデル
Fig. 1　The metamodel of external design specification

- データ要素を中心にした構成要素群
  データ要素を中心にした構成要素群は，ソフトウェアが扱うデータ構成を表す．データ実体は，属性とデータ複合実体に分類できる．データの属性には，可変な属性と不変な属性があることを定義する．データ複合実体とデータ実体は，データ要素の再帰的な構造を示している．
- データ操作要求
  データ操作要求は，要求を具体化したものであり，業務要素がデータ要素に行う操作の内容を示す．
- インタフェースを中心にした構成要素群
  インタフェースを中心にした構成要素群は，ロールのデータ操作要求への参照を，インタフェースを通して行うことを示す．インタフェースには，プログラム間の呼び出しのインタフェースを示す API（Application Program Interface），ユーザ向けのインタフェースとして，文字ベースの CUI（character user interface），図を用いた GUI（Graphical User Interface）がある．

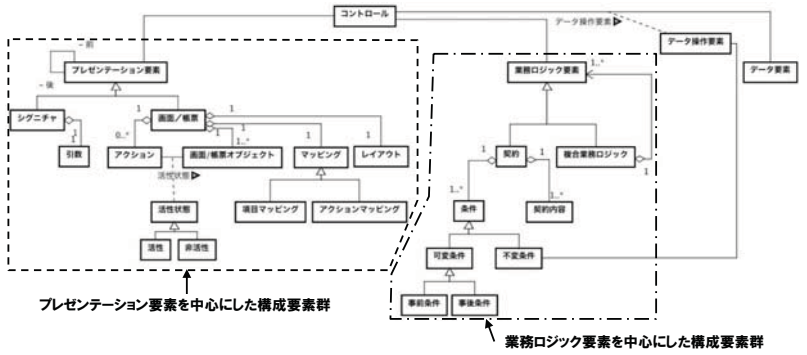

図 2　詳細設計仕様のメタモデル
Fig. 2　The metamodel of internal design specification

- ロールを中心にした構成要素群
  ロールを中心にした構成要素群は，アクターやシステムなどの操作関与者に伴う役割を示しており，それに基づきデータ操作要求の内容が変化することを示す．

以上の基本設計仕様のメタモデルは，基本設計として実施すべき設計内容を設計要素と示し，その設計要素間の関連を明示している．

3.3　詳細設計仕様のメタモデル

詳細設計仕様のメタモデルを**図 2** に示す． 詳細設計仕様のメタモデルは，プレゼンテーション要素，業務ロジック要素，データ要素，コントロールの 4 個の構成要素群から構成されている．以下に，各構成要素群を解説する．

- プレゼンテーション要素を中心にした構成要素群
  プレゼンテーション要素を中心にした構成要素群は，画面や帳票などのグラフィカルなインタフェースと，コマンドラインなどのキャラクタベースのインタフェースを表している．画面／帳票は，アクションクラス，画面／帳票オブジェクト，マッピング，レイアウトの要素に分類できる．アクションは，プログラムの動作を示し，画面／帳票とオブジェクトの活性状態を示す関連を持つ．レイアウトは，画面／帳票オブジェクトの配置を表す．マッピングは，データ要素との対応付けを示す項目マッピングと，アクションとの対応付けを示すアクションマッピングに分類される．
- 業務ロジック要素を中心にした構成要素群

契約と複合業務に分類される業務ロジック要素は，業務ロジックの内容を示す．契約は，業務ロジック内の条件を記述する要素である条件と，業務ロジック内容を記述する要素である契約内容で構成される．条件は，可変条件と不変条件の要素に分類され，条件の変化の有無を場合分けして記述する．可変条件は，事前条件と事後条件に分類され，条件の内容を示す．

- データ要素
  データ要素の構成は，基本設計仕様のデータ要素と同じ構成となる．そのため，詳細設計仕様のメタモデルでのデータ要素の構成内容は，ここでは解説を省略する．
- コントロール
  コントロールは，プレゼンテーション要素，業務ロジック，データ要素の制御と情報伝達を行う．データ操作に示す内容は，データ要素に対する操作を示す．データ操作要素は，業務ロジック要素を具体化するものであり，不変条件クラスに記述する内容に沿ったものである．

