Vol.2010-SE-171 No.31
2011/3/15

帳票の取り回しによる業務システムに最適化したユースケースとそれに適合した業務フロー図による要求分析手法

大内隆信[†1]、森下大介[†1]、丹郁夫[†1]、
寺町康昌[†2]、鶴保征城[†3]

ユースケース記述を帳票の取り回しによって運用される業務システムに最適化することを考えた。先ず、処理対象を帳票に限定することにより、処理が定型化できることが分かった。そのユースケース記述の考え方に準拠して、クライアントからの要求獲得を用意にする業務フロー図を考案した。業務フロー図は変換ツールにより、実行環境で実行可能となった。このような要求の分析手法について報告する。

Requirement Analysis Method for Enterprise Business System Development by using Usecase Descriptions and Business Process Flow Charts which are Optimized to Business Process with Ledger Sheets Handling

Takanobu Ouchi,[†1] Daisuke Morishita, [†1] Ikuo Tan, [†1]
Yasuaki Teramachi, [†2] and Seishiro Tsuruho[†3]

Business processes are realized by handling ledger sheets. Thus new type of usecase descriptions are reconsidered to just fit for this kind of business processes. In this case, processing object is limited to a ledger sheet. If object is limited, then process of object should be limited. Then, processes become stereotyped. Special business process flow chart is developed based on this new usecase description. This business process flow chart is just suitable for obtaining requirements from clients. This new system development method is introduced.

.

1. はじめに

基幹情報システムが利用されるビジネス環境は複雑化している上に、仕様変更が頻繁に行われ、さらに顧客の要求が曖昧なために、基幹情報システムの開発は困難になってきている．しかし経営の立場からは、迅速に業務運営が実施できるための情報システムが求められている．短期間でのシステム開発が可能で、仕様変更に対して、変更必要範囲を明確にし、必要工数を見積もることのできるシステム開発方式が求められている．一方、基幹業務システムは帳票の取り回しによって、業務を進めていくシステムである．実体として物やお金を直接扱うことはない．扱う対象を帳票に限定することによって、処理の種類は限定され、処理の仕方も定型化される．それによって、システムの完全性が保証できるようになる．処理対象を帳票に限定されたユースケース記述を **SVO Statement** と呼び、英文法の主語 **S** はアクタでは無く役割 **role** であり、述語 **V** は帳票に許される処理、目的語 **O** は処理対象である帳票である．この **SVO Statement** を簡単に生成できる業務フロー図であるシーケンス図は、プロトタイプ作成のための **Visual Programming Language** となっている．しかし、本方式の詳細分析以降の工程をシーケンス図作成によって得られた要求仕様を基にして、厳密に展開してゆくことによって、より精度の高い基幹業務システムの構築が可能である．同様な目的で BPMN を用いた研究もある 1)．

2. システム開発手法の構造

本方式は、図 1 に示すように分析・設計手法である方法論、各工程の成果物の導出・管理およびプロジェクトの管理をサポートするツールおよび分析・設計の成果物で構成されるアプリケーションの実行環境であるエンジンの３つの要素より構成される．

分析手法はシナリオ・ドメイン表の作成と、シーケンス図の作成の 2 段階に分かれている．ツールはプロトタイプ作成の場合は成果物の自動展開を行い、成果物の一貫性検証やプロジェクト管理の道具が用意されている．さらに、アプリケーションはプロセス、モデル、業務項目、インターフェイスおよびルールの 5 つの要素より構成される．これらは全て分析手法やツールの成果物であり、カタログとして纏められてい

[†1] アトリス株式会社
Atrris Corporation
[†2] 職業能力開発総合大学校
Polytechnic University
[†3] HAL 東京
HAL Tokyo

