ハッセ図（工場モデル）に基づくクラス図作成手法の適用性
Applicability of Class Diagram with Hasse Diagram Structure

金田重郎 †
Shigeo Kaneda†
† 同志社大学大学院　理工学研究科
†Graduate School of Science and Engineering, Doshisha Univ.

要旨

概念データモデリングにおけるクラス図・ER 図（以下,「クラス図」）作成に関して，著者らは，初期材料（エンティティ）を次々と加工する「ハッセ図（工場モデル）」としてクラス図を描く手法を，既に提案している．この既存手法では,「関連」を 1 対多関連（正確には，少なくとも一方の多重度が「1」の関連）に限定することにより，任意のクラス図は，関連を半順序として，既約の非巡回性有向グラフ（ハッセ図）となる事を利用している．加えて，ハッセ図構造として記述されたクラス図から得られる関係モデルデータベースは，第 3 正規形（3rd Normal Form，以下,「3NF」）となる．逆に，外部キーを属性値として持つ属性を含めて，テーブルに 3NF を保証する関係モデルは，クラス図に変換すると，ハッセ図となる．しかし，従来研究では，このハッセ図のアプリケーションプログラムにおける適用範囲を論じていない．問題となるのは，クラス内の複数インスタンスから計算された導出値を格納する VIEW 的なエンティティの扱いである．そこで,「花束問題」の仕様を用いて，EADD 法の適用範囲について分析した．その結果，EADD 法の適用が可能なクラス（エンティティ）と，VIEW 的なクラス（エンティティ）では，クラス図上の表現方法が異なることが示された．

1. はじめに

著者らは既に，概念データモデリングを対象として，シンプルな構造「エンティティ・アクション従属ダイヤグラム（Entity Action Dependency Diagram，以下『EADD』)」に基づいてクラス図/ER 図（以下,「クラス図」で代表する）を作成する手法（以下,「**EADD 法**」と呼ぶ）を提示している[1]．

EADD 法では，原料となるエンティティを加工して，エンティティを出力側に生成するアクションを繰り返す「工場モデル（ハッセ図)」として，対象ドメインの活動をモデル化する．EADD 法を用いると，時間的前後関係を持つ基本アクションの積み重ねとして，見通し良くクラス図を描画できる．また，1 対多関連のみでモデルを記述し，時間的前後関係を用いているので，誰が書いても，似たようなクラス図となることが期待される．得られたクラス図を変換した関係モデルデータベースは，第 3 正規形（以下,「**3NF**」と略称する）を満たす．しかし，従来研究では，工場モデルのアプリケーションプログラムにおける処理に対する適用範囲を論じていない．

問題となるのは，入力側のエンティティ中の複数のインスタンスからの計算結果に基づいて出力される「VIEW 的なエンティティ」の扱いである．そこで,「花束問題」の仕様を用いて，EADD 法の適用範囲について分析した．その結果，EADD 法の適用が可能なクラス（エンティティ）と，VIEW 的なクラス（エンティティ）では，クラス図上の扱いが異なることが示された．

以下，第 2 章では，既存手法である EADD 法の概要を示す．第 3 章では，EADD 法の適用範囲を論じる．最後に，第 4 章で結論を述べる．

2.EADD 法の概要

2.1. 処理フロー

EADD 法の処理フローを図 3 に示す．EADD 法では，最初に，クラス図のスケルトンである EADD を描く．そして，EADD をそのまま写し取り，クラス図を描く．クラス図はそのまま関係モデルのテーブルに変換できる．以下に処理フロー概要を示す．

STEP1:エンティティの明確化： 分析対象ドメインからエンティティの候補（名詞）を抽出し，そのエンティティのインスタンスを追加するのに他エンティティのデータ状態は無関係となる**独立エンティティ(独立クラス)** と，他エンティティのインスタンスの存在が前提となる**従属エンティティ(従属クラス)** に区分する．

[1]本稿は，文献 [2] とほぼ同一内容である．EADD 法の詳細については，文献 [1] を参照されたい．尚，EADD の名称は，当初，RPDD(Resource Process Dependency Diagram) と呼称していた．

