概念モデリングと連携可能なWebアプリケーションジェネレータ
A Web application generator cooperates with conceptual modeling

井田明男† 金田重郎‡
Akio IDA† Shigeo KANEDA‡

† 同志社大学
‡ 同志社大学大学院
† Doshisha University.
‡ Graduate School of Science and Engineering, Doshisha University.

要旨
インターネット利用の高度化に伴い,今日ではアプリケーションの主流は Web アプリケーションである.けれども,大学の教育現場では授業時間の制約から実習を伴う Web アプリケーション開発技術の指導が困難である．結果，学生は自らが作成したモデルに自信が持てない問題が発生している．そこで，概念レベル程度の少ない設計情報だけで Web アプリケーションを生成可能なコードジェネレータを開発した．本システムが自動生成するソースコードは可読性が高く，標準的なアーキテクチャに則ったものであるため，学習教材としても，使い捨てでないプロトタイプとしても使っていける．また，ジェネレータそのものは Ruby 言語で実装したため，非常にコンパクトであり，ユーザ自らの手で改良していける特徴を持っている．

1. はじめに

1.1. 背景

ソフトウェア産業に目を向ければ，今日ではアプリケーション開発の主流は Web アプリケーション（以下，Web アプリ）であることは，疑う余地がない．ところが，Web アプリの開発技術は一体いつどこで学べるのであろうか．

Web アプリの開発は手軽ではない．求められる要素技術についての知識の量が非常に多い上，それらは日々改訂されるため追いかけていくのが大変である．その上，Java を実装言語として選んだ場合，正しい作法でコーディングすれば相当なステップ数になってしまう．しかし，それを教えるとなるとさらに困難になる．筆者の一人は，社会人向けの 10 日間程度の技術研修でビジネス Web アプリ開発の指導を幾度が実施した経験があるが,扱うべき事柄が多いので常に時間がタイトであった．ましてや，大学の場合であれば 90 分毎に寸断されるコマの中での実施となるため授業の中で Web アプリ開発技術について実習を交えながらきっちり教えることは不可能に近い．

1.2. 問題

本稿では，Web アプリ主流の時代に，大学において，その開発技術教育が困難であることを問題として取り上げ，筆者らの取り組みを紹介する．筆者らは大学で UML を用いたオブジェクト指向分析・設計の講義を担当している．ビジネス系のシステムを題材とし，概念モデルの作成から，サーバサイド 3 層アーキテクチャにもとづいた設計モデルの作成までを指導している．

最初に直面した問題は，学生がまともなクラス図を書けないことであった．仕様文から適切なクラスを切り出せない，属性が誤ったクラスに持たされる，クラスの結びつけ方が適切でない，多重度に誤りがあるなどのモデルが多発した．そこで，筆者らは，認知言語学の知見を取り入れたモデリングの方法論を開発し[1]，授業の中に取り入れることにした．その結果，この問題に関してはかなり改善できる見通しが出てきた．

次に直面した問題は，モデルを作成しても，それが正しいのか，それをもとに実際に動くシステムへと繋げていけるのかが疑問で，学生が自分の作成したモデルについて自信を持っていないことである．この問題の特効薬は，学生が作成したモデルをもとに実装まで行い，実際の動作をその目で確認させることである．しかし，それは前述の通り時間がかかりすぎる．そこで，辿り着いたアイデアが，ソースコードジェネレータの導入である．Ruby の世界には Rails があり,その scaffolding 機能を用いれば,テーブルの定義情報を与えるだけで,そのテーブルを CRUD 操作[1]できる Web アプリケーションを瞬く間に取得できる[2]．scaffolding 機能は，使うのが簡単で，与えるべき設計情報も最小限で済むため，理想的ではあるが，生成されるコードが Ruby であることが問題であった．Java を

[1]行について新規作成（Create），読み出し（Read），変更（Update），削除（Delete）の操作の総称

学んでいる学生は多いが，Ruby を学んでいる学生は少ないからである．

他の製品も探したが，価格，製品仕様，動作プラットホームについての筆者らのニーズをすべて満たす製品は結局，見つけることができなかった．そこで，コードジェネレータを自作することにした．筆者らが求めるコードジェネレータの要件は次の通りである．

