出力データに着目したゴール指向分析による機能要求抽出方法の提案 -酒屋倉庫問題のケース-

岡野 道太郎[1,a)]　中谷 多哉子[†1,b)]

概要：システムの要求獲得を行う場合，設計・実装する上で必要な情報を出来る限り抽出することが望ましい．しかし，要求獲得にゴール指向要求分析を用いる場合，必要な要求を十分に引き出せず，設計・実装上で必要となる要求が欠如する場合がある．そこで本稿では，この問題を解決するために，ゴール指向分析を適用して機能要求を獲得する際に，出力データに着目して，設計・実装上で必要となる要求が欠如していないか判別する方法を提案する．提案する方法の有効性を検証するために「酒屋倉庫問題」に適用した．その結果，問題には示されていなかった，設計・実装上で必要となる要求が欠如している部分を指摘できた．

キーワード：要求獲得，ゴール指向要求分析，KAOS，ゴール分解，酒屋倉庫問題

A proposal of functional requirements elicitation method by goal-oriented analysis focused on output data : The case of liquor store problem

Abstract: When we elicit system requirements, we have to define requirements as much as possible including requirements for design or implementation. For example, a goal-oriented analysis method sometimes does not work well on defining requirements for design and/or implementation unambiguously. In order to solve the problems, we developed a functional requirements elicitation method by focusing on output data of each functional requirement. The method is an extended KAOS by a goal decomposition process with three stages base on the goal categories proposed by Alex von Lamsweerde. In this paper, after we introduce the method, we will show the result of evaluation of the method. The method is evaluated by applying to the common problem named "Liquor Shop Problem." In the evaluation, we could find undefined requirements for design and implementation unambiguously.

Keywords: requirements elicitation, goal-oriented requirements analysis, KAOS, goal decomposition, liquor store problem

1. はじめに

システムの要求分析手法の 1 つにゴール指向分析があり，ゴール指向要求分析手法の 1 つとして，KAOS[1] が挙げられる．KAOS では，システムの目標を最上位のゴールとして記す．この最上位のゴールはトップゴールと呼ばれる．そしてトップゴールを AND 分解または OR 分解によって分解し，下位ゴールを導く．次に下位ゴールを上位ゴールとして AND 分解または OR 分解を行う．これを繰り返し，木状のゴールモデルを作成する．

ここで AND 分解は分解されたサブゴールのすべてが達成されないとゴールが達成されない分解方法であり，OR 分解とはサブゴールのうち 1 つでも達成されればゴールが達成される分解方法と定義する．この定義だけでは，AND

[1] 筑波大学大学院ビジネス科学研究科
University of Tsukuba
[†1] 現在，放送大学
Presently with Johoshori Uniersity
[a)] okano@gssm.otsuka.tsukuba.ac.jp
[b)] tinakatani@ouj.ac.jp

分解や OR 分解をどのように行うかの明確な指針がないため，要求分析者がどのようにゴールを分解していけばよいのかわからず，設計・実装する上で必要な情報を十分抽出できないことがある．例えば筆者らが AND 分解，OR 分解の定義のみを説明した後，大学生等がゴール分解を行った実験 [2] でも，一部のゴールでは分解が進んでも，他のゴールの分解が行われないといった現象が見られた．その場合，設計・実装する上で必要な要求を十分に抽出することが出来ない．

我々は KAOS を用いてゴール分解を行う際に，要求抽出を十分に行えるように，ゴール分解の指針を開発している．[2] では「2．先行研究」で述べる Lamsweerde のゴールカテゴリに基づいたゴール分解の指針を提案した．この方法で分解は行える．しかし，設計・実装上で必要な情報が欠如していないかどうか判断できない．本稿では，[2] で提案した方法を用いて要求抽出を行った後，出力データに着目して，設計・実装上で必要な情報が欠如していないかどうか判断できる方法を提案する．特に，機能要求に着目して行う場合を提案する．また，その方法の有効性を評価するために，「酒屋倉庫問題」[3][4] に適用し，機能要求の獲得実験を行った．その結果，設計・実装上で必要な要求が欠如している部分を指摘できた．

本稿は以下の構成となっている．次の 2 章では関連研究を挙げ,3 章で提案する方法を説明する．4 章で酒屋倉庫問題にその提案を適用し,5 章で適用結果を考察し，6 章でまとめる．

2. 先行研究

ゴール指向分析は様々な手法が提案されている．特に i*[5]，KAOS[1] などの適用や研究が進んでいる．しかし先に述べたとおり，ゴール分解の詳細な分解手順には言及されていない．

Lamsweerde は，[1] の中で，AND-Refinement，OR-Refinement について言及している．しかしそこで述べられているのは，ゴールをサブゴールに分割できた時，ゴールとサブゴール間で満たすべき条件についてであり，何に着目してサブゴールに分割するのかについて述べてはいない．我々は，ゴールをサブゴールへ分割する際，何に着目して分割するかを [2] で，どこまで分割するかを本稿で提案している．