以上の構成要素は，ソフトウェアの実装構造に沿ったものとなっている．

3.4　ソフトウェアレビューのための関連モデル

基本設計仕様と詳細設計仕様のメタモデルを水平方向のモデルとしたとき，仕様の引き継ぎを確認するためのソフトウェアレビューの視点を表すモデルは，垂直方向の関連モデルとして表せる．この関連モデルは，基本設計仕様と詳細設計仕様のメタモデルの構成要素の間の依存関係を示している．

ソフトウェアレビューのための関連モデルを**図 3** に示す． 図 3 は，中央の横点線より上が基本設計仕様のメタモデル，下を詳細設計仕様のメタモデルである．各設計仕様のメタモデルの構成要素における，メタモデル間での対応を縦方向の関連線で示している．この対応は，基本設計仕様に記述されている設計仕様の構成要素が，詳細設計仕様の何処の構成要素として記述されているかを示す．例えば，基本設計仕様のメタモデルの制約に記述する内容は，詳細設計仕様のメタモデルの契約に詳細化した仕様として記述される．基本設計仕様と詳細設計仕様を対応づけることにより，各設計仕様を構成する設計要素間の対応付けが網羅的に把握でき，設計仕様の不整合を検出するためのレビューの観点が明確となる．

4.　ソフトウェアレビュー実験による設計仕様のメタモデルの評価

4.1　ソフトウェアレビュー方法の提案

複数の設計仕様書の引き継ぎが正当に行われていることをレビューすることは，すなわち，設計仕様書を構成する要素の間の依存関係を調査していることに他ならない．そこで，基本および詳細設計仕様のメタモデルと関連モデルに基づいたソフトウェアレビュー方法を，Metamodel-Based Reading(以下，MMBR と略す) として提案する．

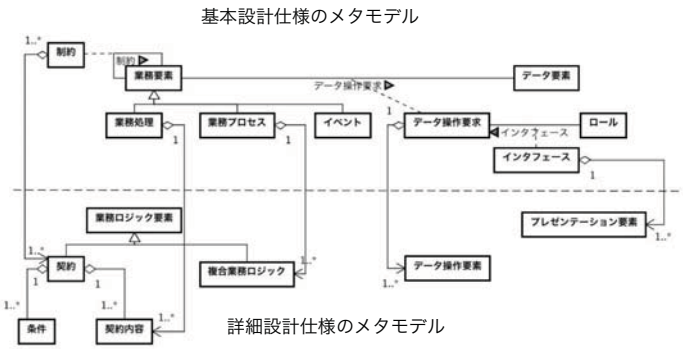

図 3　関連モデル
Fig. 3　Relationship model

このレビュー方法は，まず設計仕様のメタモデルの中からレビュー観点（perspective）となる設計仕様の構成要素を選択する．この観点がレビューの起点となる．次に，レビューの起点から，設計仕様のメタモデルに示された関連を用いて，設計仕様の構成要素間の対応付けが漏れなく定義されているか否かレビューする．ソフトウェアレビューの観点に関連している箇所に注目し，ソフトウェアレビュー対象を設定することにより，レビューの範囲を有限に設定できる．

MMBR には，基本または詳細設計仕様のメタモデルに基づく，一つの設計工程で作成される設計仕様書間のレビューと，基本と詳細設計仕様のメタモデル間の関連を示す関連モデルに基づく，設計工程を跨がった設計仕様書間のレビューがある．それぞれのレビューは，レビューアが設計仕様のメタモデルから導出したチェックリストに基づいてレビューを実施する．以下に MMBR のチェックリストの作成方法を示す．**図 4** はチェックリストの作成例である．チェックリスト例は，基本設計仕様書の一種である業務機能定義書と画面定義書の間でのレビューと，基本設計仕様書の画面定義書と，詳細設計仕様書の一種であるファンクション層（F 層）定義書の間でのレビューを実施するためのチェックリストの一部である．業務機能定義書は，開発するソフトウェアの 1 つの業務機能の概要を記述したものであり，業務機能の目的や機能説明を 5W1H 形式で記述する．画面定義書は，画面レイアウトや画