コードジェネレータに与えるべきパタメータは必要最小限の設計情報で済むこと．生成されたコードの手直しが容易であること．すなわち，MVC に基づくサーバサイド 3 層アーキテクチャに準拠した可読性の高いコードを整然と生成すること．ジェネレータおよびジェネレータが生成するコードは，Linux，Windows 双方の OS 上で動作すること．生成されたコードは，Tomcat，Java，Servlet/JSP，JDBC，MySQL 以外のコンテナ，ライブラリ，フレームワーク，およびミドルウェア製品を必要とせず動作すること．である．以下，第 2 章では本システムの紹介を行う．第 3 章では本システムの使い方を述べる．第 4 章では，本システムが生成する Web アプリの概要を紹介する．第 5 章では本システムの導入と活用について展望する．

2. 本システムの概要

コードジェネレータ自体は，Ruby 言語で開発することにした．Ruby 言語は強力な連想配列，ヒアドキュメントと文字列操作，およびフォルダやファイルに対する操作などを提供する[3]ため，少ない行数で本システムの機能を実装できると考えたからである．

2.1. コードジェネレータのコマンド構成

筆者らは約 1 ヶ月の開発期間の後，コードジェネレータの初版を作り上げた．現時点で本システムは次の 2 つのコマンドで構成される．

1)excel2mdb コマンド：Excel 表に記入されたデータベースの設計情報とサンプルデータを読み込み，MySQL データベースおよびテーブルを生成する．

2)genapp コマンド：MySQL サーバからデータベースの定義情報を取得し，そのデータベースに含まれるすべてのテーブルを CRUD 操作可能な Web アプリを自動生成する．データベースを核とするビジネス Web アプリの本質は，CRUD 操作であるため，生成された Web アプリは，そのままで完成に近い状態である．例えば，凝った EC サイトもその本質は，会員と商品とを結びつけた注文を CRUD 操作しているに過ぎない．

2.2. 少ない行数とカスタマイズの容易性

これらのコマンドは先の予想通り少ない行数で実装できた．実装の基本アイデアは，スクリプト内にヒアドキュメントの形式で生成すべきコードの雛型を記述しておき，入力された設計情報に応じて必要な箇所を文字列置換後，出力するものである．内包しているコードの雛型を除く実質的なステップ数はさらに少ない（表 1）．

当初は想像していなかったが，それぞれのスクリプトは単純なため，Ruby さえ知っていればユーザ自らがジェネレータを改良していくことも十分可能であることも分かった．スクリプトが内包するコードの雛型を修正することで，ジェネレータが吐き出すコードを容易にカスタマイズできる．

表 1:本システムのコマンド毎のステップ数

コマンド名称	コマンドから呼び出されるスクリプト名称	コマンドまたはスクリプトの機能概略	ステップ数	内包するコードの雛型を除外したステップ数
excel2mdb	excel2mdb.rb	エクセルで作成されたテーブルの定義情報からMySQLデータベースを生成する.	**289**	**289**
genapp		データベースの定義情報を取得してWebアプリケーションのソースコードを生成する.	**3991**	**2769**
	cleanup.rb	プロジェクトルートフォルダ（../）内のWebDbAppGen/を除くファイルとフォルダを削除する.	62	62
	getdbinfo.rb	コマンドライン引数で指定されたデータベースに含まれる全テーブルの定義情報を./dbinfo.ymlに出力する.	103	103
	gen_src_common.rb	../src/common/にBusinessException.java, ConvUtil.javaを生成する.	291	64
	gen_web_jsp_css.rb	../WebContent/jsp/css/にスタイルシートを生成する.	131	73
	gen_src_dao.rb	../src/dao/に<Tblname>DAO.java, DBManaget.java, DAOExcenption.javaを生成する.	511	326
	gen_src_domain.rb	../src/domain/にアクセッサを持つ<Tblname>.javaを生成する.	184	184
	gen_src_dto.rb	../src/dto/にアクセッサを持つ<Tblname>Dto.javaを生成する.	206	206
	gen_web_jsp_entry.rb	../WebContent/jsp/に<Tblname>Entry.jspを生成する. これは行への入力画面である.	233	215
	gen_web_toppage.rb	../WebContent/にindex.htmlを生成. これは管理者認証画面である.	167	72
	gen_web_jsp_list.rb	../WebContent/jsp/に<Tblname>List.jspを生成する. これは行の一覧画面である.	259	247
	gen_web_menujsp.rb	../WebContent/jsp/にmenu.jspを生成. これは管理者メニュー画面である.	193	169
	gen_web_jsp_modify.rb	../WebContent/jsp/に<Tblname>Modify.jspを生成する. これは行の変更画面である.	260	228
	gen_src_pre_cmds.rb	../src/presentation/にテーブルごとのCRUD操作に対応したコマンドクラスを生成する.	702	404
	gen_src_pre_ctrl.rb	../src/presentation/にフロントコントローラサーブレットを生成する.	218	146
	gen_src_service.rb	../src/service/にテーブルごとのCRUD操作に対応した手続きを生成する.	391	220
	gen_web_webxml.rb	../WebContent/WEB-INF/にweb.xmlを生成する.	80	50