る．各要素の説明は表１に示す．

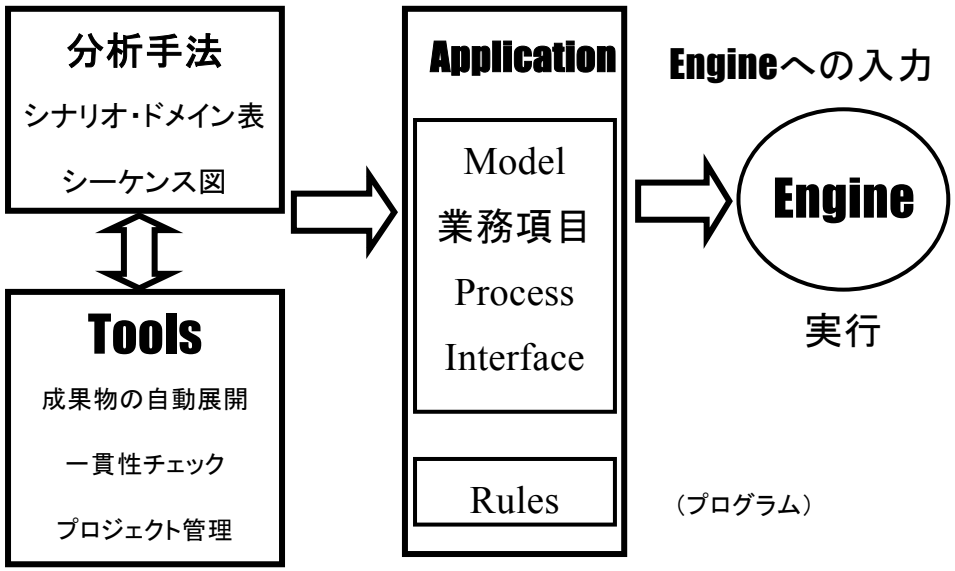

図　１　本開発手法の構造

これらの要素は全てカタログとして一括管理できる．そのため、どのプロセスがどのモデルを扱い、どの業務項目を利用しているか、どの画面に出力しているかがチェックできる．また、どこでも入力されないのに出力されている業務項目は、ルールとして計算されるべきものか、あるいはマスタとして何処で準備されているかのどちらかであることも確認できる．
また、プロセスでは汎用言語を利用することはできないので、処理のフローの流れを自由に制御することは許されない．しかし、ルールでは値だけを計算するので汎用プログラム言語を利用しても問題は発生せず、自由で豊富なプログラム機能を利用できるので、ルールのプログラミングには汎用プログラム言語が有効である．また、既存のルールエンジンを組み合わせて利用することも可能である．

表　１　アプリケーションを構成する 5 つの要素

要　素　名	意　　味
プロセス	帳票を処理してゆく手順を示す． どのモデル、業務項目をどのように扱うかが登録されている．
モデル	帳票を抽象化したもので、どのような個別のデータ（業務項目と呼ぶ）の構成によって出来上がっているかを示す
業務項目	業務中で取り扱われる情報の最小単位．プロセス、モデルから独立して、全体として一元管理される． 各々はテキスト・ウインドウで入出力するデータ、ラジオボタンのように一つの値を選択するデータ、複数の値を選択できるデータの 3 種類に分類できる 2)． また、テキスト・ウインドウで入出力するデータにはプログラム上の型も定義される．
インターフェイス	画面
ルール	その値が、他の業務項目値から自動的に計算される業務項目 プロトタイプの外にある．通常汎用言語を用いて実装される．

3.　システム開発工程

システム開発は、次の 5 つの工程に従う．

１．　業務分析
２．　詳細分析
３．　設計
４．　実装
５．　試験

通常のウォーターフォールモデルのやり方では要求分析、設計、実装、試験の分類になっている．しかし、業務の実態を正確に把握するには、業務分析に力を入れることが本質的に必要である．業務分析はシナリオ・ドメイン表作成、概略シーケンス図の作成、詳細シーケンス図の作成、プロトタイプの検証の 4 つの段階．詳細分析はプロ

セス分析、モデル分析、業務項目分析、ルール分析の 4 つの段階、設計工程はプロセス設計、インターフェイス設計、ルール設計の 3 段階、実装工程は、プロセス実装、インターフェイス実装、ルール実装の 3 段階に分かれている．すなわち、アプリケーションの 5 つの要素をエンジンへの入力可能な成果物に作り上げてゆくプロセスである．この中で、ルールはプロセスから完全に分離されて、他の業務項目値から計算によって決定される値なので､汎用プログラム言語で実装されることが多い.そのため、要求分析、設計、実装という一般的な開発工程を踏むことになる．モデルおよび業務項目の二つのデータ要素は詳細分析段階で設計に近いところまで確定され、設計、実装工程が存在しない．また、インターフェイスである画面の構成は、プロトタイプ用であれば、自動的に生成される．しかし、処理の度に新しい画面が生成されるような構成で、使い勝手が良くは無い．実用的なシステムを構築するにはしっかりした画面の設計、実装が必要になる．

