JAIST Repository
https://dspace.jaist.ac.jp/

Title	CosminexusワークフローシステムのColoured Petri Netsによるモデル化と検証
Author(s)	押手, 俊
Citation	
Issue Date	2007-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/3610
Rights	
Description	Supervisor:平石　邦彦, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修 士 論 文

Cosminexusワークフローシステムの Coloured Petri Netsによるモデル化と検証

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

押手 俊

2007年3月

修 士 論 文

Cosminexusワークフローシステムの Coloured Petri Netsによるモデル化と検証

指導教官　平石邦彦 教授

審査委員主査　平石邦彦 教授
審査委員　金子峰雄 教授
審査委員　宮地充子 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

510022 押手 俊

提出年月: 2007 年 2 月

Copyright © 2007 by Takashi OSHITE

概 要

現在，多くの業種で導入されているワークフローシステムを使用し，本学の履修システムのワークフローを作成した．日本版 SOX 法では，業務内容の明確化，業務履歴の保存が求められており設計したビジネスプロセスとドメインモデルの整合性を取る必要がある．この整合性について本学の履修システムを取り上げ，そのビジネスプロセスから形式的なモデルで検証が行なえるか検討することが本研究の目的である．そのために，ビジネスプロセスから Coloured Petri Nets への変換規則を定義し，ペトリネットモデルで検証できることを確認した．

目 次

第 1 章　はじめに **1**
1.1 研究の背景 1
1.2 研究の目的 1
1.3 論文の構成 2

第 2 章　ワークフローシステム **3**
2.1 SOA 3
2.2 ワークフローシステム 8
2.3 Cosminexus 10
2.4 他のワークフローシステム 15

第 3 章　ワークフローの設計 **18**
3.1 設計 18
3.2 ビジネスプロセスのレベル 18
3.3 履修システムのビジネスプロセス 19
3.4 メインプロセスとサブプロセス 19

第 4 章　ワークフローの実装 **26**
4.1 Cosminexus ワークフローシステムの仕様 26
4.2 ビジネスプロセスの記述 30
4.3 デプロイ 34

第 5 章　ワークフローのモデル化 **37**
5.1 方法 37
5.2 ペトリネット 37
5.3 Coloured Petri Nets 38
5.4 CPN Tools 40
5.5 Cosminexus から CPN への変換 41
5.6 モデルの記述 44

第 6 章　モデルの検証 **49**
6.1 一般的性質の検証 49
6.2 特定の仕様における帳票権限の検証 49

6.3 考察 52

第 7 章 おわりに **53**
7.1 まとめ 53
7.2 今後の課題 53

謝辞 **54**

参考文献 **56**

付 録 A ユーザ機能追加処理 **57**

図 目 次

2.1 プロセス指向アプローチの変遷 3
2.2 SOA の構造 4
2.3 J2EE による SOA サービス 5
2.4 Web サービスの基本構成 6
2.5 ワークフローシステム 9
2.6 Cosminexus システム構成 13

3.1 履修システム 18
3.2 履修システムアクティビティ図 (再掲) 19
3.3 履修届アクティビティ図 20
3.4 再履修届アクティビティ図 21
3.5 受講者通知アクティビティ図 22
3.6 休講通知アクティビティ図 23
3.7 点数登録アクティビティ図 24

4.1 ビジネスプロセスのステップ 26
4.2 ビジネスプロセスの構成 26
4.3 制御ノードの種類 27
4.4 ドメインモデルデータと RDB の対応関係 29
4.5 WorkCoordinator Definer 30
4.6 Definer で記述した履修サブプロセス 31
4.7 履修サブプロセスの作業ウィンドウ 31
4.8 メインプロセス 32
4.9 履修サブプロセス (再掲) 33
4.10 再履修サブプロセス 33
4.11 受講者通知サブプロセス 33
4.12 休講通知サブプロセス 33
4.13 点数登録サブプロセス 33
4.14 振り分けルール定義ウィンドウ 34
4.15 履修システムのディレクトリ構成 35
4.16 Eclipse Server Plug-in 35
4.17 ポータル画面 36

4.18 帳票履歴 36

5.1 ペトリネットのマーキングの推移 38
5.2 制御ノードのペトリネット表現 41
5.3 共有データを有するプレース 42
5.4 Fusion プレース 42
5.5 ガード関数による振り分けルール表現 44
5.6 メインプロセス 44
5.7 履修モデル 45
5.8 再履修モデル 46
5.9 受講者通知モデル 47
5.10 休講通知モデル 47
5.11 点数登録モデル 48

6.1 担当者割り振りの選択 49
6.2 デッドロックを現す表示 49
6.3 履修届 ASK-CTL 実行画面 51
6.4 点数登録 ASK-CTL 実行画面 51

A.1 点数登録帳票フローチャート 57
A.2 ユーザ機能付き帳票 58

表 目 次

2.1 統制された業務プロセスの実現に必要な技術 11
2.2 ワークフロー用語の対比 12
2.3 オブジェクト権限の種類 15

3.1 履修届担当者 20
3.2 履修届テーブル 20
3.3 再履修届担当者 21
3.4 再履修届テーブル 21
3.5 受講者通知担当者 22
3.6 受講者通知テーブル 23
3.7 休講通知担当者 24
3.8 休講通知テーブル 24
3.9 点数登録担当者 24
3.10 点数登録テーブル 25

4.1 振り分けルールで定義する帳票の担当者と権限 28
4.2 組織/ユーザデータ (一部抜粋) 30
4.3 履修サブプロセスの RDB テーブル 32
4.4 SQL で使用する組み込み変数 34

5.1 Cosminexus から CPN への変換規則 41

第1章　はじめに

1.1　研究の背景

現在の社会は，電子社会であり，企業活動や日常生活のあらゆるところで情報システムを基盤として構成・運用されている．安心して生活できる電子社会を構築するためには，様々な要件 (正当性，公平性，セキュリティ，進化性，耐事故・故障性，アカウンタビリティなど) を事前に検証しておく必要がある．法制度の面では，米国で不正経理を防止する SOX 法が施行されたことに続いて，日本で日本版 SOX 法 (金融商品取引法) が成立した．この法律に盛り込まれている内容に，内部統制報告書と情報技術による内部統制がある．前者の内部統制報告書は，適正な財務・企業情報の開示を確保するため財務報告に関する内部統制を評価するために提出が義務付けられている．後者の情報技術による内部統制は，組織や業務の透明性を確保するため情報技術で統制することが求められている．これは，不正を起こすことが困難な情報システムの構築や業務履歴の保存による追跡を可能にするという面で，組織の社会活動における安心性・正当性を評価する上で有効な手段といえる．このような社会の流れに加え，1990 年代初頭から情報システムを利用したビジネス改革が頻繁に行われるようになった．その中に，電子社会における情報システムの大きな流れとしてビジネスプロセスの記述に基づいたシステム構築がある．組織内あるいは複数の組織にまたがる情報システムのワークフローを形式的に記述し，それにより，ソフトウェア，データベース，人間の活動など様々なコンポーネントを統合することができる．

これまでの研究として，ワークフローのビジネスプロセスの経路が正しいかどうかの検証や，シミュレーションにより動作を確認するというフロー作成後の検証が中心であった．さらに，電子社会ではドメインモデルとワークフローの整合性が重要であるが，これに関する研究は従来はほとんどこなわれてこなかった．そこで本研究では，ワークフローを作成する段階でリレーショナル・データベース (RDB) を用いて組織のドメインモデル，担当者の属性を考慮した動的割り当て表現を可能にしたワークフローシステムを取り上げ，そのシステムで作成したワークフローに対して形式的に検証する手法について提案する．

1.2　研究の目的

本研究は，ビジネスプロセスを記述するワークフローシステム (日立製作所 Cosminexus) において組織に関するドメインモデル (内部組織構造，役割，権限，責務，承認/報告フ

ロー，業務規則など) とワークフローの整合性検証の方法を考案することである．ドメインモデルの属性割り当ての手段として RDB を用いることで，担当者の属性を考慮した割り当てを動的に行うことができる．これにより，フローの中にソフトウェア，データベース，担当者の活動など様々なコンポーネントを統合することが可能になる．さらに，ドメインモデルと RDB の連携を形式的にモデル化する方法と，それに対するモデル検査の方法について検討する．

1.3 論文の構成

第 2 章は，ワークフローシステムについて，その背景となる技術について触れ，本研究で使用する Cosminexus ワークフローシステム，そしてほかの代表的なワークフローシステムを紹介する．第 3 章では，ワークフローの例題として本学の履修システムの設計を行なう．第 4 章では，第 3 章で設計したワークフローを Cosminexus を用いビジネスプロセスの記述を行う．第 5 章では，ワークフローシステムから形式的モデルへの変換について述べる．第 6 章では，モデル化したビジネスプロセスについてツールを用いた検証を行なう．第 7 章では，本研究のまとめを行い，今後の課題について述べる．

第2章　ワークフローシステム

ワークフローの基本アーキテクチャである SOA (Service Oriented Architecture) [5] について 2.1 節で説明し，ワークフローシステムの概要について 2.2 節で説明する．2.3 節では，本研究で使用するワークフローシステムの Cosminexus について概説し，2.4 節ではそれ以外のワークフローシステムを紹介する．

2.1　SOA

2.1.1　概要

情報システムが単に機械的処理をこなす道具から社会インフラへとなって久しい．また近年では，組織改革や合理化を進めるために情報システムが利用されている．図 2.1 は，1990 年代初頭から業務を見直すための様々な取り組みを表している [7].

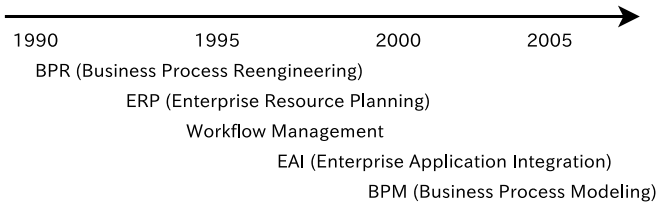


図 2.1: プロセス指向アプローチの変遷

ERP や EAI といったアプローチは全体的な組織インフラを刷新するには適しているが，組織の変化といった柔軟な対応を行うことができない．そこで，SOA が注目を浴びる理由に，社会的な面では組織内の全構成員が統一された目的意識と組織の方向性の理解を求められること，技術面ではオープンシステムの隆盛を受け J2EE の標準化技術を組み合わせることで組織の情報システムを柔軟かつ容易に実現することがあげられる [14].

SOA を初めて定義した米・調査会社によれば，SOA とはサービスとサービスのクライアントから成るアプリケーションソフトウェアのトポロジであるとされている．SOA は，サービスを定義，開発して利用するためのシステムアーキテクチャであり，そのため，どのようなサービスを定義するか決める必要がある．

SOA の基本となる構造は，サービスとサービスを利用するクライアントモジュールで構成されている．サービスが提供する機能は，WSDL (Web Service Description Language)

を用いてインタフェースファイルに記述される．クライアントモジュールはインタフェースファイルに記述されたサービスへのリクエスト方法を理解し，サービスへ機能の実行をリクエストする．それらと既存のアプリケーションやRDBなどをミドルウェアで接続し，Webサービスの通信機能やインタフェースを付加した構成をとる (図2.2)．

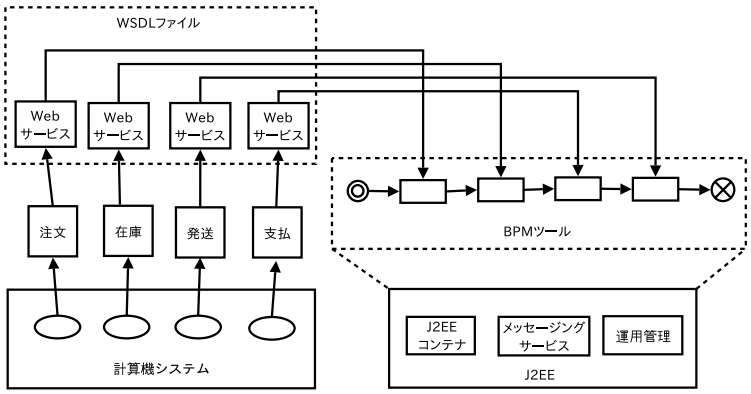


図 2.2: SOA の構造

SOAは様々な技術の組合せで構成されるが大きく分類すると「サービスを実現する技術」と「サービスを活用する技術」の2種類がある．

サービスを実現する技術には，サービスの実装をコンポーネント化する技術，サービスが外部からのリクエストを受け付ける仕組みを実現する技術，およびそのリクエストを処理するビジネスロジックを実装する技術などがある．

サービスを活用する技術には，サービスをビジネスプロセスの構成要素とするBPMやサービスを集約してポータルサイトを通じて一連のまとまった情報提供を行うポータルフレームワークがある．この技術について2.1.4節で述べる．

これらSOAを実現するにはJ2EEのアーキテクチャが適していると言われている．J2EEによって抽象度の低いサービス (ビジネスロジックや外部と接続した機能のサービス化) から抽象度の高いサービス (Webサービスの標準化技術とJ2EEの組み合わせ) まで利用できるため，多くのフレームワーク製品がJ2EEを基盤としている．

