要件定義における問題フレームの活用、評価と改善

上野, 俊也 / UENO, Toshiya

(発行年 / Year)
2007-03-24
(学位授与年月日 / Date of Granted)
2007-03-24
(学位名 / Degree Name)
修士(理学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

平成 18 年度修士論文

要件定義における問題フレームの活用、評価と改善

法政大学
情報科学研究科
情報科学専攻
学籍番号　05T0003
氏　名　　上野俊也
指導教官　　溝口徹夫

1 はじめに

1.1 研究の背景

ソフトウェアを開発する際、「どんなソフトウェアを作るか」といった「要件」を決める、「要件定義」といった工程が必要となる。要件定義の質が低いと、目的に沿わないソフトウェアが完成してしまう。また、予算やスケジュールの超過といった不都合の原因ともなる。要件定義はシステムアナリストやシステムエンジニアが行い、専門的な知識を必要とする重要な工程である。

しかし、この工程は軽視される事も多い[1]。無理なスケジュールに合わせるために要件定義が疎かにされ、結果として不良のある要件を基に開発が行われ、後から修正しなくてはならない、といった事も起こる。要件の不良の修正コストは、開発の初期であれば最も安いが、製品出荷後になると最も高い。また、要件定義の段階で要件が抜け落ちてしまう、といった事も起こる。この場合は要件のエラーの中で最も発見や修正が難しい事態となる。

要件に不良のある場合以外に、要件をなかなか凍結できないといった問題も起こる[1]。それはユーザーや顧客がどんな問題を解決したいのかよく分かっていないことが原因である最初は分かっていたつもりでも、プロジェクトが進むに従って実はわかっていないことが判明する。もしくは、最初から何も分かっていない。やるべき事が明らかな問題でもそれを解決するソフトウェアを作るのは難しいが、要件がどんどん変わる場合にはソフトウェアは開発できない。そのようなプロジェクトは途中で頓挫する。そのような場合は要件が十分に理解されていない可能性も大きい。これに対して昔は要件を変更してはならないという主張があったが、期待通りの結果を得られなかったため、今はほとんどない。現在でも、要件がなかなか決まらない問題を解決するため、色々なアプローチが行われている。

要件定義の難しさは、ソフトウェア開発の本質に根ざす物である[2]。全てのソフトウェア開発では本質的な作業として、抽象的な概念構造体を作り上げる事が含まれている。そして、その概念構造体の仕様作成、設計、テストこそ最も難しい事であるいう主張がある。本当に難しいのはプログラミング言語による表現では無い。ソフトウェアのコンセプト上の誤りに比べると、構文上の誤りはまだ軽症である。なので、「何を構築するかの決定」つまり要件定義は困難となる。ソフトウェアの製作者が顧客のために行う最も重要な仕事は、顧客の要件を繰り返し抽出し、洗練していく事となる。実の所、顧客自身も何を必要としているのか分かっていない。

よって、要件定義を効果的に、かつ問題の本質的な部分を攻略する形で行う事がソフトウェアの開発を成功させる第一歩となる。そのために、特定の言語や手法を離れ、問題そのものに視点を当てる事を考えなければならない。

1.2 研究の目的

本研究の前提として、要件定義における有効な手段を探るといった事を念頭に置く。そのために、Michael Jackson の提唱する「問題フレーム」を使用する[3]。問題フレームでは解決しなければならない問題の性質を明らかにするための記述を行う。

具体的な研究方法としては、まず要件定義の一環として実際に問題フレームを利用する。そのために、企業の技術報告に掲載されている既存のシステム 2 例を参考とし、ケーススタディとして問題フレームの図を作成する。また、問題フレームの構造を理解し易くするために、問題フレームの構造をメタモデル化する。その後、問題フレームを使う事の有用性を評価し、そして改善案の提案を行う。

1.3 論文の構成

次の 2 章で、問題フレームの下敷きとなっている要件および仕様についての解釈の仕方を述べる。3 章では、問題フレームについての説明を行う。4 章では、実際に事例ごとの問題フレームの図を作成し、ケーススタディを行う。5 章では、問題フレームのメタモデルの作成を行う。6 章では、問題フレームの評価を行い、利用する事のメリットや改善が考えられる箇所について述べる。7 章では、問題フレームの改善案を提案する。8 章では、まとめと将来の展望を述べる。

2　要件および仕様について

要件、および仕様については人によって様々な解釈があり、それがソフトウェア開発の混乱の一因ともなっている。ここでは、問題フレームの考え方の下敷きとして、問題フレームの提案者である Michael Jackson による解釈に沿ってその意味を述べる。

2.1 要件とは何か

Michel Jackson は要件について以下のように述べている[4]。

古くからソフテウェア開発においてはソフトウェアすなわち「機械」が影響を及ぼす「世界の一部」である「問題領域」（もしくは「適用領域」）を無視してきた。そして、専ら機械のみに注意を向けてきた。そのために、要件をうまく決められなくなっていた。「顧客は自分が何を欲しいかわかっていない」というのは、開発者の驕りである。

要件は問題領域の現象であり、機械に関するものではない。要件を正確に述べるためには、問題文脈の現象間に成り立つべき関係を記述しなければならない。

機械が要件を満たすことができるのは、それが問題領域といくつかの事象や状態を共有するからである。共有される事象は、機械と問題領域のどちらからでも観測可能となる。

しかし、問題文脈の全ての現象が機械と共有されるわけではなく、顧客の提示する要件と機械が直接達成できることの間には乖離が生まれる。それは、要件は機械と共有される現象のみに制限されるわけではないからである。要件は問題領域のみに関わる現象、および問題領域と機械の両方に関わる現象について記述されるので、機械と共有されない現象をふくんでいるかもしれない。しかし、プログラムは最終的に機械のみに関わる現象、および問題領域と機械の両方に関わる現象について書かれる。

最終的にプログラムが要件を満たしていると主張するには、以下のような推論を行う必要がある。

- コンピューターが性質 C をもち、プログラム P に書かれたようにコンピューターが動けば、仕様 S が満たされる。この様子を図 1 に示す。
- 問題領域が性質 D をもち、仕様 S が満たされれば、要件 R が満たされる。この様子を図 2 に示す。

D という性質は機械が何をしようが、何ができなかろうが、それとは無関係に問題領域が保有する性質である。

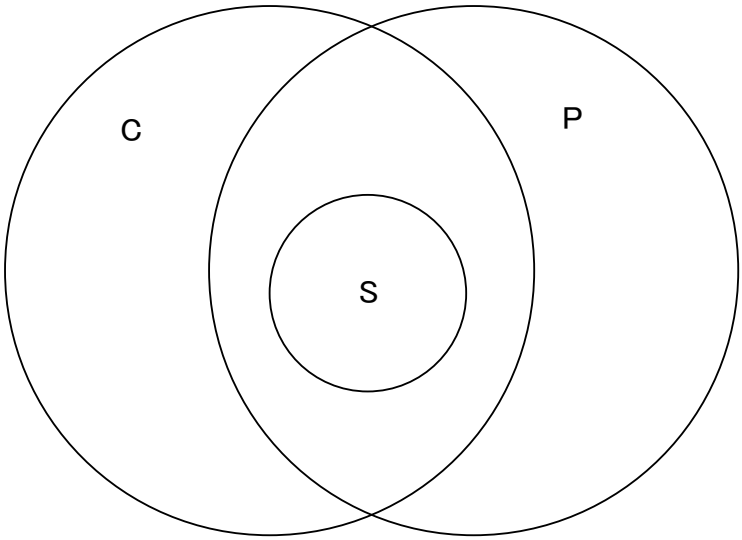


図 1　仕様が満たされるために必要な事

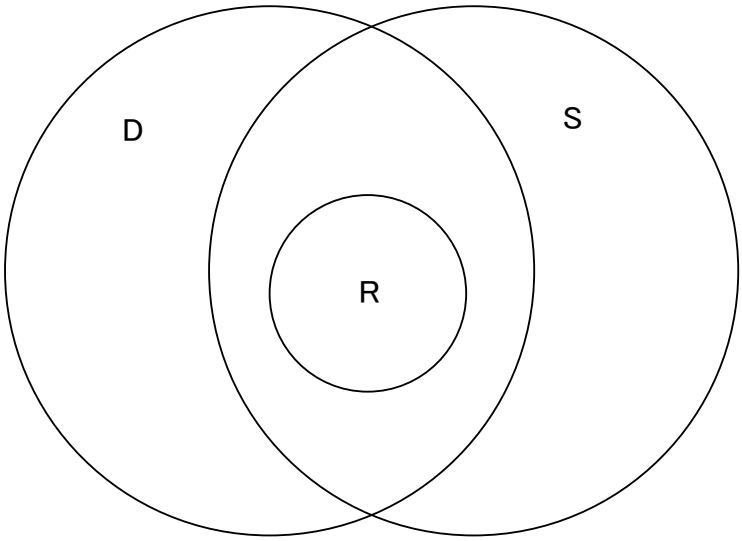


図 2　要件が満たされるために必要な事

以上のような事から、機械を望むように動かし要件を満たすためには、問題領域の性質を理解する事が重要である。

2.2 仕様とは何か

Michel Jackson は仕様について以下のように述べている[4]。

「仕様」と言う言葉の使い方はソフトウェア開発の混沌状態を反映して混沌としており、人によって使い方が大きく違う。ここではより限定的に使い、要件、仕様、プログラムという 3 つ一組の用語の一つとする。

機械は環境と相互作用する。これは、事象や状態という現象を機械と環境の間で共有する事により成り立つ。このような共有された事象や状態が機械と環境のインターフェースを作る。これが仕様インターフェースである。仕様とはインターフェース上の共有された現象のみについて語るものである。

一方、要件は環境現象についてのみ語るものである。システムの顧客は環境に関心があるのであって、機械には関心がない。顧客の関心がたまたま仕様インターフェース上の共有現象に及ぶことはあるが、要件に関する限り、それはあくまでも偶然のことである。

またプログラムは機械現象についてのみ語るものである。プログラムは機械の動作に関わる。プログラマは機械の振る舞いについてのみ関心がある。その関心の中には仕様インターフェース上の共有現象に関するものも当然あるが、これは機械の振る舞いをよりよく知りたいという欲求からのみ起こるものである。

プログラム、仕様、要件をこのように見ると、全ての仕様は形式的な意味で要件でもありプログラムでもある。環境現象について語るので要件であり、機械現象について語るのでプログラムとなる。なので、仕様は要件とプログラムの間の橋となる。

仕様は理解するのが難しいものである。顧客の要件から多くの推論段階を経て得られるものなので、環境における意味が見えにくいかもしれない。また、仕様だけを見て機械のどのような内部的振る舞いが仕様に記述された外部的振る舞いをもたらすかを知る事も難しい。