3. 本システムの使い方

ここでは，excel2mdb および gennapp の使い方を「文献カード管理システム」の生成を例題として説明する．成果物はプロジェクトのためのフォルダ（例えば，sampleprj と命名，以降，プロジェクト・ルート）を作成し，そこに格納していくものとする．

3.1. ユースケースモデル

本システムが生成する Web アプリのデフォルトのユースケース図は図 1 の通りである．今回は，文献カード管理システムを作成するため，図中の「テーブル」を「文献カード」に読み替える．

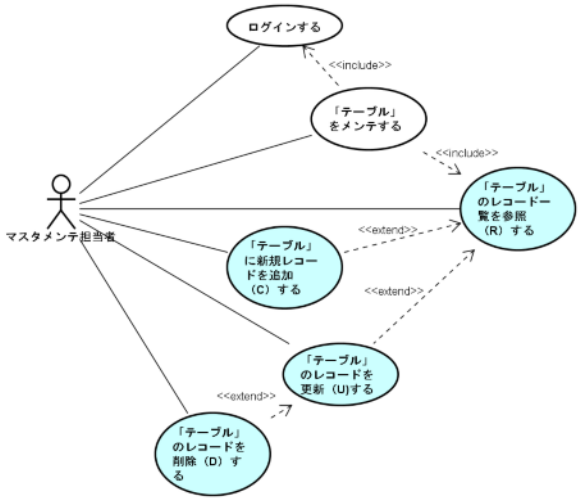

図 1:本システムが生成するアプリーションのユースケース図

3.2. ユースケースの実現

本システムが生成する Web アプリのデフォルトの画面遷移は図 2 の通りである．メニューから一覧，一覧から詳細へと流れる標準的なイベントフローを実装している．図中のログイン画面と，管理者メニュー画面以外の画面はテーブルの種類数だけバリエーションが自動生成される．例題ではテーブルが 1 個だけなので「テーブル」を「文献カード」と読み替える．ログインが無事完了すると，管理者メニューが表示される．メニューから CRUD 操作を行いたいテーブルを選択し，テーブルが選択されるとレコードの一覧を表示する．さらにレコードが選択されると，項目の入力ないし変更のための詳細画面に遷移する．詳細に関する処理が完了すると再びメニューに戻る．

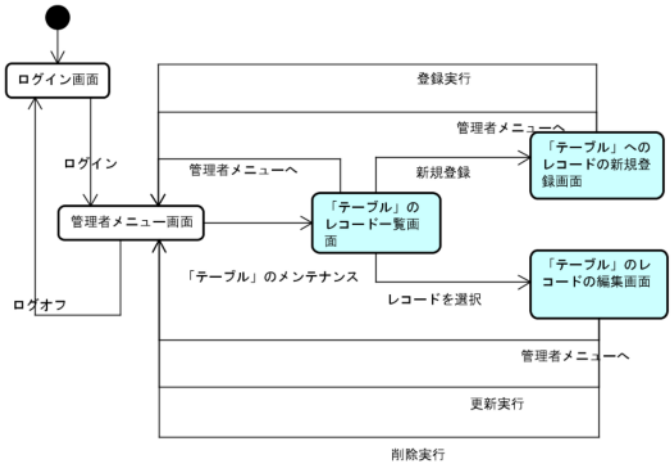

図 2: 本システムが生成するアプリーションの画面遷移図

3.3. エンティティのクラス図を作成する

今回は，文献カード管理システムを作成するため，エンティティはとりあえず「文献カード」のみである．そこで，文献カードとして求められる属性を考えて図 3 のようなエンティティのクラス図を作成した．

