ソフトウェア工学 96－2
（1994．1．20）

オブジェクト指向分析における機能モデル検証支援

大西　淳

京都大学大型計算機センター
606-01 京都市左京区吉田本町

オブジェクト指向分析における3つのモデルの正当性とモデル間の整合性を検証する手法を提案する。手法の中で特に動的に機能モデルを検証する部分を紹介する。具体的には動的なシステムの振舞いのモデルに表された状態遷移図から一連の遷移を「動作シナリオ」として導き出し、このシナリオを解釈実行することによって、ビジュアルな要求言語VRDLによって表された機能を表すモデルのデータや装置の動作をアニメーションにより表示させる。これによって、利用者は機能モデルの正当性を検証できる。手法を支援するツールを試作中であり、手法とツールの動作について「タクシーの配車」という例を用いて説明する。

A Supporting Method for Verification of Functional Model of Object-Oriented Analysis

Atsushi OHNISHI

Data Processing Center, Kyoto University
Sakyo-ku, Kyoto 606-01, JAPAN
e-mail: ohnishi@kudpc.kyoto-u.ac.jp

A verification method of the completeness and the consistency of the three models of Object-Oriented Analysis is proposed. A dynamic verification method of the functional model is mainly presented. In the dynamic verification, first a scenario can be derived from both the dynamic model described with a state transition diagram and the functional model described with a visual requirements language VRDL. Then by interpreting the scenario, dynamic behaviors of the target system are represented as an animation. Through the animation users can verify the functional model. Both the method and a supporting tool are illustrated with an example.

1 はじめに

Rumbaugh 等の OMT[10], Coad/Yourdon の OOA[2], Shlaer/Mellor の手法 [11] などに代表されるオブジェクト指向分析では、対象システムを ① システムの要素となるオブジェクトの階層構造とオブジェクト間の関連を表すモデル、② システムの動的な振舞いを表すモデル、③ システムにおけるデータの変換過程や構造と言った機能を表すモデルの 3 モデルを導くことを目標としている。これらのモデルは対象システムを 3 つの異なる観点から眺めたものであり、それぞれのモデルに誤りがあってはならないし、さらにモデル間で矛盾があってはならない。

しかしながら、単にこれらのモデルを記述しただけでは、モデルの正当性やモデル間の整合性は保証されない。人手でこれらをチェックするのも大変な労力を伴うため、モデルの正当性やモデル間の整合性を計算機支援により高める手法の確立が望まれる。

本稿では、各モデルの正当性検証手法とモデル間の整合性検証手法について述べる。次に、動的な振舞いを表すモデルからデータや装置の移動や変化を動作シナリオとして導出し、このシナリオを元にして、機能を表すモデルに表されたデータや装置の動的な変化をアニメーションとして表示することによって、❶ 機能を表すモデルの正当性を動的に検証するとともに、❷ 動的な振舞いを表すモデルの正当性と ❸ 2 つのモデル間の整合性を利用者に確認してもらう手法とその支援ツールの動作を具体例を用いて説明する。

2 オブジェクト指向分析のモデルの記述

2.1 オブジェクトのモデルの記述

オブジェクトはクラスとインスタンスを総称したものとする。クラスオブジェクトは図 1のように OMT の記法に準拠したものとする。オブジェクトとオブジェクト間の関係が明記されて、オブジェクトモデル (Object model) が形成される。本稿では関係は ① クラス間の概念的な包括関係を表す is-a, ② オブジェクト間の構造的な集約関係を表す part-of, ③ オブジェクト間の参照・利用関係を表す関連に分類されるものとする [3]。

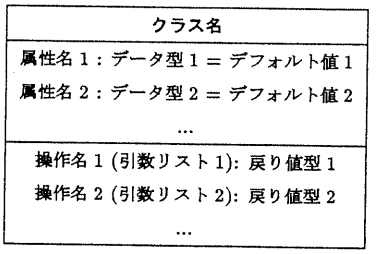

図 1: クラスオブジェクト図

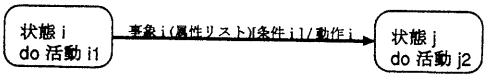

図 2: 状態遷移図

2.2 動的な振舞いのモデルの記述

動的な振舞いはある事象に伴う個々のオブジェクトの状態の遷移を図や表で表したものとする。状態遷移図の記法は図 2のように OMT の動的モデル (Dynamic model) の記法に準拠したものとする。図で事象 (event) はある時点で生じる出来事を意味しており、属性 (attribute) を持つ。属性は値 (attribute value) を持ち、属性値の組み合わせによって状態が決定する。条件 (condition) は状態遷移に当たっての条件を、動作 (action) は事象に関連して瞬時に起こる操作を、活動 (activity) は状態に関連して起こり完了までにある程度の時間を要する操作を意味する。