また、仕様の記述をできるだけ抽象化しようとする傾向があるが、仕様インターフェース上の現象は具体的なものなので、仕様を完全に抽象的なものとすることはできない。誤った場所で過大な抽象化をした場合、仕様は顧客の現実的な問題に答えるものではなくなってしまう。

さらに、仕様は機械と環境の両方に位置するものであるため、それが何を述べているかについて、範囲の問題から来る混乱が起こる事がある。これは、機械が環境の一部の「モデル」となり、環境の一部の記述が同時に機械の一部の記述となりうるために、さらに複雑になる。因果関係と制御についての考慮を全て捨象した仕様の記述を行うと、機械と問題領域間の相互作用について、混乱が起こる。

3　問題フレームについて

この章では、問題フレームについて説明する。問題フレームの背景にある Michael Jackson の考え方や問題フレームの特徴、そして問題フレームの図の読み方について説明する。

3.1 問題フレームの背景

2 章で述べたように、要件とは問題領域に関することであり、機械だけに目を向けていては要件定義はうまくできない。要件を満たすためには、問題そのものについて考え、問題領域の性質を理解しなければならない。

しかし、ソフトウェア開発において問題に焦点を当てることは難しい。その理由の一つは、ソフトウェア開発においては規則性が求められる視点と規則性の無い視点が混在するからである。規則性が求められる視点とはプログラミング等に関する事であり、規則性の無い視点とは社会学的、民族学的といった人間の事である。問題の理解や分析のためには、これら両方を公平に扱う事が必要となる。そのためには、ある面では規則的、ある面では非規則的な記述を行わなければならない。ソフトウェア開発の問題はシステムが影響を及ぼす現実世界に関する事なので、問題の分析における規則性も非規則性も直面している世界がもたらす物である。可能であれば規則的な記述を行う事は重要である。しかし、物理的な世界を忠実または適切な形で規則的に記述する事は不可能な場合もある。なので、計算不可能な現実世界の複雑性は感性的、直接的に取り扱うべきである。

問題に焦点を当てることの難しい別の理由としては、システムと周りの環境、そしてその関係は全て永続的な物だと考えられがちだという事がある。問題は常に変化していく物なのだが、ある時点での問題のスナップショットである一過性の記述しか行われない、といった事がよくある。そういった物にはソフトウェア開発に必要な「要件」が存在しないし、あっても使い物にならない。問題が流動的であっても、要件をしっかり掴み、問題を分析し構築する事が、正しいシステムの完成へと繋がる。

3.2 問題フレームの特徴

問題フレームでは、ソフトウェアによって解決されなければならない問題を図で記述し、それにより問題の性質の理解を助ける。図はソフトウェアそのものである機械領域、ソフトウェアに関連する問題領域、各問題の要件、そしてそれらの相互関係によって成り立つ。問題フレームで現実の複雑な問題を記述する場合、最初に文脈図として全ての問題を網羅する図を記述した後、整理しやすい大きさの小さくて単純な副問題へと分割する。そして、分割された小さな問題は「フレーム」というパターンに当てはめ、問題図が描かれる。フレームに当てはめることにより問題がより理解しやすくなり、各問題の解決策にも結びつけ易くなっている。また問題を完全な階層的な分割ではなく、並列的な分割を許しているので、

より自然な形で問題を整理できる
図 3 に問題図の例を示す。後述する無人ヘリコプターによる自動薬散システムの、ヘリコプターの高度誘導の問題である。

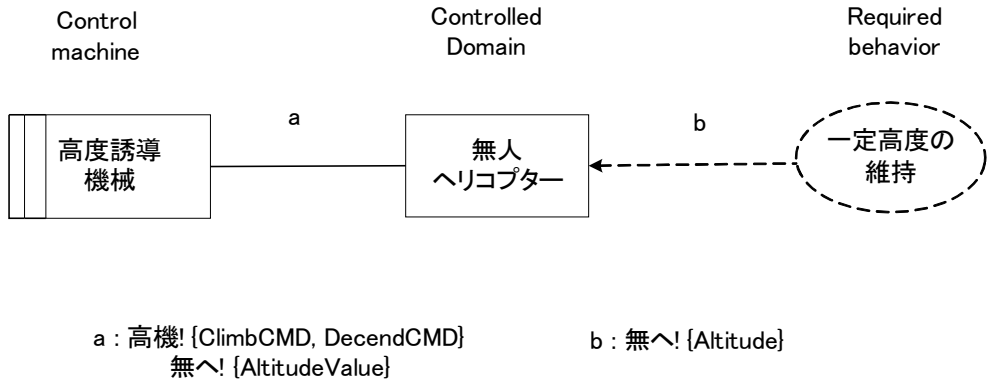


図 3　問題フレームの例：高度誘導の問題（required behavior frame）

問題フレームでは、ソフトウェア開発における特定の手法を使う事を強いる事はしない。一つの理由は、違ったフレームの問題には違った手法が有効だからである。全ての問題に有効な手法は存在しない。またそういった手法があったとしても、ある特定の問題に対しては大した効果が望めない。

もう一つの理由は、今日使われている多くの手法は解法指向だからである。こういった手法は特定の問題の解法を作り出すのには有効だが、問題の分析と構築には助けにならなかったり、むしろ妨げになったりする。また、アナリストが特定の解法を使う事を仮定すると、要件は過剰に詳細になってしまう場合がある[7]。

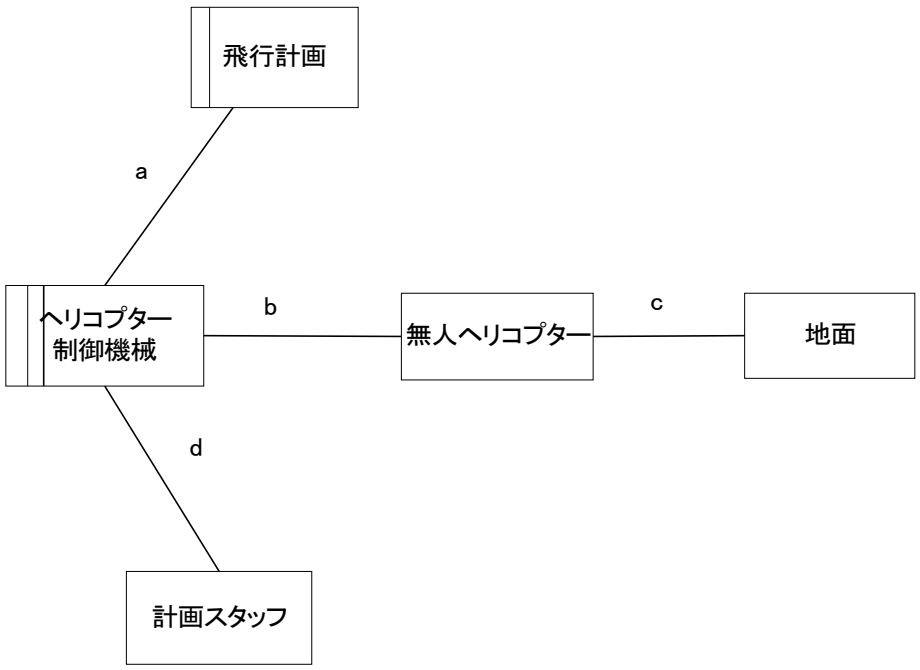
問題フレームは、問題そのものの性質を明らかにする。そして、「方法は自由だが、やらなければならない事」に対する入り口を与える。または既存のソフトウェアの拡張や修正をすべき箇所の発見に繋がる。

3.3 問題フレームにおける図の構成要素

この節では、問題フレームの図の読み方について説明する。

3.3.1文脈図の構成要素

図 4 に文脈図の例を示す。これは後述する無人ヘリコプターによる自動薬散システムの全体文脈図である。文脈図では、問題領域や機械領域の相互関係を図示する。特に、全てのシステムに一つずつ、全体文脈図として全ての範囲の問題を網羅する文脈図を描く。それ以外にも、必要に応じて、ある問題領域の内部を説明するために詳細な文脈図を描く事もある。



a : FlightPlanParam, ExceptedArea, etc

c : Apply

b : Commands, Sensors

d : EnterFlightPlan,
EnterExpectedFiled

図 4　全体文脈図の例：無人ヘリコプターによる自動薬散システム

- 図 4における「ヘリコプター制御機械」のように、左端に線が二本入っている四角形のシンボルは「機械領域」、つまり開発されるべきソフトウェアを表している。機械領域は全ての文脈図に一つずつ存在する。
- 図 4における「飛行計画」のように線が一本入っている四角形は機械領域が影響を及ぼす、または機械領域に影響を与える世界の一部である「問題領域」を表す。この場合は開発者によって設計されるべき領域である。具体的にはファイルやデータベースの場合が多い。
- 図 4における「無人ヘリコプター」等のように線の入っていない四角形も問題領域を表すが、このシンボルの場合は開発者による設計の必要が無い。つまり既存の物として与えられている問題領域を表している。
- 領域間を接続している線は、共有現象のインターフェースである。インターフェースに付いている識別子は下部の文字表現と対応している。下部の文字表現は対応するインターフェースの現象を大まかに示した物である。なお、図 5のように、3つ以上の領域を「ハイパーアーク接続」によって接続する事も可能である。

