情報処理学会論文誌　プログラミング　Vol.6　No.4　10–26 (Dec. 2013)

OTCデリバティブ商品定義を目的としたドメイン特化言語の開発と評価

松本 吉史[1,†1,a)]　久野 靖[1,b)]

受付日 2013年4月12日, 採録日 2013年6月3日

概要：ドメイン特化言語（DSL）は，特定の問題領域に対する，適切に抽象化された簡潔なプログラム記述を可能にする．これにより，ドメイン専門家がプログラム記述を理解し，開発者と効果的に対話できるようになる可能性がある．本論文は，金融商品の一種であるOTCデリバティブ取引業務を記述するDSLの開発と評価について述べている．OTCデリバティブは，(1) 業務の専門性が高く，(2) 商品が複雑かつ多様であるため，これを対象とする業務システム開発において，開発者と業務担当者の意思疎通が難しい．筆者らは，デリバティブ商品定義向けDSLを開発することで，この問題の克服を試みた．金融機関の業務担当者によるレビューの結果，開発した言語の記述力は，レビュアが担当する情報システムで対応している商品を網羅しているとの評価を得た．また，開発した言語による商品記述を既存の商品記述方式（タームシート）と比較する評価の結果，開発したDSLの可読性は既存方式と同等以上であることが示せ，将来的に業務担当者がDSLを直接レビューすることによる開発生産性の向上が期待できるという結果を得た．

キーワード：ドメイン特化言語，OTCデリバティブ

Development and Evaluation of a Domain-specific Language for OTC Derivative Instruments

Yoshifumi Matsumoto[1,†1,a)]　Yasushi Kuno[1,b)]

Received: April 12, 2013, Accepted: June 3, 2013

Abstract: Domain-Specific Language (DSLs) allow solutions to be expressed with the vocabulary of, and at the level of the problem domain. Consequently, domain experts might communicate with the software developers in a more effective manner. Development of OTC derivatives trade management systems in financial institutions is the domain that embody communication problems among domain experts and software developers. Characteristically, OTC derivatives deals with a wide variety of complex products, and requires high degree of expertise and know-how, leading to frequent communication problems between domain experts and software developers. In this paper, To improve communication between domain expert and software developers, We designed and developed Derivative Definition Language, a DSL for development of derivatives trade management systems. The language allows business users to understand DSL programs. We have evaluated descriptive ability of the language through reviews by the domain experts. The result was that the language covers all of the products which they encounter in the current system. We have also evaluated the readability of its programs. The results have shown that readability of its program is quite comparable to, and in some aspects even better than, the term sheet currently used to describe real products. From these results, We conclude that improvement in development productivity could be expected using the Derivative Definition Language.

Keywords: Domain Specific Language (DSL), OTC derivatives

[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, Bunkyo, Tokyo 112–0012, Japan
[†1] 現在，株式会社電通国際情報サービス
Presently with Information Services International-Dentsu, LTD.
[a)] ymatsu@isid.co.jp
[b)] kuno@gssm.otsuka.tsukuba.ac.jp

1. はじめに

1.1 ドメイン特化言語

ドメイン特化言語（Domain Specific-Language：DSL）は，特定の領域のみでの利用を前提とし機能を限定することで，汎用プログラミング言語では対応が難しい課題に，より優れた解決策を提供する [1]．DSL を利用する主なメリットは次の 2 つである．

(1) ドメイン専門家と開発者のコミュニケーション改善
(2) システム開発生産性の向上

効果的なシステム開発には業務知識が不可欠であるが，業務知識を持つドメイン専門家とシステムを構築する開発者は互いの背景知識が異なるため効果的なコミュニケーションが難しく [2]，理解のギャップが残されたまま開発が進んだり，後段になって手戻りが起こったりするなどの問題が生じる可能性がある．適切な DSL の利用により，ドメイン専門家がコード記述を読解できるようになれば，コード記述を媒介として開発者と効果的に対話が行え，前記の問題の軽減が期待される．

また，汎用プログラミング言語により，複雑な属性を持つドメインの問題記述を行った場合，コード記述も大きく複雑になりがちであり，その結果，開発コストの増大や品質の低下が起こりやすい．適切な DSL の利用により，コードの構造や記述内容が簡潔になれば，前記の問題も軽減される．

DSL の利用には，デメリットもある．

(1) 特定ドメインのための言語はまだ存在していないことが通常であり，その場合，新規に開発する必要がある．
(2) DSL 開発後，言語機能の追加や改善など保守コストが必要である．
(3) 特定ドメインのための言語をすでに習得している開発者は，まれであるため，学習コストが必要となる．

DSL は，古くから利用されてきたが，体系的な研究が多数行われるようになったのは 90 年代後半からである [1]．近年は，デザイン方法論 [3], [4]，実装技術 [3], [5]，および様々なドメインへの適用事例 [6], [7], [8] などについて，多くの有用な知見が実用可能なレベルで蓄積されてきており，コストに関するデメリットが軽減されつつある．

1.2 OTC デリバティブ

金融機関で取引される金融商品の種類の 1 つに OTC デリバティブがある．デリバティブとは，株式や債券，外国為替などの金融商品をベース（原資産と呼ぶ）とし，原資産の価格や指標によって相対的にその価格が決定される金融商品の総称である．デリバティブは，原資産の将来にわたる価格変動リスクを回避するために開発された取引であるが，一部のデリバティブ商品は，元本金額を実際に保持していなくても取引でき，少ない手元資金で高収益を実現できる可能性があることから，投機目的で利用される．

デリバティブ商品の取引において，取引所を介さずに，売買当事者が相対で直接条件や金額などを交渉して取引するものを OTC デリバティブと呼ぶ．取引条件や売買方法が標準化された取引所売買商品とは対照的に，OTC では売買当事者さえ合意すれば各種の条件を柔軟に定義可能であり，実際に多様な商品が組成され取引されている．また，OTC デリバティブ商品は，投機性を高めるため，支払い条件や金額の計算方法などが複雑なことが特徴である．

1.3 OTC デリバティブ取引を管理する情報システム

金融機関では金融商品の金額計算や決済処理など，各種業務プロセス遂行のために，情報システムを利用する．OTC デリバティブでは，顧客ニーズに対応して頻繁に新商品が開発されるため，情報システムの改修が継続的に必要となる．したがって，OTC デリバティブの取引管理を行う情報システムは，迅速に効率良く新商品に対応できることが望まれるが，それには以下の課題がある．

- 業務の専門性が高いため，業務担当者から開発者担当者へ要件の正確な伝達が難しい．
- 商品特性が複雑で，多様な商品に対応が必要なため，コード量が多く複雑なプログラムとなる．

デリバティブ取引は金融業務の中でも特に専門性が高い．日本の主な金融機関においては，システム開発は外部ベンダに委託して行われるため，金融機関の業務担当者とシステム開発担当者との間に大きな知識ギャップが生じる．この知識ギャップが，両担当者間のコミュニケーションを阻害し，要件の正確な伝達を難しくしている．また，OTC デリバティブ商品は投機性を高めるため，多様な条件が付加されており，契約内容が複雑である．そのため，取引業務処理を実行するプログラムも，複雑でコード記述量が多くなる．これらの要因は，情報システムの品質および開発生産性低下をもたらす．

この結果，一部の金融機関では，全商品を完全にはシステム化せず，スプレッドシートを利用して取引情報を管理している場合もある．これには，次の問題がある．

- データがスプレッドシートのファイル単位で分散し，統合的な管理が困難．
- 作業の自動化が難しく作業効率が低い．
- 手作業が多いため，入力ミスや作業漏れなどのオペレーショナルリスクが高い．

07 年の世界金融危機をきっかけに，このような状況が問題視された結果，様々な規制や監視強化が進められており，情報システムの強化が求められている [9]．金融機関が適切にシステム対応を実施していくためには，開発品質および生産性の向上が必須である．

1.4　DSL による問題解決

OTC デリバティブ取引管理システム開発における知識ギャップの問題は，プログラミング言語や開発環境が特定ドメインを前提としておらず，知識ギャップの克服をほぼ開発者に負わせていることに 1 つの要因がある．プログラミング言語が DSL としてドメイン固有の知識を含んだ形で定義されていれば，以下の 2 点を通じて上記の問題を緩和できる可能性がある．

- プログラムが簡潔に記述できる．
- 業務担当者がコードを読み，実装内容を理解できる．

ドメイン知識を適切なレベルで抽象化できれば，処理手続きの詳細を隠蔽し，簡潔にコードを記述できるため，個々の商品に対する実装量を削減できる．また，複数の商品間での共通化も容易となるため，新商品追加の際の作業量も削減できる．

また，ドメイン専門家の馴染みがある表現でコードを記述できれば，業務担当者もコードを読み内容を理解でき，この効果により，システム開発プロセスにおける業務担当者と開発担当者の知識ギャップにより発生するコミュニケーションの問題を次の形で緩和できる．

- DSL を要件定義フェーズで利用することで，仕様伝達の正確性向上と作業効率化が期待できる．
- 業務担当者がプログラム内容をレビューすることで，開発プロセスの早い段階での不具合検出が期待できる．