2.1.2 J2EE機能

J2EEに準拠するWebアプリケーションサーバはWebコンテナとEJB (Enterprise JavaBeans) コンテナを持ち，Webサービスをインタフェースとするアプリケーションを構築できる．SOAにおいてJ2EEの果たす役割は，サービスの実現とサービスを活用する技術の両方を含んでいる．ここで，J2EEの機能を紹介し，サービスの実現に用いられる機能について解説する．

J2EEによるサービスの実現は，図2.3で示す技術を基盤とする．

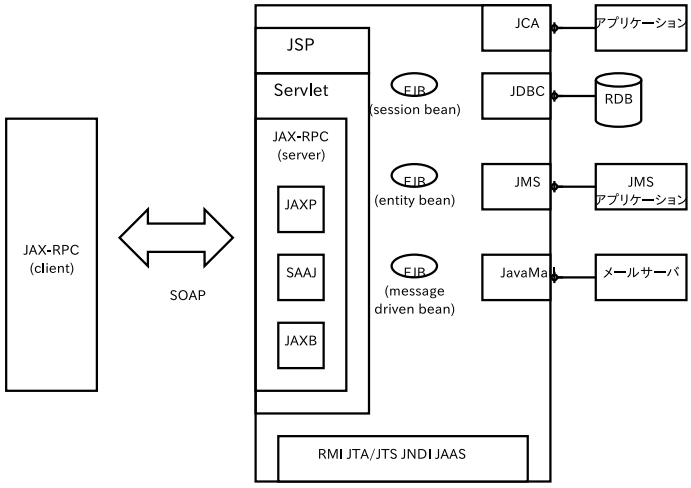


図 2.3: J2EE による SOA サービス

外部アプリケーションや RDB との接続とコンポーネントベースのビジネスロジックを実装を補う技術，Web サービスを提供する技術について説明する．

- JCA

  外部アプリケーションや RDB を J2EE と接続するには，JCA (J2EE Connector Architecture) と呼ばれる，サーブレットや EJB などの J2EE コンポーネントが EIS (Executive Information System) と接続し連携した処理をする．EIS と J2EE サーバの間のコネクタの役割を果たすソフトウェアコンポーネントであるリソースアダプタは，特定のアプリケーションと連携に際に利用する．J2EE アプリケーションはリソースアダプタを介して EIS との通信を実現し，EIS とのトランザクション連携，セキュリティ連携，データの相互通信，アプリケーションの相互呼び出しを行う．リソースアダプタでは，次の 2 種類の処理が行われる．

  1. サービス要求・応答

     サービス要求・応答とは，EIS との要求/応答通信である．クライアントアプリケーションからリソースアダプタ経由で EIS にサービス要求が発行され，リソースアダプタからクライアントに EIS 応答が返る．例として，ミドルウェアの標準 API や RDB への SELECT 文の実行要求である．

  2. イベント

     イベントは EIS から送信されるメッセージである．実行時に EIS とリソースアダプタは，要求・応答・イベントを XML ドキュメントとして交換する．リソースアダプタでは，設計時に定義されたスキーマを使用して，EIS フォーマットと XML フォーマットのデータ交換が必要になる．

- JDBC

JDBD (Java Database Connectivity) はRDBと接続を実現するためのAPIである．

- EJB

  EJBは，ビジネスタスクまたはエンティティを実装するサーバサイドJavaモジュールである．EJBには，セッションBean，エンティティBean，そしてメッセージドリブンBeanの3種類がある．セッションBeanは単一セッションをクライアントに代わって特定のビジネスタスクを実行する．セッションBeanは永続的ではないため，クライアントでセッションBeanの利用が終わるとそのBeanは消滅する．エンティティBeanはRDBのビジネスオブジェクトを表す．メッセージドリブンBeanは到着したメッセージが処理される．

この他にJ2EEは，ネーミング機能とディレクトリ機能を提供するJNDI (Java Naming Directory Service)，Javaの分散コンピューティングの標準規格であるRMI (Remote Method Invocation)，電子メールシステムを構築するJavaMailなどで構成されている．また，Webサービスを提供する技術にSOAP,WSDL,UDDIがある．

SOAP (Simple Object Access Protocol) はSOA環境において，サービスとBPM間，およびサービスとポータル間の通信に使用される．SOAPはXML形式のデータ送受信を可能にするプロトコルであり，同期型・非同期型通信で利用可能である．これまでの通信プロトコルは異なる技術間の通信は容易には実現できなかったが，SOAPはこの点を解決し異種間の通信を実現している (図2.4)．

WSDLは，Webサービス定義を記述するためのXMLの文法である．WSDLによって記述されたXMLファイルは，添付されるWebサービスの内容と呼び出し方，Webサービスが存在する場所が記載される．アプリケーション間の通信を定義しており，通信プロトコルや実装技術，実装言語に依存しない．WSDLで記述した文書をWSDL文書とよび，あらかじめ定義されたXMLタグにサービスの名称やサービスで定義されたメソッド名，引数，通信プロトコル，送受信データの定義と型，サービスの所在が記載される．

UDDI (Universal Description, Discovery and Integration) は，Webサービスの内容とアクセス方法，アクセス先を管理するレジストリに関する標準である．

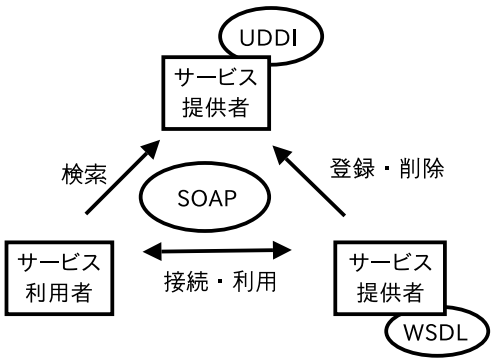


図 2.4: Webサービスの基本構成

2.1.3 J2EE による Web サービスの実装

J2EE アプリケーションにおいて Web サービスの仕組みは，JDK (Java Development Kit)，J2EE によって定義された仕様で実装される．SOAP,WSDL,UDDI を利用するために J2EE では以下の仕組みが定義されている．

- JAXP

  JAXP (Java API for XML Processing) は，XML ドキュメントの処理と変換を行うための API である．

- JAX-RPC

  JAX-RPC (Java API for XML-based Remote Procedure Call) は，異種プラットフォーム，異種言語環境での相互接続性を実現するために規定されている．JAX-RPC を利用することで，SOAP で通信するサーバ側の Web サービスと Web サービスクライアントの実装を構築できる．

- JAXR

  JAXR (Java API for XML Registries) は，B2B 取引の標準仕様である ebXML レジストリ，UDDI レジストリなどの XML ベースのレジストリに対するクライアントからのアクセス方法を定義している．

- SAAJ

  SAAJ (SOAP with Attachments API for JAVA) は，SOAP メッセージを操作する際に利用する．SAAJ API は XML データの部分的な抽出や，JAX-RPC のメッセージハンドラ内で SOAP メッセージにアクセスすることを目的として JAX-RPC によって使用される．

- Web Services for J2EE

  Web Services for J2EE は，J2EE サーバにおける Web サービスの配備と実行の仕組みを定義している．

2.1.4 サービスを活用する技術

前述の通り Web サービスを活用する技術には BPM，ポータルといったフレームワークがある．それぞれ，以下で概説する．

- BPM

  BPM は，ビジネスプロセスのライフサイクルを管理する仕組みであり，ビジネスプロセスを設計，運用し最適化を行う一連の流れとそれを実現するフレームワークの

ことである．BPM において，組織内のビジネスプロセスの設計，実行そして改善案がツールによって提供され，一連のライフサイクルとして管理・運用される．BPM を実現するソフトウェアを BPM ツールや BPMS (Business Process Management System) と呼ぶ．

- ポータル

  ポータルフレームワークを使用すると，ユーザはポートレットと呼ばれる機能別のウィンドウで各アプリケーションや Web ページを表示することができる．ポートレットは，ポータルアプリケーションの基本的な構成要素であり，アプリケーション，情報，および，ビジネスプロセスのインタフェースのウィンドウである．

2.2 ワークフローシステム

ワークフローシステムは，人と機械をベースにした活動の組合せを明確にし，その活動を管理する．WfMC (The Workflow Management Coalition) は，ワークフローシステムの標準化団体であり，これまでにリファレンスモデル WRM1.1 を発表している [8]．このリファレンスモデルで定義されたワークフローシステムの基本機能は以下の 3 分野である．

- ビルドタイム機能

  ビジネスプロセスを分析，モデリング，システム定義の技法を使うことにより，ワークフローのプロセス定義を行う．

- ランタイムプロセス制御機能

  プロセス定義を解釈し，動作環境におけるワークフロープロセスの管理と，さまざまなアクティビティをここのプロセスの一部として動作させる．

- ランタイムアクティビティ相互作用

  ワークフローのプロセスにおけるここのアクティビティは，アプリケーションプログラム等により人による操作が行われる者である．プロセス制御ソフトウェアによっては，アクティビティ間で制御を移行，プロセスの動作状態を確認，アプリケーションツールを起動，適切なデータを渡す，といった処理をする．

これらを実現するためにワークフローシステムは，ワークフローエンジンを中心に，そのプロセスを管理する様々なアプリケーションによって構成される．このプロセスとは，ある共通の目標を達成するために結合したプロセスアクティビティが，統合された構造体として表現されたビジネスプロセスの見方である．

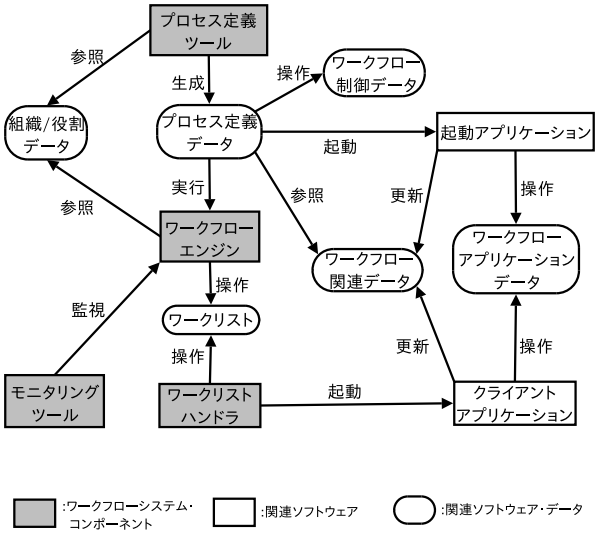


図 2.5: ワークフローシステム

これは，WfMC が定義したワークフローシステムの一般的な機能構成である．ワークフローにおける作業の単位をアクティビティと呼び，ユーザがワークフローを定義したものを「プロセス定義データ」という．プロセス定義では，アクティビティごとの作業内容，開始と終了の条件，担当者，関連するアプリケーションプログラムとデータなどを定義する．アクティビティ間の順序関係も定義する．アクティビティの担当者は人間の場合もあるが，アプリケーションプログラムの場合もある．

このプロセス定義データを作成する時に使うソフトウェアモジュールが「プロセス定義ツール」である．GUI によってアクティビティ間の順序関係を定義できる．

アクティビティの実行順序は，組織構造や人員構成に依存することがある．そこで，組織構造や人員構成などを記述したデータを別に作成しておき，それを利用してアクティビティの順序関係を定義することもできる．この組織構造や人員構成を表すデータを「組織/役割データ」と呼ぶ．このデータから，実行順序は組織や権限などの役割データによって具体的な作業の流れが決定する．

ワークフローエンジンは，プロセス定義データを解釈して実行するソフトウェアモジュールである．同一あるいは異なるワークフローを複数個，並列して実行できる．複数のワークフローを同時に実行したときにそれらを区別し制御するための「ワークフロー制御データ」と呼ぶ内部情報を使う．

個々のアクティビティにおいて，担当者が実行する具体的な作業をワークアイテムと呼ぶ．これは，決められた帳票にデータ入力することや，回覧することである．

実行中の全てのワークフローに関するワークアイテムは，担当者ごとにまとめて管理されている．処理すべきワークアイテムを担当者ごとに一覧にしたものをワークリスと呼ぶ．アクティビティが終了するごとにワークフローエンジンは，プロセス定義データの内容にしたがって，次に処理すべきアクティビティの担当者のワークリストに，実行すべき

ワークアイテムを追加する．

担当者に対する作業指示は，ワークリストハンドラと呼ぶソフトウェアモジュールが出す．ワークリストを参照して，その担当者が実行すべきワークアイテムを示す．担当者が作業を完了させたら，ワークリストからワークアイテムを削除する．これによって，ワークローエンジンはアクティビティの終了を判断する．

業務の担当者がワークフローの進行状況を知りたいときは，モニタリングツールと呼ぶソフトウェアモジュールを使う．これは，ワークフローの実行記録を保存することができる．

2.3 Cosminexus

2.3.1 概要

日立製作所の Web アプリケーション基盤ミドルウェアである Cosminexus は，金融分野に置けるバンキングシステム，産業分野における e ビジネス，そして販売管理，顧客情報管理などの業務システム，公共分野における電子申請，電子窓口，企業の従業員に対する福利厚生サービスサイト，総務業務を中心とする社員サポートシステム，地域コミュニティに対する情報提供サイトの構築などの実績がある．

Cosminexus の基盤製品群は次の通りである．

- 高信頼なアプリケーション実行基盤
- Web/Java ベース業務の容易なアプリケーション開発環境
- 既存 IT 資産や外部リソースを活用するためのシステム連携基盤
- モバイルアクセスやポータルなど利便性の高い Web フロントを実現するフロントアクセス基盤