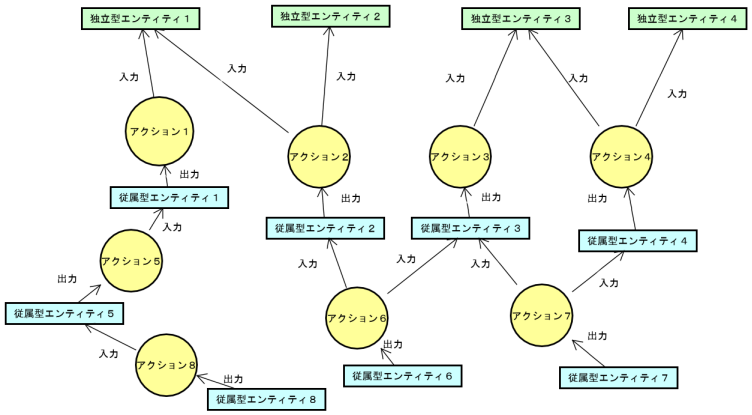

図 1: EADD の作成イメージ

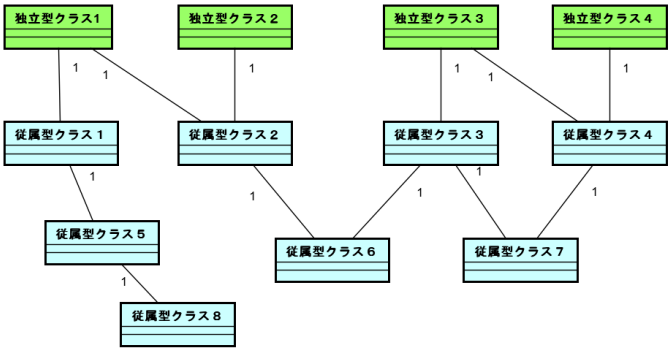

図 2: 生成されたクラス図イメージ（一対多関連を半順序とするハッセ図）

STEP2:EADD の作成： 対象ドメインの業務を図 1 に示す様な「工場モデル」で表現する．独立エンティティを最初の原材料（入力）として，それを加工するアクション[2]を定め，そのアクションの結果，従属エンティティ（出力）を作成するステップとして，EADD を描く．EADD 全体が，独立エンティティからスタートして，時間的推移のままに流れて行く様にモデル化する．作成するクラス図の「関連」は「1 対多関連[3]」のみが許されるが，図 1 において，従属エンティティから入力側エンティティに張られている矢印が 1 対多関連となる．「多重度=1」側のエンティティが時間的に先に存在する入力値であり，反対側が時間的に後に作られるエンティティである．1 対多関連を時間的前後関係を示す半順序と見なすことができ，EADD は非巡回性有効性グラフ（Directed Acyclic Graph，以下「DAG」) となる．

STEP3:EADD への規約性の付与： EADD において，抽出されたエンティティを A，C とする．AC 間に矢印（1 対多関連）が張られているとする．この時，分析対象ドメインの中に，「A から B へ，そして，B から C へ矢印が張られるべきであるエンティティB」が存在しないことを確認する．また，「A から B へ，B から C への矢印があるのに，A から C への直接の矢印がある」状態であれば，この A から C への矢印が冗長でないことを確認する．これらにより，EADD は既約な DAG（ハッセ図）となる．

STEP4:クラス図の作成： EADD の構造をそのまま写し取り，クラス図（図 2）を作成する．この時，各クラスは 3NF でなければならないので，クラスの識別子に，属性は非推移的に関数従属する様にデザインする．

STEP5:関係モデルへの変換： 図 4 に例示する様に，クラス図を関係モデルに変換する．各クラスはそのまま 1 対 1 対応にテーブルとなる．関連は，属性値として外部キーを格納する属性として表現

