RelaxerStudio

浅海　智晴 [1)]　　五嶋　剛 [2)]　　山中　崇文 [3)]　　永山　葉子 [4)]
Tomoharu ASAMI　Tsuyoshi GOTO　Takafumi YAMANAKA　Yoko NAGAYAMA

1) (有) 浅海智晴事務所 asami@AsamiOffice.com

2) (株) クレオ gotot@creo.co.jp

3) (株) クレオ yamanat@creo.co.jp

4) (株) クレオ nagayay@creo.co.jp

ABSTRACT. The RelaxerStudio projects developed RelaxerStudio, Relaxer NG engine for Relaxer, Relaxer process, a sample application, a Relaxer tutorial in English, and a Relaxer Referenc Manual in English. RelaxerStudio is a GUI model editor to edit RELAX NG models and Relaxer CDL models. Relaxer process is a object-oriented development process for Relaxer. The sample application is developed to verify availability of Relaxer process.

1　背景

SE が業務アプリケーションを構築するために必要となるスキルレベルは年々高くなっており、一般的な SE のスキルレベルとは大きな乖離が発生している。

オブジェクト指向開発プロセス技術は Unified Process (UP) によって一応の完成を見たが、コンポーネント開発への脱皮、XML 技術の統合など、新たなステージへの移行が急務となっている。

Relaxer は、XML スキーマ言語 RELAX を核にしたデータアーキテクチャと、Relaxer CDL によるコンポーネントアーキテクチャにより、分散コンポーネントをベースとした業務アプリケーションの構築を支援する。しかし、現在の Relaxer では、コマンドベースのインタフェースとなるため高いスキルを持った上級 SE でないと活用することができない。また、開発プロセスの支援がないため、組織的に Relaxer を導入することも難しい。

2　目的

Rleaxer Project は、Relaxer 活用を目的としたモデルエディタ RelaxerStudio および RelaxerStudio を核にした開発プロセスである Relaxer プロセスを提供することで、背景で説明した問題を解決することを目的としている。

これに加えて、日本発の技術を世界に発進していくという点も目的の一つである。

最新技術情報の多くが米国で開発されているため、技術情報が英語中心となっている点も日本国内の SE にとって逆風となっている。RELAX および Relaxer は、いずれも日本が中心になって開発されている技術であり、日本語においてもっとも詳細な情報が流通している。このような技術を世界に対して発信し、デファクトスタンダードとしていくことで日本のソフトウェア産業にとっての財産となる。

3　プロジェクト内容

「RelaxerStudio プロジェクト」のテーマは、プロジェクトマネージャである萩谷 PM の指示にもある通り「XML スキーマに勝て！」である。つまり、日本発の技術である RELAX および Relaxer を世界標準とすることがこのプロジェクトの最終的なミッションとなる。このため、いかに RELAX および Relaxer を普及させていくのかという点が重要なポイントとなる。

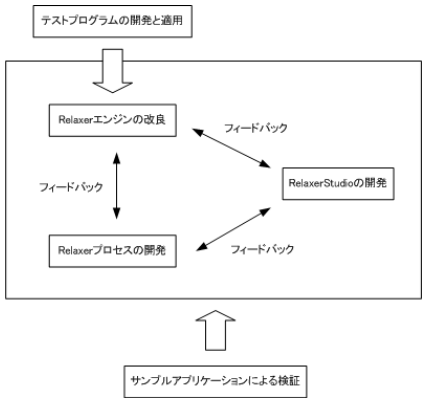

図 1: RelaxerStudio プロジェクトの活動

このミッションを達成するために、プロジェクトでは以下の作業を行った。

- Relaxer エンジンの改良 (RELAX NG 完全対応)
- テストプログラムの開発
- RelaxerStudio の開発
- Relaxer プロセスの開発
- サンプルアプリケーションの開発

RelaxerStudio プロジェクトの全体の枠組みは図 1 となる。

Relaxer エンジンの改良は、Relaxer の RELAX NG 完全対応を主なターゲットとしている。また Relaxer エンジンの改良を支援するために、テストプログラムの開発を行い、テスト結果を Relaxer エンジンにフィードバックした。