日本版 SOX 法の求めでは業務の流れを統制する必要がある．その例として，企業内の業務の流れを以下に示す．

- 営業本部：受注登録が行われ決済が完了すると，各拠点へ受注情報がタイムリーに伝達される．
- 生産本部：受注情報をもとに作業計画を立て，在庫引当，出荷指示を行い，納品実績に基づいて在庫の更新や売掛け手配が行われる．
- 製造工場：受注情報に基づいて部品の調達が行われる．
- 経理部：納品の入金後に売上げとして計上する．

このように，統制された業務プロセスとは，

- 受注から出荷、そして売上計上まで、一連の業務の流れがコントロールされている．
- 受注から生産，売掛けといった各業務処理の作業履歴が確保・保持されている．
- 不正進入や不当なデータアクセスができないようにコントロールされている．

といったことが実現されている．

表 2.1: 統制された業務プロセスの実現に必要な技術

技術	説明
ワークフロー	業務の流れの制御，他システムとの連携・タイムリーな通知，作業ログ取得，権限制御
セキュリティ	認証，不正アクセス防止
文書管理	文書の保持・保管，検索，閲覧
運用管理	業務の自動連携
データベース	データ共有・データ保管・検索

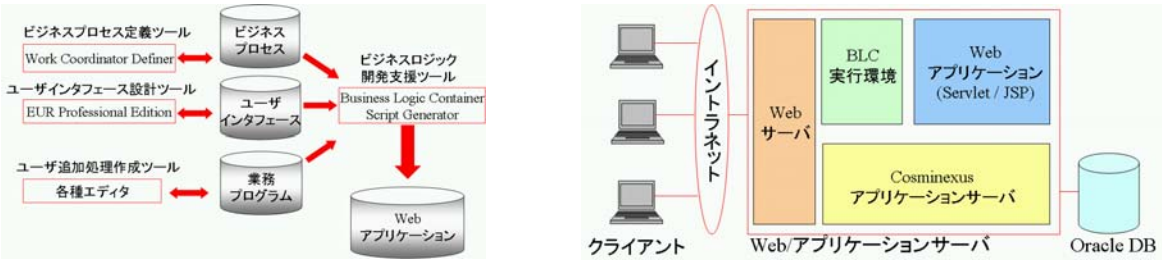
表2.2 (文献 [10] より引用) にワークフロー用語の対比として，WfMC と Cosminexus の用語を示す．

表 2.2: ワークフロー用語の対比

WfMC	Cosminexus	意味
Activity	ノード	プロセス内で一つの論理的なステップを形作っている仕事.
Business Process	ビジネスプロセス	業務目的を達成するために複数の手続きやアクティビティが結合した構造体.
Process Definition	プロセス定義	自動化された操作を目的としたビジネスプロセスの表現.
Sub Process	サブプロセス	他のプロセスによって実行されるかコールされるプロセスで，全体のプロセスの入れ子になっている.
Manual Activity	マニュアルアクティビティ	ワークフローシステムの範囲外にあるビジネスプロセス内の自動化できないアクティビティのこと.
Automated Activity	自動処理ノード	ワークフローシステムの管理の対象となる自動化できるアクティビティのこと.
Workflow Management System	ワークフロー管理システム	ソフトウェアを用いてワークフローの実行を定義・創造・管理するシステムで，ワークフローエンジンの上で稼働する.
Process Instance	ワークインスタンス	プロセスが具体的に設定されたときの表現で，プロセス定義にしたがって，ワークフローシステムによって生成・管理される.
Activity Instance	ノードインスタンス	ワークインスタンスにおけるアクティビティの表現.
Work Item	ワークリストアイテム	アクティビティにおいて処理されるべき仕事の表現.
Invoked Application	起動されたアプリケーション	ワークリストアイテム処理を支援するためにワークフローシステムによって起動されるアプリケーションのこと.

2.3.2 システム構成

Cosminexus のシステム構成は，ワークフロー開発環境およびワークフロー実行環境から構成される (図 2.6)．また，Cominexus は WfMC に準拠している．



(a) 開発環境　　(b) 実行環境

図 2.6: Cosminexus システム構成

ワークフロー開発環境は，図 2.6 (a) に示すように次のツールが含まれる．

- ビジネスプロセス定義ツール

  ビジネスプロセス定義ツールは，WorkCoordinator Definer を使用しビジネスプロセスの定義を行う．

- ユーザインタフェース設計ツール

  既存の紙ベースの帳票などをユーザインタフェース設計ツール EUR を用いることで，紙帳票から電子帳票を作成することができる．

- ユーザ追加処理作成ツール

  ユーザ追加処理作成ツールは，個別の帳票に対して追加したい機能をエディタや IDE を使用する．実装には，Java Script および，Hitachi Business Logic - Container が用意する Java クラスを使用する．この機能例として，付録 A に記載した．

- ビジネスロジック開発支援ツール

  ビジネスロジック開発支援ツールは，Hitachi Business Logic - Container - Script Generator のことであり，上記の 3 つのツールで作成した情報を入力し，Web アプリケーションの生成・出力するツールである．

ワークフロー実行環境は，図 2.6 (b) に示すように次のツールが含まれる．

- Web サーバ

  Web サーバは，Cosminexus Application Server に含まれる Web サーバ機能を使用して，Web 画面の処理をする．

- BLC 実行環境

  BLC 実行環境とは，Hitachi Business Logic - Container を使用し，受信ボックス・送信ログ・案件履歴・帳票一覧の処理を実行する．

- Web アプリケーション

  Web アプリケーションは，Servlet/JSP を使用した動的な Web ページを生成する．

- Cosminexus アプリケーションサーバ

  Cosminexus アプリケーションサーバは，ワークフローエンジン Hitachi Business Logic - Container，Web アプリケーションの処理を行う，ワークフロー実行環境のプラットフォームである．

- Oracle DBMS

  Oracle DBMS は，ワークフローシステムおよび帳票のデータ管理に使用する．データベースアクセス共通基盤 DABroker を使用し，ワークフロー実行環境と Oracle DBMS が接続され，DBMS のコントロールを行う．具体的には，DABroker を利用することで，Web サーバ，ワークフローエンジンからのデータベースアクセスを行うことができる．2.3.3 節で Oracle DMBS が管理するデータおよびそれらの権限について説明する．

2.3.3 RDB

ワークフローシステムや帳票が参照・更新を行うデータベース権限には，オブジェクト権限とシステム権限の 2 種類がある．オブジェクト権限は，特定のオブジェクトを操作する権限である．例として，表の挿入・削除・検索のことである．ユーザは自分が所有するオブジェクトに関して，全てのオブジェクト権限を持っているが，他のユーザのオブジェクトを操作するためには，対象となっているオブジェクトの権限を与える必要がある．また，表の作成・ユーザ追加などはシステム権限の 1 つであり，特定のオブジェクトに依存しない権限である．担当者が行える業務の範囲，権限の制約は，このオブジェクト権限を用いる．オブジェクト権限には表 2.3 に示すものがある．

表 2.3: オブジェクト権限の種類

オブジェクト権限	表	ビュー	順序	プロシージャ	備考
ALTER	o		o		定義変更
DELETE	o	o			データ削除
EXECUTE				o	実行
INDEX	o				索引
INSERT	o	o			データ入力
REFERENCES	o				データ参照
SELECT	o	o	o		データ検索
UPDATE	o	o			データ変更

2.3.4 導入事例

Cosminexus は，次に示すような様々な業種で採用されている．

- 信託銀行：年金業務の処理パターンをビジネスプロセス定義で対応
- 自治体：電子決済基盤，文書管理等をシステムと連携
- 電力会社：顧客問い合わせ部署，担当者の動的な割り当て
- 通信会社：事務処理効率化の並列作業機能適用
- 日立製作所：6 万人規模の大規模ワークフロー，作業振り分けによる宛て先設定
- 倉庫会社：業務の開始から終了までのステータスを一元管理
- 小売業：開発効率，作業振り分けの柔軟性
- 電力系設備会社：フレームワーク適用による開発効率

2.4 他のワークフローシステム

ここで紹介するワークフローシステムは，商用および，OSS (Open Source Software) で発表されているものについて紹介する．それぞれ，システムの特徴や動作環境についてまとめる．

2.4.1 FormWave for WebSphere

FormWave for WebSphereは，IBMのWebSphere Application Server上で動作するサーブレット，JSP，EJBなどのJ2EEに準拠したJava技術を活用したワークフローの開発を容易に実現するミドルウェア製品である．また，WebSphereは，J2EEやWebサービスなどのオープンな技術をサポートするWebアプリケーションのフレームワークである．

2.4.2 StarOffice21

StarOffice21は，NECのワークフローシステムである．業務におけるコントロールが適切に評価されていることを運用状況評価で確認し、承認を受けた運用状況評価の実行結果を評価証跡として保存するための仕組みを提供し，以下の機能を有する．

- テストの進捗管理と証跡の自動記録
- 運用評価の工数削減
- 評価の効率化
- 内部監査報告書の作成支援

2.4.3 GLOVIA/MyOFFICE

GLOVIA/MyOFFICEは，富士通のワークフローシステムである．特徴として，人事・総務の業務改革の実践で蓄積してきた技術・ノウハウおよび，100社以上の適用実績があるパッケージ “MyOFFICE” をもとに，J2EEフレームワーク・XML技術を駆使して開発されたワークフローシステムである．

2.4.4 BPEL Process Manager

BPEL Process Managerは，Oracleのワークフローシステムである．プロセス設計を行うOracle JDeveloperのプラグインとして提供され，設計ツールとOracle Application Server (J2EEアプリケーションサーバ) 上で動作する実行エンジンである．

本ワークフローシステムの特徴は次の通りである．

- 承認，アドホックプロセス，自動エスカレーションなど人のワークフローの共通パターンをサポート
- ロール，ユーザ，グループによるタスク割り当て
- タスクの監査，期限設定，委譲

2.4.5 Nautica Workflow

Nautica Worklfow は，情報処理推進機構 (IPA) 2004 年度オープンソースソフトウェア活用基盤整備事業の支援で開発を行っている．

本ワークフローシステムの特徴は次の通りである．

- WfMC 標準仕様に対応し，ワークフローシステムに最低限必要とされる機能を有する．
- JVM (Java Virtual Machine) 上で動作するため，システム稼働環境の依存がない．
- ワークフローエンジン間で連携が可能である．

第3章　ワークフローの設計

3.1　設計

本学における実際の履修プロセスと，ビジネスプロセス定義として設計したワークフローを示す．また，履修プロセスで使用する RDB のデータについて説明する．

3.2　ビジネスプロセスのレベル

ビジネスプロセスの設計では，プロセスモデルの詳細度によってレベルを分ける必要がある．まずは，全体の大まかなプロセスを記述するマクロレベルがある．そして，マクロレベルにおけるひとつひとつプロセスを階層的に表すミクロレベルがある．

- マクロレベル

  マクロ的なビジネスプロセスで作成する場合，機能は最上位レベルのプロセス，すなわちメインプロセスのみとなる．具体的には,「履修」プロセス,「再履修」プロセス,「受講者通知」プロセス,「休講申請」プロセス,「点数登録」プロセスである．この関係を表したがアクティビティ図を図 3.1 に示す.

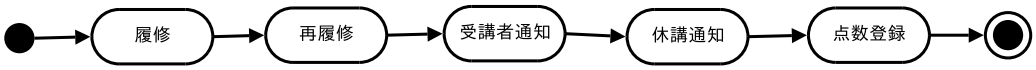


図 3.1: 履修システム

- ミクロレベル

  具体的な業務手順を記述するサブプロセスは，メインプロセスの各プロセスの階層下にある．この点において，マクロレベルとミクロレベルの中間に位置するミドルレベルのダイアグラムを描く場合，他のプロセスと連携が発生する部分は，すべてミドルレベルの段階で描いておく．例として，学生の観点から，教員や教務課との連携・連絡が発生する部分はミドルレベルで描き，学生が行う詳細な手順はミクロレベルで記述する．

3.3 履修システムのビジネスプロセス

ドメインモデルデータ

WfMC におけるドメインモデルのデータは組織/役割データとして定義されている．組織/役割データは，ワークフロー参加者のリストおよびそれらの関係である．ワークフロー参加者は種類および関連する情報によって定義される．データは，人物，組織としての役割，リソースがあげられる．

必要なデータ

- 組織内の人物リスト
- 役割単位での人物リスト
- 組織内での役職
- 人物の役割リスト

人物データ

ID，パスワードなど個人情報を集めたデータである．ワークフローに関係ある人物全員がデータの中に含まれているものとする．ここに役割は含めない．

組織データ

組織内での各人物の関係，役割を表すデータである．上長など，相対的な対象を検索するのに必要となる．

3.4 メインプロセスとサブプロセス

履修システムはメインプロセスの下に次の 5 つのサブプロセスで構成されている．

- 履修届
- 再履修届
- 受講者通知
- 休講通知
- 点数登録

履修システムのアクティビティ図を図 3.2 に示す．

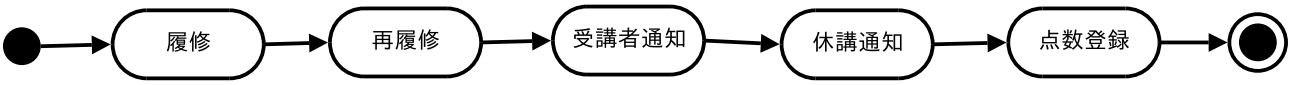


図 3.2: 履修システムアクティビティ図 (再掲)

順にそのサブプロセスについて説明する．

3.4.1 履修届

履修届は，学生が帳票入力を行い，教務係宛てには申請し，処理が完了する．履修届のアクティビティ図を図 3.3 に示す.

