社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－SE－155 (23)
2007／3／23

UMLモデリングにGUI設計の取り込みを考慮した設計支援ツールの提案

銅島 康　　上田 賀一

茨城大学

本研究では，短期化が進むソフトウェア開発サイクルに対応するために，アジャイル開発において全側面の仕様を同時に少しずつ設計することに焦点を当てて議論する．現在のソフトウェア開発では，UML設計とGUI設計はほぼ同じ工程で行われるにもかかわらず，最初からUML設計にGUI設計を取り入れた設計支援ツールは見当たらない．これではアジャイル開発において，細かい繰り返しによる修正が円滑に行われず，利点である開発期間の短縮という効果が失われてしまう．本研究では，MVCモデルパラダイムに基づき，UML設計でModelを，UI設計でViewを，UML設計とUI設計の関連付けによってControllerを記述可能にすることで，UML設計とUI設計を同時に支援するツールの開発を試みた．

A Design Support Tool which Takes GUI Design into Consideration in UML Modeling

Yasushi Doujima and Yoshikazu Ueda

Ibaraki University

In this study, it is discussed to design the specification concurrently on all points of view in the agile development. The UML modeling and the UI design are done in the same process for the design of software development. Nevertheless, there is no design support tool which adopted UI design for the UML modeling. Therefore, in agile development, the effect of shortening at the development period is lost. This study describes the follows. In the base of MVC model paradigm, the model part and the view part is described by UML modeling and UI design respectively. And the controller part is described by relating the UML modeling and the UI design.

1 はじめに

近年のソフトウェア開発では，小規模でかつ顧客も積極的に参加するようなプロジェクトにおいて，アジャイル開発を採用する場合が多い．アジャイル開発を採用することで，ウォータフォールモデルなどの従来の手法より迅速に開発を進められるという利点が得られる．そこで本研究では，短期化が進むソフトウェア開発サイクルに対応するために，アジャイル開発における全側面の仕様を同時に少しずつ設計し，かつMVCモデルに基づいて設計することに焦点を当てて議論する．

現在のソフトウェア開発では，設計にはUMLを用いるのが一般的である．設計にはUI設計を含み，UML設計とUI設計はほぼ同じ工程で行われるにもかかわらず，最初からUML設計にUI設計を取り入れた設計を支援するようなツールは見当たらない．これではアジャイル開発において，細かい繰り返しによる修正が円滑に行われず，利点である開発期間の短縮という効果が失われてしまう [1].

本研究では，MVC モデルパラダイムに基づき，UML 設計で Model 部分を，GUI 設計で View 部分を，UML 設計と GUI 設計の関連付けによって Controller 部分を記述可能にすることで，UML 設計と UI 設計を同時に支援するツールの開発を試みた．

2 関連研究

本研究を進める上でまず始めに，UML 設計と UI 設計を並行的に行え，かつ MVC 分割を明示的に行える設計支援ツールの有無を確認するため，いくつかの関連研究を調査する．

2.1 設計支援ツール共通の欠点

UML 設計支援ツールが何を行えるかを表 1 にまとめる．表 1 は，各製品が UML（構造図，振舞い図，実装図）の記述を行えるかどうか，GUI 設計を行えるかどうかを示している．表 1 では，UML 設計支援ツールが UI 設計も支援しているかどうかを確認するため，入力 GUI 設計の列に着目する．そこから，どの製品においても GUI 設計を支援していないことが分かる．

また，UI 設計支援ツールが何を行えるかを表 2 にまとめる．表 2 の見方は表 1 と同様である．表 2 では，UI 設計支援ツールが UML 設計も支援しているかどうかを確認するため，入力 構造図，振舞い図，実装図の列に着目する．そこから，どの製品においてもほとんど UML 設計を支援していないことが分かる．

