ゴール指向分析ＫＡＯＳにおける依存性を考慮した要求抽出法の考察 -酒屋倉庫問題の場合-

岡野道太郎
筑波大学大学院ビジネス科学研究科
okano@gssm.otsuka.tsukuba.ac.jp

中谷 多哉子
放送大学 情報コース
tinakatani@ouj.ac.jp

要旨

本稿は，要求仕様書に書かれるべき，設計・実装・テストに必要な情報を獲得するためにはどのような手法を用いればよいかの知見を得ることを目標とする．この目標へのアプローチとして，ゴール指向要求分析の一手法である *KAOS* を用いる．この手法はシステムの目標であるトップゴールから，ゴールの詳細化を繰り返すことにより要求を獲得する．本稿ではゴール $P \rightarrow Q$ の詳細化手法として，状態遷移→部分を詳細化するマイルストーン分解と，状態 Q を分解する要素分解を用いる．その際，状態 P については分解していないが，状態 P を分解する必要があるかについて考察した．考察を行うために「酒屋倉庫問題」のゴール分解を行った．すると，状態Ｐを分解しないと依存性に関して問題が生じることが判明した．そこで本稿では状態Ｐの部分を分解し，依存性を明記する表記法を提案する．関連研究では本稿の手法と他の要求抽出手法との関連について議論し，今後の研究において本稿で取り上げた達成ゴール以外のゴールについて述べる．

1. はじめに

ソフトウェア開発は，要求分析の後，設計・実装・テストの各工程が続く．そのため要求分析では，設計・実装・テストに必要な情報が獲得される必要がある．この「設計・実装・テストに必要な情報」は何であり，その情報を要求分析の工程で獲得するには，どのような手法を用いればよいかの知見を得ることが本稿の目標である．この目標へのアプローチとして，筆者はゴール指向分析の一手法である KAOS[1] を用いる．KAOS については２章で詳述するが，簡潔に述べると，KAOS はシステムの目標であるトップゴールからゴールの詳細化を繰り返すことで要求を獲得する手法である．我々の今までの研究ではゴールを「状態 P から状態 Q への状態遷移」とし，詳細化手法として，状態遷移を詳細化するマイルストーン分解 [1] と，状態 Q を詳細化する要素分解 [2] を利用してきた．つまり，状態 P は詳細化しなかったが，詳細化する必要がないのだろうかということについて，本稿では考察する．その結果，分解を行わないと依存性に関して問題が生じ，設計に必要な情報が欠けることが判明したので，依存性に対する提案を行い,「酒屋倉庫問題」の事例を用いて提案を評価する．

本稿の構成を以下に述べる．まず第二章で用語の説明を行う．第三章で，上述した状態Ｐの分解について「酒屋倉庫問題」を利用して考察する．第四章では，考察時に生じた問題を解決する手法を提案し，その手法を酒屋倉庫問題に適用する．第五章では，提案手法について考察する．第六章では関連研究として，本稿と i*[3],UML[4] の関連について述べる．第七章では今後の研究について述べ，第八章でまとめる．

2. 用語の説明

2.1. KAOS について

KAOS は，ゴールを要求獲得に活用する，ゴール指向要求分析の一手法である．はじめにシステムが満たすべきゴールをトップゴールとして 1 つ挙げ，そのゴールを詳細化して，いくつかのゴールに分解し（この「詳細化により分解されたゴール」をサブゴールと記す），サ

ブゴールを再度詳細化してサブゴールのサブゴールへ分解するというように，詳細化を繰り返す．すると，トップゴールでは，多くの人が関わり，実現に複雑な過程を経るとしても，最終的には 1 人あるいは 1 システム（これをエージェントと呼ぶ）が満たせる程度のゴールとなる．このとき分解を終了し，末端ゴールをエージェントに対する要求とする．このようにＫＡＯＳを用いれば，要求はゴールの詳細化によって抽出できるが，そもそも，「ゴールとは何か,「設計・実装・テストに必要な情報は要求だけなのか」,「詳細化はどのように行うのか」については説明していない．そこで，まず,「設計・実装・テストに必要な情報は要求だけなのか」について説明する．

2.2. 設計・実装・テストに必要な情報と要求の関係

「設計・実装・テストに必要な情報」は，要求仕様書に文や図としてまとめられる．図は文章で表現することも可能なので,「設計・実装・テストに必要な情報」は文で表現することが可能であるとみなすことが出来る．そして文を Lamsweerde は図 1 のように分類している [1].