[2]ここで言う「アクション」は，あくまで，ビジネス実現のための行為であり，オブジェクト指向のメソッド（手続き）ではないことに注意．

[3]正確には，一方の多重度が「1」であれば良い．多対多関連は認めないからと言って，最終的に生成される関係モデルのテーブル数が増えることもない．

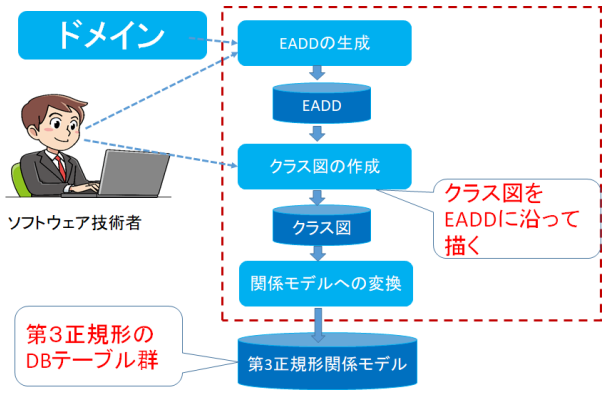

図 3: EADD 法の処理フロー概要

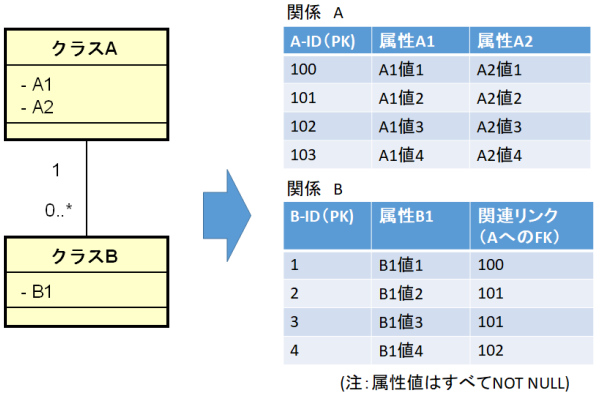

図 4: クラス図と変換後の関係モデル

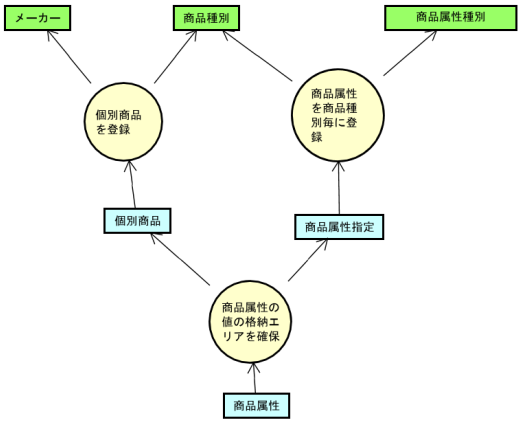

図 5: 動的属性の EADD

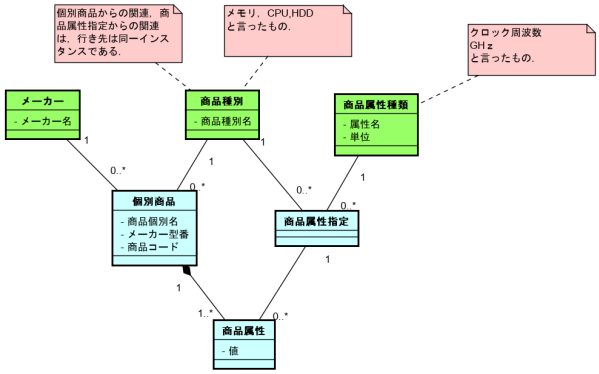

図 6: 動的属性のクラス図

する．この場合，外部キーの属性を含めて，NOT NULL とすると，生成されたテーブルは，外部キーの属性を含めて，3NF が保証される．

2.2.EADD の適用例

次に，EADD の適用例を示す．平澤章の著作中の「動的属性」を例にとる [3]．要求仕様は以下の通りである．尚，この仕様は，平澤の記述を意味的にくみ取ったものであり，平澤の仕様とは内容的にも一部異なっている．

- CPU，メモリ，HDD と言った「商品」を A ジャンク店が販売している．個別の商品には「メーカー」があり，個別の商品には，予め複数個指定されている「商品種別」から 1 つだけ選んで指定する．例えば，××電機の HDD-YY というハードディスクは，商品種別「HDD」に属している．
- 個別の商品を特徴づけるために，クロック数，メモリ容量といった「商品属性」があり，どの様な商品属性が付与されるかは，商品種別毎に決まっている．商品属性の名称と単位は，「商品属性種類」として登録されている多数の商品属性から商品種別毎に指定する．同じ商品属性が，異なる商品種別の商品に付加されることがある．
- 個別の商品には具体的な測定値が「商品属性」の属性値として個別の商品毎に記録されている．

まず，仕様からエンティティ（クラス）候補を取り出す．「商品種別」「商品」「メーカー」「商品属性」「商品属性種別」である．商品種別とメーカーは独立エンティティであることが自明である．微妙なのが，商品属性種類である．使われなくても登録をすることはあり得るので，独立エンティティに入れる．あとは，商店主になったつもりで，以下の様に，商売の準備を進める．結果として得られる EADD 図を図 5 に示す．