図 3.3: 履修届アクティビティ図

履修届帳票の担当者権限を表 3.1 に示す.

表 3.1: 履修届担当者

担当者	権限
学生	申請
教務課	受理

帳票に必要な項目を表 3.2 に示す.

表 3.2: 履修届テーブル

属性名	カラム名	制約	データ型
月曜 1 限	MON1	NOT NULL	文字列
月曜 2 限	MON1	NOT NULL	文字列
月曜 4-5 限	MON45	NOT NULL	文字列
火曜 1 限	TUE1	NOT NULL	文字列
火曜 2 限	TUE2	NOT NULL	文字列
火曜 4-5 限	TUE45	NOT NULL	文字列
水曜 1 限	WED1	NOT NULL	文字列
水曜 2 限	WED2	NOT NULL	文字列
水曜 4-5 限	WED45	NOT NULL	文字列
木曜 1 限	THR1	NOT NULL	文字列
木曜 2 限	THR2	NOT NULL	文字列
木曜 4-5 限	THR45	NOT NULL	文字列
金曜 1 限	FRI1	NOT NULL	文字列
金曜 2 限	FRI2	NOT NULL	文字列
金曜 4-5 限	FRI45	NOT NULL	文字列

3.4.2 再履修届

再履修届は，学生が帳票入力を行い，2 名の教員の承認が必要となる．承認のプロセスは，主指導教員 → 講義担当教員であるが，並列プロセスではないので主指導教員の承認が得られれば，その承認データは，講義担当教員の承認後であれば保持する必要がない．最終的に，教員 2 名の承認を得られた後に教務係に提出し，処理が完了する．再履修届のアクティビティ図を図 3.4 に示す．

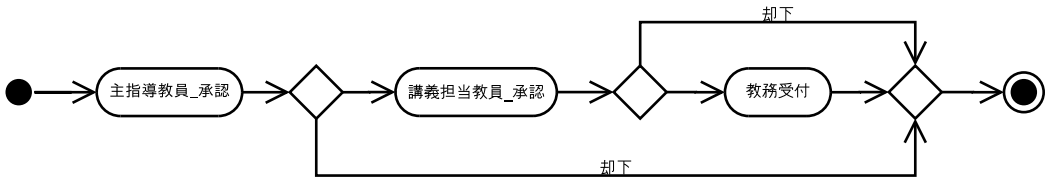


図 3.4: 再履修届アクティビティ図

再履修届帳票の担当者権限を表 3.3 に示す．

表 3.3: 再履修届担当者

担当者	権限
学生	申請
主担当教員	承認，却下
講義担当教員	承認，却下
教務課	受理

帳票に必要な項目を表 3.4 に示す．

表 3.4: 再履修届テーブル

属性名	カラム名	制約	データ型
科目名	KAMOKU	NOT NULL	文字列
ターム	TERM	NOT NULL	文字列
ターム (前回)	TERM_B	NOT NULL	文字列
教員	KYOUIN	NOT NULL	文字列
教員 (前回)	KYOUIN_B	NOT NULL	文字列
理由	RIYUU	NOT NULL	文字列

3.4.3 受講者通知

受講者通知は，教務係が帳票入力を行い教員宛てには通知し，処理が完了する．受講者通知のアクティビティ図を図 3.5 に示す.

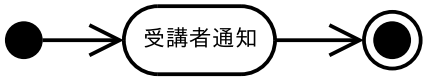


図 3.5: 受講者通知アクティビティ図

受講者通知帳票の担当者権限を表 3.5 に示す.

表 3.5: 受講者通知担当者

担当者	権限
教務課	申請
教員	受理

帳票に必要な項目を表 3.6 に示す.

表 3.6: 受講者通知テーブル

属性名	カラム名	制約	データ型
科目名	KAMOKU	NOT NULL	文字列
学生番号 1	STID1	NULL 可	文字列
学生番号 2	STID2	NULL 可	文字列
学生番号 3	STID3	NULL 可	文字列
学生番号 4	STID4	NULL 可	文字列
学生番号 5	STID5	NULL 可	文字列
学生番号 6	STID6	NULL 可	文字列
学生番号 7	STID7	NULL 可	文字列
学生番号 8	STID8	NULL 可	文字列
学生番号 9	STID9	NULL 可	文字列
学生番号 10	STID10	NULL 可	文字列
学生氏名 1	STNAME1	NULL 可	文字列
学生氏名 2	STNAME2	NULL 可	文字列
学生氏名 3	STNAME3	NULL 可	文字列
学生氏名 4	STNAME4	NULL 可	文字列
学生氏名 5	STNAME5	NULL 可	文字列
学生氏名 6	STNAME6	NULL 可	文字列
学生氏名 7	STNAME7	NULL 可	文字列
学生氏名 8	STNAME8	NULL 可	文字列
学生氏名 9	STNAME9	NULL 可	文字列
学生氏名 10	STNAME10	NULL 可	文字列

3.4.4 休講通知

休講通知は，教員が帳票入力を行い教務係宛てには通知し，処理が完了する．休講通知のアクティビティ図を図 3.6 に示す.

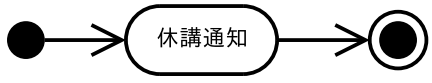


図 3.6: 休講通知アクティビティ図

休講通知帳票の担当者権限を表 3.7 に示す.

表 3.7: 休講通知担当者

担当者	権限
教員	申請
教務課	受理

帳票に必要な項目を表 3.8 に示す.

表 3.8: 休講通知テーブル

属性名	カラム名	制約	データ型
科目名	KAMOKU	NOT NULL	文字列
月 (休講)	MON_K	NOT NULL	文字列
日 (休講)	DAY_K	NOT NULL	文字列
時限 (休講)	TIME_K	NOT NULL	文字列
月 (補講)	MON_H	NOT NULL	文字列
日 (補講)	DAY_H	NOT NULL	文字列
時限 (補講)	TIME_H	NOT NULL	文字列

3.4.5 点数登録

点数登録は，教員の帳票提出 → 教務受理である．点数の訂正は，再提出で行うことができる．点数登録のアクティビティ図を図 3.7 に示す.

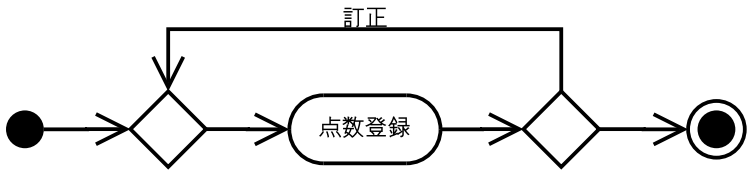


図 3.7: 点数登録アクティビティ図

休講通知帳票の担当者権限を表 3.9 に示す.

表 3.9: 点数登録担当者

担当者	権限
教員	申請，訂正
教務課	受理

このサブプロセスに必要な帳票の項目を表 3.10 に示す.

表 3.10: 点数登録テーブル

属性名	カラム名	制約	データ型
講義番号	KOUGINO	NOT NULL	文字列
学生番号	GAKUSEINO	NOT NULL	文字列
学生氏名	GAKUSEINAME	NOT NULL	文字列
点数	TENSUU	NOT NULL	整数

第4章　ワークフローの実装

本章は，前章で設計したワークフローを Cosminexus ワークフローシステムを用いて実装する．4.1 節では，Cosminexus ワークフローシステムのビジネスプロセスの仕様について述べ，4.2 節で設計と実装の対応関係を表しながら説明する．

4.1　Cosminexus ワークフローシステムの仕様

4.1.1　ビジネスプロセスの仕様

ワークフローシステムにおけるビジネスプロセスは，業務の流れを定義した情報で，ワークフローの機能で中心的な役割を担う．ビジネスプロセスの基本的なステップは次の通りである．ビジネスプロセスはソースノードから開始され，定義された遷移 (アロー) に沿って業務ステップへ進む．この業務ステップ内では，業務の条件 (担当者，業務開始・終了)・処理期限・作業の処理が行われる．途中，制御ノードによって，業務ステップが分岐・接合される．最終的に，シンクノードへ遷移し，一連のワークフロー処理は終了する．ビジネスプロセスのステップを図 4.1 に示す．

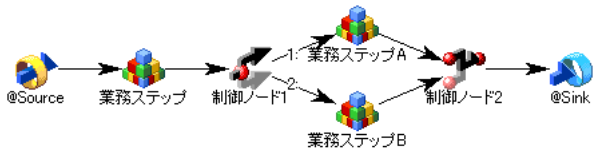


図 4.1: ビジネスプロセスのステップ

また，ビジネスプロセスは図 4.2 に示すように，階層的に構成され，図中の構成要素を次に説明する．

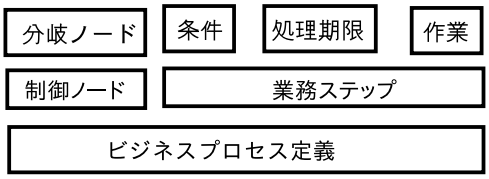


図 4.2: ビジネスプロセスの構成

- 業務ステップ

  業務の途中段階の状態を示す要素である．1 つの業務ステップに対して 1 つ以上 の作業が含まれ，内部に定義された作業がすべて完了することで，自動的に完了状態になる．しかし，独自に定義する「完了条件」というデータ条件を持つこともできる．各業務ステップの実行順序は，アローによって決められる．各業務ステップに対して，入力と出力を一つずつ与えることができる．

- 作業

  人やコンピュータによって実行される具体的な処理である．作業は業務ステップ内の要素として定義され，各作業の実行順序は特に指定しない．業務ステップが実行されると，その業務ステップに含まれるすべての作業が同時に実行される．また，作業は「発生条件」と「完了条件」の 2 つのデータ条件を持つことができる．この条件の設定によって，作業の発生と完了のタイミングを制御することも可能である．WorkCoordinator は，あらかじめ規定の動作をする「組み込み作業」を提供しており，作業は，振り分けルール定義と作業アプリケーション情報を含む．振り分けルール定義を使って作業者を割り当てたり，作業アプリケーション情報を使って実行形式ファイルを自動起動させたり，分散オブジェクトを実行させたりすることができる．

- 制御ノード

  案件の流れを制御するノードである．制御ノードには，分岐，分業，先着，待合の 4 種類がある．このうち，分岐ノードにはデータ条件を含めることもできる．制御ノードを図 4.3 に示し，それぞれの制御ノードについて説明する．

図 4.3: 制御ノードの種類

  – 分岐ノード

    次の業務ステップとしてあらかじめ定義された複数の業務ステップから，条件に従って一つの業務ステップを選択し，開始する．

  – 分業ノード

    次の業務ステップとしてあらかじめ定義された複数の業務ステップをすべて開始する．

  – 先着ノード

    直前の業務ステップとしてあらかじめ定義された複数の業務ステップのうち，どれか一つが完了した時点で次の業務ステップを開始する．

– 待合ノード

直前の業務ステップとしてあらかじめ定義された複数の業務ステップのうち，すべてが完了した時点で次の業務ステップを開始する．

- アロー

アローは，業務ステップ，制御ノード，ソースノード（案件の始点），又はシンクノード（案件の終点）に対して 1 対 1 の関係を定義する要素である．各アローには遷移元と遷移先がある．

また，作業においてワークフローシステムでは，ビジネスプロセス定義時に各作業の担当者を直接指定することはない．このため，担当者を決定するためのルールだけを指定しておき，作業の実行時にルールの内容を適用することで担当者を決定する．このルールを振り分けルールといい，次にこの定義について説明する．

4.1.2 振り分けルール定義

振り分けルールは，ビジネスプロセスとは別に定義する．振り分けルールを定義したものを振り分けルール定義と呼ぶ．

ビジネスプロセス定義時には，各作業の作業者情報として振り分けルール定義の名称を指定しておく．作業の実行時には，この名称で示された振り分けルール定義が呼び出される．呼び出された振り分けルール定義の内容に基づいて，作業者の情報が格納されている担当者データベースなどが検索され，実際の担当者が決定する．

振り分けルール定義で定義する帳票の権利を表 4.1 に示す．

表 4.1: 振り分けルールで定義する帳票の担当者と権限

帳票	担当者	権限
履修届	学生，教務課	申請，受理
再履修届	学生，教員，教務課	申請，承認，受理
受講者通知	教務課，教員	申請，受理
休講届	教員，教務課	申請，受理
点数登録	教員，教務課	申請，受理

4.1.3 RDB

ワークフローシステムが RDB と連携し処理を進める上で，作成するファイルには次の 2 種類がある．

- 外部論理データ

  論理データ項目としてインポートされ，業務設計段階で物理的な情報が決まっていないデータ条件・ルールに使用する．

- 論理・物理対応情報

  データモデルの論理モデルと物理モデルとの対応情報である．

ドメインモデルデータと RDB の対応関係を表したのが図 4.4 である．この図は履修届を例としてアクティビティ図で表している．履修届が申請されると，履修届を管轄する教務に遷移し，申請の結果と RDB の対応関係の処理が行われ一連の処理が終了する．