また，Lamsweerde は，[1] の中で，ゴールを分類し，ゴールカテゴリを定義している．それによると，ゴールは機能ゴールと非機能ゴールに分類され，さらに機能ゴールは以下の 3 種類のゴールに分類されている．

- 満足ゴール：エージェントの要求を満足させることに関する機能ゴール
- 刺激-反応ゴール：特定のイベントに対する適切な反応に関する機能ゴール
- 情報ゴール：重要なシステムの状態に関してエージェントに知らせることに関する機能ゴール

しかし，ゴール分解に際して，これらのゴールカテゴリをどのように考慮するのかといった明確な指針は示されていない．本稿はこの点も考慮している．

3. 提案する手法

3.1 本提案の概要

本提案は，大きく 3 段階に分けて分解を行う．

初めに「満足ゴール」に対して行う分解があり, 次に「刺激-反応ゴール」に対して行う分解 , さらに「情報ゴール」に対して行う分解がある．

分解法を提案する際に利用する用語を図 1 にまとめた．

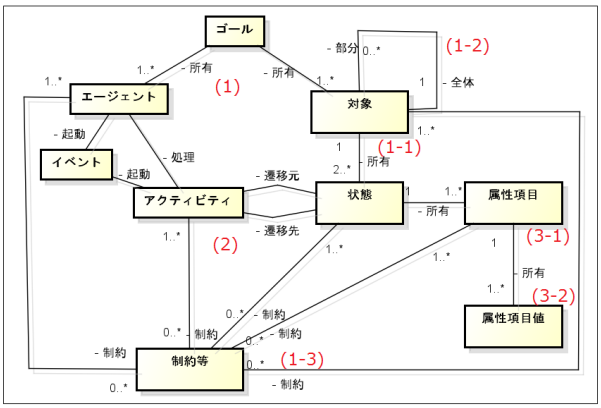

図 1 各用語の関連図

Fig. 1 The relationship diagram of terms in this paper.

「ゴール」のうち，満足ゴールは「対象」S が「状態」Q になっていることにより「エージェント」X が満足している状態と本稿では定義する．「対象」は，さらにいくつかの「対象」に部分的に分解できることがある．「対象」は 1 つあるいは複数の「属性項目」をもち，「属性項目」は「属性項目値」をもつ．「状態」は，ある「対象」の「属性項目」が特定の「属性項目値」を持つことにより表現できる．

ある「状態」から別の「状態」に遷移させる処理が「アクティビティ」である．「刺激-反応ゴール」は，特定の「状態」P のときに，特定の「イベント」が発生することにより，「アクティビティ」が起動され，適切な反応が起こり「状態」Q に遷移した状態と本稿では定義する．この「アクティビティ」は「状態」P の特定の「属性項目」の「属性項目値」を入力とし，「アクティビティ」によって「状態」Q に遷移すると，特定の「属性項目」の「属性項目値」が変更される可能性がある．この変更される可能性のある「属性項目値」が出力となる．

本稿において「情報ゴール」は，「イベント」の発生や「アクティビティ」の実行を「エージェント」が行う際に，「エージェント」にとって必要な「状態」に関する情報を知り得ている状態とする．ここでいう情報とは「属性項目」

の「属性項目値」であり，イベントの発生やアクティビティの実行時に必要な情報は，入力情報となる．

「制約等」は，「エージェント」，「アクティビティ」，「対象」，「状態」，「属性項目」間で取り得る，あるいは取り得ない関係を示し，制約と評価基準がある．制約とは，あらゆる時点で常に成立していなければならない条件で，評価基準は，状態 Q が成立しているかどうか判断する基準である．信号機のランプの例で言えば，青ランプと赤ランプは同時に点灯することはないというのが制約，いつかは青ランプが点灯し，青ランプが点灯したら少なくとも交差点を渡りきるまで点灯し続けるというのが評価基準となる．

以下，各ゴールに対する分解方法を記述する．

3.2 「満足ゴール」に対して行う分解

はじめにシステムのトップゴールを挙げる．システム開発では，システム開発を行う目的をトップゴールとする．

次に，機能要求を導くために満足ゴールへ分解する．満足ゴールは「対象 S が状態 Q になっていることによりエージェント X が満足している状態」と定義した．図 1 の（1）に対応する．この抽出法は [2] で提案したので，本稿では，その各ゴールに対して, 以下の「状態」に関する分解,「部分」に関する分解,「制約等」に関する分解を行い，必要な情報が欠如していないか確認する方法を提案する．

3.2.1 「状態」に関する分解

図 1 の（1 - 1）に対応する分解である．対象 S の状態が $\{q_1, q_2, q_3...\}$ に変化し，最終的に状態 Q に至る場合，変化する状態 $\{q_1, q_2, q_3...\}$ 毎にゴールを分割する．