2.3 機能を表すモデルの記述

機能を表すモデル (Functional model) は、筆者が開発してきたビジュアルな要求言語 VRDL (Visual Requirements Description Language) [8, 9] によって、データフローが表されているものとする。 VRDL の特長を以下に示す。

1. アイコンの形状と意味を定義できる。
2. 複合的なアイコンを容易に作成できる。
3. アイコンと矢印をエディタ上で配置していくことによって要求を記述する。
4. VRDL 記述に用いられたアイコンの動作を与えることができ、その動作記述を解釈実行することによっ

て、記述の中のアイコンの動作をアニメーションとして表示できる。

5. VRDLによる記述を標準的なアイコンを用いた記述へ変換できる。
6. VRDL記述に対して、要求定義環境 CARD [7] のツール群による検証・検索・フロー表示・設計支援ができる。

4番目の特長を利用して、VRDLによって表現された機能モデルの正当性を検証する。

VRDLのデータフローでは、流れるデータ、データの源泉、データの目標、用いられる装置(オプション)のオブジェクトが明示される。このうち、流れるデータと装置が動きの対象となる。アイコンのデフォルトの動作は源泉格に対応するオブジェクトのアイコンから目標格に対応するオブジェクトのアイコンに向かっての、データを表すアイコンの移動であり、デフォルトの動作は利用者は定義しなくてよい。

デフォルト以外のアイコンの動作を定義するために、動作記述言語が利用できる。この言語は以下の項目を記述できる。

- 動かすアイコンの指定
- アイコンを表示する位置の指定
- アイコンの表示開始
- アイコンの表示終了
- アイコン表示時間の設定

アイコンの動作は、

① アイコンを表示させる位置を少しずつずらしていく

② 形状の異なる複数アイコンを切替えながら表示する

ことによって実現している。例えば「車が移動する」という動作を、指定した座標位置から車を表すアイコンを座標を少しずつずらしながら表示したり消去したりすることを繰り返すことによって実現する。また、「電話が鳴る」という動作を、受話器が少し持ち上がったアイコン☎と平常の電話のアイコン☎とを切替えて表示することによって実現する。さらに、アイコンの移動と異なる複数のアイコンの表示の切替を組み合わせることによって、複雑な動作を実現することができる。

3 モデルの正当性検証手法

3.1 オブジェクトモデルの正当性検証

① 属性のないオブジェクトクラスはないか、② 他のオブジェクトと関係のないオブジェクトはないかを調べる。

3.2 動的モデルの正当性検証

動的モデルのオブジェクトの状態はオブジェクトの属性の内の特定の属性がとる値の組み合わせによって決定するものとしている。従って状態を決定する属性の値が変わることによって状態が遷移する。動的モデルの正当性検証では① 各状態とそれを決定する属性の値が明示されているか、② 各状態遷移での事象に伴う動作によって変化する属性の値が遷移前の状態と遷移後の状態とで矛盾していないかを調べる。

3.3 機能モデルの正当性検証

入力のない機能や出力のない機能がないか調べるといった静的な正当性検証に加えて、次のような動的な正当性検証を行なう。データの流れや装置の動きの動作を動作記述言語によるシナリオとして記述し、シナリオを実行することによって、データの流れや装置の動きを逐次的にアニメーションとして表示する。これによりデータの流れる方向とデータ処理の順序が正しいかどうかを確認できる。アニメーション表示の手順は以下の通り。