以上のことから，既存の設計支援ツールにおける共通の欠点とは，UML 設計と UI 設計の両方を支援できていないという点であることが分かる．

Xcode において補足しておくと，クラス図と GUI 画面の連携はある程度，行えている．しかし，Xcode ではクラス図以外の構造図に焦点を当てておらず，またシーケンス図などの振舞い図にも焦点を当てていない．そのため，振舞い図と GUI 画面の連携が行えない．これは Xcode が実装に焦点を当てており，設計には焦点を当てていないためだと考えられる．さらに，クラス図と GUI 画面の整合性はとれておらず，一方の図を修正しても，その情報がもう一方の図へ反映されない．以上のことから，Xcode も UML 設計と UI 設計の連携は不可能であるといえる．

類似研究が少ない理由は，UI 設計支援ツールとビジュアルプログラミング言語によって容易に UI を作成し，その UI 設計から自動でコード生成を行いできるだけコーディングの手間を省くという目的を突き詰めていったためであると考えられる．

2.2 UMLi

相互通信システムの開発では UI を考慮する必要があるにもかかわらず，既存の UML を用いて UI 設計を行う場合，UI に適した形で UML 記述できない．UML だと，UI に必要な情報（レイアウトなど）が失われてしまい，分かりにくく，全体像をイメージしにくい．そこで UMLi は，UI プロトコルのモデリングのために，図の表記法を提供する [13]．

UMLi は既存の UML メタモデルを拡張することで，UI をより適した形で UML のように設計できる．

しかし，UMLi は CUI やネットワークインタフェースに主眼を置いている．GUI 独特のレイアウト，look & feel の記述には主眼を置いていない．

3 提案手法

本章では，提案手法における MVC モデルの捉え方を示し，提案手法の全体像，既存モデルの拡張，および制約条件を示す．

3.1 MVC モデルの捉え方

既存の MVC モデルの捉え方は，図 1 の上部に示す形になっている [14] [15]．本提案手法では，この既存の捉え方を変更し，UML 設計で Model 部分を，UI 設計で View 部分を，UML 設計と UI 設計の関連付けによって Controller 部分を記述する．これにより，MVC モデルの Model，View，Controller を明示的に分割して設計作業ができる．

表 1: UML 設計支援ツール入力可能情報一覧

製品名	入力			
	構造図	振舞い図	実装図	GUI 設計
Rational Rose Enterprise Edition バージョン 2001A [2]	○	○	○	×
Konesa Client Evaluation 1.1c [3]	○	○	○	×
Pattern Weaver Standard Edition 1.1 [4]	○	○	○	×
WithClass 7.0J Enterprise Edition [5]	○	○	○	×
Describe Developer for Sun ONE Studio 4 [6]	○	○	○	×
Together Control Center 6.0.1 [7]	○	○	○	×
Jude community 3.1.1 [8]	○	○	○	×
Enterprise Architect 3.51 Professional Edition [9]	○	○	○	×
Poseidon Community Edition 1.6.0_01 [10]	○	○	○	×
EclipseUML freeEdition 2.1.0.20050927 [11]	○	○	○	×
野村らのツール [12]	△	△	×	×

表 2: UI 設計支援ツール入力可能情報一覧

製品名	入力			
	構造図	振舞い図	実装図	GUI 設計
Expression Interactive Designer May 2006 CTP	×	×	×	○
Xcode 2.3	△	×	×	○
Visual C++ 2005 Express Edition 8.0.50727.42	×	×	×	○
Visual Editor 1.1.0.1	×	×	×	○
IBM Reflexive User Interface Builder 1.0.1	×	×	×	○

3.2 提案手法全体像

本提案手法の全体像を図 2 に示す．UML において，構造図ではクラス図が，振舞い図ではシーケンス図がそれぞれ重要と考えられる．それは，クラス図ではシステム内のオブジェクトタイプと，それらの間に存在する各種の静的な関係を記述でき，これらの情報を実装で容易に利用できるからである．また，シーケンス図ではユースケースに関するいくつかのオブジェクトの典型例と，それらのオブジェクト間でやり取りされるメッセージを順序付けして示せるからである．そのため，本研究ではこの 2 つを UML 設計で扱う．また，UI 設計もアジャイル開発対象では GUI が一般的であるため，GUI 設計を UI 設計で扱う．

3.3 UML2.1 と UMLi メタモデルの拡張

本提案手法では UML2.1 を扱うが，UMLi は UML1 を拡張したメタモデルであるため，まず UMLi を UML2.1 に対応できるよう拡張する [13] [16]．