例えば，エージェントが歩行者で，交差点を渡るというゴールに対して，信号機は，青ランプが点灯する，赤ランプが点灯する状態があるので，これらに分解する．

状態が 1 つしかない場合，すでにゴールが成立しているか，永遠に成立しないことになるので，この場合は状態が欠如している．状態は，必ず 2 つ以上あるはずである．

3.2.2 「部分」に関する分解

図 1 の（1 - 2）に対応する分解である．対象 S が, いくつかの部分 $\{S_1, S_2, S_3...\}$ から構成されている場合は，$\{S_1, S_2, S_3...\}$ にゴールを分割可能である．$\{S_1, S_2, S_3...\}$ の動きが同期が取れている場合や,S_1 と S_2 を取り替えても役割が同じ場合は, 分解しなくてもよいが,S_1 と S_2 が非同期の場合は, かならず分割する．

信号機の例でいえば，信号機の赤ランプが消灯しているとき，青ランプは点灯している場合と点滅している場合があり，青ランプと赤ランプは同期していない．よって，青ランプの部分と赤ランプの部分を分割する．

分割後，分割した各部分に対して前述の「3.2.1 「状態」に関する分解」を行う．

信号機の例では，分割した青ランプには点灯，点滅，消灯の 3 つの状態がある．

3.2.3 「制約等」に関する分解

図 1 の（1 - 3）に対応する分解である．対象や状態に制約や評価基準があれば，それらをゴールとして挙げる．

3.3 「刺激-反応ゴール」に対して行う分解

図 1 の（2）に対応する分解である．「刺激-反応ゴール」は，特定の「状態」P のときに，特定の「イベント」が発生することにより，「アクティビティ」が起動され，適切な反応が起こり「状態」Q に遷移した状態と本稿では定義した．このとき，「状態 Q になっているということは，状態 Q を生成するアクティビティ A が存在し，アクティビティ A を起動するイベント E が発生したので，アクティビティ A が適切に反応し，状態 Q になるというゴールが成立した」と仮定する．

そして，アクティビティ A が起動するときには入力情報が必要であり，入力情報 I は，その情報 I を出力する状態 P が，アクティビティ A を起動させるイベント E が発生する前に実行されていると仮定する．

たとえば,「顧客に商店から商品が届いている」という状態 Q は,「商店から商品を発送されている」という状態 P がその前に起きているから成立する．さらに,「商店から商品を発送している」という状態 P を生成するアクティビティは「商品」が存在しなければアクティビティを実行できないので「商店に商品が存在している」という状態 O があるはずである．

このように考えていくと，商品が届いているという状態 Q を得るには，商品が注文されている→商品が存在する→商品を発送されている→商品が届いているという状態遷移を順番に起こさなければならない．[2] では，このように状態遷移に順序性が存在する場合があることを指摘した．本稿では，状態遷移に順序性が存在する時，未定義のアクティビティを発見する方法を示す．

「商品が届いている」という例において,「商品を発送されている」という状態が抜けているのに,「商品が届く」状態が起きていれば，商品が商店に存在しながら，顧客のところにも存在するという矛盾した状態になるので，何らかの未定義のアクティビティがあることが分かる．

つまり，初期状態 a_1 というアクティビティが起き，最終的に a_n というアクティビティが起きる場合，a_1 から $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots \rightarrow a_{n-1} \rightarrow a_n$ とアクティビティが矛盾なく辿れるかどうか確認することで，未定義の状態の有無を確認することが出来る．ここで，アクティビティが辿れるとは，既存の情報 I を入力とし，演算等の既知の処理方法を用いて，目的の状態 Q に遷移し，出力が生成されることである．

3.4 「情報ゴール」に対して行う分解

図 1 の（3-1），(3-2) に対応する分解である．本稿では，

表 1　アクティビティ・出力データマトリックス
Table 1　An activity-output data Matrix.

対象	項目	A_1	A_2	A_3
S	i_1	*1	*2	
	i_2	*1	?	
	i_3			*3
	i_4			*4

*1; 自ら設定　／　*2:= $A_1.S.i_2$
／ *3:=現在 $.S.i_2$　／ *4:= $A_1.S.i_1 * A_1.S.i_2$

イベントの発生やアクティビティの実行時に必要な情報は，入力情報であるとした．そこで入力情報を確認するために，以下「属性項目」に対する分解と「属性項目値」に対する分解を行う．