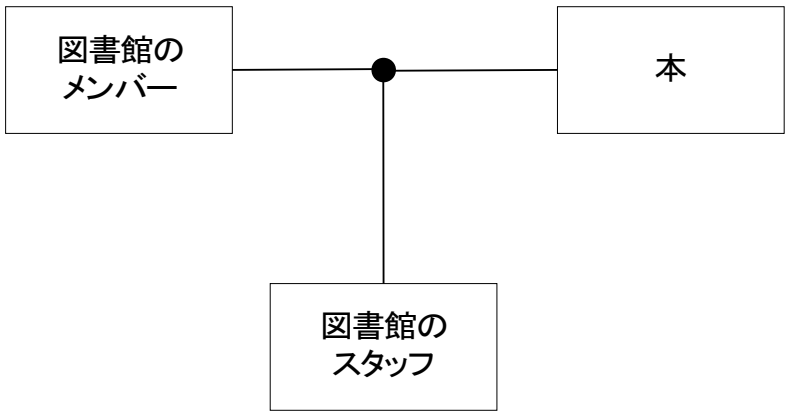
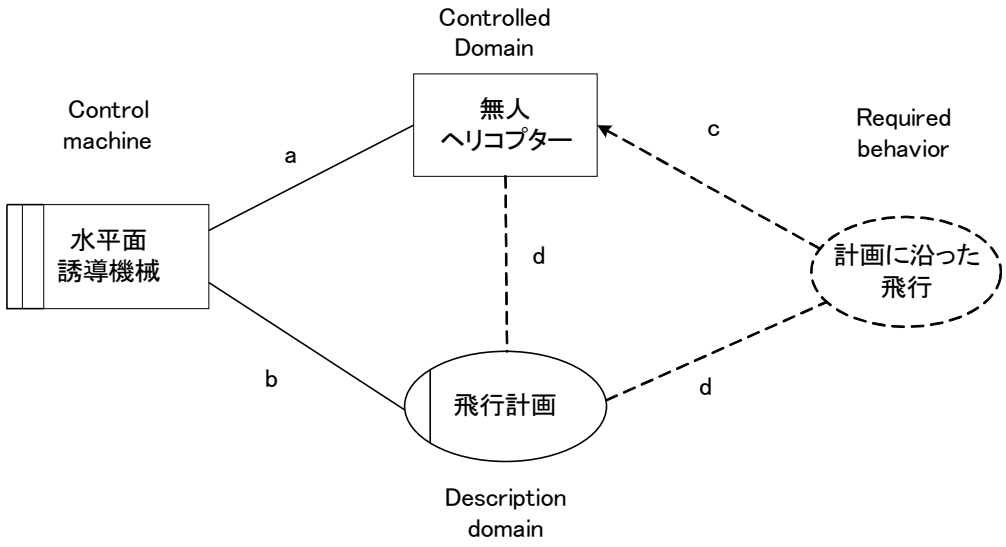


図 5　ハイパーアーク接続の例

3.3.2問題図の構成要素

問題図では文脈図の記法を継承しており、領域の表現について文脈図と共通の表現をしている。しかし、問題図独自の表現もあるので、ここではそれについて説明する。図 6 に記述領域を含んでいる問題図の例を示す。これは後述する無人ヘリコプターによる自動薬散システムの水平面誘導の問題である。なお、問題図でも機械領域は全ての図に一つずつとなる。



a : 水機! {AdvanceCMD, RotateCMD, HoverCMD}
無へ! {HorizontalPosStatus, DirectionStatus}

c : 無へ! {HorizonalPosition, Direction}

b : 飛計!{PointParameter, RDirectionParameter, HDirectionParameter, FormParameter, RadiusParameter, TimeParameter}

d: 飛計!{FlightForm, TargetPoint, RotatingDirection, RotatingRadius, HoverringTime, HoverringDirection}

図 6　問題フレームの例：水平面誘導の問題（required behavior の description variant）

- 図 6 における「計画に沿った飛行」のように、破線の楕円のシンボルは問題の「要件」を表す。要件も各問題図に一つずつである。要件は機械領域によって一つ以上の問題領域にもたらされなければならない状態を表している。
- 要件と領域を接続する破線は「要件の参照」である。ある領域のある現象に対し、要件が関係しているという事を表す。
- 先端が矢印になっている要件の参照は矢印の無い通常の物とは違い、領域の現象に対し制限を課す物である。機械はこの領域の状態や動作が要件を満たすことを保証しなければならない。
- 下部の文字表現は対応するインターフェースの現象を詳細に示した物である。図 4 のインターフェース a の「無へ!」等、領域名の 2 文字の略称に「!」を付けた表現は、どちらの領域がその現象を制御しているか、といった事を表すものである。この場合は「無人ヘリコプター」が「HorizontalPosStatus」、「DirectionStatus」といった現象を制御し

ている。

- 図 6 の「Control machine」等、領域や要件に付随している文字表現は、その問題が当てはめられている問題フレームにおける役割を表している。
- 図 6 の「飛行計画」のように実線の楕円に一本の線が加わっているシンボルは、「物理的に表現された記述領域」を表している。この場合は特に、開発者によって設計されなければならない記述領域となる。記述領域とは、後述する「Description variant」で使うものである。基本的な問題フレームの要素以外に加え、参考とするファイル等の領域を入れることによって問題図を基本形から変形させる物である。
- 図 7 のように左端に線のない実線の楕円のシンボルも記述領域を表す物だが、こちらは開発者によって設計されるものではなく、既存の物として与えられた領域である。
- 図 6 の d のように、記述領域と問題領域を接続している破線は「記述の参照」となる。その問題領域の現象に対して記述の側が参照しているという事である。この場合は、「飛行計画」が「無人ヘリコプター」の「FlightForm」等の現象を知っている事を表す。

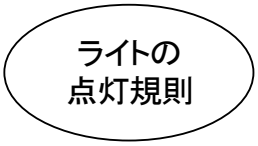


図 7　物理的な記述の例

3.4 基本的な問題フレーム(basic frame)

ここでは、5 つの基本的な問題フレームについて説明する。

3.4.1Required behavior フレーム

Required behavior フレームのフレーム図を図 8 に示す。

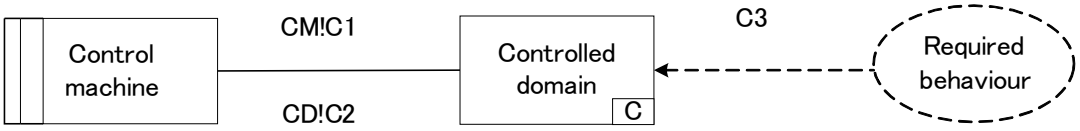


図 8　Required behavior のフレーム図

ここで「Controlled domain」に付いている「C」というシンボルは、「Causal domain」である事を示す。「Causal domain」はモーター等の動作が予測可能な領域であり、他の領域によって制御される事もあれば、他の領域を制御する場合もある。

Required behavior フレームでは、自動的な制御を行う問題を取り扱う。「Control

machine」によって制御される現象「C1」によって「Controlled domain」は影響を受ける。「Controlled domain」からの入力である「C2」の現象によってフィードバックを受ける場合もある。「C3」は最終的に制御されるべき現象であり、「C1」や「C2」とは違う場合もある。「C1」等の「C」の付いた現象は「Causal phenomena」であり、他の現象に影響を与えたり、他の現象の影響を受けたりする現象である。

Required behavior フレームに当てはまる問題としては、信号機の制御等が考えられる。

3.4.2Commanded behavior フレーム

Commanded behavior フレームのフレーム図を図 9 に示す。

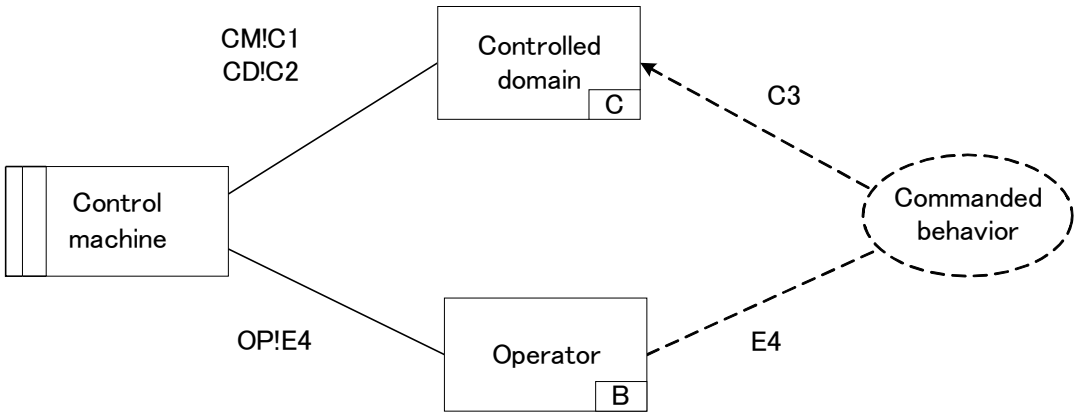


図 9　Commanded behavior フレームのフレーム図

ここで「Operator」に付いている「B」というシンボルは、「Biddable domain」である事を示す。「Biddable domain」は人間等、動作が予測できない領域を表す。

Commanded behavior フレームでは人間等の入力するコマンドによる制御を行う問題を取り扱う。ちょうど Required behavior に「Operator」を追加した形となる。「Operator」の発するコマンドである「E4」を入力とし、「Controlled domain」の制御を行う。「E4」のような「E」の付いた現象は「Event phenomena」であり、ある時点で起きるイベントを表す。

Commanded behavior フレームに当てはまる問題としては、オペレーターによるゲートの制御等が考えられる。

3.4.3Information display フレーム

Information display フレームのフレーム図を図 10 に示す。

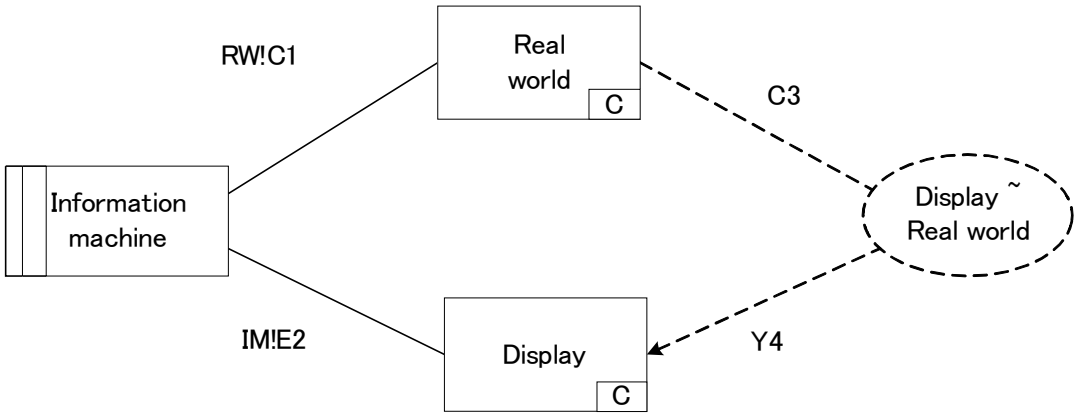


図 10　Information display フレームのフレーム図

Information display フレームでは、データ等の表示に関する問題を取り扱う。「Information machine」は「Real world」つまり現実世界の状態を表す現象である「C1」に応じて「E2」のイベントを起こし、「Display」に表示を行う。そして、表示されている「Y4」は現実世界「C3」に対応していなければならない。また、どの現象も「Real world」に対しては操作を行えない。「Y4」等の「Y」の付いた現象は「Symbolic phenomena」であり、他の現象や現象間の関係を値としてシンボル化した物である。