UMLi は UML 設計と UI 設計の関連付けを記述できないため，次に関連付けを記述できるよう拡張したパッケージ図を図 3 に示す．2 重腺で囲まれたメタクラスが拡張・修正した部分である．

3.4 制約条件

本提案手法を実現するために必要な制約条件の一部を示す．ただし，抽象ウィジェット定義は省略する．

UMLi は CUI やネットワークインタフェースに主眼を置き，GUI 独特のレイアウト，look & feel の記述には主眼を置いていない．そのため，OCL 定義によって GUI 設計に適した形へ UMLi を拡張する．また，UML 設計と UI 設計の関連付けのために拡張したメタモデルにおいて，OCL 定義により制約として関連付けを具体的に記述する．以降では，UML 設計で扱う最小要素を UML 構成要素，GUI 設計で扱う最小要素を抽象ウィジェットと呼ぶ．

3.4.1 関連名定義

UML 構成要素と抽象ウィジェット間の関連付けにおいて，関連付けをいくつかの種類に分類し，関連付けの各種類において関連名を定義する．それ

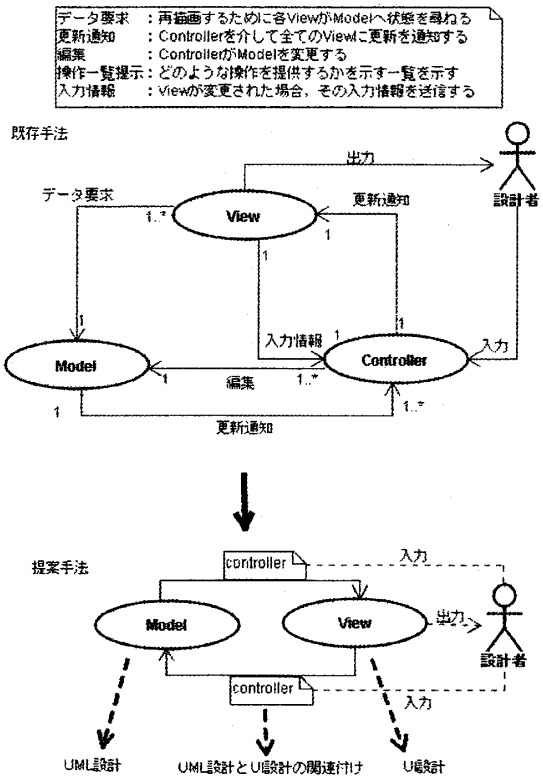

図 1: 既存 MVC モデルの修正

らの定義の一部をを表 3 に示す.

表 3: 関連名定義

関連名	簡易説明
専用: exclusive	あるクラスのみが相手クラスを識別し，アクセスできることを明示化する．ただし，一方のクラスは UML 設計中のクラス，もう一方のクラスは GUI 設計中のクラス（抽象ウィジェット）とする．
通知: advice	何らかの内部処理によって変更したモデルの保持する属性の値をビューへ反映する．
観測: observer	クラスの所有する属性やメソッドの名前，値を出力する．
行動: action	メソッド呼び出しを抽象ウィジェットから UML 構成要素へ行う．

3.4.2 関連定義

UML 構成要素と抽象ウィジェットの関連付けをその方向性において次の 4 パターンに分ける.

A1 : UML 構成要素 ———> 抽象ウィジェット
UML 構成要素から抽象ウィジェットへ制約関係.