1. 動的モデルから、特定の状態遷移を一つずつ利用者に順々に選択させて、一連の遷移の流れを導く。選択されたあるオブジェクトの状態遷移に伴う動作が別のオブジェクトの状態遷移も引き起こす場合は、別のオブジェクトでの状態遷移は自動的に選択されたものとみなされる。得られる一連の遷移の流れを「動作シナリオ」としてまとめる。
2. データフローの中に現れるアイコンでデフォルト以外の動きの定義が必要なものについて利用者にその動きを「アイコン動作定義」としてまとめさせる。特に必要がない限り、この段階は省略してよい。
3. 動作シナリオとアイコン動作定義をもとに一連の遷移の流れの中でのデータの動きと装置の動きをアニ

メーションとして表示する。

OMT ではシナリオや事象トレースを考えて、それから状態遷移を導く [10] ので、最初に考えたシナリオをそのまま用いる方法も考えられるが、このシナリオは動的モデルを導くために一例をとりあげたものに過ぎず、すべての状態遷移に対応しているとは限らない。ここでは状態遷移表現の任意の部分集合の正当性を検証することを考慮して、新たにシナリオを導出する。

タクシー会社は客から電話で配車依頼を受け、空車を検索する。見つかった空車に電話で客の現在位置と目的地からなる配車指令を出す。指令を受けた空車は予約車となり、客の現在位置という目的地に向かう。客が乗車すると賃走となり目的地へ向かう。また、空車に客が直接乗車しても賃走となり目的地に向かう。

図 3: タクシーの配車記述

4 モデル間の整合性検証手法

4.1 オブジェクトモデルと動的モデルの整合性検証

① オブジェクトモデルでのオブジェクトの持つ属性と動的モデルで状態を決める属性の間に矛盾がないか、② オブジェクトの持つ操作と状態の持つ活動と状態遷移に伴う動作の間に矛盾がないかを調べる。

4.2 オブジェクトモデルと機能モデルの整合性検証

① オブジェクトモデルにおけるオブジェクトと機能モデルのアイコンが対応するか、② オブジェクト間の関連とアイコン間のデータフローに矛盾がないか、③ オブジェクトの持つ操作とデータフローという動作の間に矛盾がないかを調べる。

4.3 動的モデルと機能モデルの整合性検証

動的モデルにおける状態遷移に伴う入出力動作で、その入出力データと機能モデルにおけるデータフローに矛盾がないかどうかを調べる。これは機能モデルの正当性検証で述べたシナリオ作成段階で調べられる。

5 例

5.1 タクシーの配車問題

例としてタクシーの配車を取り上げる。図 3に概要の記述を示す。

5.2 オブジェクトの抽出

タクシー、タクシー会社、客をオブジェクトとする。客からタクシー会社には配車依頼という関連があり、タクシー会社からタクシーには配車指令という関連がある。タクシーの属性としては目的地 (値: あり・なし)、乗客 (値: あり・なし)、型 (値: 大型・中型・小型) などが、メソッドとしては「客の現在位置に向かう」、「客の目的地へ向かう」、「客をさがす」、「客を乗せる」、「客を降ろす」などが考えられる。タクシー会社の属性として配車依頼 (値: あり・なし) が、メソッドとしては「空車を検索する」、「配車依頼を受ける」、「配車指令を出す」が考えられる。客の属性としては乗車中 (値:Y/N)、配車依頼 (値: 未・済) が、メソッドとしては「配車を依頼する」、「タクシーに乗車する」、「タクシーから降車する」などが考えられる。

これらに基づいてオブジェクトモデルが作成される。メソッドや属性を持たないオブジェクトがないか、他と関連のないオブジェクトがないか検証される。

5.3 問題の状態遷移

この問題からタクシーオブジェクトの状態遷移を図 4に、タクシー会社オブジェクトの状態遷移を図 5に示す。タクシーの属性値によって状態は表 1のように決まる。また遷移によって、属性値がどう変わるかも表 2のように明確にする。

属性値と状態の対応がとれているか、個々の状態遷移で矛盾がないかどうか検証される。

5.4 問題のデーターフロー

一方、この問題を VRDL で記述したものを図 6に示す。図で画面の左にはあらかじめ定義したアイコンが示される。これらのアイコンと矢印を指定して、マウスでクリックした位置に張り付けながら記述を進めていく。