最初に，メーカーと商品種別を決めて，商品を登録する．この個別の商品に商品属性の値を記入する所を作りたい．しかしすぐには次には進めない．商品属性の値を格納する領域を個別の商品に貼り付け

る前に，そもそも，各商品種別に対して，商品属性として何があるかのかが分からない．商品属性を決めるには，先ほど，商品を作る時に特定したのと同じ商品種別インスタンスが必要である．

この状態では，商品種類毎にどのような商品属性があるのかが分からない．しかし，商品属性の名称と単位は，独立エンティティである商品属性種別として登録されている．よって，次に商品種類と商品属性種別を材料にして，これらを関係付けるために,「商品属性指定」と言う，関連をエンティティ化したものを考える．これは多対多関連を表現するためのものであり，商品種別と商品属性種別を関係付ける．以上から，指定された商品種類毎の商品属性種類は分かる様になっているので，各個別商品毎に商品属性の実際の規格値を格納するインスタンスを作ればよい．

図 6 は，EADD から作成されたクラス図である．作成されたクラス図は，平澤の書籍の掲載例と殆ど同じである．平澤は，ひとつの商品属性種別は唯一の商品種別にのみ利用されるとしている．例えば,「クロック数」は,「CPU」にのみ利用される商品属性であると言った制約である．これに対して，本稿では,「商品種類」と「商品属性種類」との間が多対多関連のため，図 6 の商品属性指定を用いて，多対多関連を表現している．本稿の EADD の視点を入れると，クラス図を作成するための方向性が読み取れる．あとは，関係モデルに変換すればよい．

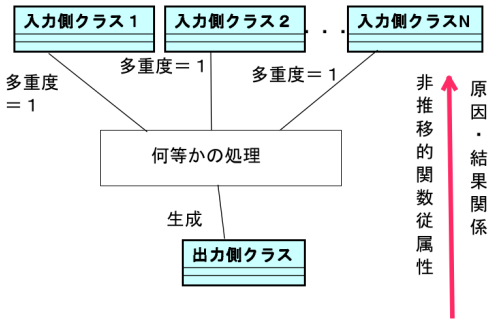

図 7: EADD 法の構成単位

3.EADD 法の適用範囲

3.1.EADD 法の構成要素

次に，EADD 法の適用範囲について考える．EADD では，その要素である構成単位として，図 7 の形で，処理が行われるものとモデル化している．この構成単位では，入力側エンティティと出力側エンティティの間には，1 対多関連のみが張られている．1 対多関連は，事実上，関数従属性と時間的前後関係（原因・結果，前提・帰結，材料・中間製品等の関係）を含意している．しかも，EADD は，既約になるように，1 対多関連は，非推移的な，基本的な関係に対してのみ張られている．これによって，生成される関係モデルのテーブルでは，外部キーを属性値とする関連を表す属性も含めて，3NF となることが保証される．逆に言えば，アプリケーションの処理内容が，図 7 が適用できなければ，作成されたクラス図の「1 対多関連」を引けるかどうか怪しくなってくる．では，図 7 では扱えないエンティティとはどの様な物だろうか．その為には，独立エンティティ，従属エンティティ以外のエンティティとして何があるかが問題となる．

椿正明は，アプリケーションを構成するエンティティを，リソース，イベント，断面，在庫，要約に分類している [4]．この区分で考察する．椿のリソースエンティティは EADD 法の独立エンティティに近く，イベントエンティティは，従属エンティティに近い．一方，断面，在庫，要約などは，あるクラス（1 クラスとは限らない）に所属する複数のインスタンスの属性値から何かの計算処理により導かれた属性値を持つエンティティであり，一種の VIEW，あるいは複雑な導出属性値を持つエンティティである．

これら VIEW 的エンティティの扱いについて,「花束問題」の事例を用いて分析する．

3.2. モデリングが難しいケース

「花束問題」は，NPO 法人「IT 勉強宴会 [5]」が公にしている要求条件であり，モデリングのベンチマークとして知られている．簡単に見えて，実際には実務で現れる扱い難い要素を含んでいると思われる．以下に著者なりの理解でリライトした，花束問題の要求仕様を示す．本来の仕様とは異なる部分が含まれていればご容赦願いたい．