3.1 一業務分析

業務側になればなるほど粒度を的確に扱うのが困難になる．本方式の業務フロー分析法も業務を実行する現場の人から見ると、ちょうど適した粒度になっている．しかし、経営層から見ると粒度が小さすぎると見做されるであろう．しかし、システム構築に直結するためには現在の粒度が最適である．業務の内容には関わらないレベルでシナリオ・ドメイン表を用いて第一段階での業務分析を行う．シナリオとは、組織内の他の影響を全く受けずに要求を発することができるサービスを言う．企業のシステムでは、社外の顧客による発注などがそれに相当する．サービス要求に対して、それを実現するために実施無ければならない業務をドメインと呼ぶ．現状のシステムの分析（AsIs 分析）を行う場合は、その企業における業務の粒度がこのドメインの粒度になる．しかし、次の段階のシーケンス図を描く段階では、そのドメインで主に扱っている帳票が確立するまでをドメインと考える．交通費申請システムでは、出張者が交通費申請書を作成し、その部局のセクレタリが受け付けて、部長が承認し、最後に社長が承認すると、その書類が有効になり、次のドメインである支払いドメインに引き渡されていく．作成から、社長承認までが一つのドメインを構成する．

交通費申請システムでは、交通費申請という単一のシナリオに対して、2 つのドメインという簡単な表になってしまう．そこで､図書館管理システムにおけるシナリオ・ドメイン表を表 2 に示す．シナリオを実現するのに必要なドメインのクロスするところに○をつけてゆく．例外的に発生するドメインには●をつけてゆく．業務分析の第一段階として先ず、シナリオ・ドメイン表が作られる．

次の業務分析工程の段階としてシーケンス図の作成段階がある．

基幹業務システムは帳票の取り回しによって、業務を進行させるためのシステムであると考え、帳票の取り回しに最適化した業務フロー図であるシーケンス図を考案した．システムとして扱うのは帳票の形をした情報であって、物や金という実態を扱うのでは無い．処理する対象を帳票に特化することにより、処理の仕方が限定され、少数の定型化された処理のみでシステム記述が可能となる．通常業務フロー図は処理

表 2　図書館システムにおけるシナリオ・ドメイン表

シナリオ＼ドメイン	利用者情報登録	発注先登録	書籍購入依頼作成	書籍発注作成	書籍寄贈受付	書籍入荷	請求受付	支払	紛失書籍抹消	書籍貸出作成	書籍返却	書籍予約依頼作成	書籍到着連絡	書籍予約貸出	書籍返却依頼作成	他図書館書籍取寄依頼	他図書館書籍取寄受付	他図書館へ返却	他図書館へ送付
利用者登録依頼	○																		
書籍購入依頼		○	○	○		○	○	○											
書籍寄贈依頼					○	○													
書籍紛失申告									○										
書籍貸出依頼									●	○	○				○				
書籍貸出予約依頼											○	○	○	○	○	○	○	○	
他図書館から検索依頼																			
他図書館への書籍予約依頼												○	○			○	○	○	
他図書館から書籍予約依頼												○							○