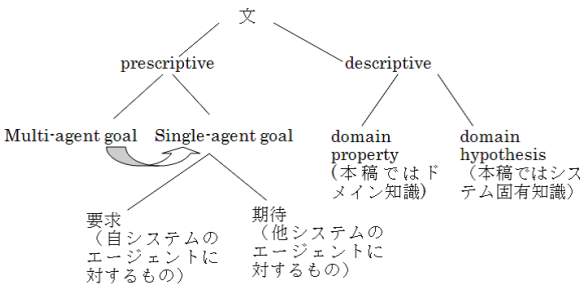

図 1. 文の分類（[1] を元に作成）

文は意図を含むか否かにより，prescriptive な文と，descriptive な文に分けられる．本稿では前者を規範的な文，後者を記述的な文と記す．記述的な文は，自然法則のようなシステム開発者の意図とは関係なく従わなければならない domain property と，本システム開発上，そのようになっているという（例えば，本社は東京にあるなど）domain hypothesis がある．本稿では，前者をドメイン知識，後者をシステム固有情報と記す．そして規範的な文はゴールであり，このゴールは多くの人が関わる Multi–agent goal から，詳細化を繰り返すことにより，一人または一システム（本稿ではこれらの人とシステムをまとめてエージェントと記す）が実現できる Single–agent goal に分解される．この Single–agent goal は，対象システムのエージェントが実現する場合「要求」となり，それ以外のエージェントが実現する場合「期待」となる．

つまり，文は意図を含むか否かにより，記述的な「ドメイン知識」,「システム固有情報」と，規範的な「要求」,「期待」とに分かれる．よって,「設計・実装・テストに必要な要求仕様書で表現されるべき文」は，対象システムに関する文なので（「期待」を除いた）「要求」,「ドメイン知識」,「システム固有情報」であるといえる．すなわち要求以外にも必要な情報はあるが，少なくとも要求は必要であると言えるので，まず要求について分析する．

そして「要求」はゴールを分解することによって得られる．次に「ゴールとは何か」について説明する．

2.3. ゴールとは何か

ゴールとは，Lamsweerde[1] によると,「エージェントの協調によって満足すべき意図を示した規範的な文」であるとしている．ゴールにはいくつかのタイプがある（詳細は「６．今後の研究」にて述べる）が，本稿では達成ゴールを取り扱う．達成ゴールは，振る舞いを表したゴールの一種で，状態遷移の連続であると [1] では定義している．よって，達成ゴールは「状態 P から状態 Q へ状態への遷移が完了している」のように記述できる．このような状態遷移を本稿では P → Q と表現する．P が状態変化が起こる前の状態（本稿では事前状態と呼ぶ），Q は状態変化が起こった後の状態（本稿では事後状態と呼ぶ），→は P から Q へ状態が遷移する過程について表している．なお達成ゴールを [1] では，様相論理を用いて P →◇ Q で表現している．本稿ではそこまで厳密な定義をせず，ある状態から別の状態に遷移すれば，その状態遷移を P → Q と表現し，それを達成ゴールとする．

よってトップゴールも達成ゴールの形式で書かれる．そして詳細化すると最終的に要求が達成ゴールの形で得られるが，どのように詳細化したら，要求が得られるかについては説明していない．そこで次に詳細化の方法について述べる．

2.4. 詳細化はどのように行うのか

達成ゴール P → Q の詳細化を，本稿では 2 つの方法で行う．一つは P → Q のうち，状態遷移（→）を分解

するマイルストーン分解であり，もう一つは Q の部分を分解する要素分解である．

マイルストーン分解は [1] で述べられている．マイルストーン分解とは，P から Q への状態遷移が起こる過程において，マイルストーン M の状態を経るのであれば P → M と M → Q に分解できるとする分解法である．

要素分解 [2] は，P から Q への状態遷移が起こったとき，事後状態 Q がいくつかの要素 $\{Q_1, Q_2, \cdots Q_n\}$ から成り立つとき，状態遷移 P → Q を事前状態 P，事後状態を各要素として分解する方法である．すると，P → Q_1,P → Q_2, ⋯ P → Q_n に分解できるが，それだけでは $\{Q_1, Q_2, \cdots Q_n\}$ という各要素が存在する状態になるだけで，それが結合した状態にならないので，$\{Q_1, Q_2, \cdots Q_n\}$ → Q という状態遷移も付加することもあるとする分解法である．例えば,「お皿の上にスパゲティが盛り付けてある」を分解するとき,「お皿がある」「スパゲティがある」だけでは，お皿の上にスパゲティがあるかどうかわからない．そこで「お皿とスパゲティがある状態から，スパゲティをお皿の上に盛り付けられている」というゴールを追加して分解する．なお要素分解における要素は，分解時に着目する視点によって様々に変わる．例えば，ユースケースに着目して，ユースケースごとに分解することも考えられるし，構成部品に着目し，部品ごとに分解することも考えられる．さらに機能に着目し，機能ごとに分解することもあり得る．

2.5. 詳細化と本稿での考察との関係

我々は既に，達成ゴールをマイルストーン分解と要素分解を用いて詳細化し，要求を導き出し，それとドメイン知識・システム固有情報をもとに実装が行えるものがあることを，電気工事の事例により示した [2]．

しかしここで疑問がある．状態遷移→と状態 Q は分解したが，事前状態の P は分解する必要はないのだろうか．本稿ではその事前状態 P の分解について考察する．

3. 事前状態 P の分解の必要性 「酒屋倉庫問題」による考察

酒屋倉庫問題とは，情報処理 [5] に記載されている「ある酒類販売会社」（以下酒屋と記す）における「受付係の仕事（在庫なし連絡，出庫指示書作成，在庫不足リスト作成）のための計算機プログラムを作成する」という問題である．酒屋倉庫問題における本稿に関係のある事項についてのみ，以下に記載する．