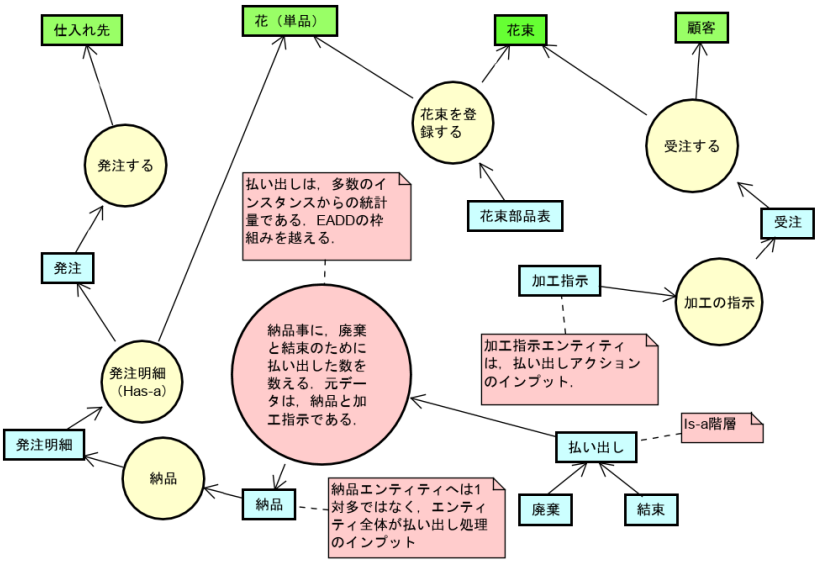

図 8: 花束問題の EADD（表現できない例）

- Web サイトで花束を販売している花屋がある．単品の花を仕入れ先から購入し，花束に結束して，顧客の指定したお届け先に送っている．花の単品の販売はやっておらず，お届も花束一つ毎に受注している．
- 単品の花は，花コードで区別される．花は専門の仕入れ先業者から購入しているが，発注から納品までは，花毎に決まっているリードタイム（日数）を必要とする．また，購入は 1 ロット何本の単位でのみ購入できる．仕入れ先は実績があり，納品が遅れることはない．
- 花は単品毎に，花束として利用してよい有効期間が設定されており，この有効期間が過ぎると廃棄する必要がある．
- 花束（商品）は花束コードで区別され，花束コード毎に，必要な花（単品）の花コードと必要数が予め決められている．
- 花束の作成は，すぐにできるので，出荷日に行うが，結束の指示書は出る．
- システムユーザはメアドを ID として，予めパスワードを登録している．
- 花の発注は自動的にはでず，花コード毎に，前々日残，前日残，当日残，当日利用数，当日廃棄数を日付毎に表示して確認できるインタフェースがあり，その表示で判断する．
- 花束の顧客からの受注分に関しては，納品日があとから修正されることがある．

以上から，クラス候補を名詞として取り出す．ただし,「発注」については,「発注」と「発注明細」を分けて考える．更に，発送指示は，配送日に花束の結束をするので，加工指示を併用する．抽出されるクラス候補は，以下の通りである.「システムユーザ」「(花の）仕入れ先」「花（単品)」「花束」「顧客」「年月日（日付)」「(花の仕入れ先への）発注」「発注明細」「(顧客からの）受注」「(花の）納品」「加工指示」．独立エンティティと従属エンティティの種別を表 1 に示す．

上記以外で検討が必要なのは，日々の在庫や廃棄数を表現するクラスの扱いと，花束の構成（どの花を何個使うか？）の表現方法である．花束と花の関係は多対多である．花束の構成を表すには，関連をクラス化した別のクラスが必要である．これを「花束部品表」と呼ぶことにする．

一方，花毎の日々の在庫・処理量・廃棄量などを表示するエンティティが必要である．仕様から見て分かるように，花の鮮度が保証されねばならない．このため,（花の）納品で古いものから順番に花を使う（加工指示による）必要がある．また，鮮度が保証されない花は廃棄する必要がある．日々の在庫・処理量・廃棄量などを考える前に，各納品事に,「廃棄」「結束」のイベントが存在するはずであり，これらを「払い出し」と呼ぶことにする．払い出しから，花毎の日々の在庫・処理量・廃棄量などを表示する必要があるが，その詳細は省略する．