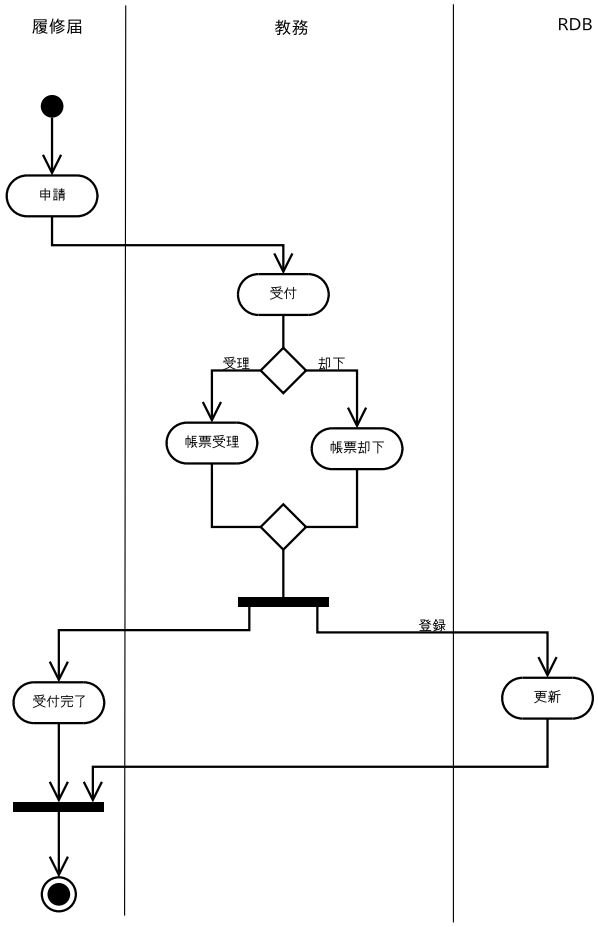


図 4.4: ドメインモデルデータと RDB の対応関係

次に，RDB の組織およびユーザデータテーブルの内容を表 4.2 に示す．ドメインモデルはこのデータを参照することによって構成され，ワークフローシステムは，この表をもとにして履修プロセスで使用するファイルを作成する．

表 4.2: 組織/ユーザデータ (一部抜粋)

組織名	研究科/課名	専攻/係名	ユーザ名	ユーザ ID	役職/課程
北陸先端大	情報科学研究科	情報処理学	小野 寛晰	I00201	教授
			浅野 哲夫	I00202	教授
			能美 一郎	I00501	後期課程
			能美 二郎	I00502	前期課程
		情報システム学	金子 峰雄	I00301	教授
			平石 邦彦	I00302	教授
			能美 三郎	I00503	後期課程
			能美 四郎	I00504	前期課程
	学生課	教務係	担当係員 A	I01401	係員
		学生係	担当係員 A	I01421	係員

4.2 ビジネスプロセスの記述

ビジネスプロセスの定義および振り分けルール定義は，図 4.5 に示す WorkCoordinator Definer (以下，Definer) を使用し記述する．

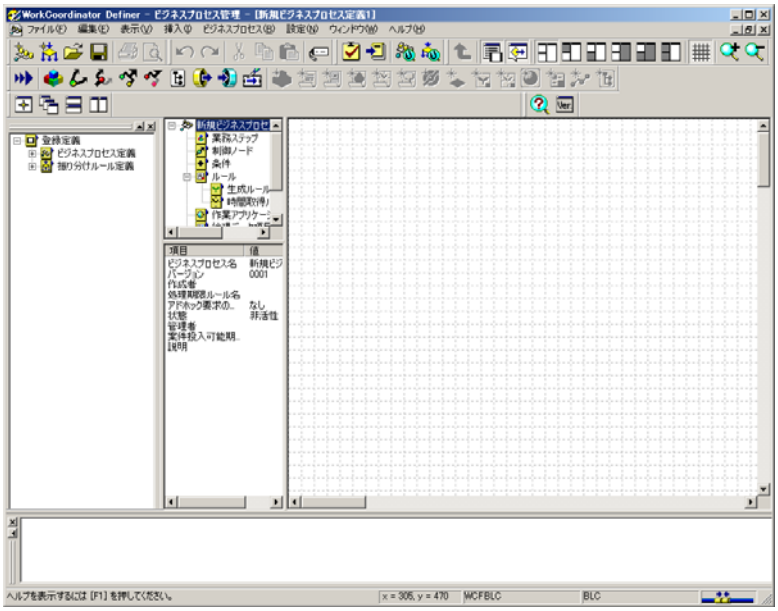


図 4.5: WorkCoordinator Definer

この Definer の機能には次の 2 点がある．

- ビジネスプロセス管理

  ビジネスプロセスを定義し，それらの運用属性を操作する．

- 案件運用操作

  案件の運用状況を監視したり，必要に応じて案件，業務ステップ，作業，ビジネスプロセス定義，および振り分けルール定義の状態を操作する．

この機能のうち，前者のビジネスプロセス管理で業務の流れを定義し，WorkCoordinator Server に登録を行うことができる．後者の案件運用操作に関しては，本研究では取り扱わないので説明を省く．ビジネスプロセス管理には，次の機能がある．

- ビジネスプロセス定義機能

  業務の流れを定義したビジネスプロセス定義を作成する．また，ビジネスプロセス定義の運用属性を参照したり，変更することができる．

- 振り分けルール定義機能

  ビジネスプロセスの各作業の作業者を決定するためのルールを定義した振り分けルール定義を作成する．また，振り分けルール定義の運用属性を参照したり，変更することができる．

4.2.1 設計したビジネスプロセスの記述

記述の説明は，履修サブプロセスについて行なう．図 3.3 をもとに，業務ステップ，および業務ステップに含まれる作業の記述を行なう．この記述を Difiner で行なったものが，図 4.6 および図 4.7 である．

図 4.6: Definer で記述した履修サブプロセス

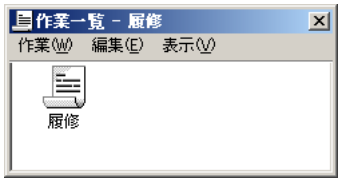


図 4.7: 履修サブプロセスの作業ウィンドウ

また，作業で使用するデータを格納する RDB テーブルを作成する．これを表 4.3 に示している．

表 4.3: 履修サブプロセスの RDB テーブル

名前	データ型	サイズ	制約
MON1	VARCHAR2	16	NULL 可
MON2	VARCHAR2	16	NULL 可
MON45	VARCHAR2	16	NULL 可
TUE1	VARCHAR2	16	NULL 可
TUE2	VARCHAR2	16	NULL 可
TUE45	VARCHAR2	16	NULL 可
WED1	VARCHAR2	16	NULL 可
WED2	VARCHAR2	16	NULL 可
WED45	VARCHAR2	16	NULL 可
THR1	VARCHAR2	16	NULL 可
THR2	VARCHAR2	16	NULL 可
THR45	VARCHAR2	16	NULL 可
FRI1	VARCHAR2	16	NULL 可
FRI2	VARCHAR2	16	NULL 可
FRI45	VARCHAR2	16	NULL 可

他のサブプロセスも同様にしてビジネスプロセス定義の記述を行なう．また，メインプロセスは，サブプロセスの階層構造となっている．各サブプロセスを記述したビジネスプロセスを図 4.8 から図 4.13 に示す．

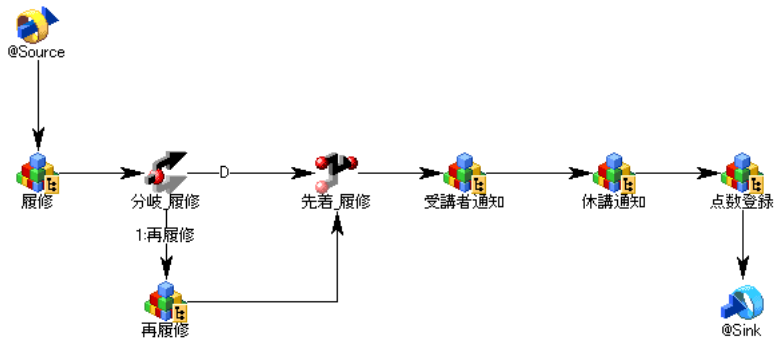


図 4.8: メインプロセス

図 4.9: 履修サブプロセス (再掲)

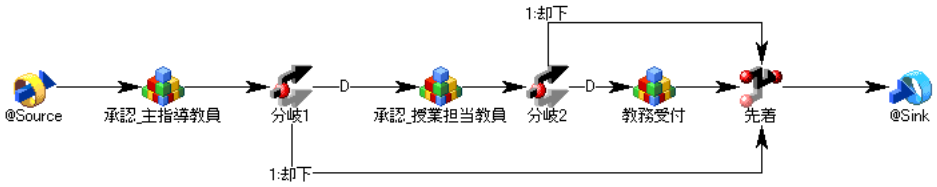


図 4.10: 再履修サブプロセス

図 4.11: 受講者通知サブプロセス

図 4.12: 休講通知サブプロセス

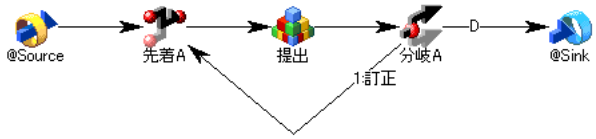


図 4.13: 点数登録サブプロセス

4.2.2 振り分けルール記述

履修届では，履修帳票を申請する担当者の振り分けルール定義を行う．振り分けルール定義は，RDB を用いて動的に担当者を割り当てる．このため，振り分けルール定義には SQL 文を入力する．SQL に指定できる項目は，基本的な項目 (テーブル，カラムなど) に

加え，各帳票に対応する担当者を動的に割り当てるために WorkCoordinator Server が用意している組み込み変数の指定を行う．この一覧を表 4.4 に示す．

表 4.4: SQL で使用する組み込み変数

組み込み変数名	意味
@PIName	案件名
@AIName	並列業務ステップ作業が子業務ステップを生成した場合に，子業務ステップに付ける属性
@WIName	並列作業が子作業を生成した場合に，子作業に付ける属性
@PIID	案件 ID

この作業を行ったのが図 4.14 である．

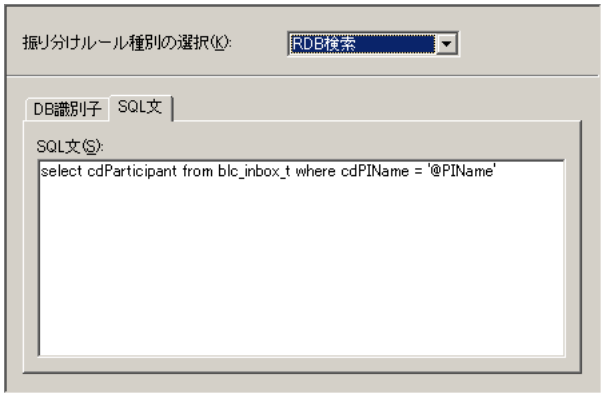


図 4.14: 振り分けルール定義ウィンドウ

図中の SQL 文は次の通りである．

```
select cdParticipant
from blc_inbox_t
where cdPIName = '@PIName'
```

これは，帳票データが納められている blc_inbox_t テーブルから，該当する案件名の担当者を選択するものである．また，他のサブプロセスにも同一の振り分けルールを定義する．

4.3 デプロイ

以上まで行ってきたビジネスプロセス記述の他に，各作業の帳票を作成する．そして，Web アプリケーションとしてワークフロー処理を行えるように，Web アプリケーションサーバに登録を行う．この登録をデプロイという．

デプロイには，ディレクトリ/ファイル配置の構成が定められており，この配置図を図 4.15 に示す.

フォルダ構成	格納物
webRishu	
/app	
/include	アプリケーション共通ファイル
/main	アプリケーションファイル
/form	
/_entryform	帳票登録情報ファイル
/common	帳票共通部品ウィジェット
/include	帳票用部品
/Hitachi	
/RegiRishu	帳票ファイル
/images	イメージファイル
/WEB-INF	
/lib	BLCライブラリファイル
/classes	
/jp	
/co	
/Hitachi	
/soft	
/blc	
/Hitachi	帳票classファイル
/src	
/jp	
/co	
/Hitachi	
/soft	
/blc	
/Hitachi	帳票javaファイル

図 4.15: 履修システムのディレクトリ構成

このように配置されたファイルを ear ファイルにし，Eclipse のプラグインである Server Plug-in を用いて Cosminexus アプリケーションサーバにデプロイする (図 4.16).