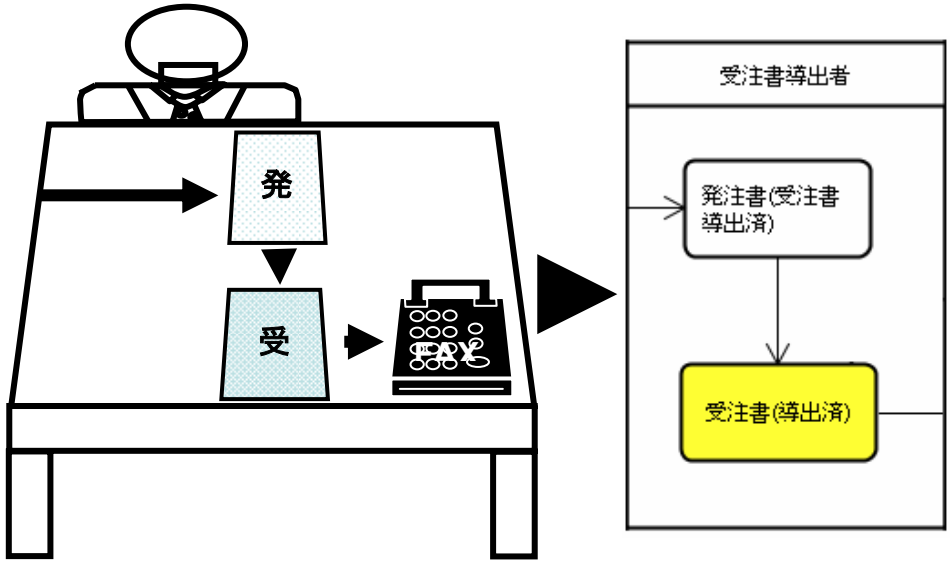

図 2　シーケンス図のイメージ

action を中心に記述されるが、本方式では処理対象である帳票を中心に記述する．例えば、ある会社の営業担当者が顧客からの発注書を受け取った場合、その発注書と顧客リストなどから与信額などを確認しながら受注書を作成してゆく．この様子を、図 2 の左の図に示してある．発注書は角丸の四角形で示し、それから受注書への矢印を描き、顧客リストからも矢印が発生している．シーケンス図は UML のアクティビティ図を作成するツールを用いて喜寿される．帳票はアクティビティ図の action を対応させている．また、帳票名に続いて、その帳票の状態が記述される．その role が終了した時点での、帳票の状態を記述する．図 2 の受注書は“導出済”になっている．その処理を担当する役割を受注書導出者のように、対象“受注書”を処理“導出”する“者”のように表現する．ここで、ある帳票を元に他の帳票を作る場合を導出すると呼んでいる．このように記述すると、実際の業務を担当している人から、どのように帳票を回しているのかという視点での要求獲得が非常にやりやすくなる．要求獲得において、文章によって記述するのではなく、シーケンス図の作成を行っている．要求分析者の文章力の違いにより、獲得された要求仕様の質にバラツキを生じる可能性を小さくすることが出来る．また、このように用語を厳しく限定することによって、業務フロー図であるシーケンス図から実行可能なプロトタイプを直接生成することができる．

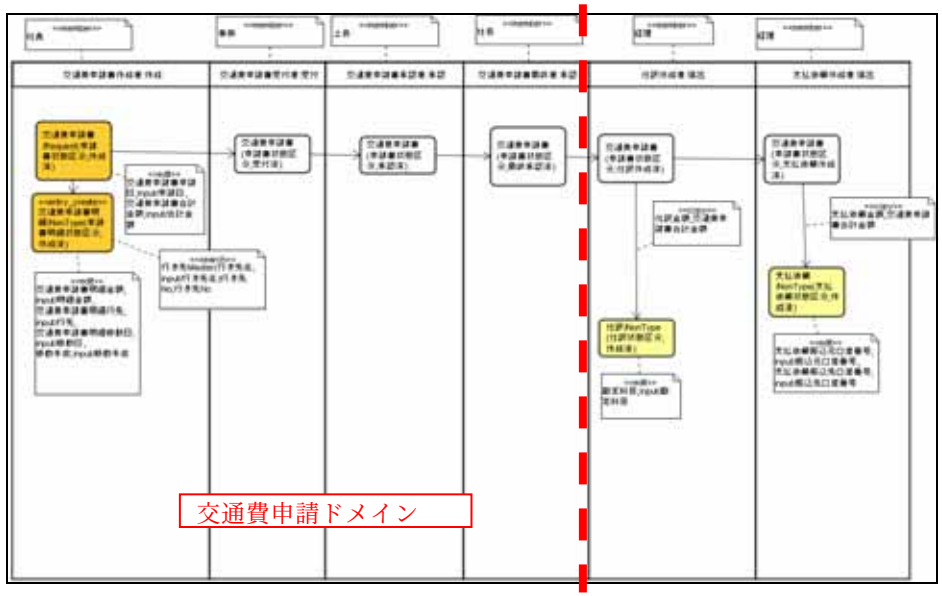

図 3　交通費申請システム

図 3 に交通費申請システムの交通費申請ドメインと支払いドメインを示す（簡単のために、交通費申請ドメインと支払いドメインを一つにして記述している．）縦のパーティションは一つの役割 Role を意味している．ちょうど図 2 の一つの机に相当していて、交通費申請書という帳票を作成するという処理をする役割という意味で“交通費申請書”を“作成”する“者”という役割名で呼ぶ．その名前を書くパーティションの最上段に記載してある．その情報にコメントとして“社員”という記載がされているが、これは参加者 member とし社員、社長などを具体的に記載されている．このように、役割に参加者を割り当てる方式では、ユースケース図で良く見られる、actor の汎化などを考慮する必要が無くなる．従来のユースケースでは、actor を中心に考えられてきた．しかし、M. Fowler はスウェーデン語の原典から英訳する際に、role とすべきところを actor に誤訳されたと指摘している 3)．この交通費申請書では role 数が 5 で、帳票数が 4、業務項目数が 22 であった．このように、actor ではなく role に徹底すると、業務の担当者が変更になった場合にも、業務フローには変化無く、参加者だけを変更すれば良いことになる．このような、担当変更は官庁でも頻繁に行われている．