3.4.1　「属性項目」に対する分解

図 1 の（3-1）に対応する分解である．満足ゴール「対象 S が状態 Q になっていることによりエージェント X が満足している状態」を満たすためには，状態 Q へ状態遷移するアクティビティ A が，入力情報をもとに出力対象 S の属性項目 $\{i_1, i_2, i_3 \cdots\}$ を生成できなければならない．このためには，出力対象 S の属性項目が，既存の情報 I をもとに，既知の演算等を使って生成できなければならない．このとき，アクティビティは既知の情報 I を入力とし，既知の演算方法を利用して出力対象 S の属性項目を出力できる．[2] では，対象 S の属性項目が，既存の情報 I と既知の演算等を利用して生成できるか確認する方法について示していなかった．本稿ではその方法として，表 1 のアクティビティ・出力データマトリックスを作成することを提案する．

表 1 の各列はアクティビティを示し，各行は出力対象の属性項目を示す．アクティビティと出力対象の属性項目の交叉するセル（以下「アクティビティと出力対象の属性項目の交叉するセル」を「交点」と記す）に，そのアクティビティで出力する属性項目の生成方法を記述する．

例えば，アクティビティ A_1 と出力対象 S の属性項目 i_3 の交点には，何も記載されていない．よってアクティビティ A_1 のときは出力対象 S の属性項目 i_3 には何も出力されない．アクティビティ A_3 と出力対象 S の属性項目 i_4 の交点には*4 と記され，*4 は「$= A_1.S.i_1 * A_1.S.i_2$」と記載されている．よってアクティビティ A_3 のときは出力対象 S の項目 i_4 は出力され，その内容はアクティビティ A_1 の出力対象 S の属性項目 i_1 にアクティビティ A_1 の出力対象 S の属性項目 i_2 を乗じたものが出力される．*3 に「現在」と書かれているが，これは，「イベントが発生し，アクティビティが起動した時点」の意味である．

「自ら設定」と記載されている交点は，アクティビティ A を実行する際に，自動的ないしは入力を促すことにより，値が設定されることを意味する．

？となっている交点は，出力は行うが，必要な情報が不明であることを示す．ここが設計・実装上で必要となる要求が欠如している部分である．

3.4.2　「属性項目値」に対する分解

図 1 の (3-2) に対応する分解である．情報 I に具体的な値を設定し，アクティビティを遷移させたときに，制約に抵触しないかどうかを確認する．

4.　事例

4.1　事例の概要

本提案を，ソフトウェア工学の共通問題である「酒屋倉庫問題」に適用した．「酒屋倉庫問題」は，情報処理 25 巻 9 号「共通問題によるプログラム設計技法解説」[3] で紹介された問題で，酒屋の倉庫の受付係のための計算機プログラムを作成する問題である．その後 25 巻 11 号 [4] で「在庫不足の商品が入庫次第発送する」と，問題の一部が仕様変更されている．

本研究では 25 巻 11 号，すなわち修正後の問題（以下これを「要求文」と記す）を元に提案した手法を適用する．

4.2　「満足ゴール」について

4.2.1　「満足ゴール」の抽出

はじめに「トップゴール」を挙げる．要求文中に「受付係の仕事のための計算機プログラムを作成せよ」という文があるので，システム対象は「受付係の仕事の電算化」となるが，受付係の仕事を電算化する目的は，「酒屋の倉庫の仕事が滞りなく行われること」であると思われるので，これがトップゴールとなる．

次にトップゴールを達成する為の満足ゴールを挙げる．要求文から，挙げるべきエージェントは「倉庫係」,「受付係」,「依頼者」であることがわかる．

「倉庫係」は，最終的にすべての仕事が行えていれば満足であるはずである．そこで「倉庫係の仕事が行えている」がゴールとなっている．「受付係」も，同様にゴールを決めている．「依頼者」は, このシステムにおいて満足が得られるのは, 酒屋が仕事をしたときなので,「依頼者が酒屋の仕事に満足している」をゴールとした．

4.2.2　「状態」に関する分解

倉庫係において, 状態 S は仕事の状態である．仕事は, 要求文によると, 入庫と出庫がある．入庫を行っているときと出庫を行っているときで倉庫係の状態，すなわち仕事の内容は違うので,「入庫が行えている」「出庫が行えている」をゴールとして挙げた．そして入庫に関しては,「積荷票を受付係に渡す」ことと「コンテナが保管できている」作業は別の仕事の内容なので，２つのゴールに分けている．

受付係は「出庫依頼を受け付けている」「在庫不足を通知できている」「在庫不足リストを出せている」「出庫指示を出せている」「空になるコンテナを指摘できている」の仕事があるので, これらをゴールとして挙げた．そして「出庫指示を出せている」に関しては在庫がある場合とない場合

で, 出庫指示を出すタイミングが異なっている．そこで, これらの 2 つは分離し,「在庫有る時出庫指示が出せている」と「在庫不足の出庫指示が出せている」に分けてゴールを挙げている．

依頼者に関しては, 対象 S は「酒屋の仕事」だが, 酒屋が行っている仕事の内容は依頼者には分からないので，酒屋の仕事の進捗に応じて状態が変わるわけではない．ただし依頼者は, 酒屋に出庫依頼をするときと, 依頼品を受け取るときは「酒屋の仕事」に影響を与える．そこで「依頼できている」「依頼分受け取っている」をゴールとして挙げた．