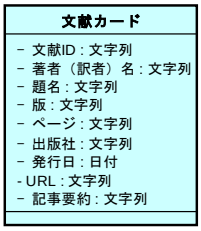

図 3: 文献カード管理システムのエンティティのクラス図

3.4. データベースのテーブル設計とデータベースの生成

本システムでは，永続化の手段として MySQL という RDB 製品を用いるため，クラス図で表現したモデルを RDB のスキーマに対応付ける（以降，O-R マッピング）必要がある．筆者らのモデリングメソッドでは仕様文を S+V+O の形式にリライト後，クラスおよびクラス間の関係を定義していく[1]ため，その成果物は第 3 正規形であることがある程度保証されている．しかし，クラス間の関係をエンティティ間のリレーションに翻訳するには，参照元の参照キー（外部キー）となる属性と参照先の識別子を明らかにする必要がある．渡辺は，２つのエンティティ間の関係は 1）親子関係，2）参照関係，および 3）派生関係のいずれかに分類できることを示し，それぞれはエンティティ間で共有する属性の位置づけによって区別されるとしている[4]．1）親子関係は，２つのエンティティがあって，一方のエンティティ（親）の識別子がもう一方のエンティティ（子）の識別子の一部になっている場合である．子のエンティティは親のエンティティなしでは単独で識別できないため，依存エンティティと呼ばれる．クラス図においてコンポジションで表記されるべき関係は親子関係に翻訳できる（図 4 の①）．2）参照関係は，参照先のエンティティの識別子が参照元のエンティティに含まれはする（ただし，クラス図では属性としてではなく，限定子または関連端名として表記される）が，参照元の識別子の一部にはなっていない場合である．クラス図におけるコンポジション以外の関連は参照関係に翻訳できる（図 4 の②）．そして，3）派生関係は２つのエンティティが同一の識別子を持っている場合である．クラス図における汎化関係は派生関係に翻訳できる（図 4 の③）．例題の場合は，クラスが 1 つのため文献 ID を識別子に指定することを決めれば，後は，おのおのの属性の型を検討するだけである．

さて，本システムでは，RDB のスキーマ定義を Excel で行うことができる．表 2 は文献カードクラスの定義をもとに属性の英文字名と型を考慮しテーブル定義（テーブル名称：literature）に落とし込んだ例である．生成したいデータベース名と同名の Excel ファイルを用意し，その中に，作成したいテーブルの数だけ，テーブル名と同名のシートを用意する．そして，各シートの 1 行目には属性名，2 行目には属性の型として SQL 言語の型，3 行目には属性の種類（主キーの場合は「PK」，外部キーの場合は「FK」，非キー属性の場合は「N」）を記述していく．なお，各シートの 4 行目以降はテーブルの初期値として格納したいサンプルデータを好きな行数だけ記述できる．これはデータベース設計時にカラムの型の妥当性確認にも役立つ．