Information display フレームに当てはまる問題としては、車のスピードの表示等が考えられる。

3.4.4Simple workpieces フレーム

Simple workpieces フレームのフレーム図を図 11 に示す。

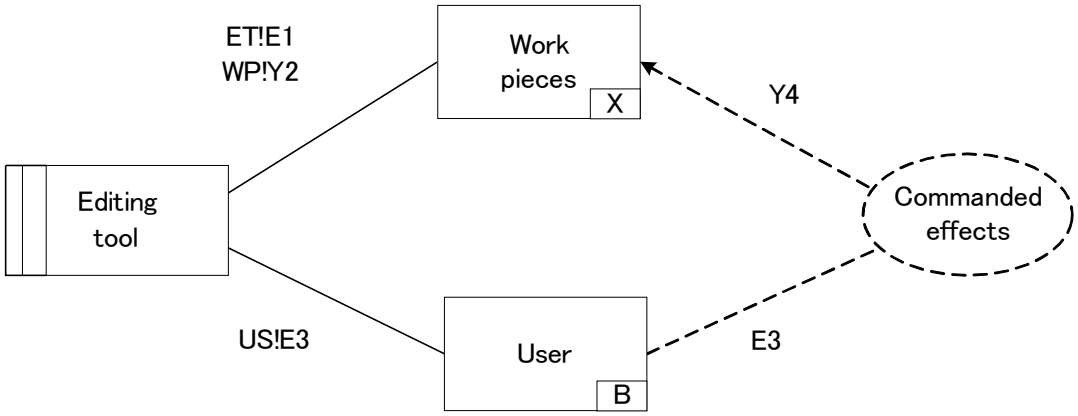


図 11　Simple workpieces フレームのフレーム図

ここで「Workpieces」に付いている「X」というシンボルは、「Lexical domain」である事を示す。「Lexical domain」はファイルやデータベース等といった、データの物理的な表現であ

る。

Simple workpieces フレームでは、ファイル等の編集に関する問題を取り扱う。「Editing tool」は「User」からのコマンドである「E3」を受け取る。そして「Workpieces」の状態である「Y2」を読み取り、「E3」と「Y2」を基にオペレーション「E1」を発し、「Workpieces」の編集を行う。「Y4」は最終的に制御されるべき現象であり、人間にとって意味のある物である場合が多い。

Simple workpieces フレームに当てはまる問題としては、テキストエディタ等が考えられる。

3.4.5Transformation フレーム

Transformation フレームのフレーム図を図 12 に示す。

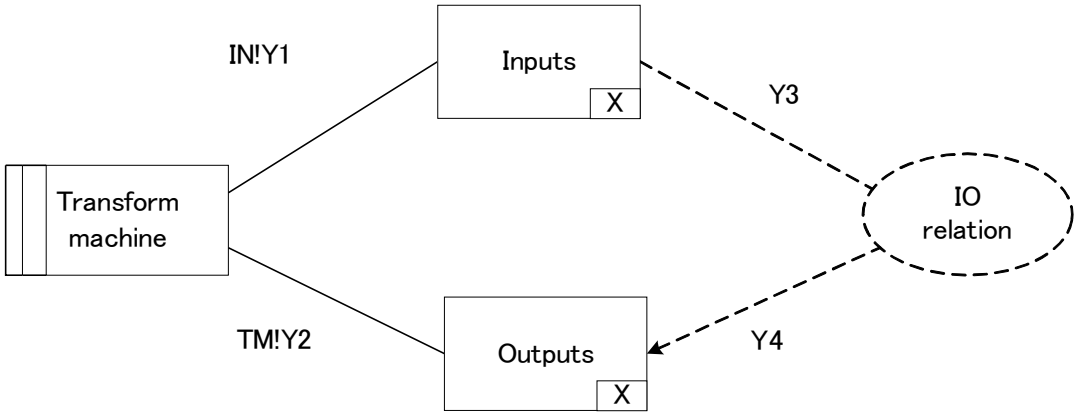


図 12　Transformation フレームのフレーム図

Transformation フレームでは、データの変換を行う問題を取り扱う。「Transform machine」は「Inputs」からの入力である「Y1」を基に「Y2」を生成し、「Outputs」へと書き込む。「Y3」は「Y1」と同じ場合もあるが、違う場合もある。同様に「Y4」も「Y2」と同じ場合もあるが、違う場合もある。これらの現象は、一般的には違った物の場合が多い。機械の側の現象はより基本的な「文字」等の場合が多いし、要件の側はより大きな「レコード」等の場合が多い。

Transformation フレームに当てはまる問題としては、コンパイラ等が考えられる。

3.5 その他の問題フレーム

ここでは、基本的な問題フレームから派生した問題フレームについて説明する。基本的なフレームから変化した物を Variant と言う。まず各 Variant について説明する。そして、通常の Variant は違った形の派生である Model building subproblem フレームについて説明する。

3.5.1フレームの Variant

現実の問題は複雑なため、単純な問題へと分割した状態でも、基本的な問題フレームだけでは問題を表現しきれない事がある。そういった場合には、 Variant frames を利用する。Variant は基本的な問題フレームを拡張し、基本的な問題フレームに適合しない問題を扱えるようにする。複数の Variant を同じ問題フレームに対して適用する事も可能である。Michael Jackson の挙げている Variant には以下のようなものがある。

- Description variant
  - 問題図に「Description」つまり記述の領域を導入することによって問題図を派生させる。記述領域はファイル等として現れ、必要な情報を機械領域に提供する。例としては、図 6 の水平面誘導の問題等が挙げられる。その場合は、「飛行計画」が、Description domain となる。
- Operator variant
  - 問題図に「Operator」の領域を導入する。「Operator」は、機械領域にコマンドを入力する人間等の表現である。 Commanded behavior フレームは Required behavior フレームの Operator Variant と考える事もできる。
- Connection variant
  - 問題図に「Connection domain」を導入する。Connection domain は問題図の中心的な問題領域と機械領域の間に入り、両者を接続する。Connection domain を入れないと問題の性質が表現できない場合に適用する。または、Connection domain の信頼性が完全でなく、故障する場合を考えなければならない場合も利用する。図 13 に例となる図を示す。この場合は「PLC」が Connection domain となる。

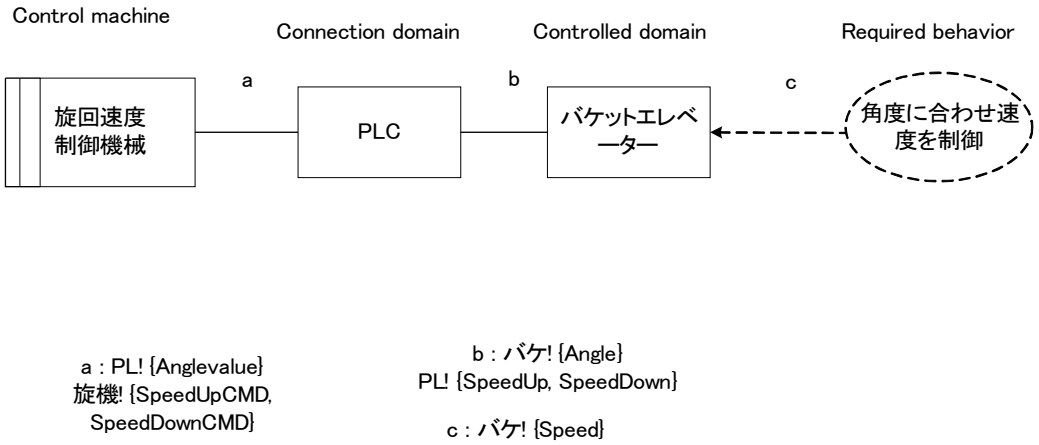
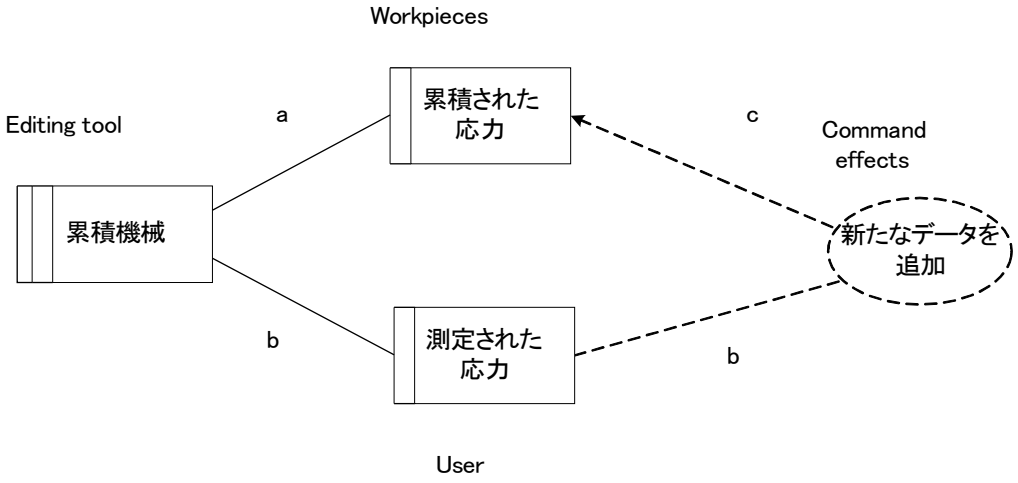


図 13　BE 旋回速度制御システム（required behavior frame の connection variant）

- Control Variant
  - Control variant の場合は新たな領域を導入するのではない。基本的なフレームのインターフェースに記述されている、制御上の性質を変化させる事によって基本的なフレームから派生する。図 14 に例となる図を示す。本来の Simple workpieces frame では User が制御の主体となり、User からのコマンドを受けて Editing tool が動作する。しかし、この場合は Editing tool が主体となり、User から能動的に値を読み込んでいる。なお、Required behavior、Commanded behavior、Information display には Control variant が存在しない。



a：累機! {AddData}　　c：累応! {CumulatedData}

b：累機! {ReadData}
測応! {StressData}

図 14　応力の累積的記録（Simple workpieces frame の control variant）

3.5.2Model building subproblem フレーム

Model building subproblem フレームでは、現実の世界を基にコンピューターの中にモデルを作成する。ここで言う「モデル」とはドキュメントの記述等の事ではなく、現実世界のような他の領域を擬似的にデータで表現した物の事である。Model building subproblem フレームは Information display フレームの Variant とも言えるが、独立に問題フレーム図が定義されており、また問題図を描くときも他の Variant とは違い、独立した一つの問題図となる。なので、他の Variant とは違った扱いとなる。