- 酒屋について
  - 酒屋には倉庫係，受付係がある
- 入庫について
  - 倉庫係はコンテナ入りの酒を倉庫に保管し，積荷票を受付係へ渡す
  - 積荷票の項目はコンテナ番号と搬入年月，日時，内蔵品目と数量の繰り返しである
- 出庫について
  - 受付係は出庫依頼を受ける
  - 出庫依頼を基に倉庫係へ出庫指示書を出す
  - 出庫指示書の項目は注文番号，送り先名，コンテナ番号，品名，数量である
  - 出庫依頼の項目は，品名，数量，送り先名である
  - 在庫がない場合は依頼者に電話連絡し，在庫不足リストに記載する
  - 在庫不足リストの項目は，送り先名，品名，数量である
  - 空になる予定のコンテナを倉庫係に知らせる（出庫指示書に記載）
  - 倉庫内のコンテナ数はできる限り最小にする
  - 出庫指示書の項目は，注文番号，送り先名，コンテナ番号，品名，数量，空コンテナ搬出マークである

なお,「在庫なし連絡」のための計算機プログラムとは，本稿では，以下の項目の「在庫なし連絡票」を作ることとする．

- 在庫なし連絡票：注文番号，送り先名，品名，注文数量，不足数量

ここで,「帳票出力が行われている」というゴール分解を行うと，図 2 のような分解を一例として挙げることができる．これ以外の分解方法も可能ではあるが，本稿では問題点が明示しやすいため，この分解を元に議論を進める．

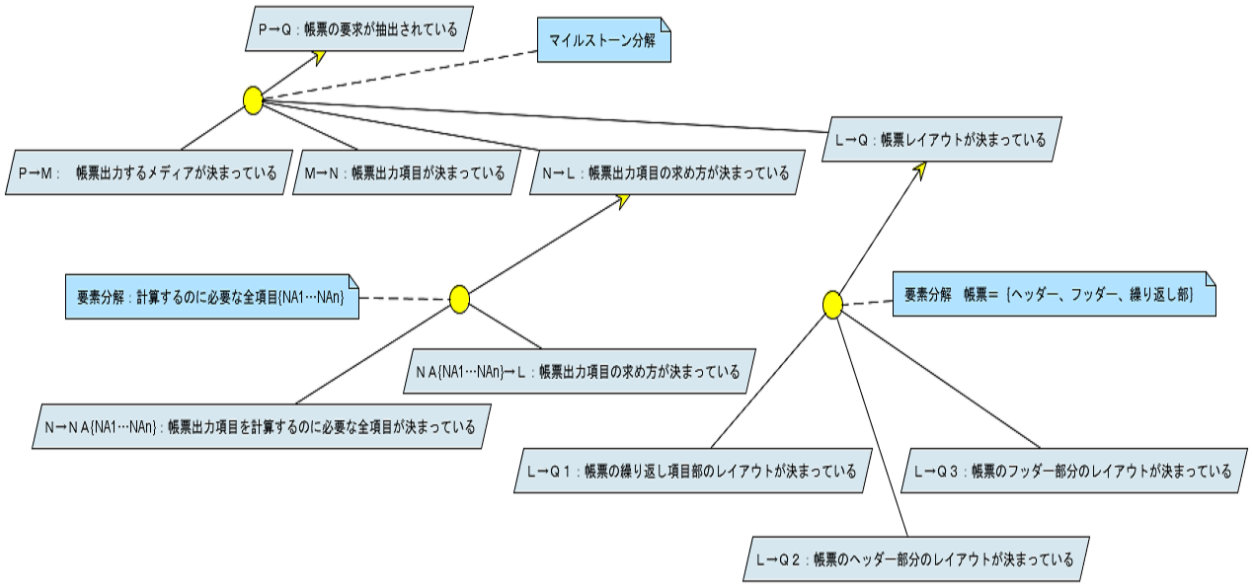

図 2. 一般的な帳票作成のゴール分解

ここで図 2 の表記法について述べる．この表記は KAOS を基にしている．すなわち，始めにトップゴールを挙げ，トップゴールをマイルストーン分解か要素分解を行ってサブゴールを挙げる．この際，マイルストーン分解を行ったか，要素分解を行ったかについては，注釈として記述している．さらに各サブゴールに対し，マイルストーン分解または要素分解を適用して詳細なサブゴールを導いている．このとき図 2 のように，ゴールとサブゴールの間にある円と，各サブゴールが線で結ばれている場合がＡＮＤ分解，ゴールから各サブゴールに線が出ている（この場合，１本の線の間に１つの円がある）場合がＯＲ分解である．ＡＮＤ分解は，サブゴールの全てが成立しないとゴールが成立しないものであり，ＯＲ分解は，サブゴールの一つが成立すればゴールが成立するものである．ゴールの記号の内部に書かれている文字列をラベルと呼ぶ．本稿では，ラベルの前半にＰ→Ｑ等，状態遷移が始めに記されている．そして，その後に「帳票の要求が抽出されている」などの状態が記されている．つまり，ラベルの前半にゴールの状態遷移，後半に最終状態Ｑの状態を表すものとする．したがって，状態Ｐの状態はＰを最終状態に持つゴールのラベルに記されている．本来，最終状態が同じゴールの場合，ラベルの後半も同じ表現になるはずであるが，同じ状態を指しても視点により表現が異なる場合がある．このため同じ最終状態のゴールでもラベルの表現が違うことがある．