Relaxer エンジンの改良と並行して、RelaxerStudio および Relaxer プロセスの開発を並行して行い、それぞれの開発内容を相互にフィードバックした。

また、Relaxer プロセスの検証を行うために、サンプルアプリケーションを開発した。

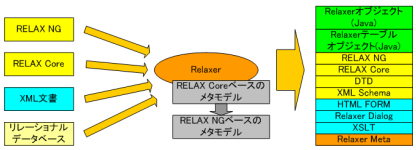

図 2: スキーマコンパイラ

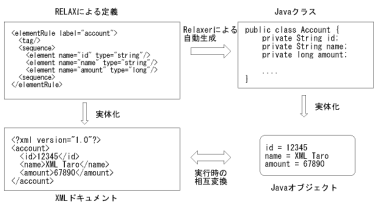

図 3: XML と Java のマッピング

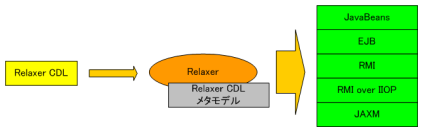

図 4: コンポーネントコンパイラ

4 Relaxer とは

RelaxerStudio および Relaxer プロセスの話に入る前に Relaxer について簡単に説明する。Relaxer は XML スキーマ言語である RELAX ベースの XML スキーマコンパイラかつ、独自コンポーネント記述言語 Relaxer Component Definition Language (RCDL) ベースのコンポーネントコンパイラである。

4.1 スキーマコンパイラ

Relaxer の持つ最重要機能がスキーマコンパイラである。Relaxer は図 2 に示すように RELAX スキーマをはじめとする入力から、さまざまな出力を生成する。

まず入力となるのは以下の 4 つである。

- RELAX NG スキーマ
- RELAX Core スキーマ
- XML 文書
- リレーショナルデータベーススキーマ

つぎに出力となるのは以下の 10 種類である。

- Relaxer オブジェクト
- Relaxer テーブルオブジェクト
- RELAX NG
- RELAX Core
- DTD
- XML Schema
- HTML FORM
- Relaxer Dialog
- XSLT
- Relaxer Meta

Relaxer オブジェクトは RELAX でモデル化された XML 文書をマッピングした Java オブジェクトである。XML 文書との相互変換が可能である。

Relaxer テーブルオブジェクトは Relaxer オブジェクトおよび XML 文書を RDBMS に格納するためのデータアクセスオブジェクトである。

また Relaxer は RELAX NG スキーマ、RELAX Core スキーマ、DTD スキーマ、XML Schema スキーマの 4 種類のスキーマを生成することができる。モデリング時には RELAX NG スキーマまたは RELAX Core スキーマを作成し、目的に応じて DTD や XML Schema に変換して利用することが可能となる。

プロジェクト開始前の Relaxer では、内部データとして RELAX Core ベースのメタモデルを使用していた。今回のプロジェクトでは、このメタモデルを RELAX NG ベースのものに更新することで RELAX NG スキーマを入力にすることが可能になった。

その他、HTML FORM や XSLT スタイルシートなどの成果物も生成することができる。

スキーマコンパイラの 1 つの応用がデータバインディングである。図 3 に示すように、RELAX スキーマから、これに対応する Java クラスを生成する機能がデータバインディングである。生成された Java クラスを用いることで、RELAX スキーマに準拠した XML 文書と Java オブジェクトの相互変換を容易に行うことができるようになる。

4.2 コンポーネントコンパイラ

Relaxer は、スキーマコンパイラの基盤の上にコンポーネントコンパイラ機能を実現している。

Relaxer のコンポーネントコンパイラ機能は図 4 に示すように RCDL 定義よりさまざまなコンポーネント実装を生成する。

Relaxer が生成するコンポーネントの種類は以下のものである。

- JavaBeans
- EJB
- RMI
- RMI over IIOP
- JAXM

JavaBeans は、クライアントサイドにおける Java の基本コンポーネントである。

EJB は、J2EE アーキテクチャ上で用いられるサーバサイドコンポーネントである。

RMI は、Java 専用の RPC メカニズムである。

RMI over IIOP は、Java 上における CORBA 分散オブジェクトの実装である。

JAXM は、XML ベースメッセージングのための API である。現在のところ、事実上 Java における SOAP API と考えてよい。つまり、JAXM を利用することで Web サービスのコンポーネントを実現できる。