項番	レビュー観点	レビュー対象	関連元	関連先	質問
1-1	業務処理	メタモデルの構成要素	基本設計仕様.業務処理.処理内容	基本設計仕様.業務処理.処理内容	入庫・仕入入力が実施できるか
1-2		設計仕様書	基本設計仕様書.業務機能定義書	基本設計仕様書.画面定義書	
1-3		設計項目	何をする	イベント定義．処理内容	
2-1		メタモデルの構成要素	基本設計仕様.業務処理.処理内容	詳細設計仕様.契約.契約内容	必須チェック内容は整合しているか.
2-2		設計仕様書	基本設計仕様書.画面定義書	詳細設計仕様書.F層定義書	
2-3		設計項目	イベント定義．処理内容．登録	イベント詳細（登録）	

図 4　MMBR のチェックリスト例
Fig. 4　The example of checklist for MMBR

面項目の属性を記述する．ファンクション層（F 層）定義書は，メソッドとして実装する処理内容を記述する．

(1)　設計仕様のメタモデルの構成要素の 1 つを，ソフトウェアレビュー観点として選択する．図 4 では，レビュー観点と表題が付いている列が，設計仕様のメタモデルの選択した構成要素を示している．例として，基本設計仕様のメタモデルの構成要素である「業務処理」を選択している．

(2)　レビューの実施対象とする設計仕様のメタモデルの選択した構成要素のサブクラス，設計仕様のメタモデルのインスタンスである設計仕様書，設計仕様書の構成要素である設計項目を対で選択し，関連元と関連先の対応付けを設定する．

同一の設計工程内のレビューの実施対象の設定は，基本設計仕様または詳細設計仕様のメタモデルを用いて，対応付けを確認する．図 4 では項番 1-1 から 1-3 が，基本設計仕様のメタモデルの業務処理クラスのサブクラスである「処理内容」クラスが，基本設計仕様書である「業務機能定義書」の設計項目「何をする」と，基本設計仕様書である「画面定義書」の設計項目「イベント定義」のサブ項目「処理内容」に対応している．

設計工程を跨がった設計仕様書間のレビューの実施対象の設定は，設計仕様のメタモデル間の関連モデルを用いて，対応付けを確認する．図 4 では，項番 2-1 から 2-3 が，基本設計仕様のメタモデルの「業務処理」クラスの「処理内容」サブクラスと，詳細設計仕様のメタモデルの「契約」クラスの「契約内容」サブクラスに対応している．さらに，基本設計仕様書である「画面定義書」の設計項目「イベント定義」のサブ項目「処理内容」が，詳細設計仕様書である「F 層定義書」の設計項目「イベント詳細（登録）」に対応している．

(3)　関連元の設計項目から関連先の設計項目へ，ソフトウェア固有の設計内容が正しく引き継がれているかを確認する内容を，チェックリストの質問として設定する．

4.2　実験方法

設計仕様のメタモデルと関連モデルがソフトウェアレビューの精度を向上させることを確認するため，ソフトウェアレビュー実験を実施し，提案するソフトウェアレビュー方法の有効性を評価した．

4.2.1　評価方法

ソフトウェアレビュー観点である設計仕様のメタモデルと関連モデルの有無が，ソフトウェアレビューの精度向上に影響するかの確認を目的とし，MMBR と開発の現場で多く実施されている Ad Hoc Reading(以下，AHR と略す) の 2 種類のリーディング方法を用いたレビュー実験を行い，両者で検出された欠陥の状況を定量的，定性的に評価した．

さらに，設計仕様のメタモデルと関連モデルがどのように欠陥検出へ有効となるかの確認を目的とし，レビュー対象とした設計仕様書に予め欠陥を埋込み，それらが正確に検出できたか，全体のうちどの程度検出できたかを評価した．