図 2 に対し，帳票が「在庫なし連絡票」,「出庫指示書」,「在庫不足リスト」であることをトップゴール直下のサブゴールにそれぞれ適用すると，図 3 のようになる．

ここで「Ｎ→Ｌ１：在庫なし連絡票の出力項目の求め方が決まっている」について議論する．このゴールは事前状態Ｎ，すなわち「帳票出力項目が決まっている」を満たしている必要がある．このゴールで示す「帳票」とは「在庫なし連絡票」,「出庫指示書」,「在庫不足リスト」の３つを指し，これらすべてが完了していないと事前状態Ｎを満たしていないことになる．しかし，Ｌ１の状態「在庫なし連絡票の出力項目の求め方が決まっている」に遷移するためには「Ｍ→Ｎ１：在庫なし連絡票の出力項目が決まっている」は満たす必要はあるが,「Ｍ→Ｎ２：出庫指示書の出力項目が決まっている」を満たす必要は必ずしもない．つまり，現状の表記法ではＮ→Ｌ１を実行するためには，Ｍ→Ｎ１，Ｍ→Ｎ２，Ｍ→Ｎ３が必要となってしまっているが，Ｍ→Ｎ２，Ｍ→Ｎ３は必ずしも必要ではなく，その分，成立条件が厳しくなってしまっている．この問題を以下「問題１」と記す．