4.2.3 「部分」に関する分解

倉庫係の担当者が 2 人いた場合, 担当者を入れ替えても仕事の内容は変わらない．よって部分によって状態が変わることなないので, 倉庫係は部分によってゴールを変える必要はない．受付係担当者や, 依頼者も同様に，２人いる場合, 入れ替えても仕事の内容は変わらない．しかし，電話と出庫依頼票の場合で出庫依頼や依頼の内容が変わるかどうかは，この段階でははっきりしないので，それぞれ，電話と出庫依頼票のゴールを作成した．

4.2.4 「制約等」に関する分解

要求文に書かれている制約を「入庫が行えている」,「出庫が行えている」のサブゴールに挙げた．ただし,「倉庫内のコンテナは最小になっている」は，このゴール 1 個だけを「出庫が行えている」のサブゴールとすると, 倉庫内のコンテナが最小になれば, いつでも出庫ができていることになってしまうので,「出庫が行えている」の評価基準である「出庫指示に基づき出庫している」も「出庫が行えている」のサブゴールにしている．

4.3 「刺激-反応ゴール」について

4.4 「刺激-反応ゴール」の抽出

まず,「最終的に満足な状態 Q」を生成するアクティビティがあると仮定し，次に，そのアクティビティの入力値を生成する,「事前に達していなければならない状態 P」を挙げる．

状態 Q と仮定したアクティビティの対応は，表 2 のとおりである．次に「最終的に満足な状態 Q」に対する前提となる状態 P を考える．

倉庫係の「最終的に満足な状態 Q」は,「入庫が行えていること」「出庫が行えていること」である．「入庫が行えていること」の前提は,「商品と積荷票を受け取っている」ことであり「出庫が行えていること」の前提は「出庫指示を受け取っていること」である．

受付係の「最終的に満足な状態 Q」は,「出庫依頼を受け付けること」「在庫不足を通知出来ている」「在庫不足リストを出せている」「出庫指示を出せている」「空になるコンテナを指摘できている」ことである．このうち,「出庫依頼を受け付けること」に関しては, 依頼者から依頼を受けなければならない．それ以外に関しては, 現在の在庫数と出庫すべき数量が確定していなければできない．

表 2　状態 Q とアクティビティの対応表
Table 2　Correspondence of state Q and activities.

No	状態 Q	アクティビティ
a1	積荷票を受付係に渡している	積荷票渡し
a2	コンテナが保管できている	入庫
a3	出庫指示に基づき出庫している	出庫
a4	電話で出庫依頼を受けている	出庫依頼 (電話)
a5	出庫依頼票で出庫依頼を受けている	出庫依頼 (依頼票)
a6	在庫不足を通知できている	在庫不足通知
a7	在庫不足リストを出せている	在庫不足リスト
a8	在庫有る時出庫指示を出せている	出庫指示 (在庫有り)
a9	在庫不足の出庫指示を出せている	出庫指示 (在庫不足)
a10	空になるコンテナを指摘できている	空コンテナ指摘
a11	電話で依頼できている	依頼（電話）
a12	出庫依頼票で依頼できている	依頼（依頼票）
a13	依頼分受け取っている	依頼品受け取り

依頼者の「最終的に満足な状態 Q」は「依頼できている」ことと「依頼分を受け取っていること」である．このうち,「依頼分を受け取っている」を実現するためには, 酒屋から依頼分が出庫されていなければならない．「依頼できている」というゴールは，事前に行うべき処理がなく，この処理から始められる．よって，前提となる状態はない．

4.5 「刺激-反応ゴール」に対して行う分解

「刺激-反応ゴール」に対して，アクティビティ図を作成し, 順番に推移しているかどうか確認する．まず，状態 Q のアクティビティと，その前提となる状態 P のアクティビティがシステム内に存在するかどうかを確認する．もし存在しない場合，未定義のアクティビティが存在することになる．ここまでの状態をまとめ，さらに状態 P のアクティビティについて記載したゴールグラフが，図 2 である．「a1 が起きた」等，a のあとに数字が振られているのは，表 2 に記載した No に相当する．例えば「a1 が起きた」は，表 2 の a1 のアクティビティ「積荷票渡し」が起きたという意味である．状態 P に対するアクティビティは図 2 中, 六角形で記載している．

状態 P に対するアクティビティがゴールグラフ中にない場合，そのアクティビティは未定義となる．図 2 中，状態 P に対応するアクティビティがないものは 2 つあった．

１つ目は「商品と積荷票を受け取っている」というゴールに対して，積荷票を発行しているアクティビティがない箇所である．しかし，倉庫係が商品と積荷票を受け取るところが本システムの範囲であり，積荷票発行は，その前に起こることなので，それについての記載がなくても，設計・実装上で必要となる要求が抜けているとは言えない．