Model building subproblem フレームのフレーム図を図 15 に示す。

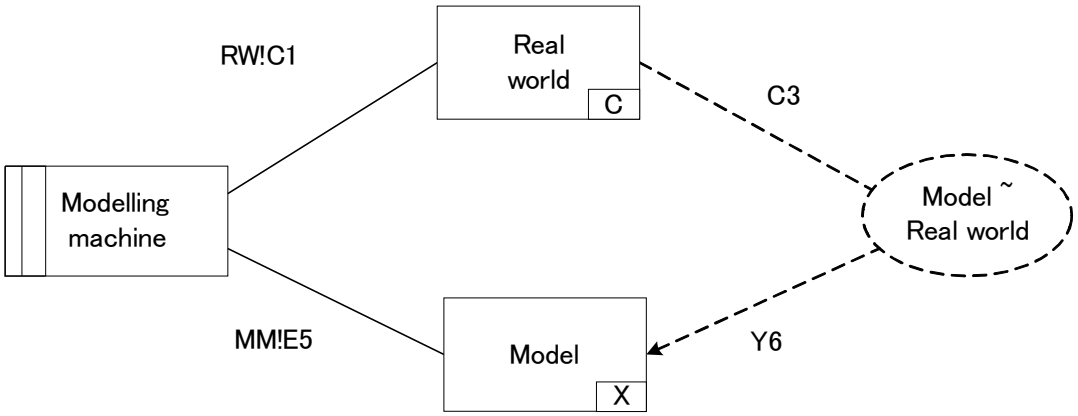


図 15　Model building subproblem フレームのフレーム図

「Modelling machine」は「Real world」を観測し、入力である「C1」を基に「E5」のイベントを発し、「Model」を作成する。現実世界の現象である「C3」に、モデル表現の現象である「Y6」は対応する。「E5」と「Y6」の関係を定義するのはモデルの設計者の責任である。「C1」と「E5」の関係を定義するのは機械領域の仕様を定義する者の責任である。「C3」と「C1」の関係は「Real world」の領域の性質によって支配される。

3.6 問題フレームの記述の際に必要な知識

問題フレームを利用する際、前提として知っておかなければならない事を以下に記す。

まず、問題の範囲を知っていなければならない。文脈図や問題図では、問題の範囲によって特定の領域を図に入れるか否かを決めるので、ソフトウェアの責任者が関知する範囲を知っておかないと、図の境界が決められない。

また、図に入れる各領域についても、特性を調査し、知っておかなければならない。それを知っておかないとインターフェースの記述ができない場合もある。また問題の境界内において無視する領域とそうでない領域を決める際も、領域の信頼性等の領域の特性の知識を必要とする場合がある。なので、領域の特性が分からない場合はそれを調査しなければならない。

3.7 問題フレームと UML の違いについて

問題領域の記述と UML には類似点があるが、両者には大きな違いがある。

UML はオブジェクト指向に基づく記法だが、問題フレームにおける図は問題領域に対して焦点を当てているのであって、オブジェクト指向に関連するわけではない。問題フレームは特定の手法や解法に繋がってはいない。図によっては UML の図と偶然似る可能性もあるが、その事については本研究の対象外である。

また、Michael Jackson は次のように述べている。UML の図におけるインターフェースはプログラミング言語における関数呼び出しのようにサービスを呼び出す事を意識した物

であるため非対称的である。ユーザーがあるサーバーのオペレーションを呼び出し、サーバーはまた別のサーバーのオペレーションを呼び出すかもしれない。しかし、あるインターフェースの一方は呼び出すだけ、もう一方はそれを受けるだけといった形は常に変わらない。

これに対し、問題フレームの図は対称的である。全ての領域に共有現象を制御する可能性がある。例えばあるインターフェースで繋がる 2 つの問題領域のうち一方はあるイベントを制御し、もう一方はある状態を制御する、といった状況も起こりうる。

4　問題フレームを使った要件定義のケーススタディ

この章では、実際に問題フレームを使い、要件定義のケーススタディを行う。事例として企業の技報に掲載されている論文2つを選び、そこで紹介されている製品に必要なソフトウェアの問題フレームの図の記述を行う。

4.1 薬剤散布ヘリコプター

SUBARU TECHNICAL REVIEW 2005 No.32 に掲載されている「無人ヘリコプタ RPH2 による全自動薬散システム」を資料とし、問題フレームの図を記述した[8]。問題は文脈図2つと問題図9つで表す事ができた。問題図は全て問題フレームに当てはめた。付録 A.1 に薬剤散布ヘリコプターの図を掲載する。

資料と図の関係を以下に示す。

4.1.1文脈図について

全体文脈図：資料全体から作成

無人ヘリコプターによる薬剤の散布についての問題なので、「無人ヘリコプター」の領域と散布対象の「地面」の領域を入れる。また、「4.1 自律飛行制御則」の「予め設定された飛行計画にしたがって自動飛行をする」との記述から、「飛行計画」の領域と、飛行計画を入力する「計画スタッフ」の領域を入れる事とする。インターフェース a の「FlightPlanParam」の詳細は「4.1 自律飛行制御則」の各種パラメーターであり、「ExpectedArea」は「4.5 散布吐出量制御則」に記述されている散布除外地の入力についてである。

無人ヘリコプターの文脈図：「Fig3 Structure of autonomous aerial application system」、「4.1 自律飛行制御則」から作成

図3の「RTK-GPS 受信機」と「GPS アンテナ」から「GPS」の領域、「ステレオカメラ高度計」から「高度計」の領域、「飛行制御装置」から「飛行制御装置」の領域、「薬散装置」から「薬散装置」の領域を入れる事とする。また「4.1 自律飛行制御則」の「目標飛行速度」という記述から、現在の速度を知る必要性がある事が分かるので、「速度計」の領域を入れる。

4.1.2問題図について

散布精度コントローラー、散布量計算：　「2. 要求仕様」、「3.3 散布装置」、「4.5 散布吐出量制御則」から、この2つの副問題が必要なことが分かる。

散布精度の制御の問題を考える際、問題フレームに当てはめられるような単純な問題にするためには2つの副問題に分割することが必要となる。問題は「散布量計算」と「散布精度

コントローラー」へと分割できる。前者は現在の速度から適正な量を計算するので Transformation フレームに当てはめられ、後者は計算された量を基に散布の精度を制御するので Required behavior フレームに当てはめられる。

「散布量計算」では、「4.5 散布吐出量制御則」の「前進速度から散布量を制御する制御則を設計した」という記述から、Inputs として「速度計」領域、Outputs として「適正量」領域を入れる事とする。

「散布精度コントローラー」では、「3.3 散布装置」の「指令値かに応じてシャッタを制御し散布量を調整する機能を持つ薬散装置を搭載した」との記述から「薬散装置」領域と散布対象である「地面」領域を入れることとした。この時「地面」は Controlled domain とし、「薬散装置」は Connection domain とする。また「2. 要求仕様」の「速度に応じた薬剤吐出量制御を行うこと」という記述、「3.3 散布装置」と「4.5 散布吐出量制御則」の同様の記述から「適正量」領域を Description domain として入れる事とする。

散布地域の制御： 「3.3 散布装置」、「4.5 散布吐出量制御則」、「Fig.4 Flight plan」からこの副問題が必要な事が分かる。

現在位置によって散布を行うか否かを制御するので、Required behavior フレームに当てはめる事とする。

「3.3 散布装置」の「飛行速度や機体の位置に応じて、散布量を制御」や「4.5 散布吐出量制御則」の「散布除外地では散布を停止する」から、「薬散装置」と「GPS」領域が必要となるが、問題フレームに当てはめやすくするためにこの 2 つは 1 つの領域とする。「薬散装置+GPS」領域は Connection domain とし、散布対象である「地面」領域を Controlled domain とする。また、「4.5 散布吐出量制御則」の「あらかじめ設定する飛行計画と共に、散布除外地を定める」との記述から、「飛行計画」領域を Description domain として入れる。ここでは散布除外地は飛行計画の一部と考えている。また「4.5 散布吐出量制御則」の「散布除外地の形状は、任意の多角形で設定可能である」との記述を反映し、インターフェース d では飛行計画から機械領域への入力は多角形として行うこととしている。

飛行計画の入力： 「Fig.4 Flight plan」「4.1 自律飛行制御則」「4.5 散布吐出量制御則」からこの副問題が必要な事が分かる。

人間による飛行計画の入力なので、Simple workpieces フレームに当てはめることとする。

とくに記述は無いが入力する人間が必要なので User として「計画スタッフ」領域を入れ、「飛行計画」領域は Workpieces とする。また、入力のコマンドであるインターフェース b や入力される内容である要件の参照 c の内容は、「4.1 自律飛行制御則」の「飛行計画については、飛行形式（前進またはホバー）、目標位置、目標飛行速度、旋回方向、旋回半径、ホバー時間、ホバー方位をパラメーターとして構成し」との記述、「4.5 散布吐出量制御則」の「あらかじめ設定する飛行計画と共に、散布除外地を定める」との記述から決めている。

速度制御：　「4.1 自律飛行制御則」からこの副問題が必要な事が分かる。

目標速度を維持する問題なので、Required behavior フレームに当てはめる事とする。

無人ヘリコプターの速度の維持なので、Controlled domain として「無人ヘリコプター」領域を入れる。「速度計」は、「無人ヘリコプター」に含まれている。また、「4.1 自律飛行制御則」の「飛行計画については、(中略)目標飛行速度(中略)をパラメーターとして構成し」との記述から、「飛行計画」領域を Description domain として入れる。「飛行計画」から機会領域への入力であるインターフェース b は目標飛行速度になる。

高度誘導：　「3.1 位置センサ」、「4.4 高度誘導」からこの副問題が必要な事が分かる。

一定の高度を維持する問題なので、Required behavior フレームに当てはめる事とする。

無人ヘリコプターの高度の維持なので、Controlled domain として「無人ヘリコプター」領域を入れる。「高度計」は、「無人ヘリコプター」に含まれている。インターフェース a において、「無人ヘリコプター」から発生する共有現象は高度の値、機械領域から発生する共有現象は高度にまつわるコマンドとなる。

離陸：　「2. 要求仕様」「Fig.4 Flight plan」、「4.1 自律飛行制御則」からこの副問題が必要な事が分かる。

任意の位置から離陸する問題なので、Required behavior フレームに当てはめる事とする。

無人ヘリコプターの離陸なので、「無人ヘリコプター」領域を Controlled domain として入れる。また、離陸に費用名情報として高度があるので、インターフェース a における機械領域への入力として高度の値を入れる。「2. 要求仕様」の「設定した任意の場所に離着陸すること」等の記述があるが、離陸時には目標位置等の情報は必要ないため、位置に関連する領域やインターフェースは入れていない。

着陸：　「2. 要求仕様」「Fig.4 Flight plan」、「4.1 自律飛行制御則」からこの副問題が必要な事が分かる。

任意の位置へ着陸する問題なので、Required behavior フレームに当てはめる事とする。

無人ヘリコプターの着陸なので、「無人ヘリコプター」領域を Controlled domain として入れる。また、「2. 要求仕様」の「設定した任意の場所に離着陸すること」等の記述から、着陸地点を設定しておく必要があるので、「飛行計画」領域を Description domain として入れる。インターフェース b での飛行計画から機械領域への入力は着陸地点や着陸時の方位に関する物となる。インターフェース a での無人ヘリコプターから機械領域への入力は実際の高度、位置、方位の値となる。