また「Ｌ→Ｇ１：在庫なし連絡票のレイアウトが決まっている」について議論すると，在庫なし連絡票のレイア

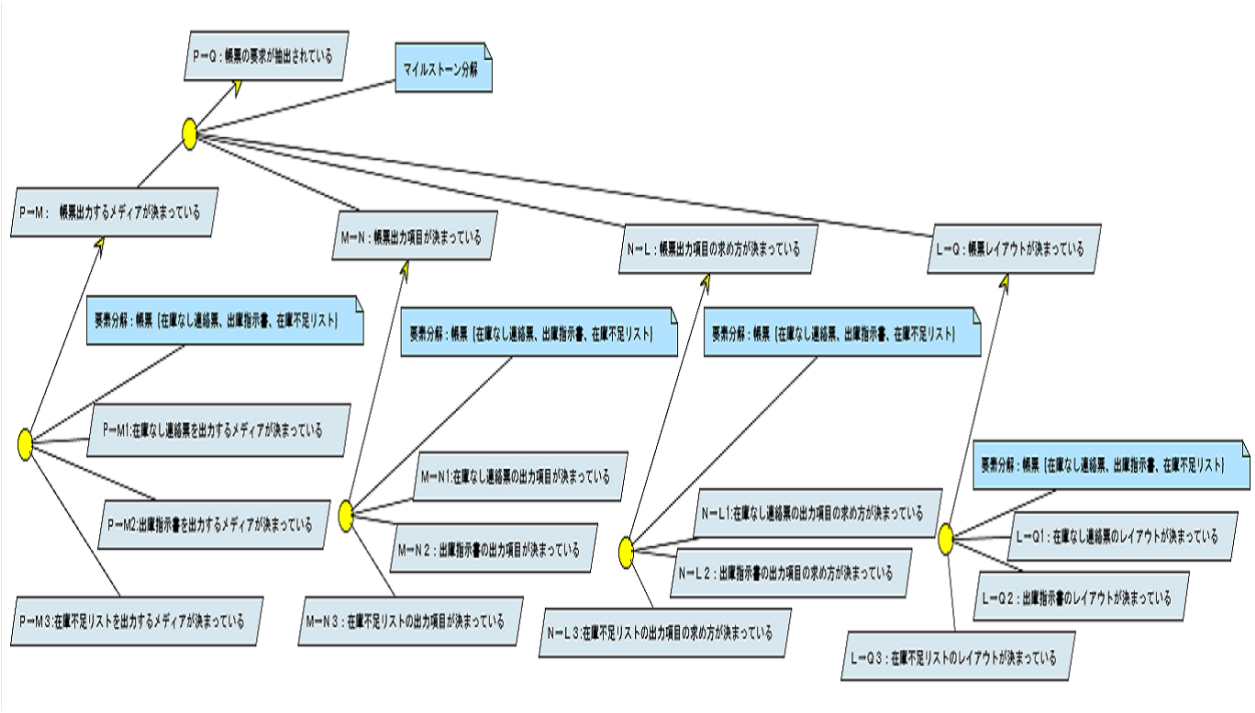

図 3. 酒屋倉庫問題のゴール分解

ウトを決めるためには,「在庫なし連絡票の出力項目が決まっている」とともに,「在庫なし連絡票を出力するメディアが決まっている」必要がある（ここでメディアとはＡ４の大きさのＰＤＦ等，出力先を示す）．そこで，Ｌ→Ｇ１のゴールの事前状態としては,「Ｎ→Ｌ１：在庫なし連絡票の出力項目の求め方が決まっている」と「Ｐ→Ｍ１：在庫なし連絡票を出力するメディアが決まっている」の両方が満たされなければならない．ただし,「Ｐ→Ｍ１：在庫なし連絡票を出力するメディアが決まっている」はレイアウトを決める直前までに決まっていればよいのであって,「Ｎ→Ｌ１：在庫なし連絡票の出力項目の求め方が決まっている」と同時並行で行ってもかまわない．一般的に出力先等，大まかな事項を決定してから出力項目等の詳細項目を決めるので，Ｐ→Ｍ，Ｍ→Ｎ，Ｎ→Ｌという遷移にしただけであり，順序は必須ではない．ところが図の表記法だと，同時並行を認めないように見えてしまい，その分，事前状態の成立条件が厳しくなってしまっている．この問題を以下「問題２」と記す．

これら問題１，２は依存性の問題といえる．本稿において依存性とは，ゴールＰ→Ｍ，Ｍ→Ｑに対し，Ｍ→Ｑを達成するには事前状態Ｍが成立しなければならず，Ｍが成立するためにはＰ→Ｍが成立しなければならない場合，ゴールＭ→Ｑが達成するか否かは，ゴールＰ→Ｍが達成するか否かの結果に影響されるといえる．このように，ゴールＭ→Ｑの事前状態の成立が，ゴールＰ→Ｍの達成結果による場合，ゴールＭ→ＱはゴールＰ→Ｍに依存しているといい，この関係を依存性という．

問題１は事前状態Ｐを分解しないことにより生ずる問題であり，事前状態の範囲，すなわち依存性の範囲を広く取りすぎている問題である．問題２は事前状態を分解したり，並行処理を認めたりした場合，それを認めなければ必ず成立していたが，認めることにより成立するかどうか確かではなくなる可能性がある場合の問題である．

依存性は,「設計・実装・テストに必要な情報」である．なぜならば，もし依存性の情報がなく要求Ｐ→Ｑが成立しているかテストを行おうとすると，事前状態Ｐが成立していない状態でテストを行う可能性もあり，その場合正しく実装されていても，テストに失敗する．したがってテストを行う者にとって，テスト対象の依存性は必要な情報といえる.「設計・実装・テストに必要な情報」を獲得することが本稿の目的なので，依存性の問題を解決しないと，本稿の目的は達成できないことになる．そこ

で，問題１，２の解決法を次の章で提案する．

4. 提案手法

ここでは，以下の表記法を提案する．
ゴールＰ→Ｑにおいて

- （問題１解決のための表記法）事前状態Ｐがいくつかの要素｛Ｐ１，Ｐ２，Ｐ３ … Ｐｎ｝に分解可能で，それらの要素の一部が成立すれば事前状態が成立したとみなせる場合，事前状態Ｐの後にっをつけ，その要素を列挙することにより，その列挙した状態が成立すれば，事前状態が成立したとみなす．例えば，事前状態Ｐが｛Ｐ１，Ｐ２，Ｐ３｝に分解可能で，Ｐ１を満たせば事前状態が成立したとみなせる場合は，Ｐっ Ｐ１→Ｑと記述する．

- （問題２解決のための表記法）事前状態の分解を認めなければ成立した状態で，事前状態の分解を認めることにより，成立するかどうかが確かではない場合や，その成立を明示したい場合は，事前状態Ｐの後に＞をつけ，その状態を列挙することにより，その列挙した状態が成立しなければ，事前状態が成立しないものとすることができる．例えば，Ｐ→Ｑが，Ｐ→Ｍ，Ｍ→Ｑと分解でき，Ｐが｛Ｐ１，Ｐ２，Ｐ３｝，Ｍが｛Ｍ１，Ｍ２，Ｍ３｝，Ｑが｛Ｑ１，Ｑ２，Ｑ３｝に分解可能な時，Ｍっ Ｍ１→Ｑ１の事前状態Ｍ１が成立する際に，Ｐ１，Ｐ２が成立する必要がある時（Ｐ３は必須でない）Ｍっ Ｍ１，Ｍ＞｛Ｐ１，Ｐ２｝→Ｑ１と記述する．

この表記法を基に図 3 を分解すると，図 4 のようになる．

本提案によって変わった点は，ゴールラベルの前半に記述される状態遷移を表す部分である．

- Ｍ→Ｎのサブゴールにおいて，このゴールはＰ→Ｍと並行して行っても問題を生じないため，Ｍ＞Ｐ→Ｎ１等，Ｐの時点すなわちはじめから実行可能であることを明記した．

- Ｎ→Ｌのサブゴールにおいて，Ｎは，それぞれの帳票に対する事前状態が成立していればよいので，Ｎっ Ｎ１→Ｌ１等，該当する帳票の状態のみを事前状態に挙げるようにした

- Ｌ→Ｑのサブゴールにおいて，このときまでに，各帳票の出力項目とその求め方だけでなく，メディアも決まっていないといけないため，Ｌっ Ｌ１等の他にＬ＞Ｍ１等も加え，Ｌっ Ｌ１，Ｌ＞Ｍ１→Ｑ１等とした