Relaxer のコンポーネントコンパイラによって実現される世界は図 5 となる。

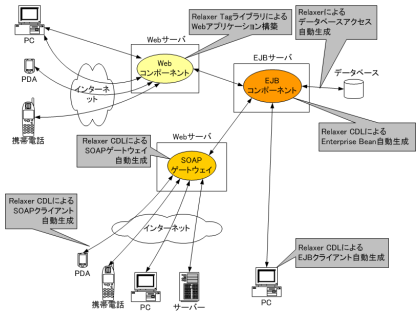

図 5: J2EE における Relaxer

図 6: プロジェクトの成果

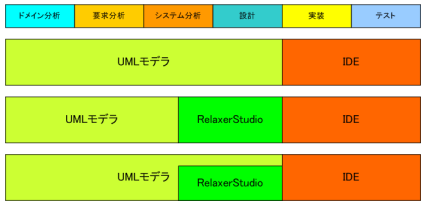

図 7: 位置付け

5 プロジェクトの成果

プロジェクトの成果は図 6 に示すように大きく以下の 3 つのテーマに分けることができる。

- Relaxer RELAX NG 対応
- オブジェクト指向開発との連携
- 普及活動

「Relaxer RELAX NG 対応」の成果物は、「Relaxer ERLAX NG エンジン」と「Relaxer テストプログラム」である。「Relaxer ERLAX NG エンジン」は、Relaxer のエンジンを従来の RELAX Core ベースを拡張して RELAX NG ベースにする機能拡張である。「Relaxer テストプログラム」は、RELAX NG ベースに拡張された Relaxer の機能テストを行うためのテストプログラムである。

「オブジェクト指向開発との連携」の成果物は、「Relaxer プロセス」と「Relaxer プロセスサンプルアプリケーション」である。「Relaxer プロセス」は、Relaxer の持つスキーマコンパイラ、コンポーネントコンパイラ機能の活用を前提としたオブジェクト指向開発プロセスである。「Relaxer プロセスサンプルアプリケーション」は、この「Relaxer プロセス」の実用性を検証するために作成したサンプルアプリケーションである。

「普及活動」の成果物は、「Relaxer リファレンスマニュアル」と「Relaxer チュートリアル」である。いずれも英文である。

そして、「Relaxer RELAX NG 対応」、「オブジェクト指向開発との連携」、「普及活動」のすべてのテーマにかかわってくるのが「RelaxerStudio」である。

Relaxer が RELAX NG を本格サポートすることを受けて、RelaxerStudio では RELAX NG エディタを核の機能としている。

オブジェクト指向開発の中で Relaxer を活用するためには、UML モデラや IDE などの CASE ツールとの連携が必須となる。RelaxerStudio は Relaxer の利用を支援する CASE ツールであり、UML モデラや IDE との連携を RelaxerStudio を通して行うことで、オブジェクト指向開発においてシームレスに Relaxer を活用できるようになると期待できる。

RelaxerStudio は、Relaxer を利用する上で必要となる以下の作業を直感的に操作できる GUI として提供する。

- RELAX NG スキーマの編集
- Relaxer の起動

現状では RELAX NG スキーマの編集は、通常のテキストエディタで行う必要がある。RELAX NG スキーマの文法を完全に把握していない初学者がテキストエディタを用いて編集作業することは簡単ではない。また、Relaxer には膨大なオプションが用意されている。これらのオプションを指定することは、初学者には難しい。RelaxerStudio は以上の問題を GUI の提供によって解決することができる。初学者に取って分かりやすい GUI を提供することで、RELAX/Relaxer 技術の普及を促進することが期待できる。