水平面誘導：　「3.1 位置センサ」、「4.3 水平面誘導」からこの副問題が必要な事が分かる。

無人ヘリコプターの位置、および方位を制御する問題なので、Required behavior フレームに当てはめる事とする。無人ヘリコプターは、「無人ヘリコプター」領域を Controlled domain として入れる。また、「4.1 自律飛行制御則」の「飛行計画については、飛行形式(前進またはホバー)、目標位置、(中略)旋回方向、旋回半径、ホバー時間、ホバー方位をパラメーターとして構成し」との記述から、「飛行計画」領域を Description domain として入れる。インターフェース b での飛行計画から機械領域への入力は上記のパラメーターとなる。またインターフェース a において、「無人ヘリコプター」から発生する共有現象は位置と方位の値、機械領域から発生する共有現象はその 2 つにまつわるコマンドとなる。

4.2 バケットエレベーター式連続アンローダー

住友重機技報 No.159 2005 に掲載されている「バケットエレベーター式連続アンローダーの予防保全システム」を資料とし、問題フレームの図を記述した[9]。問題は文脈図 2 つと問題図 12 個で表す事ができた。問題図は全て問題フレームに当てはめた。付録 A.2 にバケットエレベーター式連続アンローダー（出典元の論文に習い、以下「BE 式 CSU」とする）の図を掲載する。

資料と図の関係を以下に示す。

4.2.1文脈図について

全体文脈図：　「図 3　システム構成」を中心に資料全体から

まず、制御対象である「BE 式 CSU」の領域を入れ、機械領域に接続する。また、図 3 の「運転室」との記述や「4.1 疲労監視システム」の「運転室のモニタ画面で警報を発生する」といった記述等から、「運転室」の領域を入れる事とする。また、図 3 や「3.3 データ処理」の「「パソコン」および「メンテナンス PC」は、公衆電話回線で接続される」、「4.5 リモートアクセス機能」の「本システムは公衆回線にて CSU 上のパソコン（図 9）と、SES メンテナンス PC 間でデータ通信ができる」といった記述から、「サービスセンター」の領域を入れ、また「サービスセンター」と機械領域との間には「公衆電話回線」の領域を設ける事とする。

CSU の文脈図：　資料全体から

まず、図 3 や「3.3 データ処理」の「運搬荷役機械の制御装置(PLC)」といった記述から、機械領域へデータを渡し、また制御信号を受け付ける領域として「PLC」を入れる事とする。以下の図では、機械領域から CSU へのアクセスは全て「PLC」領域を介して行う事とする。

「4.1.1 疲労寿命推定」の「（テンションバー 2 ヵ所）で変動応力を測定し」との記述から、「テンションバー」の領域を入れる。また、テンションバーでの応力測定は「4.1.2.1 溶接型ひずみゲージ」にあるようにひずみゲージを使うので、「ひずみゲージ」の領域を入れる。また「4.1.2.2 応力頻度分布」の「計測した変動応力から応力頻度を求めるべく、アナログの計測値

（電圧）を PLC でデジタル変換し」との記述から、「ひずみゲージ」は「PLC」と「テンションバー」の間に配置する事とする。

「4.2 軸受異常監視システム」の「CSU の主要な軸受け 16 ヵ所に加速度センサを設置した」「常時計測したアナログの加速度（電圧）を PLC で変換し」との記述から、「軸受」の領域を入れ、「軸受」と「PLC」の間に「加速度センサ」を入れる事とする。

「4.3 BE 旋回速度制御システム」の「BE 旋回速度およびバケット速度をブーム直角方向で最大 20%ダウンさせる」、「BE がブーム方向（BE 旋回角 0 度および 180 度）のときに BE 旋回速度およびバケット速度をアップさせる」との記述から「バケットエレベーター」の領域を入れる事とし、制御信号を出す PLC に接続する。

「4.4 掘削力ピークカット制御システム」や図 8 から、ブームの動作、バケットエレベーターの動作、CSU の走行を制御する必要があることが分かるので、新たに「ブーム」と「脚部」の領域を入れる。また、「溶接ひずみゲージを BE ポストに取り付け」との記述から、「バケットエレベーター」と「ひずみゲージ」の間にもインターフェースを設ける。

4.2.2問題図について

精度チェック：　「4.1.2.3 荷重校正」からこの副問題が必要なことが分かる。

入力を基に出力を決める問題なので、Transformation frame に当てはめる事とする。「計測器類が初期状態を維持できているかどうかを定期的に確認する」との記述から、ここでの計測器類とはひずみゲージの事であるため、Inputs として「ひずみゲージ」の領域を入れる。また、機械領域との Connection domain となる「PLC」領域も入れる。

「初期値との差を算出し」との記述より、「初期値」の領域を Description domain として入れる。最後に、Outputs として「荷重校正値」の領域を入れる。

応力の測定：　「4.1.2 応力測定ツール」からこの副問題が必要なことが分かる。

実世界を測定しモデル化する問題なので、Model building subproblem frame に当てはめる事とする。

「テンションバーで（中略）変動応力を採取する」との記述から、Real world として「テンションバー」領域を入れる。また、「4.1.2.1 ひずみゲージ」にあるように応力の採取にはひずみゲージを使う事、また「4.1.2.2 応力頻度分布」の「アナログの計測値（電圧）を PLC でデジタル変換し」との記述から、「テンションバー」と機械領域の間の Connection domain として「ひずみゲージ」、「PLC」の領域を入れる。

「4.1.2.3 荷重校正」より、荷重校正が常に必要なので、「荷重校正値」領域を Description domain として入れる。

最後に「測定された応力」の領域を、Model として入れる。

応力の累積的記録：「4.1.2 応力測定ツール」からこの副問題が必要なことが分かる。

現在の応力を基に応力の累積データを編集する問題なので、Simple workpieces frame に当てはめる事とする。

「測定された応力」の領域を User、「累積された応力」の領域を Workpieces とする。

制御の主体は User ではなく機械領域にあるので Control variant とし、インターフェース内の共有現象は、機械領域が「測定された応力」からデータを読むイベントを起こし、それにより「測定された応力」からデータが機械領域に渡され、それに応じて「累積された応力」にデータが追加される、といった内容にする。

テンションバーの累積損傷度の計算：「4.1.1　疲労寿命推定」、「4.1.2.2　応力頻度分布」からこの副問題が必要なことが分かる。

累積された応力を基に累積損傷度を計算する問題なので、Transformation frame に当てはめる事とする。

「4.1.1 疲労寿命推定」の「変動応力を計測し、疲労設計曲線により求めた疲労寿命との比で、累積損傷度を算出・評価する」といった記述や「4.1.2.2 応力頻度分布」の数式から、「累積された応力」の領域を Inputs、「テンションバーの累積損傷度」の領域を Outputs として入れる事とする。

テンションバー以外の累積損傷度の計算：「4.1.1　疲労寿命推定」からこの副問題が必要なことが分かる。

テンションバーの累積損傷度を基にそれ以外の累積損傷度を計算する問題なので、Transformation frame に当てはめる事とする。

「4.1.1　疲労寿命推定」の「他の主要構造部位 15 ヵ所については、事前に測定したデータを基に、テンションバーとの日で累積損傷度を算出することにした。」との記述から、「テンションバーの累積損傷度」の領域を Inputs、「主要部分の累積損傷度」の領域を Outputs として入れる事とする。

疲労寿命監視システム：「4.1　疲労寿命監視システム」全体からこの副問題が必要なことが分かる。

累積損傷度を監視し、値によって警報を出す問題なので、Information display frame に当てはめる事とする。

監視対象はテンションバーを含む全体の累積損傷度なので、「テンションバーの損傷＋主要部分の損傷」の領域を Real world として入れる。

「4.1 疲労寿命監視システム」の「疲労寿命が設定値を超えると、運転室のモニタ画面で警報を発生する」との記述から、「運転室」の領域を Display として入れる。

累積損傷度の PC への表示：　「4.1.1 疲労寿命推定」からこの副問題が必要なことが分かる

累積損傷度を PC 画面に出力する問題なので、Information display frame に当てはめる事とする。疲労寿命監視システムの問題図と同様、「テンションバーの損傷＋主要部分の損傷」の領域を Real world として入れる。

「4.1.1 疲労寿命推定」の「これらの累積損傷度は、CSU 機体に設置したパソコン画面に大きい順に一覧表示する。」との記述から、「CSU 上の PC」の領域を Display として入れる事とする。

軸受異常監視システム：　「4.2　軸受異常監視システム」からこの副問題が必要なことが分かる。

軸受の異常を検知して警報を発する問題なので、Information display frame に当てはめる事とする。

「CSU の主要な軸受 16 ヵ所に加速度センサを設置した。常時計測したアナログの加速度（電圧）を PLC でデジタル変換し累積することで」との記述から、「軸受」の領域を Real world として入れ、「軸受」と機械領域の間には Connection domain として「加速度センサ」、「PLC」の領域を入れる。

警報を発する対象は資料に記述されていないので疲労寿命監視システムと同様に運転室に警報を出す事とし、「運転室」の領域を Display として入れる。

BE 旋回速度制御システム　「4.3　 BE 旋回速度制御システム」からこの副問題が必要なことが分かる。

バケットエレベーターのブームに対する角度に合わせてバケットエレベーターの旋回速度やバケット速度を制御する問題なので、Required behavior frame に当てはめる事とする。

「BE 旋回速度およびバケット速度をブーム直角方向で最大 20%ダウンさせる」、「BE がブーム方向（BE 旋回角 0 度および 180 度）のときに BE 旋回速度およびバケット速度をアップさせる」との記述から「バケットエレベーター」の領域を Controlled domain として入れる事とする。

機体の制御は PLC を介して行われているので、「バケットエレベーター」と機械領域の間の Connection domain として「PLC」の領域を入れる。

掘削力の計算：　「4.4　掘削力ピークカット制御システム」からこの副問題が必要なことが分かる。

応力から掘削力を計算する問題なので、Transformation frame に当てはめる事とする。

「掘削力は、溶接ひずみゲージを BE ポストに取り付け、発生応力と荷重校正値から算出した。」との記述から、「バケットエレベーター」の領域を Inputs として入れ、「ひずみゲージ」の領域を「バケットエレベーター」に繋がる Connection domain として入れる。また、「荷重校