なお，事前状態を分解したとき，その分解した状態が，他のゴールの最終状態として存在するときは，その状態名を使う．たとえば，Ｎっ Ｎ１→Ｌ１の分解された事前状態Ｎ１は，Ｍ＞Ｐ１→Ｎ１の最終状態Ｎ１と同じ状態なのでＮ１という状態名を使っている．もし分解された事前状態が，他のゴールの事後状態にも表れず，初期状態Ｐでもない場合は，別の状態名をつけ，その状態の内容を注釈に記載する．

5. 提案手法に対する考察

まず，この提案した表記法により，問題が解決したかを確認する．

- 問題１については，「在庫なし連絡票の出力項目の求め方が決まっている」のゴールはＮっ Ｎ１→Ｌ１となり，このゴールは在庫なし連絡票以外には依存しなくなっている．すなわち，依存する範囲を的確に示していることなり，問題は解決した

- 問題２については，「在庫なし連絡票のレイアウトが決まっている」のゴールはＬっ Ｌ１，Ｌ＞Ｍ１→Ｑ１となり，このゴールは在庫なし連絡票のメディア (出力先等）と出力項目の求め方に依存していることを明確に示している．すなわち，依存する範囲を的確に示していることなり，問題は解決した

つまり，問題１，２は本提案の表記法により適切になり，依存性が的確に示せるようになった．本稿のゴール分解手法は KAOS を基にしているが，KAOS では依存性が明白に示せなかったことは，問題１，２が生じたことからもわかる．新たに本稿が提案する記法を導入することによって，依存性が明確に記述できるようになった点に本稿の新規性，有用性がある．

ただし，本稿では事前状態Ｐを分解しているが，その際，分解の妥当性は保障していない．つまり，事前状態ＰをＰ１，Ｐ２に分解した場合，Ｐ１は，本当に事前状態Ｐに含まれているかということを保障するのは，その分解を行った者の責任である．