4.2.2　欠陥の分類

欠陥は，ソフトウェアの使用に対する利用者への影響の重要度として 2 種類に定義した．まず，高重要度の欠陥とは，利用者にとって，機能として存在していないため利用者が該当の機能を使用できない状態，または機能として存在しているが，利用者の期待する結果にならない状態を引き起こすものとした．次に，低重要度の欠陥とは，誤字等の機能としての欠陥ではないものとした．たとえば，メッセージの文言の誤字などである．

4.2.3　有効性の定義

MMBR の有効性が高ければ，設計仕様のメタモデルと関連モデルはソフトウェアレビューの観点として有効である．ソフトウェアレビュー方法の有効性を次のように定義した．

- 異なったソフトウェアレビュー方法を比較して，誤って欠陥として指摘した件数が少なく，検出された欠陥数が多いソフトウェアレビュー手法は，他方のソフトウェアレビュー方法より，有効性が高い．
- 一定のレビュー時間内で，重要度が高い欠陥が優先されて検出されるソフトウェアレビュー方法は，重要度が低い欠陥が優先されて検出されるソフトウェアレビュー方法より，有効性が高い．

以上の定義に基づき設定した，有効性を表すメトリクスは，ソフトウェアレビューでの指摘総数に対する有効な欠陥数，重要度別の欠陥数，埋込み欠陥の検出数とした．

4.2.4　設計仕様書とチェックリスト

実験で使用した基本設計仕様書 4 種と詳細設計仕様書 1 種は，本実験に向けて新規に作成した．設計仕様を策定した題材は，「酒屋の在庫問題」[8] 例題とした．この例題を基に作成された酒屋受注支援システムの要求仕様書の「在庫追加」機能のユースケースに基づき，基本設計仕様と詳細設計仕様を策定した．本実験では，異なる種類の基本設計仕様書間，および基本設計仕様書と詳細設計仕様書間での設計仕様の不整合として，基本設計仕様書には高重要度の欠陥を 5 件，低重要度の欠陥を 7 件，詳細設計仕様書には高重要度の欠陥を 8 件，低重要度の欠陥を 2 件，予め埋め込んだ．

MMBR のチェックリストは，ソフトウェアレビュー観点として，業務処理，制約，イベントの 3 観点に基づいた 42 件とした．その内訳は，基本設計仕様書間の不整合をチェックするものが 13 件，基本設計仕様書と詳細設計仕様書の不整合をチェックするものが 29 件である．作成に要した工数は，0.6 人日であった．

4.2.5　レビューア，レビューの進め方

レビューアとして 6 名に参加にしてもらった．各参加者は，ソフトウェア/システム開発の経験がある社会人であり，ソフトウェアレビュー方法に基礎的な知識を有していた．本実験で題材とした在庫管理のドメイン知識は有していなかった．レビューの技術と経験の差を平準化するため，ソフトウェア/システム開発経験の年数を考慮し，ソフトウェアレビューを行うチームを 2 つに分けた．一方を MMBR を使用するグループ，もう一方を AHR を使用するグループとした．両グループに対し，酒屋受注支援システムの概要，実験の目的，レビュー方法，制限事項，レビューにおける留意点について同じ内容を説明し，30 分間の設計仕様書の事前読み込みの時間を設定した．MMBR を使用するグループに対しては，MMBR で使用するメタモデルと，事前に作成済みのチェックリストを渡し，その内容と使用方法を説明し，チェックリストに沿ったソフトウェアレビューを実施することとした．AHR を使用するグループに対しては，チェックリストはなく，成果物の読み込み順番と方法は任意とした．レビュー時間は 1 時間と設定した．実験後，欠陥指摘の内容を精査し，分析のためのメトリクスの収集を行った．