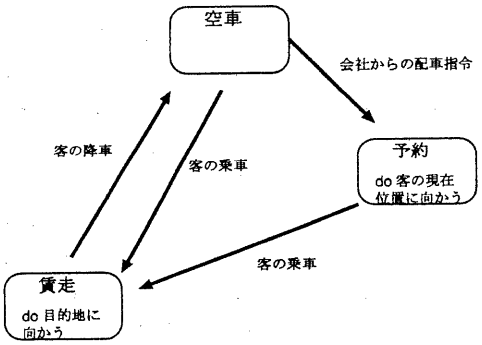

図 4: タクシーの状態遷移

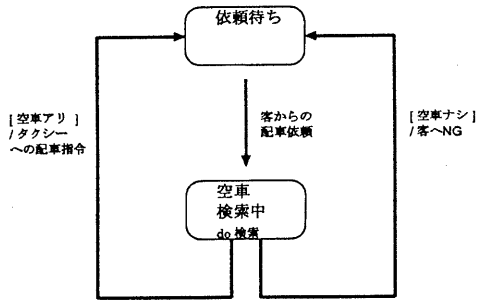

図 5: タクシー会社の状態遷移

表 1: タクシーの状態と属性値

状態	属性: 目的地	属性: 乗客
空車	なし	なし
予約	あり	なし
賃走	あり	あり

表 2: タクシーの状態遷移と属性値の変化

事象	属性: 目的地	属性: 乗客
客の乗車	なし→あり	なし→あり
客の降車	あり→なし	あり→なし
配車指令	なし→あり	なし

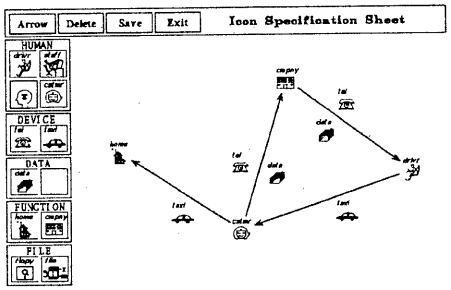

図 6: VRDL によるタクシーの配車のデータフロー

表 3: タクシーの配車でのアイコン定義

アイコンの形状	意味	
	名称	名詞の型
	運転手	人間
	客	人間
	配車依頼	データ
	タクシー	装置
	電話	装置
	タクシー会社	(機能)
	目的地	(機能)

この図で用いられているアイコンの定義を表3に示す。なお、アイコンの意味は要求フレームモデル [5] に基づいて定義されている。

この段階では、入出力データをもたない機能があるかどうかがチェックされる。

5.5 動作シナリオの導出

次に状態遷移表現から状態遷移の順序を定めたシナリオを導く。最初に各オブジェクトの属性の初期値を決めさせる。これにより各オブジェクトの初期状態が決定する。例えば、タクシーは空車状態、タクシー会社は依頼待ち状態とする。あるオブジェクトの初期状態と別のオブジェクトの初期状態が矛盾する場合はシナリオの実行時に検証される。

図 7で実線矢印はシナリオで採用された状態遷移を示し、破線矢印は採用されなかった状態遷移を示す。また

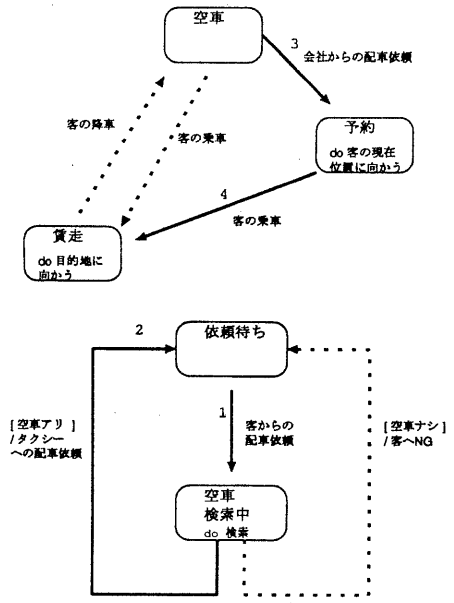

図 7: シナリオでの状態遷移

矢印に付けられた番号はシナリオでの実行順序を表す。タクシー会社の2番めの状態遷移「タクシーへの配車指令」を選択すると、自動的にタクシーの状態遷移「会社からの配車指令」が選択される。状態の遷移で条件分岐があり一意に決まらないものやループがあって実行順序が決められないものは利用者に問い合わせて決める。このシナリオでは、