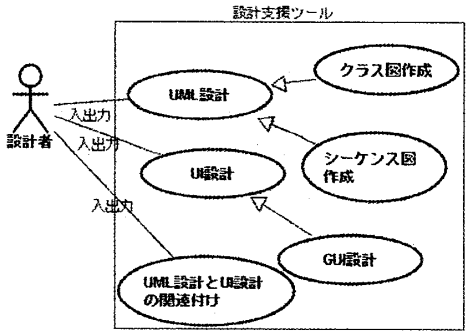

図 2: 提案手法全体像

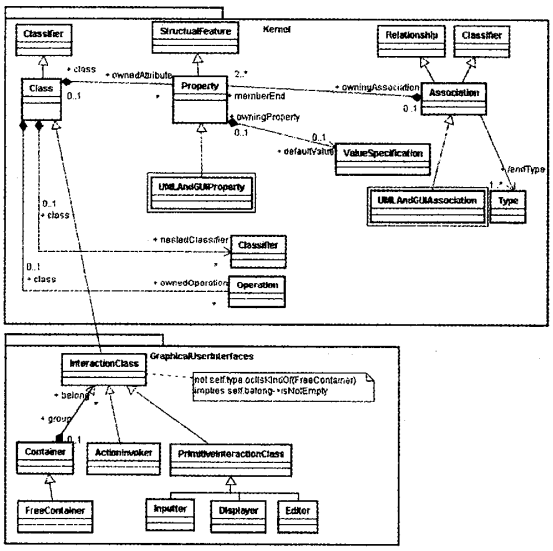

図 3: UML2.1 と UMLi メタモデルの拡張（Kernel と GUI パッケージ）

具体的には，UML 構成要素から抽象ウィジェットへのメッセージ送信など.［UML 構成要素］→［仲介クラス］→［抽象ウィジェット］の場合も含む.

A2 : UML 構成要素 <——— 抽象ウィジェット
抽象ウィジェットから UML 構成要素へ制約関係. 具体的には，抽象ウィジェットから UML 構成要素へのメッセージ送信など.

A3 : UML 構成要素 ———— 抽象ウィジェット
方向性のない制約関係．具体的には，UML 構成要素と抽象ウィジェットが同一など.

A4 : UML 構成要素 <———> 抽象ウィジェット
相互に方向性のある制約関係．具体的には，UML 構成要素の保持する属性を変更すると，それに伴い抽象ウィジェットの保持する属性も変わり，逆に抽象ウィジェットの保持する属性を変更すると，そ

れに伴いUML構成要素の保持する属性も変わる，など．
以上の関連A1～A4の一部を表4で定義する．

表4: 関連方向性定義

関連	OCL定義
A1型	UMLAndGUIAssociationと関連するUMLAndGUIPropertyの保持する誘導可能端は，方向A1のようになる． self.navigableOwnedEnd.name() <> 'UML構成要素' and self.navigableOwnedEnd.name() = '抽象ウィジェット' 「UML構成要素」,「抽象ウィジェット」にはそれぞれClass，Buttonなどの具体的な名前を記述する．
A3型	UMLAndGUIAssociationと関連するUMLAndGUIPropertyの保持する誘導可能端は，方向A3のようになる． self.navigableOwnedEnd.name() <> 'UML構成要素' and self.navigableOwnedEnd.name() <> '抽象ウィジェット'

3.4.3 クラス図のUML構成要素と抽象ウィジェットの関連付け

クラス図におけるUML構成要素と抽象ウィジェットの関連付けの定義の一部を表5に示す．シーケンス図の場合もクラス図と同様に考えればよい．

4 本支援ツールの設計

本提案手法の支援ツールのGUI設計を示し，ArgoUMLベースの実装について検討する．

4.1 本支援ツールのGUI設計

本支援ツールのGUI設計を以下に示す．

- クラス図記述機能
  図4の中央にクラス図記述ウィンドウが配置される．ツールバー内のボタンを選択し，次にウィンドウ内をクリックすることでクラスや関連を記述できる．
- シーケンス図記述機能
  図5の左側にシーケンス図記述ウィンドウが配置される．ツールバー内のボタンを選択し，次にウィンドウ内をクリックすることでライフラインやメッセージを記述できる．