本研究では上記の考えに基づき，DSL による OTC デリバティブ取引管理システムの新商品対応開発生産性向上の可能性を探究する．研究の目的は次の 2 点である．

(1) OTC デリバティブの取引管理システムの開発を目的とした，新しい DSL を開発する．
(2) 開発した DSL の有効性を評価する．

DSL の開発においては，次の 2 点を評価基準とした．

実用性
: 複雑・多様な OTC デリバティブ商品の記述が行え，情報システムの必要な機能が実現できること．

可読性
: 簡潔かつ可読性の高い記述が行え，これにより前掲の課題解決につながること．

1.5　本論文の構成

以下 2 章では，金融ドメインの DSL に関する関連研究を紹介し，本研究の位置づけを明確にする．3 章では本研究の DSL で記述対象とする範囲の検討を行った後，複雑な商品の構成要素を概説する．4 章ではデリバティブ商品定義 DSL の設計と実装について述べ，5 章で開発した DSL の評価について報告する．最後に 6 章で今後の課題を述べ，7 章でまとめを行う．

2.　関連研究

van Deursen らは，DSL とオブジェクト指向フレームワークの効果の違いについて調査を行った [10]．その中で，ドメイン特化言語 RISLA の開発・利用により，金利系デリバティブにおける新商品システム対応の開発生産性が向上したことを述べている．RISLA は，オランダの MeesPierson 銀行と Capgemini 社が共同で開発し，金利系商品の取引管理を対象ドメインとしている．MeesPierson 銀行は COBOL で実装された既存のライブラリ群を持っており，RISLA で記述された商品定義プログラムは，そのライブラリを利用する COBOL ソースコードを出力する．RISLA を用いて新商品開発を行った結果，COBOL ベースの開発と比較して，対応期間が 6 カ月から 2，3 週間に短縮された．また，業務エキスパートによる商品定義が可能となり，実装内容の検証が容易になった．この研究では，金融商品の取引管理を対象とした DSL による開発生産性向上の可能性が示されているが，OTC デリバティブにおけるような複雑な商品の記述可能性には言及されていない．

Jones らは，デリバティブ取引の契約を表現するためのコンビネータ群を関数型言語 Haskell により定義し，コンビネータを組み合わせたコードにより対象取引の価格評価を行う手法を示した [11]．この契約定義言語は，複雑な取引契約はシンプルな契約の組合せで表現できるという方針のもと，抽出したプリミティブなコンビネータの組合せにより新たなコンビネータを定義するという手法でデザインされている．この研究においては，関数型言語の特徴を活かすことで，少数のプリミティブ・コンビネータを組み合わせて多様な契約を表現できる可能性が示されている．ただし，DSL の主要な利点であるプログラムの可読性については評価されていない．

Frankau らは，株式のエキゾチック・デリバティブ商品を表現するための内部 DSL-FPF（The Functional Payout Framework）を関数型言語 Haskell で開発し，金融機関の Barclays Capital で利用した事例について報告している [12]．Barclays Capital では，従来，新商品の対応において命令型スクリプト言語を利用していたが，同様のコードを何度も記述する必要があるなど，開発が非効率であった．FPF の利用は，開発期間削減に大きな効果があったと述べている．この研究は，実際の金融機関で利用されていることから，複雑な商品も記述可能だと推測される．しかし，コードの可読性向上は FPF の目的とされておらず，可読性評価も行われていない．

金融ドメインにおける DSL の先行研究を整理したものを**表 1** に示す．van Deursen らの研究は複雑な商品への適用について未評価である．RISLA 言語の詳細は公開されていないが，金利以外の原資産を利用した商品や，近年に開発されたエキゾチックオプションには未対応だと推測

表 1　金融ドメインにおける関連研究の整理
Table 1　Related works on DSLs for financial domain.

項目	van Deursen らの研究（RISLA）	Jones らの研究および Frankau らの研究（関数型言語 DSL）
DSL 種類	外部 DSL	内部 DSL
実装言語	不明（COBOL コードを出力）	Haskell
対象金融業務	取引ライフサイクル管理	価格計算（バリュエーション）
対象金融商品	金利系商品	様々な金融取引契約
可読性評価	○	未評価
複雑な商品への適用	未評価	○

される．また，Jones らと Frankau らの関数型言語を利用した研究は，可読性について未評価である．これらの論文では，シンプルな契約プログラム例が示されているが，内容を理解するためには Haskell の知識が必要であり，可読性について改善の余地が大きいと考える．以上をまとめると，「可読性」「複雑な商品への適用」をともに実現した金融ドメインの DSL は，筆者らの知る範囲では開発されていない．

3. 対象ドメイン分析

3.1 OTC デリバティブ取引管理システムの機能

本研究の目的に基づき，開発する DSL が提供する機能は，OTC デリバティブに固有の以下の 3 つが候補となる．

取引登録

取引内容を登録し，それを検索・照会可能にする．

取引ライフサイクル管理

取引ライフサイクルで発生するイベント情報やキャッシュフローデータを生成し，期日に適切な処理を行う．

バリュエーション

日々変動する取引の価格やリスクを計算する．

これらのうち，特に複雑となる取引ライフサイクル管理について，「ヨーロピアン・コール・為替オプション」という商品を例に説明する．

（例）ヨーロピアン・コール・為替オプション

オプションとは，あらかじめ決められた将来の一定の日（権利行使日）において，一定の価格で取引する権利を売買する取引である．この例で取引されるのは，「半年後を権利行使日として，1 ドルを 100 円で購入する権利」とする．オプション行使はあくまでも権利であり，行使日の為替相場によっては行使しないこともある．

この取引のライフサイクルで発生するイベント，およびキャッシュフローを**図 1** に示す．取引ライフサイクル管理機能は，契約内容に応じてこれらのイベント情報を生成し，処理を行う．「プレミアムの受取」イベントでは，金額を計算しキャッシュフローデータを生成する．「オプション行使」イベントでは，行使の判定を行い，行使された場合は，為替取引のイベントを実行する．

イベント日	イベント内容	キャッシュフロー	
		支払	受取
2013/04/01	取引約定		
2013/04/03	プレミアムの受取		5円
2013/10/01	オプション行使 （市場価格＞105円/1ドルの場合，オプション行使する）		
	為替取引 （オプション行使された場合）	1ドル	100円

図 1　ヨーロピアン・為替オプション取引のライフサイクル
Fig. 1　Trade lifecycle of european forex option.

取引登録・取引ライフサイクル管理・バリュエーションの各機能はそれぞれ，次の 4 要素からなる．

ユーザ・インタフェース（UI）：

ユーザがデータの登録や照会を行うためのインタフェース．たとえば取引ライフサイクル管理においては，取引イベント一覧照会画面などである．

データ生成・更新処理：

ユーザの入力データやシステム設定値などから，データの生成および更新を行う処理．たとえば取引ライフサイクル管理機能においては，入力された取引契約情報からイベントデータ（実行日，イベント種類，実行ステータスなどを持つレコード）を生成する処理などである．

データ永続化処理：

UI で入力されたデータや，システム内部で生成されたデータを永続性のある媒体に保存する処理．一般的にはデータベースなどのミドルウェアを利用しハードディスクにデータを保存する．

外部システム連携処理：

他の情報システムと連携する処理．たとえば取引ライフサイクル管理においては，外部の決済システムにキャッシュフローデータを送信する処理などである．

DSL は，機能を限定することで簡潔な表現を可能としており，それがメリットとなる．そのため，対象とするシステム機能について，適切な範囲を設定とすることが重要である．本研究では，各機能の処理要素単位で，以下の方針により対象とする機能を決定した．

(1) 処理内容が商品特性に依存しており，新商品追加のたびに開発が必要となる処理を対象とする．

表 2　DSL の対象システム処理
Table 2　Scope of system function defineded by DSL.

	UI	データ生成・更新	データ永続化	外部システム連携
取引登録		○		
ライフサイクル管理		○		
バリュエーション				

(2) 各金融機関のシステム資産や環境に依存しない処理を対象とする．

(1) について，初期開発で一度実装してしまえば新商品追加による開発が不要な機能については，DSL で実現するメリットがないため，本研究で開発する DSL の対象から除外する．(2) については，たとえばデータ永続化処理は，利用するデータベースや，フレームワークに実装が依存するため，金融商品定義の DSL を利用することが最適ではない可能性がある．そのため，本研究の DSL においては環境依存の処理を対象外とする．

これらの方針に従い決定した本研究の対象機能を，**表 2** に示す（○が記載されているものを対象とする）．関連研究で示した，関数型言語による契約定義の研究 [11], [12] においては，バリュエーション機能のデータ生成・更新処理が対象となっていた．この処理は，上記の (1) および (2) に該当するが，金融機関における業務エキスパートが実装を行うことが一般的であり，本研究で課題としている業務担当者と開発担当者のコミュニケーションの問題は発生しないため，本研究の対象外とした．

3.2　OTC デリバティブで取引される金融商品の構成要素

本研究においては，DSL の対象領域を明確にし，その領域が実用的に問題がないか評価する必要がある．このため，金融商品の特性を決定する主要な構成要素を抽出し，それらに基づいて対象商品を整理した．具体的に抽出した構成要素は以下の 4 つである．

- 原資産の種類
- エキゾチック・オプション
- ペイオフ計算方法
- 発生キャッシュフロータイプ

次節以降で 4 つの構成要素それぞれについて，本研究の DSL が対象とする内容を説明する．なお，この 4 つの構成要素について業務専門家のレビューを実施することで，対象ドメインの妥当性評価を行う．