6 RelaxerStudio

オブジェクト指向開発の中での RelaxerStudio の位置付けを図 7 に示す。

図の上側は UML モデラと IDE で構成されたオブジェクト指向開発における現在の CASE ツールの構成である。

図の中側と下側が RelaxerStudio が想定する新しい CASE ツールの構成である。RelaxerStudio は上流工程で用いる UML モデラと下流工程で用いる IDE との間に位置し、UML モデラによるオブジェクトモデルと IDE による Java プログラムの間を XML を用いて接続するための機能を提供する。

図の中側は上流工程の UML モデラと下流工程の IDE の中間の作業を RelaxerStudio が完全に行う構成である。小規模な開発の場合、この構成でも十分機能すると考えられる。

図の中側は中流工程において UML モデラと RelaxerStudio を併用する構成である。RelaxerStudio でモデル化できない詳細部分を UML モデラでモデル化し、2 つのモデルを統合したものを下流工程の IDE につなげていく。

7 RelaxerStudio の仕組み

RelaxerStudio は DOM(Document Object Model) をベースにした XML マルチビューエディタフレームワークをエンジンとして利用している。

この XML マルチビューエディタフレームワークを利用して図 8 に示すように以下の 4 つの編集ビューを提供している。

- グラフビュー
- コンテナビュー
- テキストビュー
- フォームビュー

8 RelaxerStudio の機能

RelaxerStudio の機能について説明する。

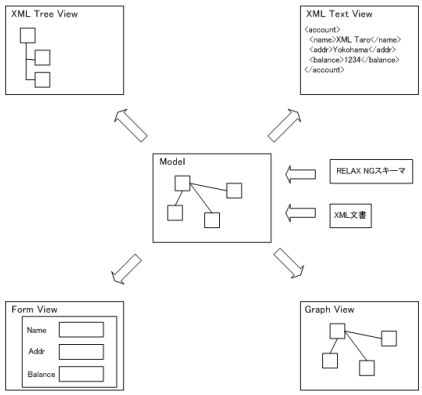

図 8: XML 編集モデル

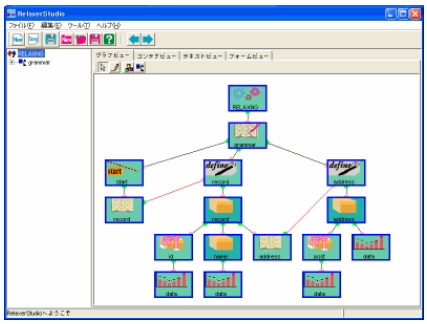

図 9: グラフビュー

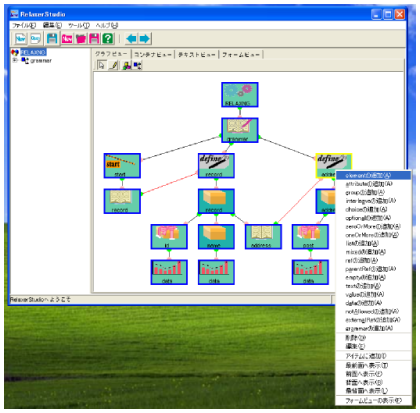

図 10: グラフビューでの編集

8.1 インストール

RelaxerStudio は Java Web Start (JWS) アーキテクチャに則った構造を持っており、JWS を用いて配布される。JWS は、Java が提供する標準のアプリケーション配布機能である。JWS を用いることで、アプリケーションのダウンロードとインストールを Web ブラウザを経由して簡単に行えるようになる。さらに JWS の持つバージョン管理機能により最新バージョンへの自動的なアップグレードも行うことができる。

JWS のサポートにより RelaxerStudio のインストールや管理の手間が大幅に削減されることが期待できる。

8.2 RELAX NG エディタ

RELAX NG エディタは、RELAX NG スキーマを編集するためのエディタである。グラフビュー、コンテナビュー、テキストビュー、フォームビューの 4 つのビューから構成されており、同一のスキーマを様々な観点から編集することができる。これらのビューは同期されているため、どのビューで編集作業を行っても、結果は他のビューに即時に反映される。