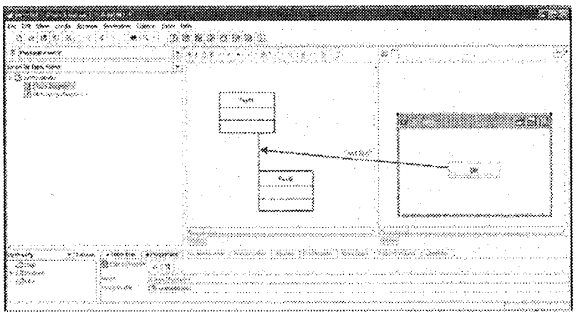

図4: GUI設計 メインウィンドウ

- GUI設計記述機能
  図4の右側にGUI設計記述ウィンドウが配置される．ツールバー内のボタンを選択し，次にウィンドウ内をクリックすることでFrameやButtonを記述できる．
- UML-GUI関連付け機能
  図4の“クラス図の関連”と“GUI設計のButton”の間に引かれている線が“UML-GUIの関連”である．GUI設計記述ウィンドウが保持するツールバー内の“UML-GUIの関連ボタン”を選択し，次に“GUI設計のウィジェット”から“UML設計の構成要素”へドラッグ&ドロップすることで“UML-GUIの関連”を記述できる．

UML設計とGUI設計の関連付けは以下の10種類に分類される．

- 補足 (addition)
- 通知 (advice) （図5参照）
- 連鎖 (chain)
- 編集 (edit)
- 専用 (exclusive)
- 観測 (observer) （図6参照）
- 唯一 (singleton)
- 同期 (synchronous)
- 行動 (action) （図4参照）
- 識別 (identifier)

表 5: UML 構成要素と抽象ウィジェットの関連付け（クラス図）

構成要素名	抽象ウィジェット	関連	関連名	関連付けが意義を持つか否かについての判断理由，関連の型についての判断理由
クラス	Frame	A1 型	連鎖	[関連付けする理由] 関連付けしたクラスのインスタンス生成時に，ウィンドウも同様に生成できるようにするため． [関連の型についての理由] クラスのコンストラクタ呼び出し時に，メソッド呼び出しがクラスから Frame へ行えるようにするため．
			唯一	[関連付けする理由] 生成できるウィンドウを一つのみに限定する制約をかけられるようにするため．
		A2 型	連鎖	[関連の型についての理由] Frame のコンストラクタ呼び出し時に，メソッド呼び出しが Frame からクラスへ行えるようにするため．
			唯一	[関連付けする理由] 生成できるウィンドウを一つのみに限定する制約をかけられるようにするため．
関連	Button	A2 型	行動	[関連付けする理由] 誘導可能性がある場合，一方のクラスがもう一方のクラスへメッセージを送信できる．そのメッセージと Button を関連付けることで，メッセージの送信タイミングを抽象ウィジェットで管理することを明示化できるようにするため． [関連の型についての理由] メソッド呼び出しが Button から関連先クラスへ行えるようにするため． [誘導可能性] 一方のクラスがもう一方のクラスを参照できるかどうか（メッセージを送信できるかどうか）を表す．

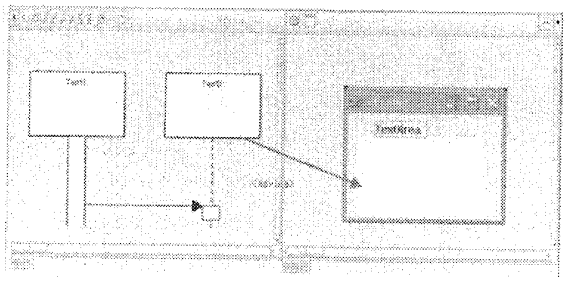

図 5: GUI 設計 通知関連

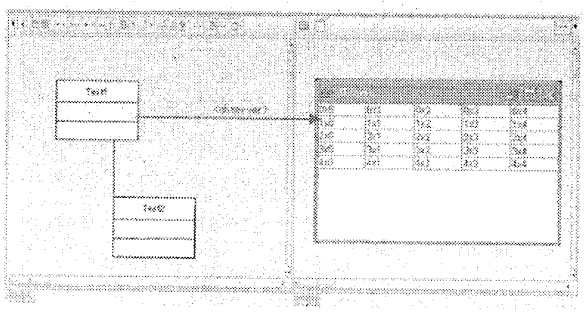

図 6: GUI 設計 観測関連

4.2 ArgoUML ベース

ArgoUML はオープンソースの UML モデリングツールであり，UML1.4 に準拠している．また，Java プラットホームで実行でき，10ヶ国語をサポートする．ArgoUML は BSD ライセンスに準拠し，拡張方法のマニュアル群も充実しているため利用しやすい．UMLi も，ArgoUML を拡張して ARGOi という支援ツールを実装している．そのため，本ツールも ArgoUML ベースの実装を行う [17].