3.3　原資産の種類

デリバティブ取引では，株式，為替，金利などの金融商品だけでなく，コモディティと呼ばれる原油・金・小麦・大豆・アルミなど市場で売買されている商品，あるいはクレジット・デリバティブと呼ばれる企業のデフォルトリスク（信用）などが取引される．OTC デリバティブにおいて，原資産に制約はないため，近年は天候や災害リスクなども原資産として利用されている．また，複雑な商品では，複数の原資産が利用される場合もある．たとえばある時点の株価によって参照金利を変更するような商品がある．

本研究においては，為替，および金利を扱えるように実装した．その他の原資産については容易に実装を追加できるように，拡張を考慮した設計・実装を行った．

3.4　エキゾチック・オプション

エキゾチック・オプションは，通常のオプション条項に様々な付加条項が付与されたものや，特殊な条件でキャッシュフローが計算・生成されるような契約事項の総称である．「定義された条件に該当する（しない）場合に，あるアクションを行う」と表現できる．文献 [13], [14] や金融機関が公表している情報をもとに整理を行い，本研究で対象とするエキゾチック・オプション一覧を作成した（**表 3**）．

たとえば「ターゲット・リデンプション」は，債券のクーポン（利息）の総額に対し，上限を設定するものである．クーポン金額が，変動する原資産により決まる商品において，ターゲット・リデンプションが付加されている場合，クーポン支払いのつどその金額を累積していき，上限（元本金額の 10%，など）に達したところで債券は償還される．

複雑な商品では，これらの条項が複数組み合わされたり，条件がカスタマイズされて利用される．

3.5　ペイオフの計算方法

ペイオフとは決済（額）のことである．OTC デリバティブではクーポンや償還金額などが決済されるが，この金額は様々な式で定義される．文献 [13], [14] や金融機関が公表している情報をもとに整理を行い，本研究で対象とするペイオフ計算方法の一覧を作成した（**表 4**）．

たとえば「レンジ・アクルーアル」は，変動する原資産の値が，1 カ月の間に何日間ターゲットレートを超えたかによりペイオフを決定するものである．たとえば，金利を原資産としてターゲットレートを 3%と定めておく．そして日々変動金利を観測し，ターゲットレートを超えた日数をカウントしていく．1 カ月で超えた日数が 15 日だった場合，ベースレートを 5%とすると，$5 * 15/30 = 2.5\%$ をペイオフ計算レートとする．

複雑な商品では，さらにキャップ/フロア（上限/下限）オプションがついたり，2 つの計算式を利用し結果が大きい方，小さい方，または平均を金額としたりするものがある．

3.6　発生キャッシュフロー・タイプ

OTC デリバティブは商品内容に応じて多様なキャッシュ

表 3　エキゾチック・オプション条項
Table 3　Exotic options.

オプション条項	内容
キャップ	金額やレートに上限を設定する
フロア	金額やレートに下限を設定する
コーラブル	発行体が期限前償還できる権利
ノックアウト・バリア	原資産が一定の価格に達するとオプションの権利が失効する
ノックイン・バリア	原資産が一定の価格に達するとオプションの権利が発生する
オート・トリガ	原資産が一定の価格に達すると早期償還する
ターゲット・リデンプション	クーポン（利息）の総額が一定金額を超えた場合に早期償還する

表 4　ペイオフ計算式
Table 4　Payoff equations.

種類	計算式（$_{(t)}$ は t 時点での参照値を表す）
フロータ	参照レート $_{(t)}$ + x%
リバース・フロータ	x% − 参照レート $_{(t)}$
スプレッド	参照レート $1_{(t)}$ − 参照レート $2_{(t)}$
フリップ・フロップ	当初 5 年：参照レート $_{(t)}$ + x%，以降：y%
レンジ・アクルーアル	x% ∗ N / 30（N = 1 カ月で参照レートが y%を超えた日数をカウント）
パワー・リバース・デュアル	外貨固定金利 x% ∗ (為替レート $_{(t)}$ / 為替レート $_{(init)}$) − 円固定金利 y%
スノーボール	前回クーポン金額 − 参照レート $_{(t)}$ + x%
デジタル	if (株価 $_{(t)}$ > z 円) then x% else y%
ラチェット	前回クーポン金額 + if (株価 $_{(t)}$ > z 円) then x% else y%

フローが発生し，その一般的な区分は存在しない．本研究ではキャッシュフローを発生パターンに基づきオプション型，債券型，スワップ型の 3 種に分類して扱う．

オプション型は，契約時にオプション購入金額（プレミアムと呼ばれる）を支払い，オプションが行使された場合に，対象の商品売買が行われキャッシュフローが発生する．複雑な商品では，オプション原資産の商品が，以降で説明する債券型であったりスワップ型であったりする．

債券型は取引開始時に元本金額と債券を交換する．その後満期を迎えるまで定期的にクーポンが発行者から債券保有者に支払われる．そして満期時に発行者から償還金額が支払われる．早期償還のオプション条項が付与された商品では，条件を満たした場合に満期時期が早まる．また償還金額は条件に応じて変動したり，別の資産（元本は日本円で支払ったが，株券で償還される，など）で支払われる商品もある．このような債券型のキャッシュフローを持つ複雑な商品は仕組債と呼ばれる．

スワップ型は，基本的には想定元本の支払いは発生せず，償還もない．開始から満期までの間，定期的に金額の交換が発生する．交換する金額は，異なる参照レートや計算式から算出した金利の利息であったり，異なる通貨の金額であったりする．スワップ型においても付与されたオプション条項により早期終了する場合がある．また稀に取引終了時に契約で定められた条件に基づいてキャッシュフローが発生する商品もある．

3.7　複雑な OTC デリバティブ商品への対応

OTC デリバティブ商品ついて，「原資産」「エキゾチック・オプション」「ペイオフ計算方法」「キャッシュフロー・タイプ」の 4 つの切り口で本研究の対象を整理した．OTC デリバティブの複雑性は，この 4 つの要素とその組合せによる複雑性といえる．本研究では，上述の 4 つの切り口で示した対象要素を記述できるだけでなく，複雑な商品への対応として，要素を複数組み合わせて組成された商品を表現できるものとする．

3.8　可読性の検討

OTC デリバティブ金融商品取引では，商品の内容や契約条項を記載したタームシートにより顧客と交渉・合意確認を行う．取引について，そのタームシートを見ることで，金融商品の内容を把握できる．タームシートのフォーマットは規定されているものではなく，金融機関ごとに異なるが，ほぼ同様の記述形式となっている．業務担当者は，タームシートの記述に慣れているため，本研究では DSL の可読性の指針としてタームシートを利用する．実際に契約で利用されたタームシートの例を**図 2** に示す．

タームシートには次の特徴がある．

- 契約条項が，項目名とその値（内容）という形式により，箇条書きで記載されている．
- 各契約条項の内容の表記は，複雑な条項の場合，自然言語で記載される．また，金額計算は曖昧性がないように式で定義される．

We are pleased to provide summary terms and conditions under which you have issued a 10 year Lifetime Cap Inverse Floater MTN, to be issued under the [redacted] Bank N.V. USD20,000,000,000 Debt Issuance Programme dated 4th September, 2003.

Together with the associated swap transaction, ***the issue will provide the Issuer with a cost of funds (excluding expenses) of 6 month USD LIBOR minus 0.15% per annum***, as outlined more fully below.

Final Terms and Conditions of the MTN

Issuer:	[redacted] Bank N.V.
Dealer:	[redacted] Bank N.V.
Principal Amount:	USD 6,700,000
ISIN:	X[redacted]
Trade Date:	7th November 2003.
Payment Date:	14th November 2003.
Maturity Date:	14th November 2013.
Issue Price:	97.50% of Par.
Net Proceeds:	USD 6,532,500.
Redemption Price:	100.00% of Par.
Coupon:	Year 1: 8.00%; Year 2-10: {10.00% - 2 × 6m$L In Arrears};
	Coupons are payable semi annually in arrear on a 30/360 (unadjusted) day basis, and subject to a Mininimum of 0.00% and a Lifetime Cap as defined below.

図 2　タームシートの例（抜粋）
Fig. 2　Example term sheet.

- 契約条項の各項目の内容記載において，他の項目が参照される場合がある．
- 専門用語が使用され，その用語にドメイン固有の性質が暗黙的に含まれるケースがある．
- 商品の基本的なイベントは，商品特性を示す用語により定められるため，明記されないケースがある．

図 2 の例では，取引日（Trade Date），元本金額（Principal Amount），発行価格（Issue Price）という項目は定義されているが，「取引日に元本金額 * 発行価格を支払う」というイベントは明示されていない．このイベントは，MTN という商品名で定義されるためである．

タームシートは OTC デリバティブ商品取引を行う金融機関とその顧客により利用されるものであるが，システム開発においても，金融機関とシステム開発ベンダの間で，仕様を説明するために利用される．

4. デリバティブ商品定義 DSL の実装

4.1 実装方針

本研究の対象であるデリバティブ商品定義 DSL は，以下の方針に基づいて開発した．

デリバティブ商品単位でのプログラム記述：
汎用プログラミング言語による情報システムの開発では，機能単位でプログラムを分割することが一般的である．しかしプログラムの分散は内容の理解を困難にするため，今回は業務担当者にとっての可読性に配慮し，商品単位で記述する．

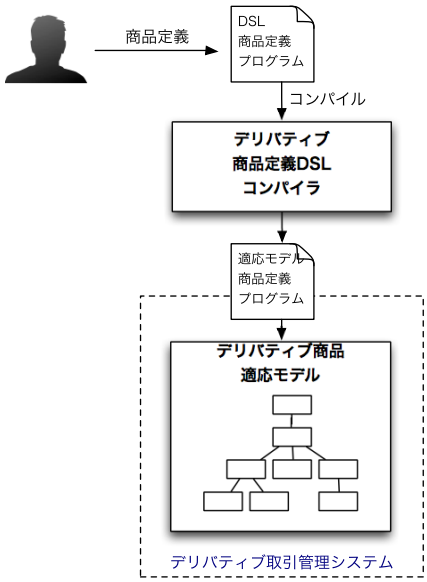

図 3　デリバティブ商品定義 DSL の実装概観
Fig. 3　Architecture of derivative definition DSL.