8.2.1 グラフビュー

グラフビューを図 9 に示す。グラフビューでは、グラフ構造として表現された各ノードおよびノード間のリンクをダイレクトに編集することができる。

グラフビューでの編集の様子を図 10 に示す。ポップアップメニューにより、ノードの操作を直感的に行うことができる。

8.2.2 コンテナビュー

コンテナビューを図 11 に示す。XML 生来のデータ構造であるツリー構造を直接コンテナという形で表現している。

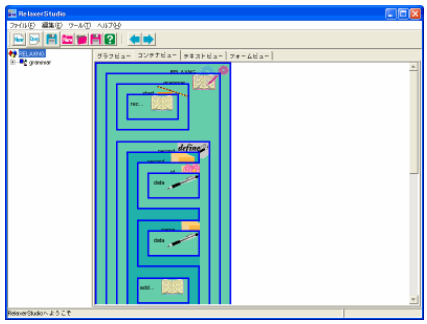

図 11: コンテナビュー

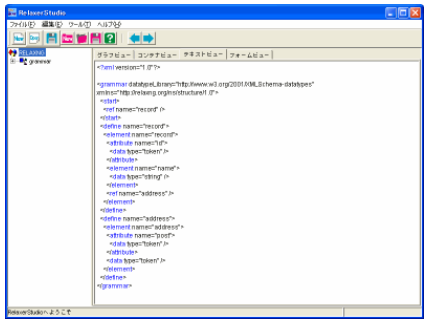

図 12: テキストビュー

8.2.3 テキストビュー

テキストビューを図 12 に示す。テキストビューは RELAX NG スキーマを XML 文書のテキストとして編集するためのエディタである。

テキストビューによる編集の様子を図 13 に示す。通常のテキストエディタとしての編集機能に加えて、図に示すように RELAX NG の文法に従った補完機能を備えている。

8.2.4 フォームビュー

フォームビューを図 14 に示す。フォームビューを利用することで、帳票形式で RELAX NG スキーマの編集を行うことができる。

もちろん、ツリー構造編集のすべての局面で帳票形式の編集がうまく機能するわけではないが、現実に用いられる構造の多くは、表形式の範囲内にあり、このような構造に対して帳票形式に特化した編集機能を大きな威力を発揮するだろう。

8.3 Relaxer CDL エディタ

Relaxer CDL エディタは、Relaxer CDL 定義を編集するためのエディタである。RELAX NG エディタと同様にグラフビュー、コンテナビュー、テキストビュー、フォーム

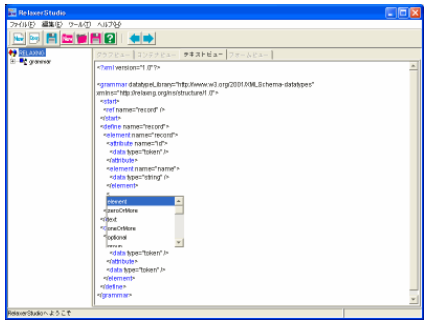

図 13: テキストビューによる編集

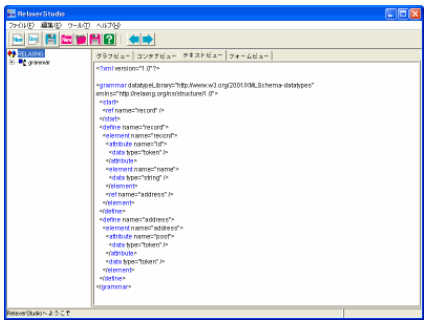

図 14: フォームビュー

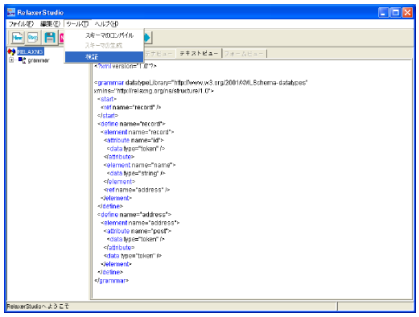

図 15: ツールの起動

ビューの 4 つのビューを通して編集を行うことができる。

8.4 ツールランチャ

RelaxerStudio は、RELAX NG スキーマを操作するためのツールを起動する機能を持っている。現在のバージョンでは図 15 に示すとおり、以下のツールを起動することができる。