5 評価

ビデオレンタルシステムを例に取り上げ，本ツールを用いたソフトウェア設計の流れに沿いながら評価目的，手順，考察を示す．

評価目的は，並行的に作業できるか否か，MVC の分割を明示化できるか否かである．

並行作業の評価目的を具体的に述べると，本ツールを用いて UML 設計と GUI 設計を並行的に行うことで，設計図間の整合性と品質が向上したかを評価する．また，設計を円滑に進められ時間短縮できたかも評価する．

次に MVC 分割明示化の評価目的を具体的に述

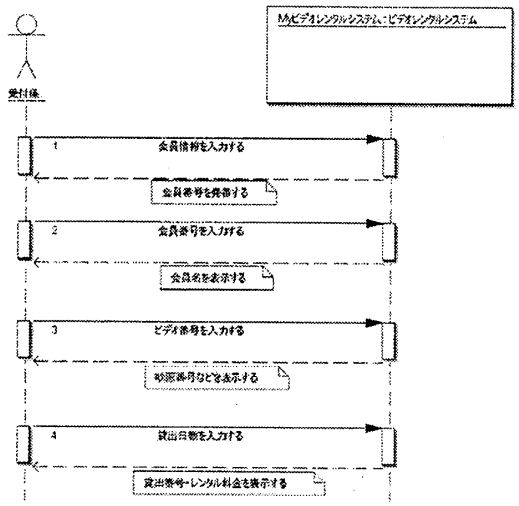

図 8: ビデオレンタルシステム シーケンス図

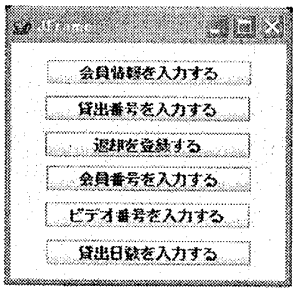

図 9: ビデオレンタルシステム GUI 設計

べると，本ツールを用いて MVC を明示的に分割して設計することで，Model，View，Controller の分類分けを直感的に理解でき，各設計図の理解容易性が向上したかを評価する．また，設計の再利用性が向上したかも評価する．

評価のために，本ツールを用いてビデオレンタルシステムを設計する．クラス図の記述，シーケンス図の記述，GUI 設計の記述，UML 設計と GUI 設計の関連付けを記述の 4 つの作業を並行的に行う．本ツールに入力するクラス図を図 7 に，シーケンス図を図 8 に，GUI 設計を図 9 に示す．

最後に，本ツールの評価を考察する．

並行作業においては，本ツールを用いることで，アジャイル開発を意識しなくても，反復開発（修正の繰り返し）を行い，品質を高めることができた．それは，UML 設計と GUI 設計を並行的に行えるようにすることで，2 つの設計の矛盾点に気が付きやすくなるためである．その矛盾点に気が付くたびに修正を行うことで，設計図間の整合性と品質が向上したと考えられる．

以上の理由により，設計後に発見された欠陥は少なく，加えて設計段階の欠陥は見つからなかった．

また，本ツールを用いることで設計段階により多くの欠陥を発見し修正することができたことにより，ビデオレンタルシステムに要する開発時間を大幅に減らすことができたと考える．

MVC 分割明示化においては，本ツールを用いることで，MVC モデルの Model を UML 設計で，View を GUI 設計で，Controller を UML 設計と GUI 設計の関連付けで記述でき，MVC を明示的に分割して設計することができた．そのため，設計においてどれが Model，View，Controller なのかを即座に見分けることができる．

また，MVC モデルに基づいて開発を行うことで，“Model” に関係したクラスと “View と Controller のセット” に関係したクラスの独立性を高め，さらに再利用性も高められるということが分かっている [15]．本ツールを用いることで MVC を明確に切り分けて設計できたため，“Model” と “View と Controller のセット” を容易に取りかえることができ，設計の再利用性が向上したと考える．