２つ目は「現在の在庫数が確定している」というゴールに対してである．在庫数を確定するには，出庫数と入庫数

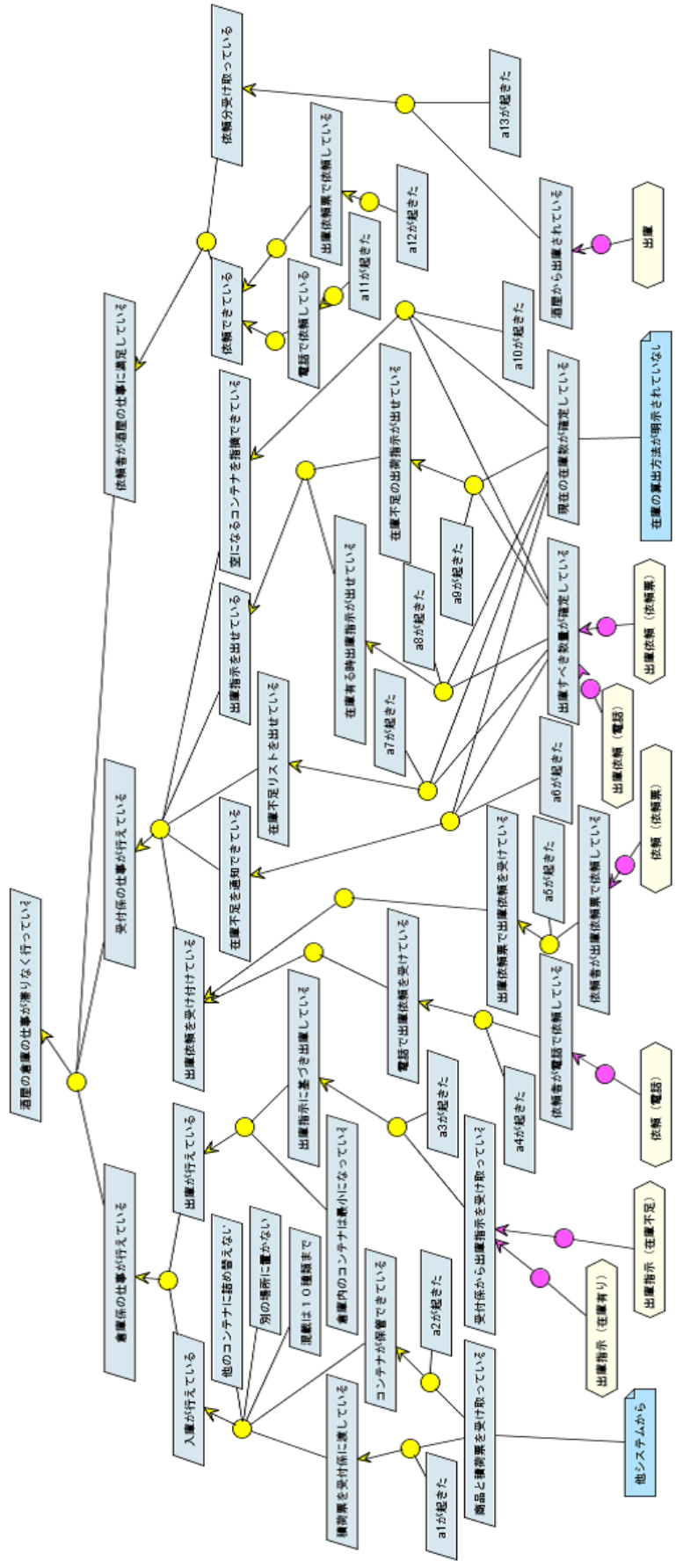

図 **2** 「酒屋倉庫問題」のゴールグラフ

Fig. 2 Goal graph of "liquor store problem"

を知らなければならない．出庫数は出庫指示から知ることが出来る．入庫数は積荷票に記載されているはずである．しかし積荷票は，受付係が倉庫係から受け取った後，どうするか書かれていない．つまり積荷票に記載されている入庫数を，受付係がシステムに入力することが書かれていない．ここが，設計・実装上で必要となる要求が欠如している部分である．

次に状態 Q の前に前提となる状態 P が来るようにアクティビティを並び替える．それをアクティビティ図を使って確認したものが図 3 である．図 3 を見ると，在庫不足リスト出力からの遷移先が不明になっている．そして入庫が起きている．そのため在庫不足リストが何のために出力され，入庫が何をもとに行われているのかが不明である．この仕様が抜けている．ここが設計・実装上で必要となる要求が欠如している部分である．

在庫不足リストは，それを元に仕入先から仕入れるために存在するものと推測される．したがって，在庫不足リスト出力後は，仕入先に発注をし，仕入先では発注を受けて，商品発送を行うと推測できる．さらに，仕入先ないしは，酒屋内で，積荷票が作られると推測できる．発注は他システムなので，それについての記載がなくても，設計・実装上で必要となる要求が抜けているとは言えない．