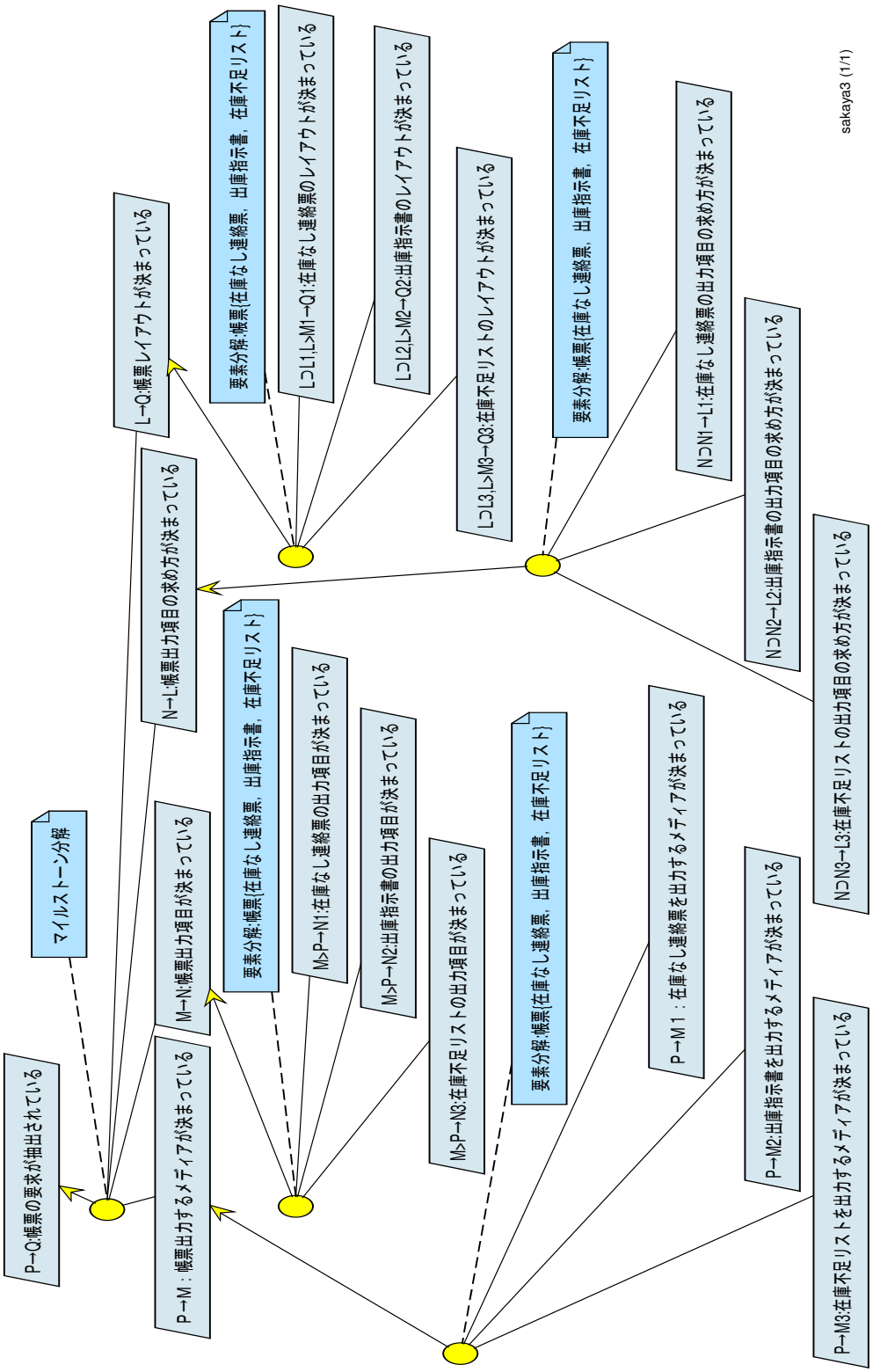

図 4. 提案手法による酒屋倉庫問題のゴール分解

本稿で行っているマイルストーン分解は，状態遷移に基づく分解であるが，状態遷移はプロセスによって実現されるので，プロセス分解を行っているといえる．また，要素分解は，最終状態の構成要素に着目して分解を行っているが，それは構造を分解していることになる．そして,「酒屋倉庫問題」の事例では，マイルストーン分解と要素分解の両方を用いている．したがって，今回の事例はプロセス分解のみを行っているのでも，構造による分解のみを行っているのではなく，両方の分解を適切に切り替えることによって分解が実現できているといえる．一般にシステムは，一連の情報処理プロセスが続いて実行される．この際，各プロセスの入出力データにはデータ構造が存在する．よってシステムを分析するには，プロセスの分析と各プロセスの入出力データのデータ構造分析の両方が行える必要がある．マイルストーン分解と要素分解を適切に組み合わせることにより，このプロセス分析とデータ構造分析が適切に行える．

次の章では，この依存性とプロセス分解を表現する振る舞い図や構造による分解を表現する構造図について，既存の他の研究と本稿の関係について議論する．

6. 関連研究

本稿はゴール指向要求分析のＫＡＯＳを基にしている．ゴール指向要求分析では，この他に i*[3] がある．i* では，ゴールとアクター間の依存性をＤの記号を用いて表すことにより明示している．ＫＡＯＳでは要求をエージェントに割り付けることにより，要求 (ゴール）とアクターを含むエージェントの関係を明示することができる．さらに本稿の研究により，要求（ゴール）間の依存性も明記できるようになった．i*は KAOS のようにトップゴールと要求の関係を明示できないので，本稿の表記法は，i*で表記できる依存性に加え，トップゴールからの関係もゴールの詳細化により明示できる点で i*よりも多くの情報を表記できるといえる．ただし，as–is 分析の場合，各エージェントが行っているゴールがトップゴールを意識しているかどうかは定かではないため，この場合，トップゴールを明示しない i*のほうが描きやすい可能性はある．しかし to–be 分析においては，各エージェントのゴールはトップゴールに貢献すべきであるので，トップゴールから各自のゴール，要求との関係が示せる本稿の表記法のほうが望ましいといえる．

また，本稿ではマイルストーン分解と要素分解によりゴール分解を行っている．マイルストーン分解は，振る舞いゴールである達成ゴールを分解していることから，振る舞いの分解といえる．そして本稿の記述法により振る舞いゴール間の依存性を示すことができるようになった．そこで，依存性を辿っていくことにより，一連の振る舞いの流れを辿ることができる．このような振る舞いの流れを記述した図が「振る舞い図」であり，ＵＭＬ [4] においては，アクティビティ図等が，振る舞い図に相当する．

要素分解は最終状態を各要素に分解している．その際，最終状態のデータ構造に着目し，データを構成する各要素に分解して，データ構造を表すことも出来る．そのような構造を表す図には，ＥＲ図やＵＭＬのクラス図等の構造図がある．

つまり本稿の表記法は，構造図と振る舞い図の両方を１つの図の中に統合している．実際の要求分析は，本稿の酒屋倉庫問題の事例で示したように，プロセス分析と構造分析が適度に切り替えられて行われる．したがってこの分析結果は別々の図ではなく，統合した１つの図に書かれることによって，その関係性が明示できる．その点で，本稿の表記法は優れている．ただし，実際にデータベースのテーブルを定義するときは，プロセスは不要であり，データベースの構造のみが必要となる．そのような場合は，テーブル構造のみを明記できるＥＲ図がふさわしい．

関連研究と本稿の関係についてまとめると，本稿はＫＡＯＳに新たな表記を加えることにより，依存性が明記できるようになり，これとマイルストーン分解，要素分解により，依存性，振る舞い，構造を１つの統合した図で示せるようになった．前章で述べたとおり，システムを分析するには，プロセスの分析と各プロセスの入出力データのデータ構造分析の両方が行える必要がある．プロセスの振る舞いと依存性，入出力データの構造を１つの図で示せる点で，本稿の提案は優れており，有用性はある．ただし，場合によっては，それらのうちの一部の情報のみが必要な場合があり，その場合には，依存性を示す i*や振る舞いを表すＵＭＬの振る舞い図 (アクティビティ図等)，構造を示すＥＲ図やＵＭＬの構造図（クラス図等）のほうがふさわしい．