図 8 は，作成した EADD の例である．最初に，花（単品）の仕入れ先へ発注する．発注は発注明細を持つ．発注エンティティは，少なくとも 1 つの発注明細エンティティを持ち，発注明細のひとつひとつには，花と本数が記載される．発注に対応して花の納品が続く．発注明細を材料として，何年何月何日に花が入荷するとの予定が作成できる．納品エンティティである．

表 1: 花束問題におけるエンティティ区分

区分	エンティティ名
独立エンティティ	「システムユーザ」「（花の）仕入れ先」「花（単品）」「花束」「顧客」「年月日（日付）」
従属エンティティ	「（花の仕入れ先への）発注」「発注明細」「（顧客からの）受注」「（花の）納品」「加工指示」

次に，顧客から受注する．受注の材料となるエンティティは，顧客エンティティと花束エンティティである．1 回の受注に 1 つの花束しか考えない．この際，花束がどの様な花（単品）から構成されるかが問題となる．この関係は，多対多関連であるので，図 8 に示すような，関連のクラス化で対応する．受注すると，それに応じて，加工指示が出される．加工指示で指定された日に，同時に出荷が行われるが，情報量が同一のため，今回のモデルでは省略した．

納品された花からは，花束の結束指示があるたびに，花が消費される．鮮度保証期間が過ぎれば廃棄する．このエンティティを「払い出し」エンティティとした．払い出しのサブクラスに，結束と廃棄があり，それに応じて納品エンティティ中の花の数は減少する．この場合，古い納品から順番に処理する必要がある．

この「払い出し」エンティティを出力するアクションを EADD に追加しても上手くゆかない．結論的には，払い出しエンティティから納品エンティティに 1 対多関連を引くことは可能である．しかし，この 1 対多関連は，EADD の 1 対多関連とは異なる性格を持つ．払い出しエンティティへとひかれた関連はアクションの原料を網羅していない．

払い出しエンティティを生成するには，1 対多関連で指定される入力側エンティティとして，ある特定の花のある特定の納品は指定されねばならない．しかし，このエンティティの詳細を決めるには，当該花種の過去の納品の履歴，当該花種の注文履歴，花束の部品表が必要であり，しかも，一歴については，生成されるべきインスタンスに依存して，個数も変化する．この様に，VIEW 的なエンティティの場合には，既存のクラス図の関連で，原料を指定することは難しい．この状況では，大半の意味的な関係をクラス図として描くことを放棄するしかない．結果として，どの納品インスタンスに対する払い出しエンティティかを指定するインデクシングのみを表現し，入力側エンティティを網羅することを放棄している.. つまり，クラス図の表現方法は，表現されるエンティティの種類に依存している．

4. 終わりに

著者らのクラス図を描くための方法論 EADD 法の適用範囲について分析した．EADD 法では，入力側エンティティから（クラス毎に）1 インスタンスを指定し，出力側エンティティも 1 個のみ出力する．そのことが適用範囲を規定している．この範囲では，1 対多関連のみで任意のクラス図を表現可能であり，結果的に，椿正明が主張するエンティティ5 種類中の,「リソース，イベント」が表現できる．

これに対して，椿が「断面，在庫，要約」と呼ぶ VIEW 的なエンティティについては，入力をクラス図の関連リンクとして表現することは難しい．単一のクラスから複数のエンティティが入力として選ばれ，しかも，個数が変化するからである．このため，出力側エンティティを管理するのに都合が良い，入力側エンティティの一部に関連を張って，モデルは表現するしかない．クラス図を描くモデラーは，エンティティの種別によって，クラス図を描く際の方法に違いがあることが示唆される．

参考文献

[1] 金田 重郎，井田 明男，ハッセ図構造を持つ実体関連モデルと正規形データベース，情報システム学会論文誌，Vol.16, No.1，2020 年 9 月）
[2] 金田重郎，ハッセ図（工場モデル）に基づくクラス図作成手法の適用性，電子情報通信学会・知能ソフトウェア研究会，2020 年 9 月
[3] 平澤章,「UML モデリングレッスン」，日経 BP，2008 年 1 月，228 ページ.
[4] 椿正明，名人椿正明が教えるデータモデリングの “技”，翔泳社，2005 年 11 月
[5] NPO 法人「IT 勉強宴会」https://www.benkyoenkai.org/