このシーケンス図からユースケース記述の一つである SVO Statement を作成することができる．英文の SVO 形式でユースケースを記述する．主語 S は role で、述語 V は“どのような処理”をするかを示し、処理対象 O は帳票を示している．これはユースケース記述の一種であるが、処理対象 O を帳票に限定している．ファイルが対象であれば、CRUD(Create, Reference, Update, Delete)の 4 種類だけ良い．しかし、業務分析のためには、承認する(ACCEPT)や保留する(SUSPEND)などの 20 用語も必要である．role である S は ACTIVE、PASSIVE、REFERENCE と CONTROL の 4 つのロールカテゴリに分類でき、さらに、PASSIVE ロールは PRODUCE、ACCEPT などの 6 種類のタイプに分類できる．各々のロールタイプごとに実行できる処理が限定される．交通費申請の SVO S tatement の表を表 3 に示す．例えば、ACTIVE ロールでは、CREATE 作成という処理が中心となる．CREATE 作成が必要であれば、必ず UPDATE 更新と REMOVE 削除という処理が必要になる．同様に、ACCEPT に対しては、UNACCEPT、SUSPEND、UNSUSPEND、REJECT の 5 つの処理が必要であることを型紙として示している．もちろん、より細かな粒度の業務項目については<<edit>>で示された機能コメントで詳細に指示する必要がある．その点も図 3 に示されている（表の一部のみ表示）．しかし、業務の流れだけを記述し、コメント部は記載しない．業務フローだけを考える場合は、この方が見やすく、概略シーケンス図と呼ぶ．業務項目まで記載したものを、詳細シーケンス図と呼ぶ．プロトタイプ化するには、これが必要である．ここで、交通費申請書は複数回分の交通費を同時に申請できるようにするため、ヘッ

ダー部と明細部に分割されているので、二つの帳票を扱うように記載されているが、一つの書類として処理される．この作られた書類が交通費申請書作成者から交通費申請書受付者へ渡り受け付けられる．次に、交通費申請書承認者（上長）に渡って承認され、交通費最終承認者（社長）に渡って、この書類が確定し、次のドメインである支払いドメインに渡される．

表 3　SVO Statement の表（部分）

ID	ロール	ロールタイプ	参加者	イニシアティブターゲット	操作名	操作タイプ
ST_001010	交通費申請書作成者	ACTIVE	社員	交通費申請書	生成する	CREATE
ST_001011					更新する	UPDATE
ST_001012					削除する	REMOVE
ST_001020	交通費申請書受付者	PASSIVE_ACCEPT	事務	交通費申請書	受付／承認する	ACCEPT
ST_001021					受付取消／承認取消する	UNACCEPT
ST_001022					保留する	SUSPEND
ST_001023					保留解除する	UNSUSPEND
ST_001024					返却する	REJECT

業務分析の段階で、ある系列の帳票は、ある法則に則った処理を行うことが知られているパターンが存在する場合がある．これをビジネス・テンプレイトと呼んでいる．ビジネス・テンプレートは現在 6 種類見つかっている．そのテンプレイートに従って処理フローを考えると、効率良く処理が進められ、開発も効率的になる．しかし、ここでは省略する．

3.2　詳細分析

帳票という粒度で考えると、帳票の作成は先ず、型紙を用意し、その個別のデータ項目の入出力の記載、そして COMMIT で登録する必要がある．詳細分析の段階では、

表 4　アクティビティ・シート（部分）

プロセス名 ProcessName	項目種別 Item Type	ラベル名 Label	業務項目名 Business Item	入力 Input	任意 Optional	値 Value
●						
交通費申請書を作成する	CREATE	交通費申請書	交通費申請書			
交通費申請書を編集する	EDIT	交通費申請書	交通費申請書			交通費申請書:1
	Input	申請書状態区分	申請書状態区分			作成済
	Input	申請日	交通費申請書申請日			input/申請日
	Input	合計金額	交通費申請書合計金額			input/合計金額
	Input	Remark	Remark	○	○	
行き先 Master を検索する	SEARCH	行き先 Master	行き先 Master			
	BeforeCondition	EXIST(Input/~行き先 No)				&Skip
	Search	行き先名	行き先名			input/行き先名
	AfterCondition	EXIST(行き先 Master:3)				入力データが行き先 Master にありません