6 まとめ

本研究では，設計時の並行作業と MVC 分割明示化を主テーマとした．24 の設計支援ツールを調査した結果，並行作業と MVC 分割明示化を支援するような設計支援ツールが存在しないことを確認した．そのため，UML 設計と UI 設計の並行作業と MVC 分割明示化を支援するツールを提供すべきであると考えられた．

そこで本研究では，MVC モデルパラダイムに基づき，UML 設計で Model 部分を，UI 設計で View 部分を，UML 設計と UI 設計の関連付けによって Controller 部分を記述可能にした．次に，UMLi を UML2.1 に対応し，関連付けを記述できるよう拡張した．また，OCL 定義により，GUI 設計に適した形に UMLi を拡張し，UML 設計と UI 設計の関連付けを具体的に記述した．最後に，これらを満たす設計支援ツールを開発した．

本提案手法により，検討項目に漏れがないかのチェックや全体や各部の整合性の確保を容易に行えると考える．

今後の課題には，設計情報からスケルトンプロ

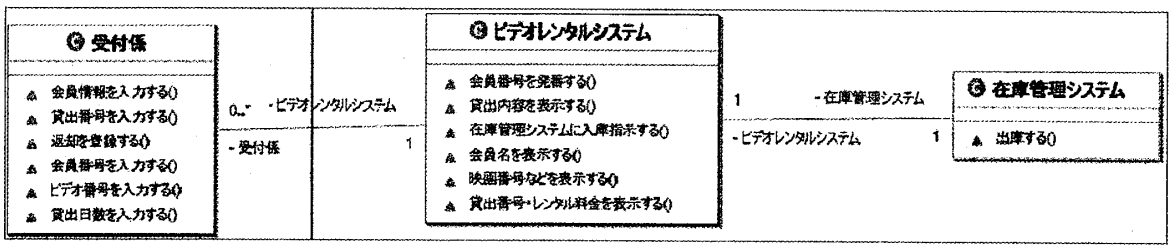

図 7: ビデオレンタルシステム クラス図

グラムを生成することで，実装作業を軽減することがあげられる．また，UML 設計において取り扱える図の追加や，UI 設計における CUI 設計，ネットワークインタフェース設計も議論することがあげられる．

参考文献

[1] 銅島康, 上田賀一. UML 記述に UI 設計を連携させた開発支援ツールの提案. 情報処理学会 研究報告 (SE), Vol. 2005, No. 29, pp. 1–8, March 2005.

[2] IBM Japan. http://www.ibm.com/jp/.

[3] OGIS-RI, 2002. http://www.ogis-ri.co.jp/.

[4] Pattern Weaver, 2001. http://www.ogis-ri.co.jp/.

[5] Grape City. http://www.grapecity.com/japan/.

[6] 日揮情報ソフトウェア株式会社, 2004. http://www.jsys-products.com/.

[7] Borland, 1994. http://www.borland.com/.

[8] Jude UML modeling tool . http://www.esm.jp/jude-web/index.html.

[9] Sparx Systems Japan , 2002. http://www.sparxsystems.jp/.

[10] Gentleware. just model , 2000. http://www.gentleware.com/.

[11] OMONDO , 2002. http://www.eclipseuml.com/.

[12] 野村幸司, 小飼敬, 上田賀一. アクティビティ図主導のモデリング支援ツールの開発. 情報処理学会 研究報告 (SE), Vol. 2004, No. 30, pp. 25–32, 2004.

[13] Paulo Pinheiro da Silva. OBJECT MODELING OF INTERACTIVE SYSTEMS:THE UMLi APPROACH. マンチェスター大学 博士課程論文, 2002.

[14] Trygve Reenskaug. MODELS - VIEWS - CONTROLLERS. *Xerox PARC technical note*, 1979.

[15] Trygve Reenskaug. The Model - View - Controller(MVC) Its Past and Present. *University of Oslo Draft*, 2003.

[16] Object Management Group 著, 西原裕善監訳. UML2.0 仕様書 2.1 対応. オーム社, 東京都千代田区神田錦町 3-1, 初版, November 2006.

[17] Tigris.org. http://argouml.tigris.org/.