4.3　評　　価

MMBR と AHR の各グループでの，メトリクスの結果を**表 1** に示す．誤って欠陥とした指摘とは，欠陥の分類から除外した文法間違い・記述間違い，設計仕様上の制限事項に該当するものである． 評価の結果，MMBR と AHR の有効な欠陥検出数は同程度であったが，MMBR は AHR より重要度の高い欠陥を多く検出していた．また，基本設計仕様書と詳細設計仕様書の埋込み欠陥については，AHR は MMBR より検出された欠陥数が多いが，重要度の低い欠陥に集中していた．以上のことより，MMBR の方が，チェックリストに示されたとおりに，重要度の高い欠陥を優先して検出していたことが明らかになった．

表 1　欠陥検出の結果
Table 1　The result of defect detection

分類	MMBR（件）	AHR（件）
ソフトウェアレビューでの指摘総数	26	37
誤って欠陥とした指摘を除いた，有効な欠陥検出数	24	26
高重要度の有効な欠陥数	16/24	2/26
低重要度の有効な欠陥数	8/24	24/26
基本設計仕様書間の不整合である欠陥数	1/24	17/26
基本設計仕様書と詳細設計仕様書間の不整合である欠陥数	13/24	0/26
設計仕様書の不整合以外の欠陥数	10/24	9/26
高重要度の埋込み欠陥 13 件中で，検出した欠陥数	2/13	0/13
低重要度の埋込み欠陥 9 件中で，検出した欠陥数	1/9	7/9

限られたレビュー時間で設計仕様書の品質を高めるためには，重要度の高い欠陥を優先して検出する必要がある．ソフトウェアレビューにより設計仕様書の品質が向上したかを評価するためには，単に欠陥検出数の大小を比較するだけでは不完全である．検出した欠陥の重要度に着目する必要がある．MMBR は設計仕様書間での設計仕様の引き継ぎ漏れや処理内容の不整合である，記述内容の理解が必要な差異を，高重要度の欠陥として低重要度の欠陥より 8 件多く検出していた．また，埋込み欠陥の総数 22 件に対して，MMBR は 3 件，AHR は 7 件を検出した．AHR は，検出した欠陥の大半が，設計仕様の属性情報や設計項目の件数の差異となっており，検出しやすく，重要度の低い欠陥が検出されていた．検出した欠陥の内容としては，設計仕様書の不整合以外の欠陥は，MMBR と AHR 共に設計仕様自体の誤りであった．さらに MMBR は，埋込み欠陥ではない，我々が認識していなかった重要度の高い欠陥を新たに 14 件検出していた．

MMBR は設計仕様のメタモデルと関連モデルをソフトウェアレビューの観点としたチェックリストを使用したため，レビューアが処理内容に踏み込んで確認することができ，設計仕様書間での設計仕様の引き継ぎ漏れや処理内容の不整合を検出できた．

また，MMBR は基本設計仕様書と詳細設計仕様書間の不整合を検出したが，AHR は検出しなかった．レビューのプロセスを観測した結果，MMBR は制限時間内にチェックリストで指定されたレビューを完了したが，AHR は基本設計仕様書のみのレビューが完了しただけであった．設計仕様のメタモデルと関連モデルをソフトウェアレビューの観点とした

チェックリストにより，レビューの効率を向上できた．

5. 考　　察

設計仕様のメタモデルと関連モデルより，ソフトウェアレビューのためのチェックリストを導出でき，そのチェックリストを使用したMMBRの有効性が確認できた．設計仕様のメタモデルと関連モデルは，ソフトウェアレビューの観点が，詳細化された項目で構成され，階層構造を持った関係があることを示している．ソフトウェアレビューの観点の詳細項目間の関係と，設計仕様書の構成要素の対比により，設計仕様書間での設計仕様の分割，または詳細化での不整合の検出が可能となった．したがって，提案した設計仕様のメタモデルと関連モデルは，ソフトウェアレビューの観点として有効であり，ソフトウェアレビューの精度の向上を図ることができた．