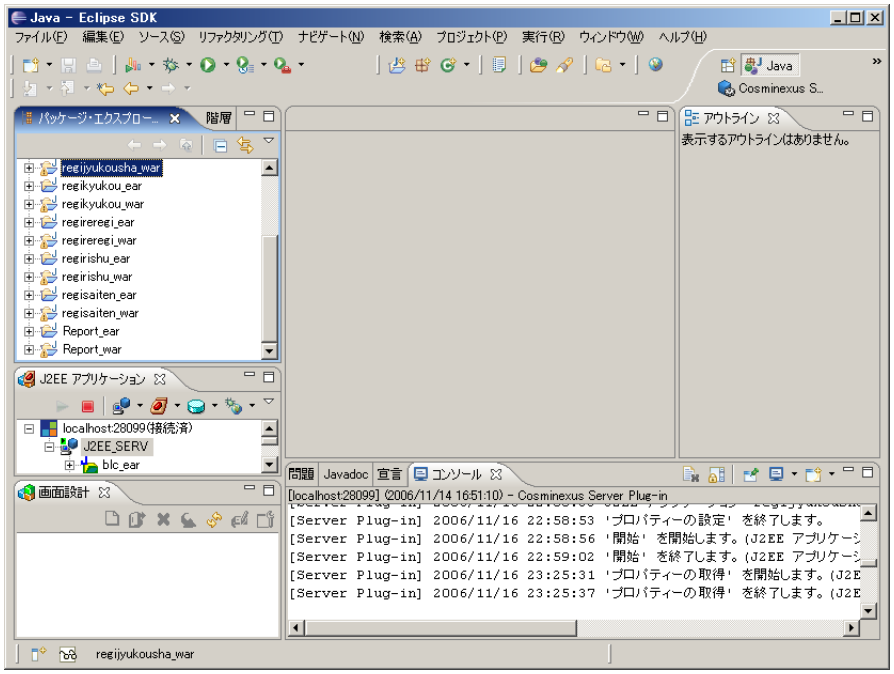


図 4.16: Eclipse Server Plug-in

そして，Web ブラウザで図 4.17 に示す Web ポータル画面を開き，帳票の記入・回覧等を行う．

図 4.17: ポータル画面

帳票が回ってきた日時は，帳票履歴ページから確認することができる (図 4.18)．

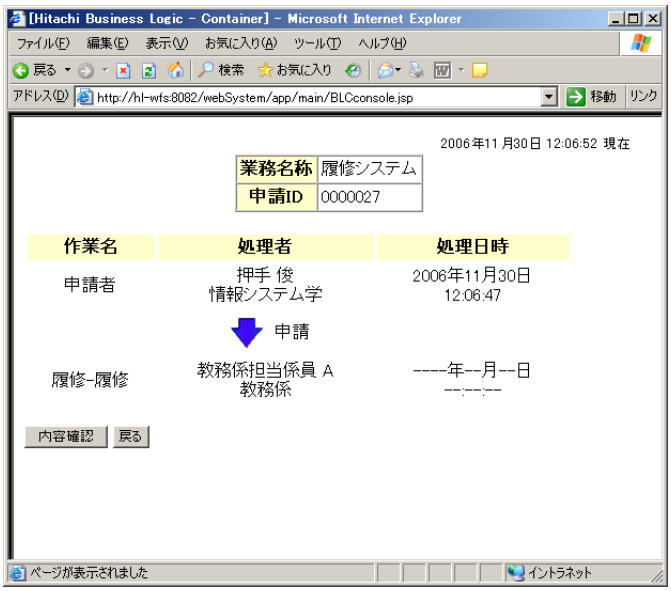


図 4.18: 帳票履歴

第5章　ワークフローのモデル化

5.1　方法

ワークフローのモデル化には，ペトリネットを用いる．ペトリネットは図的に表現されるため，分散システム，コンカレントプログラムの検証に利用されている．ワークフローは離散事象システムととらえることができ，ペトリネットの表現力と解析力でワークフローを検証することに適している [13]．そして，ペトリネットのツールを利用することでワークフローのペトリネットモデルのシミュレーションおよび検証が可能である [2]．本研究では，ツールとして CPN Tools を利用し，このツールに適応する検証方法のモデル化を行う [11].

まず，5.2 節にてペトリネットの基礎について触れ，5.3 節で高水準ペトリネットであるカラーペトリネットを説明する．そして，CPN Tools 上のモデルについて検証および評価を行う．

5.2　ペトリネット

C.A. Petri によって提唱されたペトリネットは，次の表現が可能なモデルである [4,12].

- 数学的なモデル
- 図的なモデル
- 実行可能なモデル

ペトリネットの基本となるネットは，プレースとトランジションからなる，システムのイベント関係を表すプレース/トランジションネット (P/T net) である．

5.2.1　ペトリネット表現

- 形式的表現

  ペトリネットは 4 項組 $PN = (P, T, A, M_0)$ である．
  ここで，

$P = \{p_1, p_2, \ldots, p_m\}$	プレースの有限集合
$T = \{t_1, t_2, \ldots, t_n\}$	トランジションの有限集合
$A = (P \times T) \cup (T \times P)$	アークの有限集合
$M_0 : P \to \mathbb{N}$	初期マーキング
$P \cap T = \phi$ でかつ $P \cup T = \phi$	

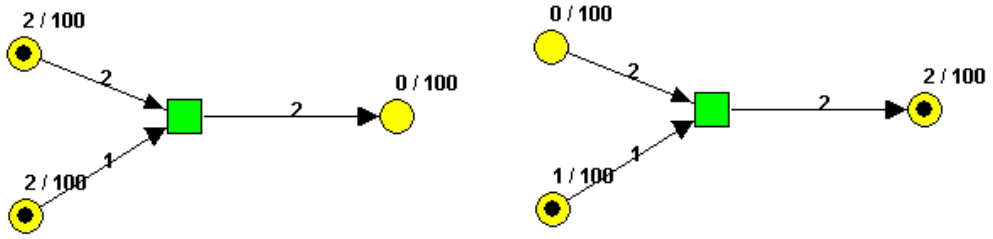
である．初期マーキングを除いた $N = (P, T, A)$ をネット構造という．

- グラフ表現

  ペトリネットは，プレースとトランジションの 2 種類のノードを持つ有向 2 部グラフである．“◯” はプレース，“□” あるいは “|” はトランジション，“→” はアークを表す．ペトリネットでは，現在の状態をプレースにトークンを配置することで表す．この配置をマーキングと呼び，マーキングは，トランジションの発火により遷移する．

  トランジションは，トランジションのすべての入力プレースについて各プレースにおけるトークンの個数が，そのプレースからトランジションへの入力アークの本数以上であるとき，発火可能である．

  トランジションの発火により，その入力プレースのトークンはアークの本数だけ減少し，出力プレースのトークンの個数はアークの本数だけ増加する．このマーキングの推移を図 5.1 に示す．



(a) 発火前　　(b) 発火後

図 5.1: ペトリネットのマーキングの推移

5.3 Coloured Petri Nets

Coloured Petri Nets (以下，CPN) は，P/T net を拡張したものであり，高水準ペトリネットに分類される [9]．トークンにカラーをつけて，各カラートークンに属性を持たせ，発火規則を各カラーに独立に定めることができ，P/T net より簡潔な表現が行える [3]．CPN は 9 項組 $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ で表される．この定義は次の通りである．

(1) Σ は以下のカラー集合である.

型名 (カラー) が $int, string, bool, \cdots$ などからなる有限集合である.

(2) P はプレースの有限集合.

(3) T はトランジションの有限集合.

$P \cap T = \phi$

(4) A はアークの有限集合.

$P \cap A = T \cap A = \phi$

(5) N はプレースとトランジションの接続を表すノード関数であり，定義は以下の通りである.

$A \rightarrow (P \times T) \cup (T \times P)$

また $N(a) = (p, tr), p \in P, tr \in T$ のとき,

$In(a) = p$
$Out(a) = tr$

とする.

(6) C はプレースに属するカラーの集合を定めるカラー関数.

$P \rightarrow \Sigma$

(7) G はトランジションの発火を制御するガード関数であり，定義は以下の通りである.

$T \rightarrow expr_B$

$expr_B$ は $true$ と $false$ の 2 値を取る論理式である.

(8) E はアークの発火条件を表現するアーク関数.

型 t の多重式の集合を MS とすると,

$$MS = \{\sum_{e \in S} n_e `e | s \subseteq t, n_e \in expr_{int}, e \in expr_t\}$$

t をこの多重式の型と呼び，n_e は多重式の係数である.

$$A \rightarrow MS_\Sigma$$

$$MS_\Sigma = \bigcup_{t \in \Sigma} MS_t$$

$E(a)$ の型と $In(a)$ の型，または $E(a)$ の型と $Out(a)$ の型が同じでなければならない．

$In(a) \in P$ のとき，$Type(E(a)) \in Type(In(a))$.
$Out(a) \in P$ のとき，$Type(E(a)) \in Type(Out(a))$.

(9) I はマーキングを定める初期関数.

MS_σ 中に，変数を含んでいないものを MS'_σ とすると，
$P \rightarrow MS'_\Sigma$
$I(p) = MS'_\Sigma$ のとき，$C(p) = t$ となっていなければならない.

プレースに属するカラー集合の要素と，アークに与える発火の条件であるアーク関数の記述を同時に与える．また，トランジションにガード関数を与えることにより，ガード関数の条件が真であるときのみ発火させることができる．あるトランジションにおいて，トランジションに向かって接続しているすべてのプレースの持つトークンがアーク関数の条件を満たしているとき，そのトランジションは発火可能であるという．トランジションが発火すると，トランジションに向かって接続しているすべてのプレースのトークンがアーク条件に従って失われると同時に，トランジションから出て接続しているすべてのプレースのトークンがアーク条件に従って加えられる．

本研究で CPN を用いる理由を次にあげる．

- ワークフローはプロセスが並行して処理されるため，並行性を表現し解析することが必要である．
- 階層プロセスおよび並行プロセスの表現は，階層的なネットの記述を CPN に導入した階層的 CPN (HCPN) によって解析することができる．
- CPN のシミュレータツールを使用することにより，P/T net の到達可能グラフに相当するオカレンスグラフを生成し，それによって，システムの動的な挙動に関する性質を検証可能である．

5.4 CPN Tools

CPN Tools は，CPN を解析するためのツールである [1]．次のような特徴があげられる．

- CPN の定義を GUI で行なうことができる．
- 定義した CPN は，関数型言語である ML を拡張した CPN ML に翻訳された後に実行される．実行過程におけるトークンの遷移は画面上で確認できる．CPN ML が備えている抽象データ型の機能を利用したカラー集合の定義が可能である．
- トランジションに CPN ML のコード記述ができる．これは，そのトランジションが発火したときに実行する．

- トランジションが発火してから出力プレースにトークンが出現するまでの時間を定義できる．この時間は実時間ではなく，トークンに付随する数値として与えられる．

5.5 CosminexusからCPNへの変換

Cosminexus から CPN へ変換するための規則を定義する．Cosminexus と変換規則で生成される CPN の対応を表 5.1 に示す.

表 5.1: Cosminexus から CPN への変換規則

Cosminexus	CPN
アクティビティ	プレース
帳票操作	トランジション
担当者，帳票項目データ，帳票遷移	トークン
帳票項目	カラー集合
振り分けルール	ガード関数，ノード関数

また，制御ノードの変換に関して図 5.2 に表す.

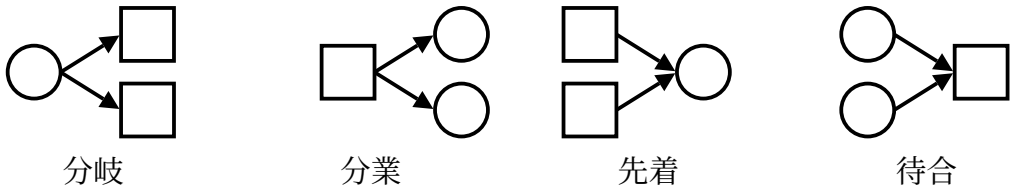


図 5.2: 制御ノードのペトリネット表現

帳票データは，ワークフローの各帳票で使用するデータから CPN のカラー集合へ変換する．履修帳票は次のように表現される．カラー集合 FORM は文字列型と文字列型の積のデータ構造になっており，曜日および時限ごとの履修科目を扱うことができる.

```
colset FORM = product STRING * STRING;
val regfrm = (``DAYTIME'',``Kamoku'')
```

5.5.1 階層構造

ビジネスプロセスの設計では，プロセスをメインプロセスとサブプロセスに分けた．これは大まかなプロセスと具体的な作業を表すプロセスを表現するために，階層化の概念を取り入れた.