タームシートを参考にした記述方式：
業務担当者はタームシートの記述に馴染みがあるため，これに近い表現でコードを記述する．

汎用的な要素の組合せによる複雑な商品の定義：
Jones らの研究では，プリミティブな契約事項を組み合わせることで，より複雑な金融商品を表現できることが示されている [11]．本研究でもこれに基づき，基本要素の組合せで複雑な商品を定義する．

4.2 実装の概観

本研究で開発した DSL の実装概観を**図 3** に示す．
実装は，次の 2 つの部分からなる．

- デリバティブ商品適応モデル
- デリバティブ商品定義 DSL コンパイラ

デリバティブ商品適応モデルは，取引管理システムの機能を実装するオブジェクト指向フレームワークであり，オブジェクト群の関連付けに基づき各オブジェクトの機能が組み合わさることで，複雑・多様な商品のための機能を実現する．DSL コンパイラは商品の宣言的記述を入力とし，適応モデルのオブジェクト群を生成し関連付けを行うコードを生成する．Fowler [15] は，このような適応モデルと DSL の組合せにより，構造把握が容易で抽象度の高い記述を行うことの利点を指摘している．

4.3 デリバティブ商品適応モデルの実装

本研究で開発したデリバティブ商品適応モデルの規模を**表 5** に示す．**図 4** は為替取引の商品を同モデルで実装した例である．デリバティブ商品は「契約条項」と，取引ライフサイクルで発生する「イベント」という 2 つの要素でモデル化できる．為替取引では契約条項として，元

表 5　デリバティブ商品適応モデルの規模
Table 5　Size of derivative adaptive model.

項目	内容
Java クラス数	110 クラス
総コード行数	5,153 行（実行行のみ）
開発期間	約 6 カ月（約 180 時間）

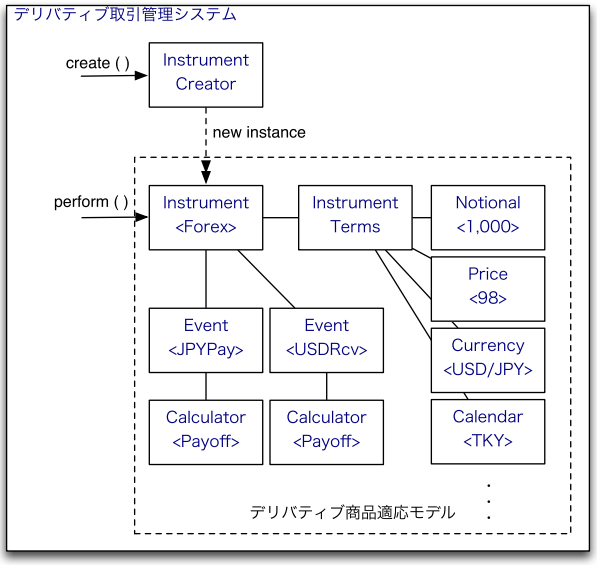

図 4　デリバティブ商品適応モデル
Fig. 4　Derivative adaptive model.

本（Notional）や為替レート（Price）などの値が定義される．デリバティブ商品適応モデルでは，この契約条項を「InstrumentTerms」オブジェクトで保持する．またイベントとしては，ある通貨の金額を支払うイベントと，異なる通貨の為替レートで換算した金額を受け取るイベントがある．デリバティブ商品適応モデルでは，このイベントを「Event」オブジェクトで表し，Event オブジェクトは，その処理を行う「Calculator」オブジェクトの参照を保持する．Calculator は処理ごとに複数の実装がある．

取引管理システムにおいて新しい商品に対応する場合は，商品定義プログラムとして，「InstrumentCreator」クラスを実装する．このクラスの create メソッドの中で，Event や Calculator オブジェクトのインスタンスを生成し，複数のオブジェクトを組み合わせて Instrument オブジェクトを生成するプログラムを記述する．

取引管理システムでこの商品を選択し取引登録を行う場合，対象商品の InstrumentCreator クラスの create メソッドを呼び出すことで，Instrument オブジェクトを生成する．この取引についてキャッシュフローデータの一覧を照会する場合は，生成した Instrument オブジェクトの perform メソッドを呼び出すことで，構成要素の Calculator オブジェクトのメソッドが実行され，データが生成される．

デリバティブ商品定義 DSL コンパイラは，この InstrumentCreator クラスの create メソッドのコードを DSL 記述から自動生成する．

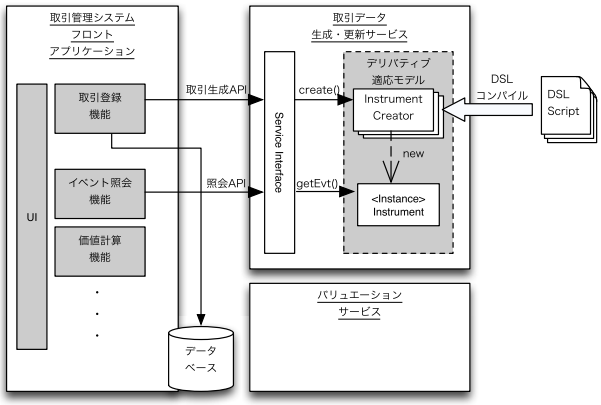

図 5　適応モデルを利用した業務システム例
Fig. 5　Example of system architecture using adaptive model.

本研究では，取引登録やキャッシュフローデータ照会を CUI インタフェースにおいて行えるテスト用アプリケーションを開発し，デリバティブ商品適応モデルが，処理イメージのとおり正しく動作することを確認した．

4.4　デリバティブ商品適応モデルを利用した業務システムの例

デリバティブ商品適応モデルを利用した業務システムの例を図 5 に示す．「取引管理フロントアプリケーション」はユーザインタフェースやデータ永続化機能を持つ，各金融機関で個別に開発するアプリケーションである．たとえば取引登録はこのアプリケーションから「取引データ生成・更新サービス」の API を呼び出すことで商品種類に応じた取引レコードを生成しデータベースに登録する．「取引データ生成・更新サービス」は内部でデリバティブ商品適応モデルを利用しデータの生成を行う．商品の価値やリスク指標の計算は「バリュエーションサービス」の API を呼び出すことで行う．

このように，デリバティブ商品適応モデルを利用したデータ生成・更新機能を 1 つのサービスとすることで，各金融機関の既存のアプリケーションや業務パッケージソフトウェアと組み合わせて本研究のデリバティブ商品定義 DSL を利用することが可能となる．

4.5　デリバティブ商品定義 DSL コンパイラの実装

本研究では可読性を重視したため，専用の言語処理系を開発する外部 DSL [1] を選択し，コンパイラの実装には SableCC [16] を利用した．SableCC は Java で実装されたパーサジェネレータで，トークン定義と BNF による文法定義を記述したファイルを入力に，パーサフレームワークの Java プログラムを出力する．出力されたフレームワークに準拠した意味処理クラスを記述することでコンパイラ

表 6　デリバティブ商品定義 DSL コンパイラの規模
Table 6　Size of derivative definition DSL compiler.

項目	内容
文法定義（SableCC）行数	253 行
意味解析（Java）行数	2,982 行（実行行のみ）
開発期間	約 2 カ月（約 60 時間）

が完成する．SableCC は次の特徴を持つ．

- 拡張 Visitor パターンによる，意味処理とパーサの分離
- 日本語の文字をプログラム記述に使用可能

本研究では日本語記述は用いていないが，将来的に必要となった場合は対応可能である．

DSL の開発は，基本水準の適応モデルとそれを使用する言語記述を実装した後，3 章で示した水準の商品要素が使用可能になるまで適応モデルと言語記述を拡張していく形で実施した．**表 6** にコンパイラの規模と開発期間を示す．

4.6　デリバティブ商品定義 DSL の記述方法

4.6.1　デリバティブ商品定義プログラムの構成

デリバティブ商品定義 DSL による商品定義プログラムは次の要素で構成される．

- プログラム宣言
- 契約項目定義
- イベント定義

プログラム宣言は，次の形でプログラムの範囲を表す．1 つの宣言が 1 つの商品定義となる．

```
商品名 {
  //ここに「契約項目定義」を記述
  //ここに「イベント定義」を記述
}
```

商品名の部分は，英字アッパーキャメルケースで，商品ごとに適切な名前を定義する．

契約項目定義は，その商品が持つ契約項目とその値を表現する．また，イベント定義はその商品の取引ライフサイクルで発生するイベントの詳細を表す記述である．

プログラム宣言，契約項目定義，およびイベント定義について，SableCC 文法定義の一部を**図 6** に示す．SableCC 文法定義は BNF 記法を利用して記述される．

4.6.2　契約項目定義

契約項目定義は，対象商品において契約に必要な項目を定義するもので，たとえば，取引開始日，満期日，想定元本，参照レート，営業日カレンダなどを定義し，次の形式で記述する．

```
# 項目名　：　値;
```

項目名と値の記述例を**表 7** に示す．金融商品契約で利用されるタームシートも項目と値のペアの箇条書きとなっており，ほぼ同様の語彙，表現を使って記述できるようにしている．項目名には，種類に応じてサフィックスを付加する（日付項目の場合は，.Date を，金額や数量項目の場合は，.Amount を付加する）．項目定義により宣言された項目名は，他の契約項目定義やイベント定義中で参照できる．

表 7　契約項目定義の記述例
Table 7　Scripting examples of contract definition.