ただし，設計仕様のメタモデル，関連モデル，および設計仕様書の設計項目の対応付けに基づき，チェック項目を設定する方法は，チェックリストの作成者に依存しており，明確には定義されていない．設計仕様書から設計仕様のメタモデルからレビューすべき箇所とレビューの順番を，チェックリストとして書き出すプロセスが不明確である．このプロセスを確立できれば，自動化が可能となり，レビューの準備の生産性を高くし，レビューの精度を向上させることが可能となる．

チェックリストの使用により，レビューのスキルが低いレビューアでも，レビューの実施が可能になる．しかし，ソフトウェアレビューの実施内容は，レビューアがチェックリストの記述内容を解釈するスキルに依存する可能性が高い．チェックリストが設定されていたにも関わらず，実験でMMBRの埋込み欠陥の検出が少なかったのは，チェックリストへのレビューアの理解が低かったことが原因と考える．チェックリストの記述内容に対し，レビューアのスキルへの依存を低くする表現や形式を，検討する必要がある．

重要度が低い欠陥もソフトウェアのリリースまでには検出し，除去すべきである．したがって，設計仕様書をレビューするときは，重要度の低い欠陥はアドホックなレビューで検出し，重要度が高い欠陥はMMBRで検出するといった，目的によってレビュー方法を切り替える方法も効果的である．

提案した設計仕様のメタモデルは，業務アプリケーションソフトウェアにドメインを限定している．業務アプリケーションソフトウェア以外の分野（例：組み込みシステム）へのMMBRの適用方法が明らかではない．開発の現場に，提案した設計仕様のメタモデルを適用し，精緻化および新たな設計仕様のメタモデルの検討が必要である．

6. ま と め

本稿では，ソフトウェアの設計品質を向上させるためのソフトウェアレビュー方法を提案するために，業務アプリケーションソフトウェアの設計仕様の構造と構成要素間の関連を調査し，設計仕様として定義すべき事項と構造を示す，基本および詳細設計仕様のメタモデルを示した．その設計仕様のメタモデルと，基本および詳細設計仕様のメタモデル間の関連を表す関連モデルをレビューの観点としたソフトウェアレビュー方法を，Metamodel-Based Reading（MMBR）として提案した．実験により，MMBR法を用いてソフトウェアレビューを行うことで，重要な欠陥をソフトウェアレビューで見落とすことを防止できることを確認した．これより，提案した設計仕様のメタモデルがソフトウェアレビューの観点として有効性が高いことを確認できた．今後は，提案した設計仕様のメタモデルを開発の現場へ適用し，設計仕様のメタモデルの精緻化を図ることを検討する．

参 考 文 献

1) Boehm, B. and Basili, V.R.: Software Defect Reduction Top 10 List, *Computer*, Vol.34, pp.135–137 (2001).
2) Shull, F., Rus, I. and Basili, V.: How Perspective-Based Reading Can Improve Requirements Inspections, *Computer*, Vol.33, pp.73–79 (2000).
3) Ramesh, B. and Jarke, M.: Toward Reference Models for Requirements Traceability, *IEEE Transactions on Software Engineering*, Vol.27, pp.58–93 (2001).
4) 大橋恭子，栗原英俊，田中ユカ，野津昌弘，山本里枝子：ビジネスアプリケーション開発における追跡可能性構築の取り組み，ソフトウェアエンジニアリング最前線2010 — ソフトウェアエンジニアリングシンポジウム2010（SES 2010）予稿集，pp.155–160 (2010).
5) 森崎修司，　野中誠，込山俊博，清田辰巳，細川宣啓，　永田敦：ソフトウェアレビュー/ソフトウェアインスペクションと欠陥予防の現在，情報処理，　Vol.50, No.5, pp.375–417 (2009).
6) Thelin, T., Runeson, P. and Wohlin, C.: Prioritized Use Cases as a Vehicle for Software Inspections, *IEEE Software*, Vol.20, pp.30–33 (2003).
7) Travassos, G., Shull, F., Fredericks, M. and Basili, V.R.: Detecting defects in object-oriented designs: using reading techniques to increase software quality, *OOPSLA '99: Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pp.47–56 (1999).
8) 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).