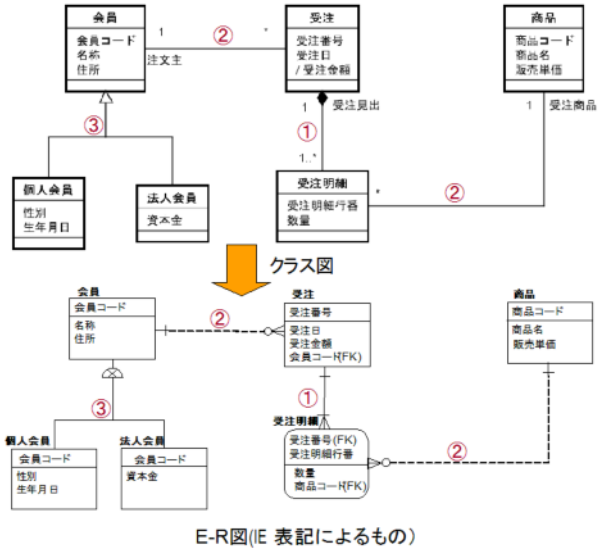

図 4:概念クラス図から E-R 図(IE 表記)への翻訳例

表 2: エンティティのクラス図からのテーブル設計例

literature_id	author_name	title	version	page	publisher	issue_date	url	capitulation
varchar(20)	varchar(500)	varchar(500)	varchar(50)	varchar(50)	varchar(500)	date	varchar(500)	varchar(4000)
pk	n	n	n	n	n	n	n	n
9784797347784	ロバート・C・マーチン（瀬谷 啓介）	アジャイルソフトウェア開発の奥義	第2版	683	ソフトバンク クリエイティブ	2008/7/8		アジャイル開発の基本原則について，特に設計原則についてはページを割いて説明している.
9784274066962	デイブ・トーマス，デビッド・H・ハンソン（前田 修吾）	RailsによるアジャイルWebアプリケーション開発	第2版	663	オーム社	2007/10/25		Webショピングカートアプリケーションをscaffoldで作成してから，肉付けしていくプロセスを辿りながら学ぶことができる.
9784774146638	山田 祥寛	Ruby on Rails3 アプリケーションプログラミング	初版	469	技術評論社	2011/6/10		ルーティングやヘルパーといったTipsを概要から詳細へという流れで解説している.

Excel ファイルの作成ができたならば，本システムの excel2mdb コマンドを実行する．コマンドライン引数にて，作成した Excel ファイル名，接続先のデータベースのユーザ名，当該ユーザのパスワードなどを指定する．すると，データベースとテーブルが自動的に生成される．

3.5. Web アプリケーションの生成

前節で作成したデータベースから Web アプリケーションを新規作成するには，本システムに含まれるコマンド genapp を実行する．その際，コマンドライン引数で，接続先のデータベース名，データベースのユーザ名，当該ユーザのパスワードなどを指定する．

genapp は MySQL サーバと通信し，指定されたデータベースに含まれるすべてのテーブル定義を取得して，各々のテーブルに対して，CRUD 操作を行うことができる Web アプリの Java ソースコード（Servlet/JSP を含む）を生成する．それらはプロジェクトのルートフォルダ内の src フォルダ配下に出力される．genapp はまた，JSP やスタイルシート，および web.xml などの設定ファイルも自動的に生成する．これらのファイルは，プロジェクトのルートフォルダ内の WebContent フォルダ配下に出力される．

3.6. 生成された Web アプリケーションの実行

生成された Web アプリケーションを実行するには，ビルドとデプロイが必要である．ant か Eclipse を用いてそれらを行う．Eclipse を用いる場合は，先程のプロジェクトのルート・ファルダをプロジェクト・コンテンツとした動的 Web プロジェクトを作成する．アプリケーションサーバに作成したプロジェクトをデプロイし，サーバを再起動する．そして，ブラウザから Web アプリの URL を指定する．具体的には，ie などのブラウザを立ち上げて，アドレスバーにプロジェクトの URL を入力すると Web アプリが起動され，ログイン画面が表示される．

4. 生成される Web アプリケーションの概要

本システムが生成するアプリケーションは，使い捨てでないプロトタイプである必要がある．すなわち，アプリケーション全体の理解が容易で，あとあとの手直しでも劣化しにくい構造である必要がある．また，教育用であるため，Struts や Hibernate などのフレームワークは一切使用しない．そのため，画面表示部分（presentation），ビジネスロジック部分（service），データ（domain）および永続化部分（DAO）をレイヤ分けし，パッケージにて明確に分離した．画面表示部分は，JSP，Servlet，永続化処理は，JDBC を用いることとし，使用するデータベース製品は MySQL を用いるが，他のデータベース製品への変更にも対応しやすい構造にしている．また，各層では，GOF のデザインパターン[5]および Core J2EE パターン[6]の中から Web アプリケーションに適したものを注意深く選んで適用している．例えば，プレゼンテーション層では，Front Controller パターン[6]と Command パターン[5]を採用している．Front Controller パターンは，JSP からの呼び出し毎に異なるサーブレットがリクエストを受け付けるのではなく，リクエストを集中的に受付けるサーブレットを導入する．それによって，リクエスト受付の共通的な処理を一元管理することができる．また，URL とサーブレットの対応関係を記述した web.xml の内容もシンプルにできる．Command パターンは，フロントコントローラが受け取ったリクエストの種類に応じてコマンドクラス側に処理を委譲する．これによって，フロントコントローラクラスの実装が簡素化できる．また，新たな機能を追加する際にもその機能に対応したコマンドクラスを追加するだけで対応できる．なお，プレゼンテーション層とビジ