1. 客からの配車依頼があり、
2. タクシー会社は空車を検索し、タクシーへ配車指令を出す。
3. タクシーはタクシー会社からの配車指令によって、客の現在位置に向かう。
4. 客が乗車すると目的地に向かう。

ことが示されている。

このシナリオで動作するアイコンはデフォルトの解釈では道具格の「電話」とデータフローの流れる対象となるデータである「配車依頼」と「タクシー」である。「電話」の動きは2.3節で示したように、2つのアイコンの切り替え表示で実現し、データの流れは源泉格から目標格へのデータを表すアイコンの移動で実現する。

シナリオでの最初のデータフロー文である「客からタクシー会社へ電話で配車依頼が送られる」という部分のアイコンの表示と動作については、動作記述言語で書かれた図8のような「動作シナリオ」が生成される。図でlocate文はアイコンの表示位置を設定するものであり、display文はアイコンを表示したり、表示を停止する文である。この動作記述言語により、デフォルト解釈以外の動作をアニメーションとして描くことができる。

5.6 アニメーション表示

「動作シナリオ」に基づいて、1つのデータの流れごとにデータの動きがアニメーションとして表示される。図9～12はアニメーションのスナップショットを示したものである。図9は初期状態を示しており、動かないアイコンが表示される。図10は1番目のデータフローのアニメーションの開始を表しており、源泉格である「客」と目標格である「会社」のアイコンの間にデータフローを示す矢印が表示される。また動く対象の「配車依頼」を表すアイコンが「客」アイコンの側に表示される。この例では電話が使われていることを示すために、「客」と「会社」の両方のアイコンの近くに「電話」アイコンが表示される。このアイコンは切り替え表示によって電話中であることを表す。図11は「配車依頼」が途中まで送られた様子を表している。図12は「配車依頼」がすべて送られた状態を表している。このようなスナップショットが10枚連続的に表示され、客から電話で配車依頼が会社に渡る様子がアニメーション化される。このように、「動作シナリオ」に基づいて、矢印で示される各々のデータフロー単位でのデータの動きがアニメーションとして表示されていく。なお、動作記述言語によって、アニメーションの動作速度を変更できる。

6 終りに

オブジェクト指向分析におけるモデルの正当性とモデル間の整合性検証のための手法を紹介し、特に機能モデルと動的モデルの2つからデータの流れをアニメーションとして表示させ、動的に機能モデルの正当性を検証する手法を提案した。現在のところVRDLの処理系と「動

```
tel_anime(ACT src, ACT goal, ACT data)
{
    ACT tel1, tel2;
    int i;
    ICON icon1, icon2;
    LINE line;

    icon1 = ''telep1'';
    icon2 = ''telep2'';
    tel1 = {icon1, icon2};
    tel2 = {icon1, icon2};
    line = LINESTYLE1 ;

    locate(tel1, 50, 50);
    locate(tel2, 450, 100);
    actfor(i=0;i<11;++i){
        locate(data, 50 + 40*i, 50 + 5*i);
        if(i-i/2*2 == 0){
            display(tel1[0], ON);
            display(tel2[1], ON);
        }
        else {
             display(tel1[1], ON);
             display(tel2[0], ON);
        }
        display(line, ON);
        display(data, ON);
        display(src,  ON);
        display(goal, ON);
    }
    display(tel1, OFF);
    display(tel2, OFF);
    display(line, OFF);
}
```

図 8: 動作シナリオ例

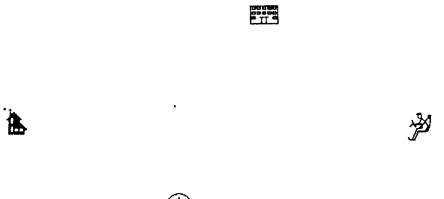

図 9: アニメーションのスナップショット (1)

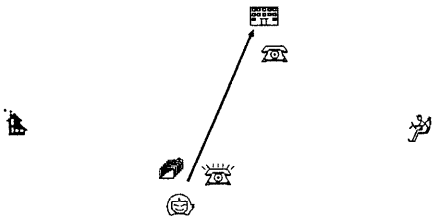

図 10: アニメーションのスナップショット (2)