- RELAX NG スキーマの検証
- RELAX NG スキーマの生成
- Relaxer の起動

8.4.1 RELAX NG スキーマの検証

RELAX NG バリデータ jing を起動し、RELAX NG スキーマの正当性を検証する。

8.4.2 RELAX NG スキーマの生成

Relaxer の機能を使い、任意の XML 文書から、XML 文書に適合する RELAX NG スキーマを生成する。

8.4.3 Relaxer の起動

「Relaxer の起動」は、RELAX NG スキーマおよび Relaxer CDL エディタから、Relaxer を経由して Java プログラムなどの成果物を生成するための機能である。

Relaxer は多くのオプションを持っているが、このオプションをダイアログから容易に指定することができるようになっている。

図 16 は、Relaxer の起動画面である。Relaxer がサポートしているオプションがダイアログから設定することができる。

9 Relaxer プロセス

本プロジェクトの目的の一つは、Relaxer を活用した作業手順のノウハウとしてオブジェクト指向開発プロセスである Relaxer プロセスの開発を行うことである。この目的で Relaxer プロセスを使用したサンプルアプリケーションである「図書館システム」を開発した。

9.1 Relaxer プロセスの流れ

図 17 が Relaxer プロセスの大まかな流れである。

一般的なオブジェクト指向開発プロセスとの相違点は以

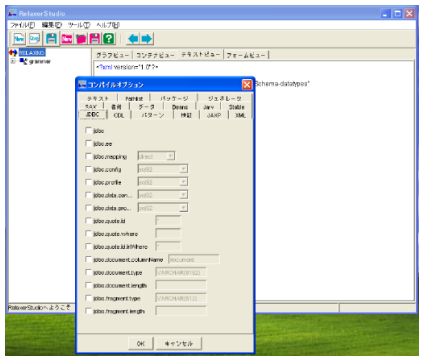

図 16: Relaxer 起動

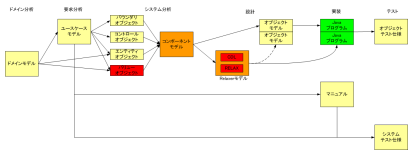

図 17: Relaxer プロセス

下の 3 点である。

- バリューオブジェクトの導入 (システム分析)
- コンポーネントモデルの作成 (システム分析)
- Relaxer モデルの作成 (設計)

9.2 システム分析

Relaxer プロセスにおけるシステム分析では、図 18 に示す分析オブジェクトを使用する。

- アクター
- バウンダリオブジェクト
- コントロールオブジェクト
- エンティティオブジェクト
- バリューオブジェクト

この中で、アクター、バウンダリオブジェクト、コントロールオブジェクト、エンティティオブジェクトは一般的なオブジェクト指向開発においてすでに利用されている分析オブジェクトである。