ネス層の間では，Data Transfer Object(DTO)パターン[6]を採用し，2 つの層の間でのデータの授受を簡素化している．また，インテグレーション層では，Data Access Object(DAO)パターン[6]を採用している．データアクセス部分の実装を製品毎に変更しやすい構造とするため，仕様と実装を分離する．具体的には，各 DAO クラスのインタフェースを抽出し，特定のデータベース製品向けにそれらを実装した DAO を定義する．別の製品向けの DAO クラスが必要になった場合にも，同様に DAO のインタフェースを実現するクラスを定義し，それを実装するだけで対応できる．

5. 本システムの導入と活用についての展望

本システムを教育の現場に導入することにより，従来は時間的な制約で困難であった Web アプリケーション開発の実習が現実的な時間で実施できるようになる．エンティティのクラス図からデータベースの設計情報を記述すれば，以降は，使い捨てでないプロトタイプを得るまでに 1 分もかからないからである．残った時間を，アプリケーション固有の機能を実現するための差分プログラミングに充当することができる．学生はアプリケーションの動作とコードの両方を実際に確認しながらアプリケーションに手を加えて行けるため，その過程において高い学習効果が期待できる．今後，筆者らの概念モデリング・メソドロジ[1]と本システムとを組み合わせた授業を実現し，効果を測定したいと望んでいる．

さらに，筆者らは，本システムを実際のアジャイル開発においても，実装ノウハウの共有の仕組みとして活用できる，と考えている．Web アプリ開発という仕事に行為的に関わることでのみ獲得される現場の暗黙知を体系化し，独自の操作知へと昇華できる組織だけが競争優位性を保つことができる．そのためには野中・竹内の提唱する SECI プロセス[7]を回していくことが大切である．実際には，実装ノウハウが増えれば増えるほどそれらの共有や移転が難しくなる問題があるが，本システムは，そうしたノウハウを製品自体に蓄積していくことができるため，ユーザと共に成長できるコードジェネレータとして利用していける可能性がある（図 5）．

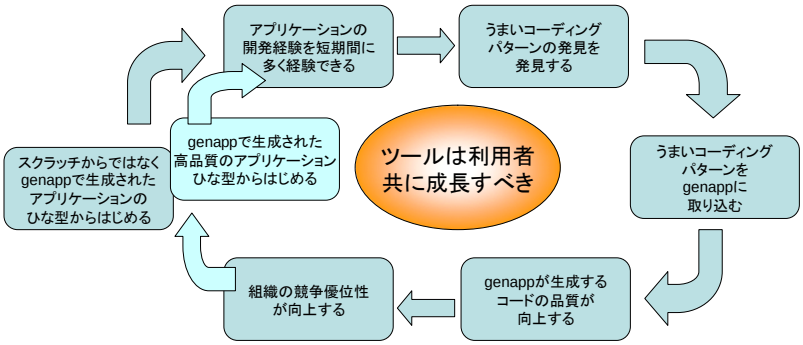

図 5: 本システムを実装ノウハウ共有の仕組みとして利用する

参考文献

[1] 金田重郎・世古龍郎：認知文法に基づくオブジェクト指向の理解，電子情報通信学会・知能ソフトウェア研究会（SIG-KBSE），2012 年 1 月
[2] Sam Ruby(著)，Dave Thomas(著)，David Heinemeier Hansson(著)，et al. 前田修吾(訳)：Rails によるアジャイル Web アプリケーション開発第 4 版，オーム社(2011).
[3] まつもとゆきひろ(著)，David Flanagan(著)，卜部昌平(監訳)，長尾高弘(訳)：プログラミング言語 Ruby，オライリージャパン(2009)
[4] 渡辺幸三（著）：業務システムのための上流工程入門，日本実業出版社(2003)
[5] Erich Gamma(著)，Richard Helm(著)，Ralph Johnson(著)，John Vlissides(著)，本位田真一(監訳)，吉田和樹(監訳)：オブジェクト指向における再利用のためのデザインパターン改訂版，ソフトバンク(2002)
[6] Deepak Alur(著)，John Crupi(著)，Dan Malks(著)，近棟稔（監訳)，吉田悦万(監訳)，　吉田悦万(監訳)：J2EE パターン　第２版，日経 BP(2005)
[7] 野中郁次郎(著)，竹中弘高(著)，梅本勝博(訳)：知識創造企業，東洋経済新報社(1996)