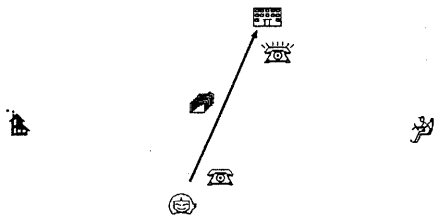

図 11: アニメーションのスナップショット (3)

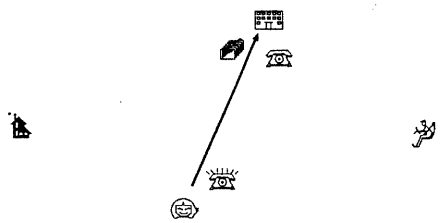

図 12: アニメーションのスナップショット (4)

作シナリオ」の解釈実行系それぞれのプロトタイプを開発済みであり、これらに加えて状態遷移図からの「動作シナリオ」導出部分のプロトタイプの開発と評価を進めている段階である。

今後の課題としては、

1. オブジェクト指向分析のもう一つのモデルであるオブジェクトのモデルを記述できるようにする。

2. ビジュアルな要求言語VRDLはファイル処理システムのデータフローを記述するために開発されたため、オブジェクト指向分析で扱える広い分野の記述には向かない点があり、拡張・改善する必要がある。

3. モデル表現の誤りを発見した場合の対処法を確立する。

があげられる。課題の1番目については筆者が開発してきた日本語要求言語[5]による要求記述からオブジェクトのモデルを作成できるように研究を進めていきたい。その際には、要求フレームモデル[5]を再検討しなければならないが、要求フレームモデルの拡張により課題の2も解決できると思われる。3番目の課題については、一つのモデルを修正・変更した場合、他のモデルへの変更の波及を検出する仕組みを研究したいと考えている。

謝辞 シナリオ導出系の開発に当たり、本学大学院工学研究科応用システム科学専攻修士課程2回生の服部芳明君に感謝する。

参考文献

[1] Chang, S.: "A Visual Language Compiler," IEEE Trans. Softw. Engnr. Vol.15, No.5, 1989, pp.506-525.

[2] Coad, P., Yourdon, E.: Object-Oriented Analysis, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ, 1991.

[3] 本位田真一、山城明宏: 「オブジェクト指向システム開発」、日経BP社, 1993.

[4] 情報処理学会: 「オブジェクト指向分析・設計チュートリアル資料」、情報処理学会, 1993.

[5] 大西 淳、阿草清滋、大野 豊: 「要求フレームに基づいた要求仕様化技法」、情報処理学会論文誌31巻2号, 1990 (pp.175-181).

[6] 大西 淳: 「要求定義のためのコミュニケーションモデル」、情報処理学会論文誌33巻8号, 1992 (pp.1064-1071).

[7] Ohnishi, A., Agusa, K.: "CARD: A Software Requirements Definition Environment," Proc. of IEEE Int'l Symp. Requirements Engineering, San Diego, CA, U.S.A., Jan. 1993, pp.90-93.

[8] 大西 淳: 「ビジュアルなソフトウェア要求仕様化技法」情報処理学会研究報告, Vol.93, No.13, SE90-8, 1993年2月, pp.57-64.

[9] Ohnishi, A.: "A Visual Software Requirements Definition Method," Proc. of IEEE Int'l Conf. Requirements Engineering (ICRE'94), Colorado Springs, CO, U.S.A., Apr. 1994, (to appear).

[10] Rumbaugh, J., Blaha, M., Premerlani W., Eddy F., Lorensen W.: Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991. (羽生田栄一監訳: オブジェクト指向方法論OMT、トッパン、1992.)

[11] Shlaer, S., Mellor, S.: Object-Oriented Systems Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1988, (本位田真一、山口亨訳: オブジェクト指向システム分析、啓学出版、1990.)

[12] Shu, N.: "Visual Programming," Van Nostrand Reinhold Co. New York, 1988.

[13] St-Denis, R.: "Specification by example using graphical animation and a production system," Proc. IEEE 23rd HICSS, Vol.2, 1990, pp.237-246.

[14] Thayer, R., Dorfman, M.: "System and Software Requiremenst Engineering," IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.

[15] Wang, W., Hufnagel S., Hsia, P., Yang S.M.: "Scenario Driven Requirements Analysis Method," Proc. IEEE Int'l Conf. Systems Integration(ICSI), 1992, pp.127-136.