このように、SVO Statement をさらに詳細に展開してアクティビティ・シートを作成する．PASSIVE_PRODUCE であれば、導出すべき帳票の型紙が用意され、元になる帳票

表 5　モデル・シート（部分）　交通費申請書 DML _000004

	現象型名	項目タイプ	値タイプ	Identified 現象型名
	交通費申請書 No	PRIMARY_KEY	COLUMN	
genarated by 000000000000 交通費申請書作成.asta 交通費申請書作成者	申請書状態区分	STATUS	COLUMN	
genarated by 000000000000 交通費申請書作成.asta 交通費申請書作成者	交通費申請書申請日	TIMESTAMP	COLUMN	
genarated by 000000000000 交通費申請書作成.asta 交通費申請書作成者	交通費申請書合計金額	BUSINESS	COLUMN	
Stereotype 共通:Request	Remark	OPERATOR	COLUMN	
Model 共通	Creator	OPERATOR	COLUMN	
Model 共通	CreateDatetime	TIMESTAMP	COLUMN	
Model 共通	LastUpdator	OPERATOR	COLUMN	
Model 共通	LastUpdateDatetime	TIMESTAMP	COLUMN	
Model 共通	RemovedFlag	STATUS	COLUMN	
Model 共通	Remover	OPERATOR	COLUMN	
Model 共通	RemovedDatetime	TIMESTAMP	COLUMN	
Model 共通	VersionNumber	PEXA	COLUMN	
Model 共通	ValidityFlag	STATUS	COLUMN	
Model 共通	EntityName	PEXA	STATIC	
	交通費申請書明細 List	ENTRY	ENTRY	

を SEARCH し、さらに SELECT し、新帳票の内容を編集し、元と新の二つの帳票を確定させる．このうに、自動展開が可能なことを示している．PRODUCE には CANCEL が対になっているので、こちらも用意する必要がある．この様に 24 種類の処理が用意されている．role を明確に分類し、各々の role ごとに用意すべき処理が確定していることから、システムの完全性を保証できることを示している．この SVO S tatement を詳細に展開したアクティビティ・シート（部分）を表 4 に示す．EDIT の詳細内容が 4 つの入力より成り立っていることを示している．　アクティビティ・シートが完成すると、扱う帳票の入出力項目が確定する．どの帳票に、どういう入出力項目すなわち業務項目が存在するかを記載したものがモデル・シート（部分）で、表 5 に示す．データベースに関する情報もモデル・シートに記載
される．誰が何時作ったかなどの全帳票で必ず必要な情報は Model 共通として扱われている．業務項目の実体については 3 つの表からなるビジネス・カタログで一元的に管理されている．

詳細分析においては、ルールの一覧表を作成する．これはアクティビティ・シートなどから、何も入力されていない出力項目を選り出すことによって、ルールの一覧化を行う．業務を担当している人にとって、ルールは当然の常識になっているので、要求獲得の際に抜け落ちている可能性が高い．それを拾い出すには、この方法が優れている．

プロトタイプ作成においては、シーケンス図を書いて、業務分析だけを行えば、成果物は自動生成されて、実行可能となる．詳細分析以降の設計、実装工程は必要ない．しかし、実際に基幹情報システムとして使用するシステムを作成する場合には、詳細分析の成果物を参考にしながらも、開発者が手作業で各成果物を作成していく．また、詳細分析段階の成果物はエクセル形式で構築される．

3.3　設計と実装

この段階では、画面の設計・実装とルールの設計・実装が中心になる．画面規約を作成し、アクティビティ・シートから必要とされる画面を抽出し、画面設計を行う．エンジンに読み込み可能とするため、Java 形式の画面デザイン書と XML 形式で書かれた画面定義書を用意する．

ルールは業務項目の値から必要とされる値を算出するプログラムを作成する作業である．プログラミングで作成されるが、制御フローの構造変化を起こすような心配は無い．

次にプロセスの設計・実装である．プロセスはアクティビティ・シートから本当に実装する部分を決定し、さらに、エンジンに入力しやすい形式であるサービス形式に

変換する．サービスという PEXA のエンジンに入力できる言語に変換されたプロセスであるサービス定義に変換する．

4. プロトタイプシステムの実行

プロトタイプ画面での交通費清算システムの実行状況を図 4 に示す．左画面に起動したユーザが処理できる仕事がリストアップされている．ここで、“ 交通費申請書作成者が交通費申請書を作成する ” をクリックすると右画面が表示される．