つまり、バリューオブジェクトが Relaxer プロセスで新規に導入された分析オブジェクトとなるわけである。

以上の分析オブジェクトを使用したクラス図の例が図 19 である。

9.3 設計

設計において Relaxer モデルを作成する点が、Relaxer プロセスの特徴である。

Relaxer モデルは RELAX スキーマおよび Relaxer CDL 定義から構成されており、Relaxer への直接の入力となる。

Relaxer が生成した Java プログラムは Relaxer による自動設計が行われていると考えることができる。

9.4 大きな流れ

システム分析から実装への流れは図 20 となる。

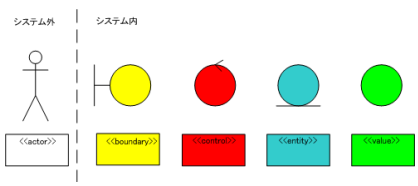

図 18: 分析オブジェクト

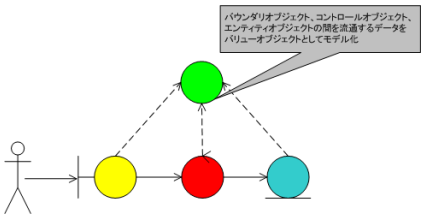

図 19: 分析オブジェクトの使用例

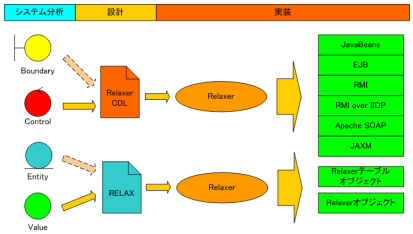

図 20: システム分析から実装への流れ

システム分析では、バウンダリオブジェクト、コントロールオブジェクト、エンティティオブジェクト、バリューオブジェクトの 4 つの分析オブジェクトが生成される。問題はこれらの分析オブジェクトをどのようにして Java プログラムに落とし込んでいくかである。

Relaxer プロセスにおける基本的な流れは以下の 2 つである。

- コントロールオブジェクトから Relaxer CDL 定義を作成する
- バリューオブジェクトから RELAX NG スキーマを作成する

RCDL 定義を作成することで、コントロールオブジェクトの実装として各種のコンポーネントを Relaxer が生成してくれる。

RELAX NG スキーマを作成することで、データを表現するための Java オブジェクトを Relaxer が生成してくれる。

9.5 図書館システム

Relaxer プロセスの実用性を検証するため、サンプルアプリケーションとして図書館システムを開発した。図書館システムの成果物は仕様書とプログラムから構成される。実際に動作するアプリケーションと、オブジェクト仕様開発プロセスの成果物としての仕様書である。

成果物の一部を図 21 に示す。この図は、仕様書から抜き出した 3 つの UML ダイアグラムを、工程の進捗方向に相関関係を明記したものである。

10 まとめ

RelaxerStudio プロジェクトでは、以下の開発を行った。

- Relaxer エンジンの改良 (RELAX NG 完全対応)
- テストプログラムの開発
- RelaxerStudio の開発
- Relaxer プロセスの開発
- サンプルアプリケーションの開発

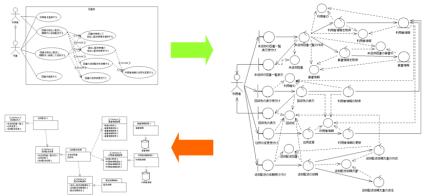

図 21: 図書館システムの成果物

また、以上の開発に加えて以下の英文ドキュメントを作成した。

- Relaxer チュートリアル
- Relaxer リファレンスマニュアル

RelaxerStudio により、一般の SE が実務に Relaxer を適用することが容易になった。

また、サンプルアプリケーションの開発によって Relaxer プロセスの効果を実証できた。Relaxer プロセスの具体的な方法論はまだ文書化されていないが、今回の成果をもとに今後文書化の作業を進めていく予定である。

加えて、今回の活動により英文ドキュメントを整備することができた。このことにより、海外での普及活動につながると期待できる。

11 参加企業および機関

本プロジェクトは参加者個人による活動であるが、以下の企業からプロジェクトの遂行上のサポートを得た。

株式会社クレオ　プロジェクト管理組織

参考文献

[1] Bill Shannon. Java 2 Platform Enterprise Edition Specification, v 1.3. Sun Microsystems, 2001.

[2] I. Jacobson, et al. The Unified Software Development Process. Addison Wesley, 1999.

[3] James Clark and MURATA Makoto. RELAX NG Tutorial : Committee Specification 3 December 2001. OASIS, 2000.

[4] James Clark and MURATA Makoto. RELAX NG Specification : Committee Specification 3 December 2001. OASIS, 2001.

[5] 浅海智晴. Relaxer : Java/XML による Web 開発. ピアソン・エデュケーション, 2001.