契約事項	記述例
日付	#start.Date : 2013/1/1;
日付（相対指定）	#end.Date : start.Date + 10years;
金額	#notional.Amount : USD 100,000,000;
変動金利	#float.Price : USD/“6m$L”+5% 30/360;
固定金利	#fixed.Price : 0.15% 30/360;
為替レート	#spotFx.Price : USD/JPY BID;
レート計算式	#coupon.Price : “0.15 - fx2.Price * 0.1”;
営業日カレンダー	#calendar : TKY;
休日調整方法	#dayConvention : Following;

4.6.3　イベント定義

デリバティブ商品におけるイベントとは，取引ライフサイクルで発生する，業務処理を表す．「変動金利についてマーケットデータを参照し値を確定する処理」「決定した変動金利の値で決済金額を計算する処理」などが例である．デリバティブ商品定義 DSL では，曖昧性なくイベント処理を定義できるように，次の 3 項目を指定する．

EventDate 項目
　イベントが発生する日付を定義する．

Type 項目
　イベントで実行される処理タイプを指定する．決済，オプション行使，レート参照，など．

Description 項目
　イベント処理内容の詳細を定義する．たとえば，イベントタイプが決済の場合，金額の計算式を定義する．

Type と Description は関連しており，Type に応じて Description の値の記述方法が決定する．

イベント定義は，次の形式で記述する．

```
イベントブロック名 {
  イベント名 ---
    EventDate    :  イベント日付;
    Type         :  イベントタイプ;
    Description  :  イベント詳細;
  ---
}
```

イベント定義は，イベントブロック中にイベントを定義する構造となっている．イベントブロックは，業務的に関連のある複数のイベントを構造化するための構文である．1 つの商品定義の中に複数のイベントブロックを記述でき，イベントブロック中に複数のイベントを記述できる．

EventDate の記述方式を**表 8** に示す．イベントには，特定の日付に 1 回だけ実施するものと，繰り返し実行するものがある．特定日付のイベントの場合は，契約事項定義と

```
prog =
  {instrument}        ident lbra instrument_body* rbra
  ;
instrument_body =
  {term}              term_declaration |
  {event_stream}      pending_annotation? ident [start]:lbra event_stream_body* [end]:rbra
  ;
term_declaration =
  {term}              term_ident colon term_expression semicolon
  ;
term_expression =
  {date}              date_expression |
  {amount}            amount_expression |
  {price}             price_expression |
  {schedule_price}    schedule_price_expression |
  {period}            period_expression |
  {businessdays}      businessdays |
  {dayconvention}     dayconvention |
  {string}            string_literal
  ;
event_stream_body =
  {term}              term_declaration |
  {event}             ident [start]:event_delimiter event_body [end]:event_delimiter
  ;
event_body =
  {default}           event_date_expression event_type_expression event_description_expression?
  ;
event_date_expression =
  {date}              eventdate colon date_expression semicolon |
  {datelist}          eventdate colon lsqbra date_expression additional_date_expression* rsqbra semicolon |
  {scheduler}         eventdate colon scheduler_expression semicolon |
  {var}               [keyword]:eventdate colon ident dot [suffix]:eventdate semicolon
  ;
event_type_expression =
  {simple}            eventtype colon string_literal semicolon |
  {basic}             eventtype colon [calculator]:string_literal minus gt [argument]:string_literal semicolon
  ;
event_description_expression =
  {basic}             eventdescription colon string_literal semicolon
  ;
```

図 6　デリバティブ商品定義 DSL の SableCC 文法定義（抜粋）
Fig. 6　Excerpt from SableCC specification of the derivative definition DSL.

表 8　イベント定義の EventDate 記述例
Table 8　Scripting examples of EventDate.

イベントタイプ	記述例
特定日付	start.Date + 2days
繰返し	Frequency Semi-Annually (2013/1/1 to 2023/1/1) EndOfPeriod - 10days
オプション行使	Option.European(start.Date to end.Date)

同様の方式で，絶対日付と相対日付が指定できる．繰返しイベントの場合は，表 8 の記述例のように，Frequency 定義を使用する．記述例は，2013/1/1 から 2023/1/1 までの間，半年間隔（semi-annually）で発生するイベントで，イベント日は，半年間隔で分割した各期間の，最終日の 10 営業日前を表している．オプション行使イベントの日付については，繰返しイベントや特定日付の指定を利用することで表現可能であるが，「ヨーロピアン・オプション」，「アメリカン・オプション」という用語を使うほうが，業務担当者が直感的にイベント行使日を理解でき，簡潔に記述できるため，「Option.Europian」「Option.American」という記述で定義可能とした．

次に Type と Description の記述方式を**表 9** に示す．Type は，3 章で示した対象の商品要素を網羅的に記述できる，必要最低限のものを抽出した．ここに示した Type により，対象の原資産，エキゾチック・オプション，ペイオフ計算式を網羅的に表現できる．

表 9　イベント定義の Type と Description 記述例
Table 9　Scripting examples of various event types.

Type	Description 記述例	イベント概要
“Payoff”	“jpy.Amount = notional.Amount * init.Price”	決済イベント．指定した計算式で金額計算．
“Fixing”	“reference.Price”	レート確定イベント． 指定したレートについてマーケットデータを参照し確定．
“PlainOption”	“Call.Forex, OP001/FxOption/Manual”	オプション行使確認イベント． 指定した識別子のオプション行使有無判定．
“ExoticOption”	“IF　spot.Price >init.Price THEN Execute.Redemption”	エキゾチックオプション処理イベント． 条件にマッチする場合に，指定したイベントを実行．この例では，Spot レート >初期レートの場合，償還イベントを実行する．
“Formula”	“Formula : coupon.Price = 0.1 - 2 * ref.Price Cap　: cap.Price Floor　: 0.1%”	計算処理イベント．指定した式を計算． この例では，計算した coupon レートの上限値（Cap）と 下限値（Floor）を設定している．
“Conditional”	“IF　spot.Price >z.Price THEN spread.Price = x.Price ELSE　spread.Price = y.Price”	条件付き計算処理イベント． 条件にマッチする場合に，指定した式を計算する．この例では，Spot レート >z の場合，spread レートに x を，それ以外は y を設定．
“Ranger”	“Condition : spot.Price >100 Range : 1M ObservationPeriod : daily, 1-20”	レンジアクルーアル・カウントイベント． 指定した期間に条件にマッチした日数の割合を計算．この例では， 1 カ月の間に spot レートが 100 円/ドルを超えた日数の割合を計算．

5.　デリバティブ商品定義 DSL の評価

5.1　実用性評価

5.1.1　評価方法

実用性評価として，次の 2 項目について確認した．

(1) 複雑なデリバティブ商品の記述

(2) 対象ドメインの妥当性

複雑なデリバティブ商品の記述評価は，実際の金融機関で取引されている複雑な商品を表現できることを確認した．対象ドメインの妥当性は，3 章で示した対象商品・システム機能が実用において不足がないか確認した．以下で，それぞれの評価結果を説明する．

5.1.2　評価結果

複雑なデリバティブ商品の記述

評価を行うための対象商品として，「リバースフローター MTN」という商品を選択した．リバースフローター MTN は，次の要素の組合せで構成される．

- 原資産：金利
- オプション条項：
  ターゲットリデンプション + キャップ・フロア
- ペイオフ計算式：
  リバースフローター + フリップフロップ
- キャッシュフロータイプ：債券型

複数のオプション条項とペイオフ計算式が組み合わされた複雑な構成である（タームシートを**図 7** に示す）．

この商品は債券型のキャッシュフローが発生し，契約時に額面金額を受け取ったあと，定期的にクーポンの支払いがある．そして満期日に償還金額の支払いがある．オプション条項としてはターゲットリデンプション条項が付いており，支払われた累積クーポンが 12%となったら早期償還となる．またクーポンは 1 年目のみ固定 6%であるが，2 年目以降は変動レートで，定義された式に従い決定される．

```
Issuer                AAA Bank
Principal Amount      USD 6,700,000
Trade Date            2012/11/1
Payment Date          2012/11/8
Maturity Date         2022/11/8
Issue Price           97.50% of Par.
Redemption Price      100.00% of Par.
Coupon                Year 1        6.00%
                      Year 2-10     {10.00% - 2 * 12m$L}
                        Semi annually in arrear
                          on 30/360 day basis.
                        Minimum 0.00%,
                        Lifetime Cap あり（以下に記載）
12m\$L In Arrears     USD 12 カ月 libor レート
                        各クーポン期間の最終日の 5 営業日前，
                        Telerate P3750 11 時 (TKY) 表示レート
Lifetime Cap          12.00%
                        クーポン累積金額の Cap
                        LifetimeCap に到達した場合，自動早期償還
                        最終クーポンとして
                        （LifetimeCap - 前回まで累積クーポン）
                        が支払われる．
Early Redemption      クーポン支払日に LifetimeCap に達すると
                      早期償還
Coupon Payment Date   Annually,
                        Payment Date の 12 カ月後から
                        Maturity Date まで
Business Days         Tokyo
Day Convention        Following
```