4.6 「情報ゴール」について

4.6.1 「属性項目」に対する分解

出力項目が，既存の情報で生成できるどうか，アクティビティ・出力データマトリクスを作成して確認する．

「酒屋倉庫問題」では，出力される帳票と，その項目について明記されている．そこで，「酒屋倉庫問題」の文章もとに, アクティビティ・出力データマトリクスを記述する．それが，図 4 である．図中，アクティビティ「積荷票渡し」は，積荷票を渡すだけで，なにも書き出さないので省略した．また依頼と出庫依頼は，電話と出庫依頼票で違いが読み取れなかったので，分けずに「依頼」「出庫依頼」で 1 アクティビティとした．出力先に対する入力項目をマトリックス上に埋めると，以下の埋まらない箇所がある．

- 在庫がある場合の出庫指示書の注文番号
- 在庫がない場合の出庫指示書の注文番号
- 在庫不足で電話通知する際の電話番号
- 出庫時の送り先住所

出庫指示書の注文番号は在庫がある場合もない場合も，一意の番号を発行しなければならない．一意にするためには，在庫がある場合もない場合も，共通のモジュールで排他制御して注文番号を発行しなければならない．この条件を満たすためには，出庫依頼を受けたとき，注文番号を発行する必要がある．

在庫がある場合は，出庫指示書に注文番号を書き出すことは可能である．出庫依頼を受けた直後に出庫指示を呼び出し，そのとき出庫指示のアクティビティの引数として注文番号を渡せばよい．

しかし，在庫がない場合は，現状の仕様では出庫指示書に注文番号を書き出すことが出来ない．なぜならば，出庫指示書は積荷票が到達したときに在庫不足リストと照合し，どの注文に対する商品が入庫したかを確認して発行されるはずである．したがって積荷票又は在庫不足リストに注文番号を含んでいないと出庫指示書に注文番号を書き出せないが，どちらにも注文番号はないからである．

そもそも依頼側に注文番号を通知しなければ，依頼者は手元に商品が届いた時，どの注文に対する商品が届いたのか確認することが出来ない．依頼者が注文番号をもとに届いた商品と注文を対応可能にするには，以下の仕様を追加する必要がある．

- 出庫依頼を受けたとき，注文番号を発生する
- 注文番号を依頼者に通知する
- 在庫不足リストに注文番号を記入する
- 出庫時にどの商品を届けたものであるか，依頼者に分かるようにするため，注文番号を文書等で通知する

この仕様が抜けている．ここが設計・実装上で必要となる要求が欠如している部分である．

また，依頼者の電話番号や送り先住所は，出庫依頼時に毎回通知してもよいが，はじめて出庫依頼する際に，電話番号，住所，名称などを通知すれば，出庫依頼時に毎回通知しなくてもよい．本稿でも，その方式を採用したとみなし，はじめて出庫依頼する前に得意先登録を行い，電話番号，住所，名称等を登録するものとする．この場合，得意先登録は「酒屋倉庫問題」の対象とするシステムとは別のシステムであるともいえる．別システムとすれば，それについての記載がなくても，設計・実装上で必要となる要求が抜けているとは言えない．

4.6.2 「属性項目値」に対する分解

ここで，具体的に値を当てはめて，制約との矛盾がないか確認する．本稿はここまでで本提案の有効性が示せている為，紙面の都合上，割愛する．

5. 考察

5.1 本提案の有効性

本提案は，設計・実装上で必要となる要求が欠如している部分を抽出できる方法を提案することを目的としていた．その箇所は以下のとおりである．

- 積荷票を入力することがはっきり書かれていない点
- 在庫に関する未定義の要求部分
- 注文番号に関する要求が未定義の部分

以上 3 点が指摘できたことにより，本提案は設計・実装上で必要となる要求が欠如している部分を抽出することに対して有効であるといえる．

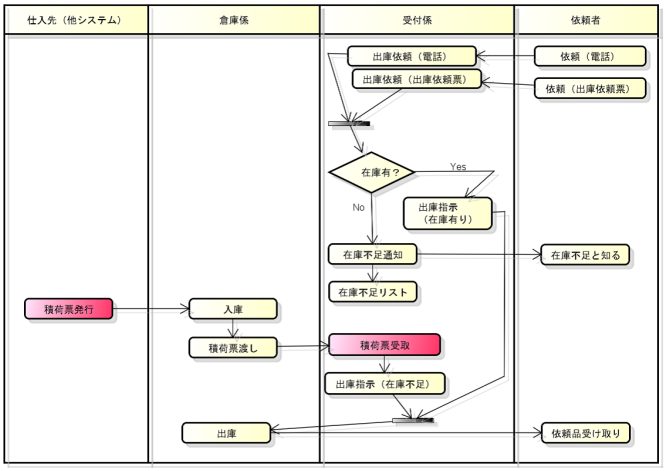

図 3 「酒屋倉庫問題」のアクティビティ図
Fig. 3 Activity diagram of the "liquor store problem"