7. 今後の研究

本稿では達成ゴールのゴールの詳細化について議論した．しかしゴールには達成ゴール以外のゴールもある．ゴールは，図 5 に示すように分類される [1]．すなわち，

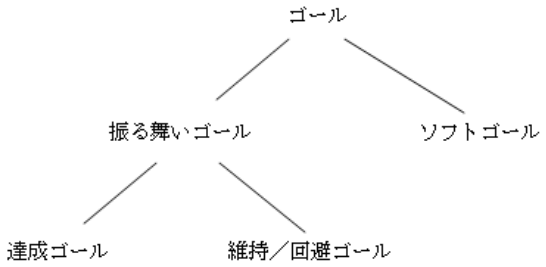

図 5. ゴールの種類（ゴールタイプ）（[1] を元に作成）

まずはじめにソフトゴールと振る舞いゴールに分かれ，振る舞いゴールは本稿で取り上げた達成ゴールの他に，維持ゴールや回避ゴールがある．ソフトゴールは,「レスポンスが 3 秒以内にあること」のような望ましい状態を示したゴールであり，非機能要求の中には，ソフトゴールで表現できるものもある（セキュリティ等は回避ゴールで示すものもある）．また，達成ゴールは最終状態である「よいことがいつかは起こる」形のゴールであるが，回避ゴールは「悪いことは起こらない」という形のゴールであり,「不正アクセスが起こらない」といったセキュリティ要求を表現することができる．維持ゴールは「よいことが起こり続ける」であり,「システムが稼働し続けている」という運用・保守に関する要求を表現することができる．非機能要求やセキュリティ要求，保守・運用の要求は，設計上必要な情報なので,「要求仕様書に書くべき内容」であるが，本稿ではそれらについて触れられていない．今後研究する必要がある．その際に，達成ゴールと i*,UML 等は関連があったが，他のゴールとの関連性はどうなのかについても研究する必要がある．達成ゴール以外のゴールもＫＡＯＳでは表現できるが，もし i*やＵＭＬでは表現できない場合，ゴール抽出をＵＭＬのみで行うと，要求仕様書に書くべき要求が抜けることになるからである．

また，本稿では酒屋倉庫問題の一部しか取り上げていない．酒屋倉庫問題のゴール分解を完成させ，トップゴールから要求が抽出できることを確認する必要がある．我々は [6] において，酒屋倉庫問題は，以下の点が抜けていることを指摘した．

- 注文番号は出庫指示書の出力項目として挙げられているが，どのように生成されるか明示されていない
- 積荷票からどのように出庫を割り当てるか，明示されていない

ゴール分解中に，これらの点が抜けていることを発見できるか確認することも必要となる．

8. まとめ

本稿は，要求仕様書に書かれるべき，設計・実装・テストに必要な情報を獲得するためにはどのような手法を用いればよいかの知見を得ることを目標とした．要求仕様書に書かれるべき情報は，要求，ドメイン知識，システム固有情報であるが，本稿ではこのうち要求を獲得する手法について対象とした．要求を獲得するアプローチとして，ゴール指向要求分析の一手法である KAOS を用いた．この手法はシステムの目標であるトップゴールから，ゴールの詳細化を繰り返すことにより要求を獲得する．本稿ではゴール Ｐ → Q の詳細化手法として，状態遷移→部分を詳細化するマイルストーン分解と，状態 Q 部分を分解する要素分解を用いた．その際，状態Ｐについては分解していないが，状態Ｐを分解する必要があるかについて考察した．考察を行うために「酒屋倉庫問題」のゴール分解を行った．すると，Ｐの部分を分解しないと，依存性に関して問題が生じることが分かった．そこで本稿では状態Ｐの部分を分解し，依存性を明記する表記法を提案し，酒屋倉庫問題で確認した．本提案により，依存性，振る舞い，構造を 1 つの統合した図で示せるようになった．既存の手法でも i*は依存性，ＵＭＬは振る舞いや構造を示せるが，依存性・振る舞い・構造の関連を 1 つの図で示せる点で，本稿の提案は優れており，有用性がある．本稿で取り上げなかったゴールから，非機能要求やセキュリティ要求，保守・運用の要求が抽出できる．これらも設計･実装･テストに必要な情報なので，今後研究する必要がある．また，本稿は酒屋倉庫問題の一部しかゴール分解していないので，ゴール分解を完成させ，設計・実装に必要な要求が欠けていることが検出できるかどうか確認することも今後の課題である．

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, 状態と状態遷移に着目したゴール指向要求分析手法の考察, 知能ソフトウェア工学研究会 (KBSE) , 電子情報通信学会技術研究報告 : 信学技報 Vol.116 No.493,2017,pp25–30

[3] *i* homepage*, http://www.cs.toronto.edu/km/istar/(2017/3/12)

[4] *WHAT IS UML*, http://www.uml.org/what-is-uml.htm(2017/3/12)

[5] 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).

[6] 岡野道太郎・中谷多哉子, 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- , ソフトウェア エンジニアリング シンポジウム 2015, 情報処理学会 ソフトウェア工学研究会,pp107 – 114