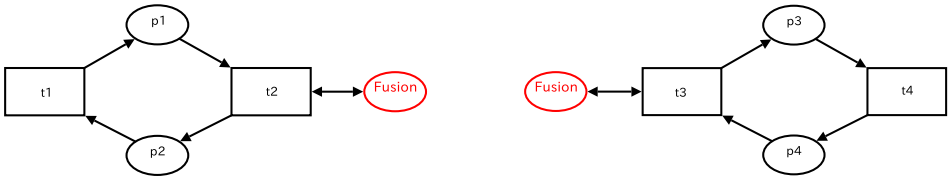
これまで述べてきたペトリネット，CPN には，このような階層化の概念がない．階層化の概念を採り入れたモデルが HCPN である．HCPN を用いることにより，サブプロセスを階層に持つビジネスプロセスの記述が行える．また，HCPN はひとつの CPN へまとめることが可能なため，CPN と同様の解析能力を持つ．

CPN Tools での記述は，5 つのサブプロセス (履修届，再履修届，受講者通知，休講通知，点数登録) をそれぞれ別のトランジションで表す．すなわち，メインプロセスは，この 5 つのトランジションを接続した構造になっており，メインプロセスのトランジションが発火するとトークンは推移し，サブプロセス内の処理が実行される．そして，サププロセスの処理が終了すると，メインプロセスに戻りトークンは次のプレースへ遷移する．

5.5.2 RDB のデータを表現する Fusion プレース

RDB のテーブル項目や帳票の参照のような，共有データを参照する表現が必要である．この表現に用いられるのが Fusion プレースである．例を図 5.3 に示す．subnet A と subnet B は共にプレース Fusion を共有しているとする．トランジション t2 が発火すると，トークンはプレース Fusion およびプレース p2 に入る．これと同時に subnet B のプレース Fusion へ同じトークンが入る．

Fusion プレースには Fusion の印付があり，この印付のあるプレースは同一として扱われる．



(a) subset A　　(b) subset B

図 5.3: 共有データを有するプレース

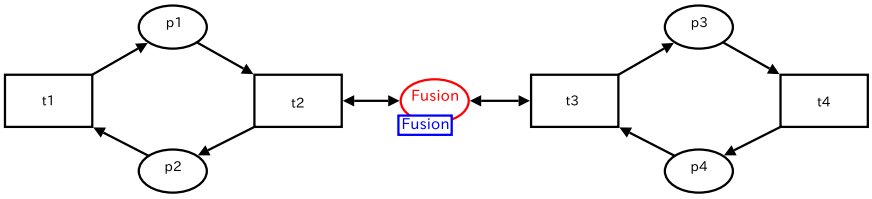


図 5.4: Fusion プレース

5.5.3 振り分けルール

振り分けルールには，作業の担当者別に次にあげるものがある．

- 学生の権限
  - 履修届の申請
  - 再履修届の申請
- 事務の権限
  - 履修届の受理
  - 再履修届の受理
  - 受講者通知の申請
  - 休講通知の受理
  - 点数登録の受理
- 教員の権限
  - 再履修届の承認
  - 受講者通知の受理
  - 休講通知の申請
  - 点数登録の申請
  - 点数登録の訂正

この振り分けルールは各担当者の業務権限であり，この組織に属する担当者は，担当の権限を越えて作業は行うことはできない．これらの内規の表現は，各帳票の権限が必要となるトランジションにガード関数を用い，条件が真ならば発火できるようにする．履修帳票の例であれば，申請を表すトランジションで権限のチェックを行い，学生以外の権限を持つ担当者が履修申請をすると発火不能となる．

振り分けルールについて，ワークフローシステムで利用する RDB から CPN の変換について説明する．この例は履修サブプロセスの申請の部分について，RDB の SQL を CPN のガード関数に変換する．SQL の内容を次に示す．

```
select cduser
from blc_user_info_t
where cdjobtitle = '00710' or cdjobtitle = '00720'
```

これは，組織/ユーザデータが納められている blc_user_info_t テーブルから，申請が許可されている担当者を選択するものである．where 句で指定している cdjobtitle 列は，役職や課程のデータが含まれている．ここでしている 00710 および 00720 は，それぞれ博士後期課程，博士前期課程の学生であることを表す．

次に，SQL からガード関数へ変換した図を図 5.5 に示す．プレース RID_1 には帳票の申請を行おうとする担当者のトークン情報が，プレース RDB にはその申請に対して許可されている担当者，すなわち学生のトークン情報が含まれている．そして，トランジション Login_1 は，2 つのプレースのトークン情報が合致した場合のみ発火が可能となり，申請を行なう担当者のを振り分けが可能となる．

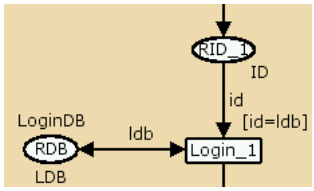


図 5.5: ガード関数による振り分けルール表現

5.6 モデルの記述

変換規則を用いて CPN Tools でモデルを記述した．履修システムのメインプロセスを図 5.6 に示す．この記述では，各トランジションが階層構造になっており，トランジションがサブプロセスに対応している．

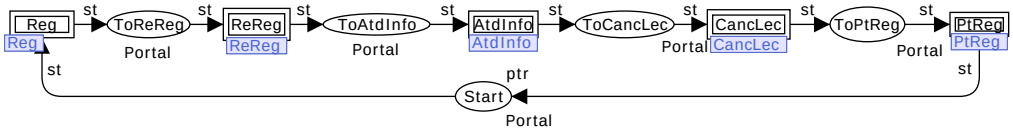


図 5.6: メインプロセス

履修サブプロセスを図 5.7 に示す．履修サブプロセスの流れは，まずログイン ID を入力し，その ID が学生の権限を持つものであればトランジションが発火できる．そして，帳票表示・入力を行ない，帳票の受理では，教務課所属の担当者のみがその帳票の受理を行なうことができる．そして，受理されると履修サブプロセスは処理を終え，メインプロセスの次のプレースにトークンが推移する．

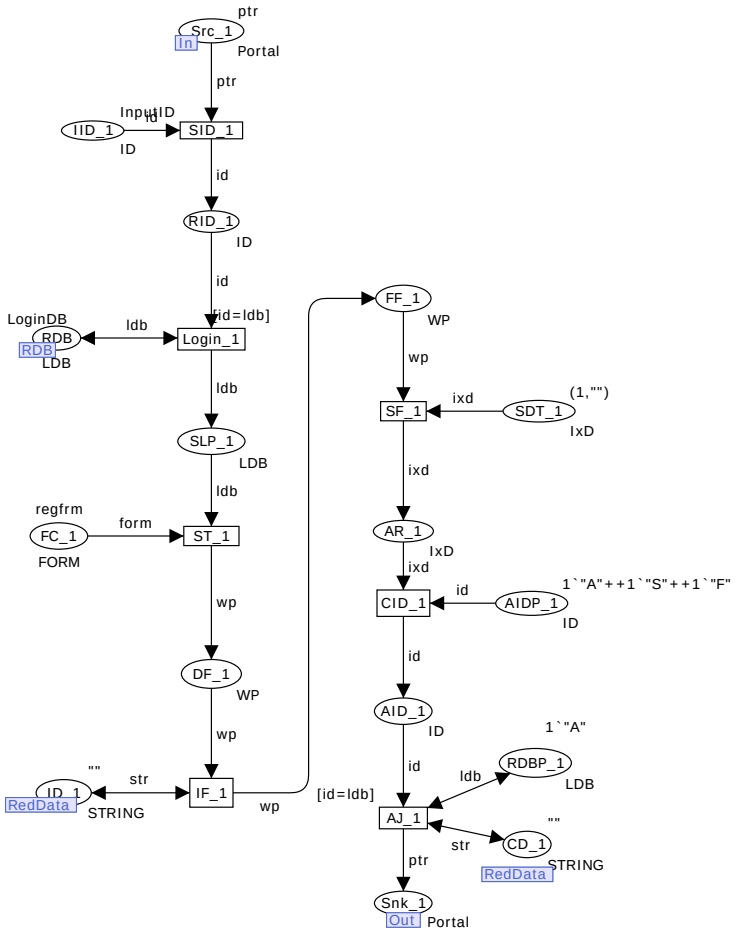


図 5.7: 履修モデル

再履修サブプロセスを図 5.8 に示す．再履修サブプロセスの流れは，履修サブプロセスと同様に，学生が申請を行なうことができ，まずこの権限を確認する．そして，帳票の表示・記入が終わると主指導教員および講義担当教員へ帳票が回覧される．いずれかの教員から承認が得られないと処理が終了する．2 教員から承認されると，帳票が送信され，受理者である教務課の担当者の権限を確認した後，帳票が受理され，再履修サブプロセスの処理が終了する．

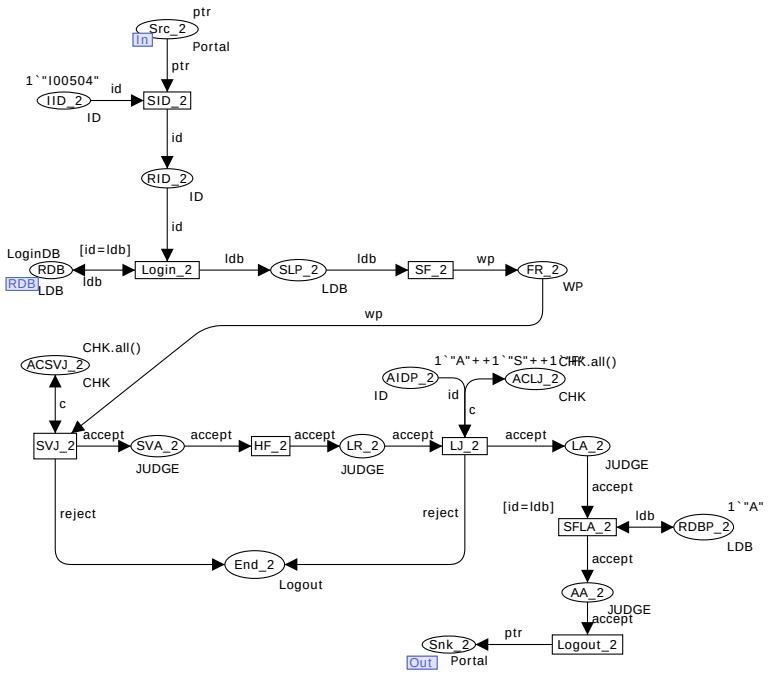


図 5.8: 再履修モデル

受講者通知サブプロセスを図 5.9 に示す．受講者通知サブプロセスは，教務課に所属する担当者であるか確認し，帳票の表示・記入を行なう．そして，受理者の権限が教員であることを確認して，帳票を開示することができ，このサブプロセスの処理を終える．

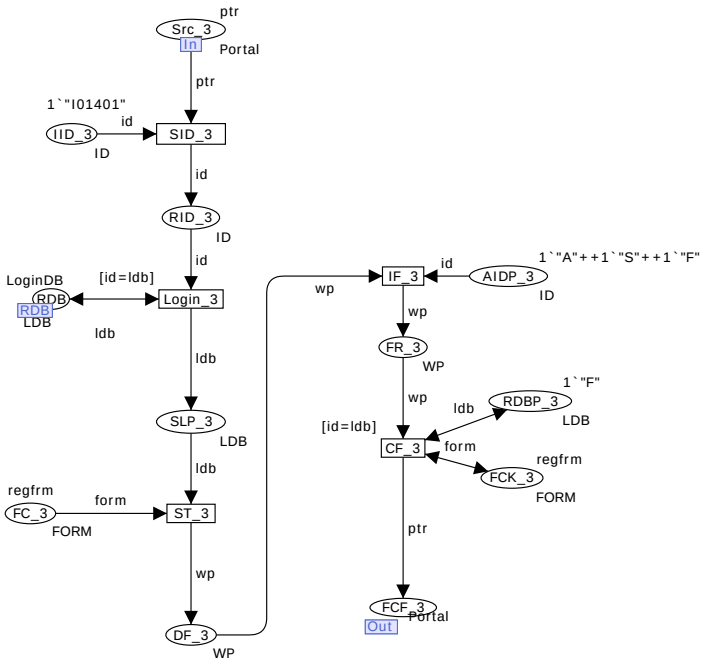


図 5.9: 受講者通知モデル

休講通知サブプロセスを図 5.10 に示す．休講通知サブプロセスは，申請者が教員であることを確認し，帳票の表示・記入を行なう．そして，受理者の権限が教務課の担当者であることを確認して，帳票を受理し，このサブプロセスの処理を終了する．

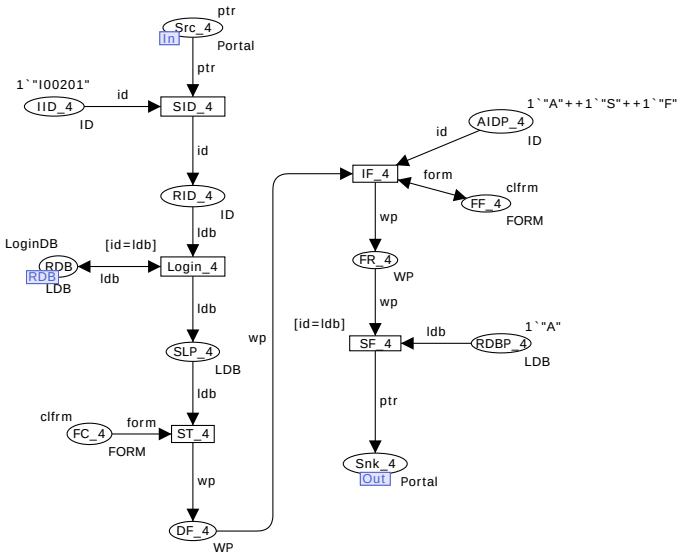


図 5.10: 休講通知モデル

点数登録サブプロセスを図 5.11 に示す．点数登録サブプロセスは休講通知サブプロセ

スと同様に，申請者が教員であることを確認し，帳票の表示・記入を行なう．そして，受理者の権限が教務課の担当者であることを確認して，帳票を受理し，このサブプロセスの処理を終了する．