正値」の領域は Description domain として入れる。

「4.1.2.2 応力頻度分布」の「計測した変動応力（中略）アナログの計測値（電圧）を PLC でデジタル変換し」との記述にあるように、ひずみゲージでの計測値は PLC を介して機械領域に送られているので「ひずみゲージ」と機械領域の間の Connection domain として「PLC」を入れる事とする。

以上の領域から得られた値を基に掘削力が算出されるので、Outputs として「掘削力」の領域を入れる。

掘削力ピークカット制御システム：　「4.4　掘削力ピークカット制御システム」「図 8 掘削力ピークカットフロー」からこの副問題が必要なことが分かる。

掘削力のピークカットは機械領域による自動制御で行われるが、リセット操作は人間が行うので、Commanded behavior frame に当てはめる事とする。

図 8 や「4.4　掘削力ピークカット制御システム」中の(1)(2)(3)より制御対象はバケット速度、ブーム旋回速度、走行速度、BE 旋回速度、起伏、スイングである。それらをまとめ、Controlled domain として「バケットエレベーター＋脚部＋ブーム」の領域を入れる。また機体の制御は PLC を介して行われているので、「バケットエレベーター＋脚部＋ブーム」と機械領域の間に「PLC」の領域を Connection domain として入れる。

図 8 や「4.4　掘削力ピークカット制御システム」中の(1)(2)(3)より掘削力に応じて制御を行うので、「掘削力」の領域を Description domain として入れる。

図 8 より異常時にリセット操作を人間が行うので、「オペレーター」の領域を Operator として入れる。

サービスセンターへのリモートアクセス：　「図 3　システム構成」「3.3　データ処理」、「4.5　リモートアクセス機能」からこの副問題が必要なことが分かる。

資料中では具体的にどんなリモートアクセス機能があるか明言されていないが、CSU からメンテナンス PC へのデータの移動は必ず必要になるので、今回はその問題のみ取り扱う。CSU 上の PC 上のデータからメンテナンス PC 上のデータを作成するという解釈をし、Transformation frame に当てはめる事もできるが、図 3 の「DB」という記述を勘案するとデータベースの更新といった形になるので、Simple workpieces frame に当てはめる事とする。

図 3 や「4.5 リモートアクセス機能」の「本システムは公衆電話回線にて CSU 上のパソコン（図 9）と SES メンテナンス PC 間でデータ通信ができる」との記述と「CSU 上の PC からリモートアクセスでデータベースを更新する」という想定された問題の内容から、「CSU 上の PC」の領域を User として入れる。また出力側には「メンテナンス PC」の領域、および「メンテナンス PC」と機械領域を接続する Connection domain である「公衆電話回線」の領域を入れる。データの格納場所となる領域が「メンテナンス PC」の他に必要なので、ここではまだ「メンテナンス PC」は Workpieces としない。

図 3 より、メンテナンス PC 上にデータベースが存在するので、データの格納場所として「データベース」の領域を入れ、これを Workpieces とする。「メンテナンス PC」の領域は Connection domain とする。

5　問題フレームのメタモデルの作成

ケーススタディの後、問題フレームの構成要素について整理するため、問題フレームにおける図のメタモデルを作成した。メタモデルはUMLのクラス図に類似した記法で表記した。ある程度までは矛盾や表記漏れなくメタモデルにできたが、記述領域と通常の問題領域とのインターフェースでもハイパーアーク接続による3つ以上の領域の接続が可能か、といった疑問が残り、そこはどちらでもいいような定義を行わざるを得なかった。また、制約を課す要件の参照は一つの問題図に常に一つだけ存在するのか、それとも二つ以上存在する事が許されているのかも明記されていない。Michael Jacksonの挙げているは例では制約を課す要件の参照は常に一つの問題図に一つだけとなっているので、それに習ってここでは一つだけとしてメタモデルを作成した。この面では、Michael Jacksonの説明不足が明らかになった。付録Bにメタモデルを掲載する。

以下にメタモデルの説明を示す。

全体文脈図のメタモデルについて

一つの「全体文脈図」に対し、「機械領域」は常に一つ。「機械領域」は一つ以上の「機械と問題領域のインターフェース」に接続され、各々の「機械と問題領域のインターフェース」は一つ以上の「問題領域」に接続される。「問題領域」同士は、「問題領域間のインターフェース」で接続される場合もある。「機械領域」とのインターフェースは持たず、「問題領域」とのインターフェースしか持たない「問題領域」も存在する場合がある。

多くの場合インターフェースは2つの領域を接続するが、3つ以上の領域を一つのインターフェースで接続する場合もある。

分割された問題領域のメタモデルについて

ある「分割前の与えられた問題領域」に対し、「分割された問題領域の文脈図」が必要なければ作成しなくて良い。必要な場合は、「分割された問題領域の文脈図」を作成するが、さらに分割が必要な場合もあり、その場合はある「分割後の与えられた領域」を別の分割における「分割前の与えられた問題領域」とし、再帰的に分割を行う事になる。

分割後の一つの文脈図に対し、「分割後の与えられた領域」は2つ以上存在する。それらの領域は互いに「内部の問題領域間のインターフェース」で接続されている場合がある。

また「分割後の与えられた領域」は「外部の領域とのインターフェース」を持っている場合がある。全ての領域が「外部の領域とのインターフェース」を持っている必要はないが、一つの文脈図の中で「外部の領域とのインターフェース」は一つ以上存在しなければならない。

物理的な領域のサブクラスについて

「物理的な領域」は「問題領域」と「記述領域」に分けられる。「問題領域」は更に「与えられた

問題領域」と「設計される問題領域」に分けられる。「記述領域」は更に「通常の記述領域」と「設計される記述領域」に分けられる。ある問題図で「記述領域」だったものが別の問題図では「問題領域」となる場合もある。

問題図のメタモデル：機械領域と物理的な領域について

一つの「問題図」に対し、「機械領域」は一つ、「物理的な領域」は一つ以上存在する。「機械領域」と「物理的な領域」は「機械と物理的な領域のインターフェース」で接続されるが、「機械領域」とのインターフェースを持たない「物理的な領域」が存在する場合もある。

問題図のメタモデル：物理的な領域の間のインターフェースについて

2つ以上の「問題領域」が「問題領域間のインターフェース」で接続される場合がある。1つ以上の「記述領域」と1つ以上の「問題領域」が「問題領域と記述領域のインターフェース」で接続される場合がある。1つの「記述領域」から1つの「問題領域」が「記述の参照」で参照される場合がある。

問題図のメタモデル：要件と物理的な領域について

一つの問題図に対し、要件は一つ存在する。「要件の参照」は「通常の要件の参照」と「制約を課す要件の参照」に分けられ、一つの要件に対し「制約を課す要件の参照」が必ず一つ存在する。また、一つの要件に対し「通常の要件の参照」は存在しない場合と一つ以上存在する場合がある。

一つの物理的な領域に対し、参照している「要件の参照」は一つ存在する場合と存在しない場合がある。

全体文脈図と問題図の関係について

一つの「全体文脈図」を並列的に分割し、「問題図」が作成される。

「全体文脈図の機械領域」を分割し複数にしたものが「問題図の機械領域」である。

ある「全体文脈図の問題領域」に対応する「問題図の物理的な領域」は一つ以上存在し、複数の問題図に現われる場合がある。「全体文脈図の問題領域」と全く同じものが「問題図の物理的な領域」として現れる場合がある。「全体文脈図の問題領域」の分割された一部分が「問題図の物理的な領域」として現れる場合もある。

一つの「全体文脈図の機械と問題領域のインターフェース」に対し、「問題図の機械と物理的な領域のインターフェース」は複数存在し、複数の問題図に現れる場合や、インターフェースが分割されて現れる場合がある。「全体文脈図の問題領域間のインターフェース」と「問題図の物理的な領域間のインターフェース」の関係も同様である。

6　問題フレームの評価

この章では、ケーススタディを基に、問題フレームという手法の評価を行う。まずメリットについて述べ、それから改善が考えられる部分について述べ、その後利点や欠点以外にも分かった事について述べる。また、問題フレームにおける記述をユーザーとのコミュニケーションに利用できる可能性について考える。

6.1 メリットの部分

メリットについては、以下の事が考えられる。

ケーススタディではどちらの例も組み込みシステムを扱ったが、組み込みシステムではコンピューター以外の現実世界が重要になる。なので、要件の記述には現実世界への大きな関心が必要となる。問題フレームでは問題領域を記述することによって現実世界を記述しているので、組み込みシステムに向いている。

問題に対して並列的な分割をするので、分割後のある問題で出てきた領域が他の問題でも出てくるといった表現ができるため自然な形で問題の理解ができる。従来のような完全な階層的な分割だと、これはできない。

問題の分割自体も比較的楽に行える。効果的な問題の分割法についてもある程度はMichael Jacksonによって示されている。

既にあるフレームに当てはめる、といった形で問題を定義するため、何も無いところから始めるよりもアイディアが生まれやすい。

機械についての記述だけではなく、機械に関わる領域全てを記述に含めるため、問題が世界のどこに位置しているのかを表現できる。機械に関する記述だけでは、機械と機械の外の世界、あるいは機械の外同士の相互関係が理解できない。

6.2 改善が考えられる部分

改善が考えられる部分としては、以下の事がある。

問題フレームに当てはめる際に、問題の性質に曖昧さがあり、どのフレームに当てはめるかが分かりにくい場合がある。TransformationフレームとModel building subproblemフレームには類似性があるため、どちらに当てはめるか迷うことも考えられる。このように当てはめるフレームの候補が複数出てきた場合にどちらにすればいいかという方針はMichael Jacksonは示していない。

逆に、問題の意味を考えてあるフレームに当てはめたくとも、フレームの定義上は別のフレームの方が正しいため、そのフレームには当てはめられない、と言った事も起こりうる。既存のデータに対し新たな入力によってデータを追加していく問題は Simple workpieces フレームに当てはめられる。しかし入力が現実世界の現象を感知するセンサーだった場合等は、Model building subproblem フレームと考える事の方が自然である。現実的には問題は両フレームの中間に位置する性質だが、Michael Jackson の本の中で述べられている限りでは前者となる。

なので、敢えて Model building subproblem フレームに当てはめたい場合には、一旦センサーからの入力を感知する部分を Model building subproblem フレームにあてはめ、そのモデル領域を別の図で Simple workpieces フレームの User 領域に当てはめ、データの追加を行う部分を記述する、といった冗長な形となる。このように、フレームに正確に適合させることを考えると、細かすぎる問題へと分割する必要性も生まれてくる。

2 つのフレームの両方の性質を持つ事を表現する方法が必要となる。基本形が同じ形をしている問題フレームはこの問題が起こりやすい。

問題が全て図によって表現されているため、ドキュメントを作成した本人以外は問題について理解できないかもしれない。各領域について、補足的な説明が必要になる場合もある。また、インターフェースの内容については、文章での説明が無いと領域以上に難解になってしまう場合がある。