		依頼者	受付係						出庫係		他システム		
		依頼	出庫依頼	在庫不足通知	在庫不足リスト	出庫指示（在庫有り）	出庫指示（在庫不足）	空コンテナ指摘	積荷票入力	入庫	出庫	発注	仕入先受注	商品発送
積荷票	コンテナ番号													*1
	搬入年月、日時													*1
	品名													*1
	数量													*1
出庫依頼	品名	*1	*2											
	数量	*1	*3											
	送り先名	*1	*4											
出庫指示	注文番号					?	?							
	送り先名					*4	*10							
	コンテナ番号					*6	*6							
	品名					*2	*11							
	数量					*7	*12							
	空コンテナ搬出マーク					*8	*13	*27						
在庫不足リスト	送り先名				*4									
	品名			*2	*2									
	数量			*5	*5		*14							
（出庫）	送り先住所										?			
	送り先名										*21			
	品名										*22			
	数量										*23			
（在庫不足通知）	依頼先電話番号			?										
（入庫）	数量確認									*20				*26
在庫	コンテナ番号								*16					
	品名								*17					
	数量					*9	*15		*18					
発注	品名											*24		
	数量								*19			*25		

*1 自ら設定 /*2 = 依頼.出庫依頼.品名 /*3 = 依頼.出庫依頼.数量 /*4 = 依頼.出庫依頼.送り先名 /*5 = 出庫依頼.出庫依頼.品名 − 現在.在庫.数量 /*6 = 出庫依頼.出庫依頼.品名と同じ現在.在庫.品名中の現在.在庫.コンテナ番号
/*7 = 出庫依頼.出庫依頼.品名 /*8 = if(現在.在庫.数量−出庫依頼.出庫依頼.数量 = 0,"*","") /*9 = 在庫.数量−出庫依頼.数量 /*10= 在庫不足リスト.在庫不足リスト.送り先名 /*11 = 在庫不足リスト.在庫不足リスト.品名
/*12= MIN(現在.在庫.数量,在庫不足リスト.在庫不足リスト.数量) /*13= if(現在.在庫.数量−在庫不足リスト.在庫不足リスト.数量 = 0,"*","") /*14= MIN(在庫不足リスト.在庫不足リスト.数量 − 現在.在庫.数量,0)
/*15= MIN(現在.在庫.数量 − 在庫不足リスト.在庫不足リスト.数量,0) /*16 = 自ら設定（入力対象の積荷票.コンテナ番号） /*17 = 自ら設定（入力対象の積荷票.品名） /*18 = 自ら設定（入力対象の積荷票.数量） /*19 = 現在.発注.数量−入力値.積荷票.数量
/*20 自ら設定（入荷した数量と積荷票の数量が一致なら真） /*21 = 出庫指示.出庫指示書.送り先名 /*22 = 出庫指示.出庫指示書.品名 /*23 = 出庫指示.出庫指示書.数量 /*24 自ら設定（現在.在庫不足リスト.品名を含む）
/*25 自ら設定（在庫不足リスト.数量以上） /*26 自ら設定（商品を発送） /*27 出荷指示書作成に委譲

図 4 「酒屋倉庫問題」のアクティビティ・出力データマトリクス
Fig. 4 The activity-output data Matrix for the "liquor store problem"

5.2 今後の課題

今回は提案方法で有効性が示せたが，常にこの提案方法が適切かどうか検証することは，今後の課題となる．また，今回は機能要求について考察したが，非機能要求に関してはどのように行うかという課題もある．

6. まとめ

本稿は，設計・実装上で必要となる要求が欠如している部分を抽出する方法を提案することを目標とし，その方法を提案した．そして提案方法に基づき「酒屋倉庫問題」をゴール分解したところ，その部分を発見した．よって，目標に対する有効性を示せた．

参考文献

[1] Axel van Lamsweerde: *Requirements Engineering: From System Goals to UML Models to Software Specifications*, Wiley(2009).
[2] 岡野道太郎, 中谷多哉子：要求のヌケ・モレを防ぐためのゴール分解方法の提案と実験-ソフトウェア・シンポジウム 2014　WG3 の事例-，ソフトウェア・シンポジウム 2015(2015)
[3] 山崎利治：共通問題によるプログラム設計技法解説，情報処理 Vol25 No9(1984).
[4] 山崎利治：共通問題によるプログラム設計技法解説（その 2），情報処理 Vol25 No11(1984).
[5] *i* homepage*, 入手先 ⟨http://www.cs.toronto.edu/km/istar/⟩(2015/5/1)