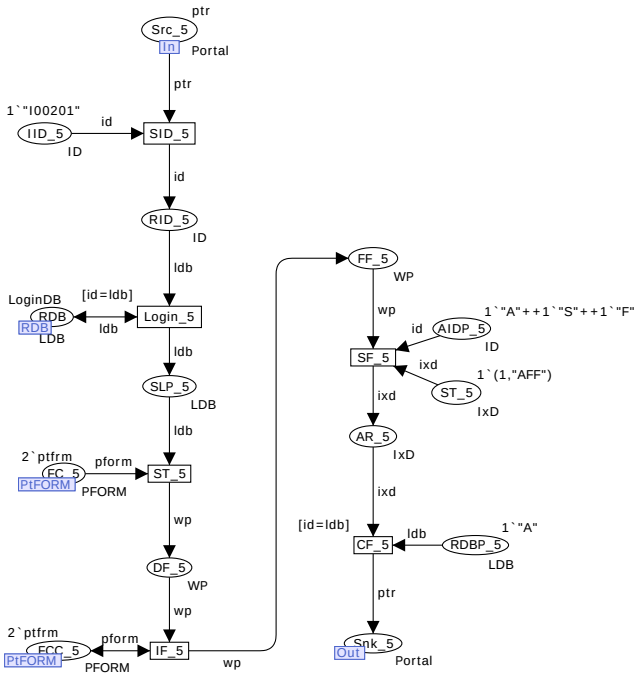


図 5.11: 点数登録モデル

第6章　モデルの検証

6.1　一般的性質の検証

生成したオカレンスグラフによって，到達可能性，およびデッドロックの検証を行なうことができた．

到達可能性は，到達可能グラフによってトランジションを発火させた場合のマーキングの変化を初期マーキングから到達可能な全てのマーキングを得られた．

デッドロックについては，到達可能グラフにおける活性の判定により，デッドノード，つまりそのマーキングにおいて，それ以上トランジションが発火不能であるというノードを判定することができた．このノードとは，図 6.1 に示す担当者の権限が帳票の処理を行なおうとすることによって，デッドロックになった．

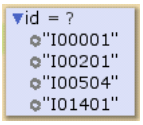


図 6.1: 担当者割り振りの選択

CPN Tools ではデッドロックになると図 6.2 の表示が現れる．

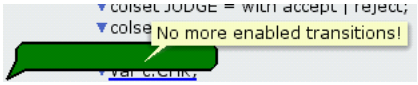


図 6.2: デッドロックを現す表示

6.2　特定の仕様における帳票権限の検証

ビジネスプロセス設計において，人為的なミスをフォローする処理が作りこまれていないと次のような例外処理が起こる．なお，各帳票における権限は，表 3.1, 3.3, 3.5, 3.7, 3.9，および 5.5.3 節に示した通りである．

- 帳票の宛先ミス

  帳票の宛先を誤ると，ワークフローシステムやビジネスプロセス定義は正しい処理であっても，担当者が意図する作業を進めることができない．

- 担当者権限の逸脱

  誤った宛先の帳票を受信した担当者がその帳票の権限を有さなくても，帳票の承認を行なうことができる．

上記の項目について検証するにはCTLモデル検査を行なう [6,15].

6.2.1 CTLモデル検査

CTLモデル検査をCPN Tools上で検査する環境がASK-CTLである．CTL式は次のものがある．アーク関数AFは，アークに関してのbool値を取る．状態論理式MODALは，トランジションに関してのbool値を取る．状態論理式POSは，あるパス中でいずれかの状態においての値を取る．また，CPNの状態空間においてモデル検査を行なうための記述は次の通りである．なお，PageName, PlaceName, TransNameは，それぞれCPN Tools上のページ名，プレース名，トランジション名を表す．

- プレース

  PI.<PageName>'<PlaceName>

- トランジション

  TI.<PageName>'<TransName>

- マーキング

  Mark.<PageName>'<PlaceName>

- 束縛要素

  Bind.<PageName>'<TransName>

6.2.2 CTLを使用した検証

ASK-CTLを用いて次の2例について検証する．

1. 履修届サブプロセスにおいて，学生が帳票を提出し，教務係が受理を行えるかの権限について，下記のASK-CTLコードで評価する．

```
fun AuthCheck a =
 (Bind.Reg'AJ_1 (1, {id=``A'', ldb=``A'', str=``''}) =
 ArcToBE a);

val myformula =
 POS(MODAL(AF(``'', AuthCheck)));

eval_node myformula InitNode;
```

このコードについて説明する．関数 AuthCheck は，Reg ページの AJ_1 トランジションにおけるアークの束縛要素について値を返す関数である．引数の id および ldb の値は，A (affair), S (student), F (faculty) のいずれかを取る．

この結果を図 6.3 に示す．この権限は許可されているので，返り値は真である．

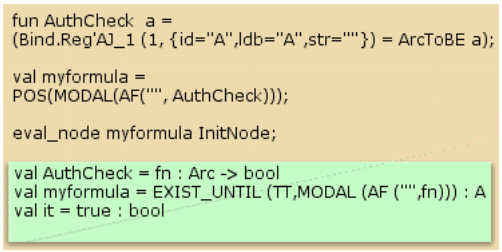


図 6.3: 履修届 ASK-CTL 実行画面

2. 点数登録サブプロセスにおいて，学生が帳票を提出し，教務係が受理を行なえるかの権限について，下記の ASK-CTL コードで評価する．

```
fun AuthCheck a =
 (Bind.PtReg'CF_5 (1, {id=``S'', ldb=``A'', ixd=(1,``AFF'')}) =
 ArcToBE a);

val myformula =
 POS(MODAL(AF(```, AuthCheck)));

eval_node myformula InitNode;
```

1 で評価した内容と同等であり，関数 AuthCheck は，点数登録帳票である PtReg ページの CF_5 トランジションにおけるアークの束縛要素について値を返す関数である．

この結果を図 6.4 に示す．学生の権限を持つ担当者は点数登録を行えないので，返り値は偽である．

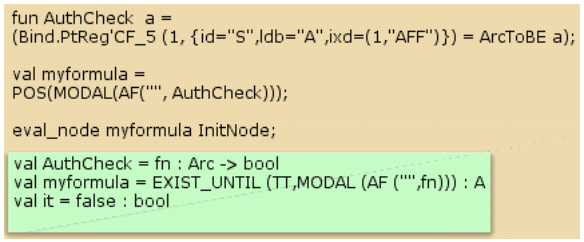


図 6.4: 点数登録 ASK-CTL 実行画面

6.3 考察

定義した変換規則を用い，ワークフローシステム上に記述したビジネスプロセスを CPN モデルに変換することができた．その CPN モデルをを CPN Tools で記述し，このモデビジネスプロセスがソースノードからシンクノードへ到達可能なこと，各サブプロセスの帳票に対する権限において担当者を識別し，申請・承認の可否を行なうか，についてのワークフロー処理をシミュレーションした．

担当者の権限について，ワークフローシステムでは RDB のデータ参照することによって，各帳票に対して動的に権限を許可している．この点に関して，6.2 節で検証した，人為的なミスをフォローするビジネスプロセスの定義が求められる．そのためにも，ワークフローシステムでビジネスプロセス記述を行なう前に形式モデルによる状態空間の解析を事前に行う必要がある．これによって全状態の推移を認識し，許可されない権限や担当者の配置を探索することができる．

第7章　おわりに

7.1　まとめ

ワークフローシステムを使用し，履修システムのビジネスプロセスを定義した．そして，この履修システムのモデルを CPN へ変換する規則を定義した．この変換規則を使用し，履修システムのビジネスプロセスを CPN モデルに変換を行い，シミュレーションした．

この提案した変換規則を使用することでワークフローシステムでビジネスプロセスを記述する前に，ビジネスプロセスと対応関係のとれる CPN モデルを作成し，事前に検証することが可能であり，提案手法の有効性を示した．

7.2　今後の課題

本研究ではワークフローから CPN への変換規則を定義した．今後の課題として，この変換規則を用いた，ワークフローシステムで記述したビジネスプロセスから CPN へ自動変換するツールがあると検証時間の削減が期待される．また，検証の正当性について言及しておらず，これについての証明が必要である．

謝辞

本研究を進めるあたり，終始変わらぬ御指導を賜わりました平石邦彦教授に深く感謝致します.

最後に，研究室の皆様には多くの助言，励ましを頂き有意義に研究に励むことができました．誠にありがとうございました.

参考文献

[1] CPN Tools. http://www.daimi.au.dk/CPNTools/.

[2] Wil van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, Vol. 8, No. 1, pp. 21–66, 1998.

[3] Wil van der Aalst, K.M. van Hee, and G.J. Houben. Modelling and analysing workflow using a Petri-net based approach. In G. De Michelis, C. Ellis, and G. Memmi, editors, *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pp. 31–50, 1994.

[4] 青山幹雄, 内平直志, 平石邦彦. ペトリネットの理論と実践. 朝倉書店, 1995.

[5] David A. Chappell. エンタープライズサービスバス - ESB と SOA による次世代アプリケーション統合. O'reilly, 2005.

[6] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999.

[7] K. Hiraishi. Modeling and Verification of Workflows for e-Society, 2005. JAIST 21st Century COE Symposium 2005 "Verifiable and Evolvable e-Society".

[8] David Hollingsworth. WfMC: The Workflow Reference Model, 1995. http://www.wfmc.org/standards/docs/tc003v11.pdf.

[9] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts.* Springer-Verlag, 1992.

[10] 戸田保一, 速水治夫, 飯島淳一, 堀内正博. ワークフロー - ビジネスプロセスの変革に向けて. 日科技連出版社, 1998.

[11] Irene Vanderfeesten, Wil van der Aalst, and Hajo A. Reijers. Modelling a product based workflow system in CPN tools. In *Proc. of CPN'05*, pp. 99–118, 2005.

[12] 奥川峻史. ペトリネットの基礎. 共立出版, 1995.

[13] 山口真悟, 葛崎偉, 田中稔. ペトリネットのワークフローへの応用. 計測自動制御学会第 30 回離散事象システム研究会講演論文集, pp. 9–16, 2002.

[14] 日本 BEA システムズ. SOA サービス指向アーキテクチャ - 企業システム全体最適化への実戦的アプローチ. 翔泳社, 2005.

[15] 米田友洋, 土屋達弘, 梶原誠司. ディペンダブルシステム - 高信頼システム実現のための耐故障・検証・テスト技術. 共立出版, 2005.

付 録A　ユーザ機能追加処理

ユーザ機能を追加することによって，帳票の処理を任意の条件に基づき変更することができる．次の例では，学生の権限を持つ担当者がログインすると点数登録を行えない処理を追加した．このフローチャートを図 A.1 に示す．

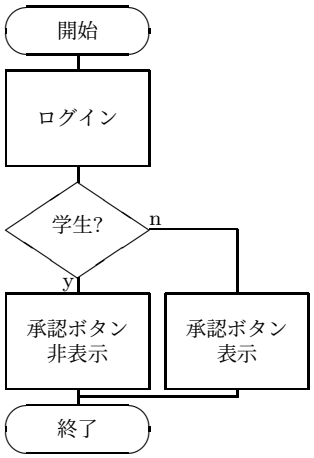
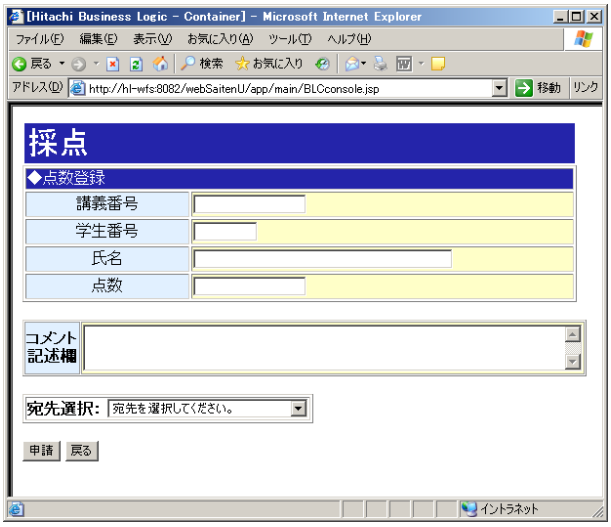
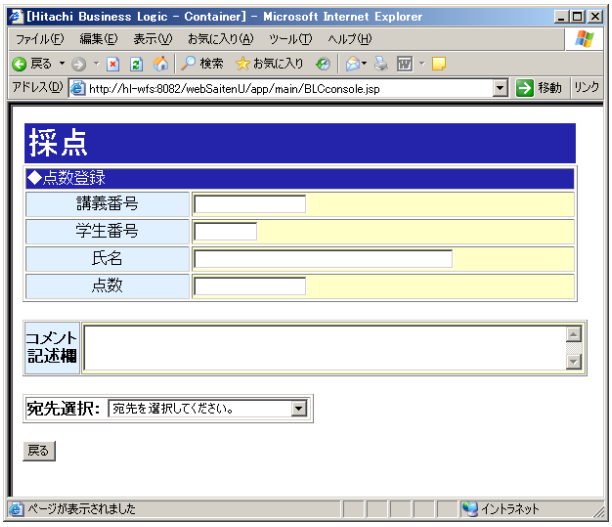


図 A.1: 点数登録帳票フローチャート

次に，ユーザ機能を付加した帳票を図 A.2 に示す．学生以外の権限を持つ担当者がログインの様子が図 A.2 (a) であり，帳票の承認を行なうことができる．また，図 A.2 (b) は，学生の権限である担当者がログインを行なうと，帳票の承認ボタンが表示されず，承認することができない．



(a) 承認ボタン表示



(b) 承認ボタン非表示

図 A.2: ユーザ機能付き帳票