図 7　リバースフローター MTN のタームシート
Fig. 7　Term sheet of Reverse Floater MTN.

```
ReverseFloaterMtn {
    #principal.Amount       :   USD 6,700,000;
    #payment.Date           :   2012/11/8;                          #maturity.Date          :   2022/11/8;
    #calendar               :   TKY;                                #dayConvention          :   Following;
    #issue.Price            :   97.50%;                             #redemption.Price       :   100.00%;
    #refference.Price       :   USD/"12m$L" 30/360;
    #year1Coupon.Price      :   6.00% 30/360;
    #year2to10Coupon.Price  :   "0.1 - 2 * refference.Price" DC30_360;
    #floorRate              :   0.00%;
    #lifeTimeCapRate        :   12.00%;
    CouponYear1 {
        CouponPayment ---
            EventDate       :   Frequency Annually (payment.Date _to_ payment.Date+1year) EndOfPeriod;
            Type            :   "Payoff";
            Description     :   "coupon.Amount = principal.Amount * year1Coupon.Price";
        ---
    }
    CouponYear2to10 {
        CouponPriceAggregation ---
            EventDate       :   Frequency Annually (payment.Date+1year _to_ maturity.Date) EndOfPeriod-5days;
            Type            :   "Formula";
            Description     :   "Formula  :  aggregateCoupon.Price = SUM(year1Coupon.Price) + SUM(year2to10Coupon.Price)";
        ---
        RefferenceRateFixing ---
            EventDate       :   Frequency Annually (payment.Date+1year _to_ maturity.Date) EndOfPeriod-5days;
            Type            :   "Fixing";
            Description     :   "refference.Price";
        ---
        CouponCalculation ---
            EventDate       :   Frequency Annually (payment.Date+1year _to_ maturity.Date) EndOfPeriod;
            Type            :   "Formula";
            Description     :   "Formula  :   year2to10Coupon.Price = 0.1 - 2 * refference.Price;
                                 Cap      :   lifeTimeCapRate - aggregateCoupon.Price;
                                 Floor    :   floorRate";
        ---
        CouponPayment ---
            EventDate       :   Frequency Annually (payment.Date+1year _to_ maturity.Date) EndOfPeriod;
            Type            :   "PayOff";
            Description     :   "coupon.Amount = principal.Amount * year2to10Coupon.Price";
        ---
        TargetEarlyRedemption ---
            EventDate       :   Frequency Annually (payment.Date+1year _to_ maturity.Date) EndOfPeriod;
            Type            :   "ExoticOption";
            Description     :   "IF    SUM(coupon.Amount) >= ( principal.Amount * lifeTimeCapRate )
                                 THEN  Execute.Redemption";
        ---
    }
    Redemption {
        RedemptionPayment ---
            EventDate       :   maturity.Date;
            Type            :   "PayOff";
            Description     :   "principal.Amount * redemption.Price";
        ---
    }
}
```

図 8　デリバティブ商品定義 DSL によるリバースフロータ MTN 商品定義プログラム
Fig. 8　A DSL program for Reverse Floater MTN.

この商品を対象としたデリバティブ定義 DSL による記述を図 8 に示す．その特徴は以下のとおり．

- 1 年目とそれ以降でクーポンの金利計算が異なる．
- ターゲットリデンプション条項により累積クーポンの総額が決まっており，超えた場合は早期償還する．

1 点目については，イベントを分割することで対応し，1

年目のクーポン，2 年目以降のクーポン，償還という 3 つのイベントブロックで定義した．2 点目については，次のように複数のイベントを定義することで対応した．

(1) CouponPriceAggregation（Formula）：クーポンの累積レートを計算
(2) RefferenceRateFixing（Fixing）：変動金利の確定
(3) CouponCalculation（Formula）：クーポンの計算（(1) で計算した累積レートが上限を超えないように Cap を定義する）
(4) CouponPayment（Payoff）：クーポンの支払い
(5) TargetEarlyRedemption（ExoticOption）：早期償還の確認，実行

このように，複数のイベントを組み合わせることで，リバースフロータ MTN を記述でき，正しく動作することが確認できた．他の OTC デリバティブ商品も，原資産，オプション条項，ペイオフ計算式，キャッシュフロータイプの組合せで表現できるため，デリバティブ商品定義 DSL は，複雑な商品の記述において問題ないと評価する．

対象ドメインの妥当性

対象ドメインの妥当性については，専門家レビューを実施し，得られた意見から評価した．レビューは，業務担当者[*1]1 名とシステム担当者 1 名に依頼し実施した．レビュー対象とした資料は，デリバティブ商品定義 DSL の説明資料，複雑な商品の商品定義プログラム，および動作するテスト用アプリケーション，である．

まず対象商品の妥当性について，3.2 節に示した 4 つの構成要素の分類が妥当であるか，そして，その内容が実際の金融機関で取引されている OTC デリバティブ商品をどの程度カバーしているかが評価基準となる．レビュー結果において，4 つの分類は妥当であり，レビュアが担当している情報システムで対応しているデリバティブ商品をデリバティブ商品定義 DSL で網羅的に表現できるとの判断が示された．このレビュアの判断だけでは，すべての金融機関で取り扱うデリバティブ商品の網羅性を示すことにはならないが，必要最低限の網羅性はあると評価する．

次に対象機能の妥当性について，実務で利用するうえで不足がないか，そして不足がある場合に容易に拡張が可能であるかが評価基準となる．レビュー結果において，レビュアからは一部の機能不足の指摘があった．指摘された機能不足は日付展開に関するものである．

- 1 つの商品の中での複数の営業日カレンダの指定
- 繰返しイベントでの初回または最終回の期間調整

1 つ目の指摘は，次のようなケースである．商品の中で複数のマーケットデータを参照しており，そのマーケットの都市が異なる場合，それぞれの都市の営業日でイベント日を決定する必要がある．この指摘事項について，デリバティブ商品定義 DSL の拡張は容易であり，言語の構文や適応モデルの大幅な変更は不要である．そのため，デリバティブ商品定義 DSL の機能妥当性は問題ないと評価する．

2 つ目の指摘については，たとえば，取引期間が 13 カ月で，イベント間隔が 6 カ月であるケースが該当する．13 カ月は 6 カ月間隔で割り切れないため，余りの月を考慮する必要がある．この場合，以下を選択できる必要がある．

- 最初の期間を 1 カ月とする．
- 最初の期間を 7 カ月とする．
- 最後の期間を 1 カ月とする．
- 最後の期間を 7 カ月とする．

この指摘は，機能追加を行わなくてもイベントを分割して定義することで対応できる（最初の 1 カ月を 1 つのイベントとして，残りの 12 カ月を 6 カ月間隔繰返しのイベントとして定義するなど）．これは，ドメイン知識を十分抽象化できていない部分であるため，デリバティブ商品定義 DSL の機能として指定できることが望ましい．ただし，現機能でも回避方法があり，致命的な問題ではない．そのため，この指摘についても機能妥当性に問題ないと評価する．

5.2　可読性評価

5.2.1　評価方法

可読性の評価は，評価者に記述式の可読性テストを実施してもらう形式で行った．以下に，実施方法について記す．

実施概要

可読性テストは，評価者と対面で実施した．実施時間は特に定めず，すべての回答が終了するまでとした．

評価者

OTC デリバティブ取引業務の経験者をドメイン専門家として選定した．金融機関業務担当者[*1]1 名，業務コンサルタント 1 名，システム開発担当 2 名の合計 4 名が評価を実施した．

事前説明

評価を実施する前に，デリバティブ商品定義 DSL の概要と記述方法について，資料（パワーポイント，スライド 8 枚）を配布し説明した後，サンプルアプリケーションを利用し商品定義がどのように動作するかデモンストレーションを実施した．すべての評価者に対し，同じ資料を使用し，同じ内容を説明した．説明時間は約 20 分とした．

可読性テスト内容

記述式のテストを準備し，評価者に記入してもらう形式で実施した．テスト用紙は A3 横の用紙に，左側に商品定義，右側に問題と回答の構成となっており，左側の商品定義を読んで，右側の問題に回答する．問題は大問が 2 題あり，タームシートによる商品定義を読んで回答する問題と，デリバティブ商品定義 DSL の商品定義プログラムを読んで回答する問題の構成となっている．それぞれの大問は，さらに 2 題の小問で構成されており，取引ライフサイクル

[*1] 前職の金融機関において OTC デリバティブ業務を担当．

表 10　可読性テスト結果一覧
Table 10　Result of readability test.