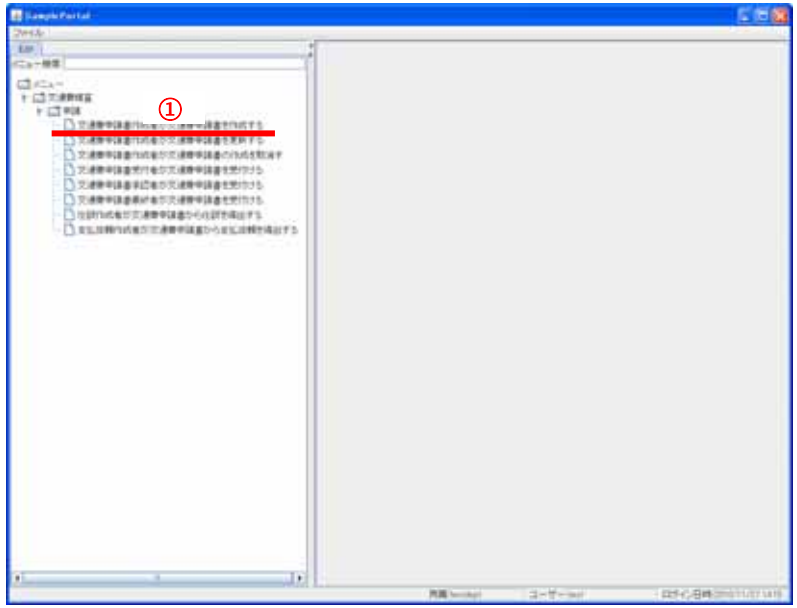

図 4　交通費申請書作成画面の起動

図 5 の右画面の申請日は日付選択大やログ機能を利用して入力し（②）、合計金額をタイプインする．③をクリックして交通費申請書明細ウインドウが開く（図 6）．図 6 で明細金額、行先移動日、移動手段などを記入することができる．このように、自動生成される画面を用いても十分プロトタイプとして利用できる．図 5 の右下の OK ボタンをクリックすると、シーケンス図での右のロールに移動して行くことができる．基本的には、OK と Cancel の二つのボタンしか無いので、右に進むのか、留まるかの二つしか選択肢が無い．

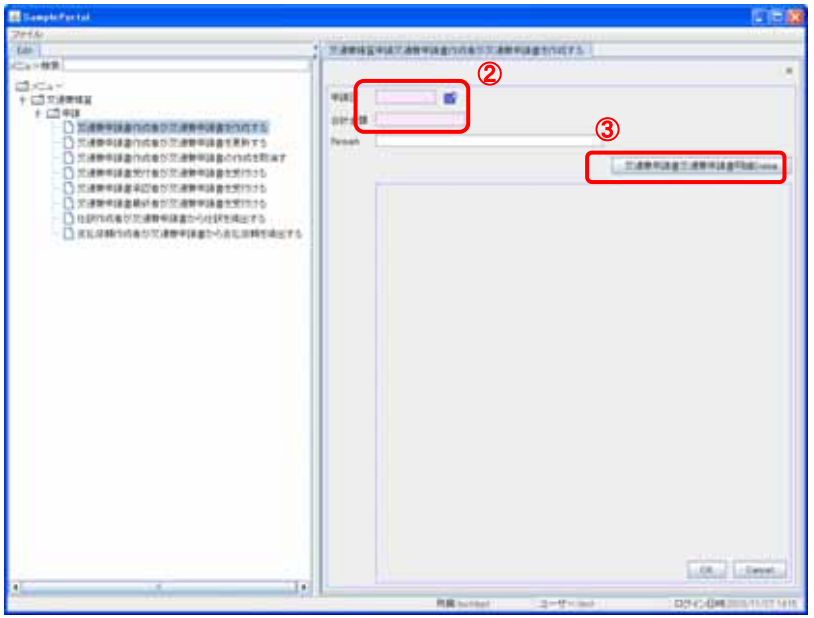

図 5　交通費申請書作成画面

5. 考察

基幹業務システムを帳票の取り回しによって、業務を運営するシステムであると考えて、システムを考えてみた。情報システムで直接扱うことができるのは帳票という形をした情報の塊である。そこで、ユースケース記述の処理対象は帳票だけであると限定する。処理対象が帳票に限定されると、処理の種類は限定されてくる。帳票という古来から使われていたデータ形式で、その処理方式は洗練されている。さらに、個々の帳票の意味を捨象すると、処理方式は定型化され少数に限定されてくる。これは、ちょうど表という使い慣れたデータ形式を対象とした関係データベースが成功した原因もここにあると考えられる。比喩的に言うと、本方式の考え方は帳票のアセンブラか、帳票の SQL を考えていることになる。帳票という粒度では定型的に処理する方式

が出来るようになった。次の段階として、複数の帳票が互いに関連して、より大きな粒度での業務を処理しているのが、ビジネス・テンプレートの考えである。まったく異なる分野の、異なる名称の複数の帳票群が実質的に同じ業務を遂行している。しか

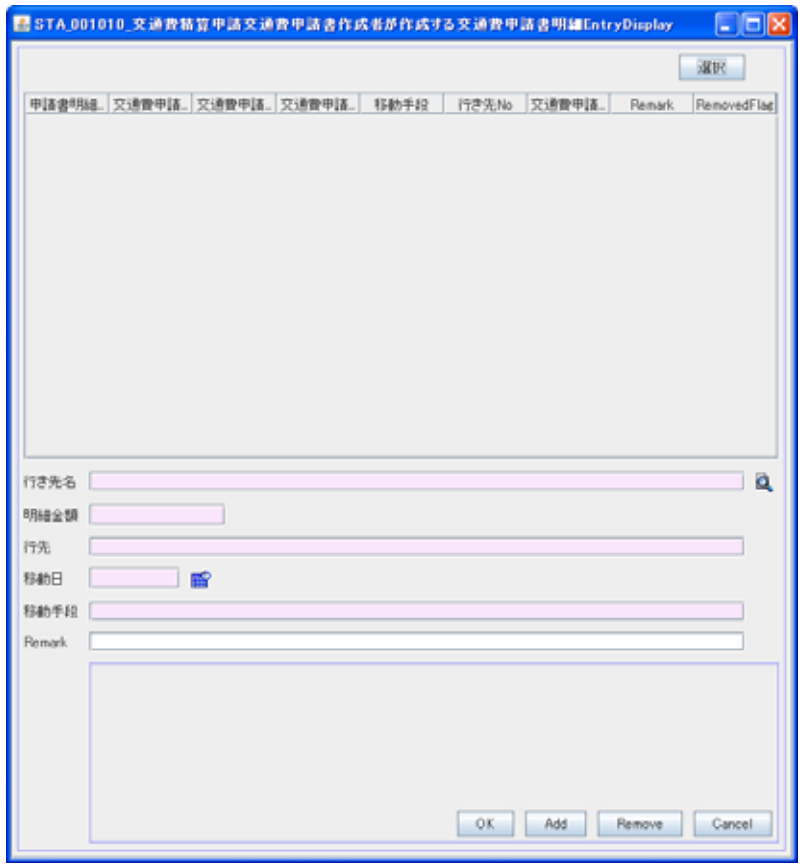

図 6　交通費申請書明細ウインドウ

し、6 つのビジネス・テンプレートしか見つかっていない。経営にかかわる人たちの目から見ると、このビジネス・テンプレートの粒度でも未だ粒度が細かいようである。それらを繋いでゆく努力が求められていると考えられる。

さらに、本方式ではモデルと業務項目という要素によってデータの管理をカタログ化によって一元化を図っている。

今回紹介した交通費申請のシステムでは、role 数は 5、帳票数は 4、業務項目数は 22 であった。また、学生の練習用として作成した大学図書館システムは、role 数が 17 で、帳票数が 11、業務項目数が 81 であった。一方、実システムとして構築された、あるデータセンターの基幹システムでは、ロール数が 132 で、SVO Statement 数は 270、帳票数が 309、業務項目数は 1995 である。実際に稼動システムと学生が勉強できるシステムとでは、その規模が大きく異なっている。

謝辞　サンプル・システム構築に協力してくれた卒業研究生の平良君と中村君の二人に感謝します。.

参考文献

1)　小野勇介、片岡信弘：ビジネスプロセスの表現に BPMN を用いた MDA 開発の研究、電子情報通信学会、技術資料　SWIM2007-32(2008-03)、p.49-54
2)　Martin Fowler:. アナリシス・パターン、ピアソン・エデュケーション　(2002) .
3)　Martin Fowler:：UML モデリングのエッセンス、翔泳社、(2000)
.