		金融業務担当者（元） A氏	業務コンサルタント B氏	システム開発担当者 C氏	システム開発担当者 D氏
	問題順序	問題1→問題2	問題1→問題2	問題2→問題1	問題2→問題1
問題1 タームシート	回答時間	約14分	約15分	約10分	約12分
	小問1 結果	○	× 誤答3カ所	× 誤答1カ所	○
	小問2 結果	○	× 誤答1カ所	○	○
問題2 デリバティブ 商品定義DSL	回答時間	約15分	約19分	約20分	約18分
	小問1 結果	○	○	○	○
	小問2 結果	○	○	○	○

で発生するキャッシュフローを記述する問題と，早期償還のオプション条項を記述する問題がある．問題に使用しているタームシートの商品定義とデリバティブ商品定義 DSL による商品定義は，それぞれ別の商品だが，同等の複雑さを持つ商品とした．なお，複雑さについては，関連研究の RISLA [10] では対応していないと考えられるターゲットリデンプションを含む商品を選択している．大問の 2 題については，評価者によって実施順序を変更し，2 名がタームシートの問題を先に実施し，2 名がデリバティブ商品定義 DSL の問題を先に実施するようにした．

デリバティブ商品定義 DSL についてのアンケート

可読性テスト終了後，評価者に選択式のアンケートに回答してもらった．その概要は，以下のとおり．

- テスト問題の難易度
  （やさしい/どちらかといえばやさしい/どちらかといえば難しい/難しい）
- 言語構文の理解しやすさ
  （分かりやすい/どちらかといえば分かりやすい/どちらかのいえば分かりにくい/分かりにくい）
- プログラミングまたはコードレビューが可能か
  （複雑な商品で可能/簡単な商品で可能/不可能）

インタビュ

可読性テストおよびアンケート終了後に，主に，全体的な感想，対象ドメインの妥当性，言語の良い点/改善点などについて，フリートーク形式でインタビュを実施した．

5.2.2　可読性テスト評価結果

可読性テスト結果を**表 10** に示す．

- デリバティブ商品定義 DSL の問題は全員が正解した．タームシートの問題は一部の評価者で誤答があった．
- 回答に要した時間は，タームシートの問題より，デリバティブ商品定義 DSL の問題の方が長かった．

可読性テスト実施中の評価者の観察結果，および質問事項について整理する．

- タームシートの問題において，評価者から商品定義の内容についての質問はなかった．
- デリバティブ商品定義 DSL の問題において，評価者から構文の意味について同じ質問がいくつかあった．
  – 式の中の 0.1 は，%を表すのか，実数を表すのか？
  – 式の中の SUM は，どの期間の合計か？
  – Formula イベントの中の Cap（上限）と Floor（下限）はどの値に対するものか？

タームシートによる商品定義の問題は，読解時間は短いが，いくつか誤答があった．一方でデリバティブ商品定義 DSL のテストは，読解時間は若干長かったが，誤答がまったくないという結果になった．タームシートの読解は，テスト中の質問がまったくなかったことからも，1 つ 1 つの契約事項の意味は理解しやすかったと考えられる．しかしその一方で，タームシートでは，当然発生すると業務上認識されているイベントは明確に記述されていなかったり，契約事項の構造化が十分行われていなかったりするため，関連が分かりにくいという面がある．そのため誤答があったと考えられる．テストで利用したタームシートの例では，以下の記述から，「TradeDate において Principal Amount に Issue Price を乗じた金額のキャッシュフローが発生する」ことを読み取る必要があるが，このキャッシュフローを回答できなかった評価者が 2 名いた．

```
Principal Amount    :  JPY 1,000,000
Trade Date          :  2007/7/20
Issue Price         :  100.00% of Par.
```

タームシートでは，契約で明確にすべき日付，金額，レートなどは曖昧性なく定義されているが，正しく理解するためには，一定以上の業務知識が必要となることが分かる．タームシートによる業務担当者とシステム開発担当者のコミュニケーションでは仕様齟齬が発生してしまう可能性が高いことが示唆される．

デリバティブ商品定義 DSL の問題で回答に時間がかかった理由は，タームシートに比べて記述量が多いこともあるが，テスト時の質問で示されるとおり，記述の表す意味が

分からない部分があったためであると考えられる．一方で，発生するイベントについては，イベント定義という形式で明示的に記述されているため，タームシートのテストのようにイベントが認識できずに漏れてしまうことがない．これらのことから，質問が発生し理解に時間がかかったものの，正しい回答ができたと考えられる．

デリバティブ商品定義 DSL の可読性について，記述の表している意味が分からない部分が存在したことは課題である．しかし，当テストにおいては，言語の説明が 20 分だけで，文法や記述方式についてすべてを説明していおらず，上述の質問だけですべて正答できたことを考慮すると，一定水準の可読性はあったといえる．マニュアル・チュートリアルの整備とトレーニングにより，この課題は改善できると考える．

当テスト結果において，デリバティブ商品定義 DSL は，タームシートと同等以上の可読性を示したと評価できる．

5.2.3　アンケート結果と評価

アンケート結果について項目ごとに検討する．

商品定義の読解難易度

すべての評価者において，タームシートの難易度とデリバティブ商品定義 DSL の難易度が同じとなった．金融業務担当者は「A. 易しかった」と回答しており，業務担当者がプログラムをレビューできるという本研究の目標について，ポジティブな回答を得られた．

デリバティブ商品定義 DSL の表記，構文の分かりやすさ

ほとんどの項目について「B. どちらかといえば分かりやすい」以上のポジティブな回答が得られた．「C. どちらかといえば分かりにくい」と回答があったのは，契約事項定義の項目名の表記について，1 名のみで，「D. 分かりにくい」と評価された項目は皆無であった．

商品定義 DSL の問題における商品定義プログラムは，元となる商品のタームシートをベースに記述した．その際，タームシートに記載がある契約事項の項目名については，商品定義 DSL のプログラムでも同じ名称を利用したが，イベントを明示的に記述する必要があるため，タームシートで名称が定義されていない項目が現れる．当可読性テストにおいては以下のケースである．

```
Coupon  :  Year 1      6.00%
           Year 2-10   {10.00% - 2 * 12m$L}
```

タームシートでは Coupon レートがこのように定義されているが，デリバティブ商品定義 DSL は，以下のように明示的な項目名を付与する必要がある．

```
#year1Coupon.Price     : 6.00%;
#year2to10Coupon.Price: "0.1-2*ref.Price";
```

この例において，名前が長く分かりにくいという感想があった．業務では一般的に英語のタームシートが利用されているため，本研究のデリバティブ商品定義 DSL は，日本語などのマルチバイトに対応していない．しかし，このようなケースでは，日本語で名前を付けた方が簡潔で分かりやすい可能性があり，今後の改善ポイントとして検討の余地がある．

デリバティブ商品定義 DSL でプログラミング可能か

「B. 簡単な商品ならばプログラミングできると思う」と回答したのが 3 名，「A. 複雑な商品でもプログラミングできると思う」と回答したのが 1 名であった．

評価を依頼した 4 名について，勤務先の業務においてプログラミングを行っている者はいない．デリバティブ商品定義 DSL による商品定義プログラムの記述は，従来どおりプログラマが開発することを想定しているため，プログラマではない評価者による，この回答は想定以上にポジティブなものであった．「できると思う」ことと，実際にプログラミングができることは異なるため，この回答により業務担当者がプログラムまでできるということはできないが，その可能性があることを示唆している．

デリバティブ商品定義 DSL コードのレビュー可能性

「B. 簡単な商品ならばレビューできると思う」と回答したのが 1 名，「A. 複雑な商品でもレビューできると思う」と回答したのが 3 名であった．

金融業務担当者は A と回答しており，本研究における業務担当者がプログラムをレビューできるという目標に適う結果であった．1 名，B の回答があったが，読解テストでは正答しており，ある程度のトレーニングを行えば，実際にはレビュー可能な可能性は十分にあると考える．

5.2.4　インタビュの意見と評価

インタビュにより得られた意見を以下に示す．金融機関業務担当者の意見には（業）を，システム開発担当者およびコンサルタントの意見には（シ）を末尾に付記した．

- デリバティブ商品定義 DSL の良い点
  - タームシートと表記方法が似ており違和感がない．（業）
  - 日付の表記が分かりやすい．（シ）
  - 新商品のシステム対応は，大きなシステム改修が必要になることが多いので，この発想はとても良い．（シ）
- デリバティブ商品定義 DSL の改善点
  - レートの数値は，%表記か実数表記か統一した方がよい．（シ）
  - 計算式の SUM の使い方が分かりにくい．（業，シ）
  - タームシートと同じ項目を，同じ位置に表記することができればレビューしやすいと思う．（シ）
  - イベント定義の表記は分かりにくいとは思わないが，括弧の使い方などプログラミング言語的なので，業務担当者によってはそれだけで拒否反応があるかもしれない．（シ）
- その他，感想など
  - タームシートが読める人ならば，デリバティブ商品定義 DSL も読めると思う．（業，シ）

– 文法が分かれば難しくない.（業）
– 取引入力画面も DSL から生成できると良いと思う.（シ）
– データ永続化も商品に登録する属性が異なるので，同じように対応できる良いのではないか.（シ）
– 文法や表記については自分で考えても同じようになると思う.（シ）
– デリバティブ金融商品は取引ライフサイクルでデータが更新されていくため，何かイベントがあるたびに他システムとのデータ連携が必要．データではなくこの商品定義を連携するようにすれば，イベント発生の度に連携しなくてもよくなるかもしれない.（シ）

デリバティブ商品定義 DSL のコンセプトや文法に大きな変更が必要となるような否定的な意見はなく，全体として好評価であった．改善点についていくつか意見がでたが，現状問題のある点については，レートの数値表記や計算式の SUM 表記の指摘のみで，「現状問題ないがこうすればより良くなるのではないか」，という指摘が多数であった．このことから，改善の余地はあるものの，DSL の方向性としては有効であることが示唆される．

5.2.5　可読性評価まとめ

可読性テスト，アンケート，およびインタビュから，可読性について次のことが示された．

- タームシートと同等以上の可読性を示した．
- 業務担当者がプログラムレビューできる可能性が高い．

可読性テストにおいて，タームシートより正答率が高い結果となり，表記が分かりにくい部分があったもののタームシートと同等以上の可読性があったといえる．また，アンケートおよびインタビュにおいて，評価者の大半が，タームシートと同様の難易度で，複雑な商品でもレビュー可能，また簡単な商品であればプログラミングまでできると感じた．このことは，実際にレビューできることを直接示していないが，可読性テストの結果と合わせてレビューできる可能性が高いと判断できる．

これらの効果の一方で，いくつかの課題も明らかになった．

- 一部の表記において，それが表している意味を理解しにくいものがある．
- 日本語の利用など，改善が期待できる事項が明らかとなり，さらなる評価が必要であることが示された．

表記については，計算式における SUM 関数など，意味が理解しにくい表現がいくつか明らかとなった．また，日本語の利用など，改善の可能性がある項目がいくつか示された．その点については，さらなる評価により効果を明らかにする必要があると考えている．

以上のことより，いくつかの課題があるものの，本研究で目標としている可読性について，デリバティブ商品定義 DSL は有効であると評価できる．

5.3　DSL 設計指針についての考察

デリバティブ商品定義 DSL では，業務担当者がコードレビューできることを目標としタームシートを参考に言語設計を行った．業務専門家による可読性評価結果から，DSL の一般的な設計指針について次のことが示唆される．

- 対象ドメインで利用されているドキュメントや用語・表現と近い形式の記法とすることで，ドメイン専門家による可読性向上が期待できる．
- ただし，ドメイン知識が過度に抽象化されている用語・表現の利用はドメイン知識が不十分な開発担当者の読解を妨げるため，適切なレベルで具体化・詳細化することが有効である．

ドメイン専門家とシステム開発担当者の両者が読解可能な DSL を作成するためには，この 2 点について適切なバランスで言語設計を行うことが重要であると考える．

また，上記以外の観点として，プログラマの実装における生産性や品質への影響についての考慮が必要であるが，本論文では評価を実施しておらず今後の課題である．

6.　今後の課題

本論文には以下の課題が残されている．

- デリバティブ商品定義 DSL 一部の表記において，表す意味を理解しにくいものがあり，改善が必要．
- 本論文の評価ではタームシートと同等以上の可読性が示されたが，金融機関業務担当者の評価件数が少なく，さらなる評価が必要．
- 業務担当者が実際の業務でプログラムをレビューできるかについてはさらなる検証が必要．
- デリバティブ商品定義 DSL による実装について，汎用言語より高い生産性が得られるかについては評価していない．
- デリバティブ商品定義 DSL の改善が期待できる項目が明らかとなり，その対応と評価が必要．

本論文の評価においては，致命的ではないものの，明らかに理解しにくい表記が示された．この点は今後改善する必要がある．また，可読性評価の実施件数が少なく，金融業務担当者は 1 名のみである．本論文では否定的な評価はなかったが，さらなる評価が必要である．実際の業務において金融業務担当者がでコードレビューできるかについても，アンケートおよびインタビュにより可能性が示唆されたまでであり，より詳細な評価が必要である．本研究の最終的な目的はシステム開発生産性向上であるが，業務担当者がコードレビューすることによる生産性の向上は，要件齟齬が少なくなり，品質が向上することにおける副次的な効果を期待したものである．もし，プログラミング作業自体の生産性が汎用言語よりも高ければ，さらに有効性が示せることになる．このことからプログラミングの生産性評価は取り組みたい課題である．本論文においては評価者に

よる貴重な意見，テスト結果により，様々な改善ポイントが示唆された．この情報を有効に活用しさらなる改善に取り組む必要がある．

7. まとめ

本論文においては，OTC デリバティブ取引管理システムの開発生産性が低くシステム化が十分実施されていないという問題について，汎用プログラミング言語ではなく，DSL を利用するというアプローチを検討した．DSL のメリットを最大化するように対象ドメイン分析を実施し，デリバティブ商品定義 DSL と呼ぶ DSL を開発した．そしてこの言語について，実用性と可読性について評価を実施した．

評価結果において，デリバティブ商品定義 DSL が対象ドメインに不足なく対応し実用性に問題がないこと，そして，この言語による商品定義プログラムを業務担当者がレビューできる可能性が高いことを示した．DSL によるアプローチは有効であるが，多くの課題が残されており，最終的な目標である開発生産性向上の実現には，今後さらなる改善と評価が必要である．

謝辞　DSL の設計・開発において，貴重なアドバイスや有益な情報をご提供いただいた有識者の方々，そして，言語評価に快くご協力いただいた評価者の方々に，深く感謝の意を表します．また，株式会社電通国際情報サービスの上司，同僚の方々のご理解とご協力により本研究を遂行できました．深く感謝いたします．

参考文献

[1] van Deursen, A., Klint, P. and Visser, J.: Domain-specific Languages: An Annotated Bibliography, *ACM Sigplan Notices*, Vol.35, No.6, pp.26–36 (2000).
[2] Al-Rawas, A. and Easterbrook, S.M.: *Communication Problems in Requirements Engineering: a Field Study*, National Aeronautics and Space Administration (1996).
[3] Mernik, M., Heering, J. and Sloane, A.: When and How to Develop Domain-specific Languages, *ACM Computing Surveys* (*CSUR*), Vol.37, No.4, pp.316–344 (online), DOI: 10.1145/1118890.1118892 (2005).
[4] Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M. and Völkel, S.: Design Guidelines for Domain Specific Languages, *The 9th OOPSLA Workshop on Domain-specific Modeling*, Citeseer (2009).
[5] Spinellis, D.: Notable Design Patterns for Domain-specific Languages, *Journal of Systems and Software*, Vol.56, No.1, pp.91–99 (online), DOI: 10.1016/S0164-1212(00)00089-3 (2001).
[6] Visser, E.: Domain-specific Language Engineering — A Case Study in Agile DSL Development, *Development*, No.Mark I (2007).
[7] Visser, E.: WebDSL: A Case Study in Domain-specific Language Engineering, *Generative and Transformational Techniques in Software Engineering II*, pp.291–373 (2008).
[8] Bunch, C., Chohan, N., Krintz, C. and Shams, K.: Neptune: a Domain Specific Language for Deploying hpc Software on Cloud Platforms, *Proc. 2nd International Workshop on Scientific Cloud Computing, ScienceCloud '11*, pp.59–68, ACM (online), DOI: 10.1145/1996109.1996120 (2011).
[9] Ding, C.: OTC Derivatives and Structured Product Pricing Practices: Trends and Technology Strategies for the Coming Market Reformation, Technical report, CELENT (2009).
[10] van Deursen, A.: Domain-Specific Languages versus Object-Oriented Frameworks: A Financial Engineering Case Study, *Smalltalk and Java in Industry and Academia, STJA' 97*, pp.35–39 (1997).
[11] Jones, S. and Eber, J.: Composing Contracts: an Adventure in Financial Engineering (Functional Pearl), *Proc. 5th ACM* (2000).
[12] Frankau, S., Spinellis, D., Nassuphis, N. and Burgard, C.: Commercial uses: Going Functional on Exotic Trades, *Journal of Functional Programming*, Vol.19, No.01, p.27 (online), DOI: 10.1017/S0956796808007016 (2008).
[13] Hull, J.C.: *Options, Futures and Other Derivatives*, 8th edition, Pearson Education (2011).
[14] みずほ証券マーケット研究会：デリバティブ・証券化商品入門，東洋経済新報社 (2008).
[15] Fowler, M.: *Domain-Specific Languages*, Addison-Wesley Professional (2010).
[16] Gagnon, E. and Hendren, L.: SableCC, an Object-oriented Compiler Framework, *Proc. Technology of Object-Oriented Languages, TOOLS 26* (*Cat. No.98EX176*), pp.140–154, IEEE Comput. Soc (online), DOI: 10.1109/TOOLS.1998.711009 (1998).

松本 吉史　（正会員）

1974 年生．1999 年慶應義塾大学大学院理工学研究科修士課程修了．同年株式会社電通国際情報サービス入社，金融機関向けの受託システム・ソフトウェア開発に従事．2013 年筑波大学大学院ビジネス科学研究科修士課程修了．

久野 靖　（正会員）

1956 年生．1984 年東京工業大学大学院理工学研究科博士後期課程単位取得退学．同年東京工業大学理学部情報科学科助手．1989 年筑波大学講師．1990 年同助教授．2000 年同教授．理学博士（東京工業大学）．プログラミング言語，ユーザインタフェース，情報教育に関心を持つ．ACM，IEEE-CS，日本ソフトウェア科学会，日本情報科教育学会各会員．