特に、図の読み方が分からないユーザーとのコミュニケーションでは、問題図そのままではなく、ユーザーにとって理解しやすい形での提示が必要となる。

具体的な記述内容は Frame concern として説明されているが、図との対応関係を整理する必要が出てくる。

以下の場合は Michael Jackson の本の中のフレームには当てはめ辛いので、新しい問題フレームが必要となる。

- Michael Jackson の本の中の問題フレームにはデータの転送を問題にしているものが無い。なので、BE 式 CSU の「サービスセンターへのリモートアクセス」はデータベースの更新という事で、Simple workpieces フレームに強引に当てはめている。Connection variant での表現もできるが、データ転送自体を問題とする場合には問題を表現できない。データ転送自体の問題を強いて既存のフレームに当てはめるなら、転送元の機械 A を開発する問題と転送先の機械 B を開発する問題の 2 つの全体文脈図を作っておき、そこから副問題に分割していき、機械 A の副問題では Required behavior で問題領域としての機械 B を制御し、機械 B の副問題では問題領域としての機械 A から入力があるとして用途に応じたフレームに当てはめる、と言った事になる。この場合はクライアントサーバー型の問題の場合等は都合がいいが、今回のような場合は分かりづらくなってし

まう。

- あるセンサーでの入力から直接は関係ない環境のアクチュエーターを制御する物は Michael Jackson の本の中のフレームには当てはめ辛い。例えば、BE 式 CSU では「掘削力の計算」と「掘削力ピークカットシステム」では一旦応力から掘削力を計算する都合上、センサーの領域とアクチュエーターの領域が別の図になっているが、もし何らかの理由で同じ図にした場合はフレームに当てはめにくくなる。この問題の一つの対策としては、薬剤散布ヘリコプターで「散布地域の制御」の問題の「薬散装置」領域と「GPS」領域を一つの領域にまとめてあるように、センサーとアクチュエーターを一つの領域とするか、「無人ヘリコプター」等のより大域的な領域を使って Required behavior フレームとする事が考えられるが、領域を分割する必要がある場合には不都合となる。

Control variant については、他の Variant フレームが領域の追加という basic frame との明確な違いをつけているのに対し、インターフェースの制御上の性質が違うという抽象的な物のため、使いにくい。しかし、これを使わないと表現できないものもある。

問題図の記述だけでは分割後の各問題の間の関係を表す事ができない。ある共通の問題領域が複数の図に出てくることやある図での機械領域が別の図で問題領域として出てくる事で間接的には表現できるが、直接的な表現方法は無い。Composite frame として相互関係を示す方法も紹介されているが、あまりに多くの問題を一つの図に描くと図の可読性や問題の単純性を損なう。他の関連研究における「Composition problem frame」も問題間の関係を表すことに利用できる[6]。しかし、この場合も入れる領域が多い場合は複雑な図になってしまう。

なので、より単純な方法で問題間の関係を表す方法が求められる。

問題フレーム中で表される要件は簡単な名前と関連する問題領域の図示によって表されているが、要件の詳細な部分の表現には文章表現などが必要な場合も多い。Michael Jackson の本にもあるとおり、これは、領域の記述とは別に行うべきである。要件を記述したドキュメントを問題図とは別に作成し、問題図での要件との対応を認識できるようにしておくことで、要件の問題中の場所と要件そのものの両方が記述されることとなる。

大きい問題になると図そして要件の数が増えるため、要件の記述との対応をドキュメントにしておくことが必要となる。

問題図の中に現れる領域は基本的には物理的な物であり、概念的な領域は入れないとの事になっているが、物理的な領域だけだと記述しきれない問題もある。「Conceptual flavors」で、物理的でない概念的な物はインターフェースに現れる現象として定義するべきだとされている。組み込みシステムの場合は物理的な領域が多くなるため比較的記述しや

すいが、コンピューターの内部で自己完結しているようなシステムでは物理的な領域とみなす事ができるのがファイル等に限られるので、問題が記述し辛い。その場合は「物理的」という事を拡大解釈し、適切な領域を入れるようにしなければならない。

何も無いところから全体文脈図や問題図を描く際、どんな領域を入れ、どんな領域を入れないかは熟練者でないと分かりにくい。Michael Jackson の本の中では入れる領域によって問題の境界を決めていると書かれているが、慣れていない者にとってはどこに境界線を引くべきかの判断が難しい。また、ある領域を一つの領域とするか、または分割して別々の領域にするかといった判断も難しい。さらに、問題をどこまで分割するか、どんな問題を独立した一つの問題とするか、といった事も個人の裁量次第となる。薬剤散布ヘリコプターの「散布精度コントローラー」と「散布量計算」は一つの問題とする事もできたが、散布量を計算する事を独立した問題とするために今回は 2 つの問題に分割している。なので、図の記述にまつわる一定のガイドラインを作るべきである。Particular concerns として、一定の方針は示されているが、これだけではソフトウェアの要件定義および問題フレームによる記述に不慣れな者が図を描くのは難しい。

6.3 その他

ここでは、ケーススタディを通して分かった利点や欠点以外の問題フレームの性質について述べる。問題フレームの記述を利用してユーザーとのコミュニケーションを行う際、どうすればいいかといった考察も行う。

6.3.1実際に問題フレームを利用する際の Variant の多さ

現実の問題では単純に Basic frame に当てはまる問題は少ないため、Variant frame を利用する機会が多い。Variant を組み合わせすぎて問題図の可読性が損なわれる場合でも、Description variant や Operator variant ならば問題を更に分割することで対処できる場合もある。BE 式 CSU の「軸受け異常監査システム」の問題は Commanded behavior つまり Required behavior の Operator variant だが、オペレーターからの命令を受け付ける問題と自動制御を行う問題が混在しているので、もし必要ならさらに 2 つの問題へと分割することができる。

特に機械の制御を行う問題では Connection variant が増える。部品に信頼性がある場合や初期化の問題が起きない場合等、別の問題が発生しない場合でも、Connection variant を使った方がより正確に問題を記述できる場合が多い。そういった問題で Connection variant を省いてしまうと、問題の性質が分かりにくくなってしまうことは Michael Jackson も述べている。なので、可読性や単純性を損なわない限り、記述の正確性のために必要な場合は Connection variant を使うべきである。なお、Michael Jackson は Connection variant は Required behavior 、 Commanded behavior 、 Information display のみの

Variant だと述べているが、現実的にはそれ以外のフレームでも使わなければならない場面が出てきた。

6.3.2問題フレームを使ったユーザーとのコミュニケーション

問題フレームにおける各種の図はエンジニアのためのドキュメントなので、ユーザーとのコミュニケーションにそのまま使用する事には不向きである。しかし、問題を並列的に記述しているので、ユーザーが分割された各問題を理解するためには便利でもある。なので、作成したドキュメントをユーザーとのコミュニケーションに使うために工夫が必要となる。

また、ユーザーとアナリストにとっては「何をするか」つまり問題と「どのようにして行うか」つまり解法の認識が異なる[5]。ユーザーにとっての「何をするか」とは問題からシステムの目的を見つけ出す事であるが、アナリストにとっては目的を基にシステムの動作を定義する事である。また、目的を基にシステムの動作を定義する事はユーザーにとっての「どのようにして行うか」だが、アナリストにとっての「どのようにして行うか」は定義された動作を基に設計や実装を行う事である。このようにして、アナリストの思考はユーザーより一歩先へ進んでいるので、ユーザーに問題を理解してもらうためにはこの事を念頭に置く必要がある。問題フレームでは、問題領域や要件の部分にまつわる記述はユーザーにとって「問題」となるが、機械領域の仕様の記述は「解法」となる。

一つの方法として考えられるのは、問題フレームの機械領域以外の部分の記述を利用する方法である。機械領域はエンジニアだけが知識を持っており、ユーザーにとっては「どのようにして行うか」つまり解法の部分なので、ユーザーとの対話には不要である。残った部分の中で、ユーザーにとって「何をするか」つまり問題となる問題領域と要件の部分を用いて、ユーザーとの対話に利用することを考える。要件の内容を記述したドキュメントだけでこれができる場合はいいが、図に Remote domain があり、要件とのインターフェースを持たない問題領域がある場合は説明が不十分になる。要件の記述は離れた問題領域について触れていないはずなので、ユーザーは問題の全体図が掴めなくなってしまう。なので、その場合は要件から離れている問題領域に関する記述を補足的に要件のドキュメントに追加した、ユーザー向けのドキュメントが必要となる。これらのドキュメントは問題フレームの図に付随するものとして書かれる物なので、流用することで省力化につながる。

7　問題フレームの改善

この章では、ケーススタディによって分かった事を基に、問題フレームの改善案の提唱を行う。まずケーススタディで記述した図に対する補完を行う。また、ケーススタディの図から離れ、一般的な改善案の提唱も行う。また、改善案に至った背景や、改善案が必要となるケースについても説明する。

7.1 ケーススタディで記述した図の補完

ケーススタディの図に対し、以下のような改善を行った。なお、今回は各改善案の効果を分かり易くするため、改善案 7.1.1 は独立して行い、改善案 7.1.2 と 7.1.3 は同時に行った。

7.1.1アーキタイプの定義

Transformation + Simple workpieces 等のように複数のフレームの性質を持つ問題に対しては、問題をより抽象化された上位の問題フレームに当てはめることを考える。基本形が同じ形状の問題フレームごとに、合成のテンプレートとして問題フレームの上位版（アーキタイプ）を作る。これは、同じ形状の問題フレームは概念的に共通の部分があるからである。そして、それを基に中間の問題フレームを作成する事とする。

現在あるフレームを基にアーキタイプを 2 つ作成した。Linear archetype と Diamond archetype である。図 16 に Linear archetype の形状を示す。

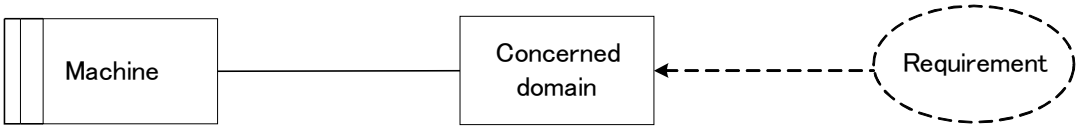


図 16　Linear archetype

Linear archetype に当てはまる問題フレームは今の所 Required behavior だけである。今後新たな問題フレームを定義した場合に、 Linear archetype がその問題フレームの上位となる可能性がある。

Concerned domain に当てはまるのは Controlled domain となる。Linear archetype は、問題に関連する問題領域が一つだけの場合の物となる。

図 17 に Diamond archetype の形状を示す。

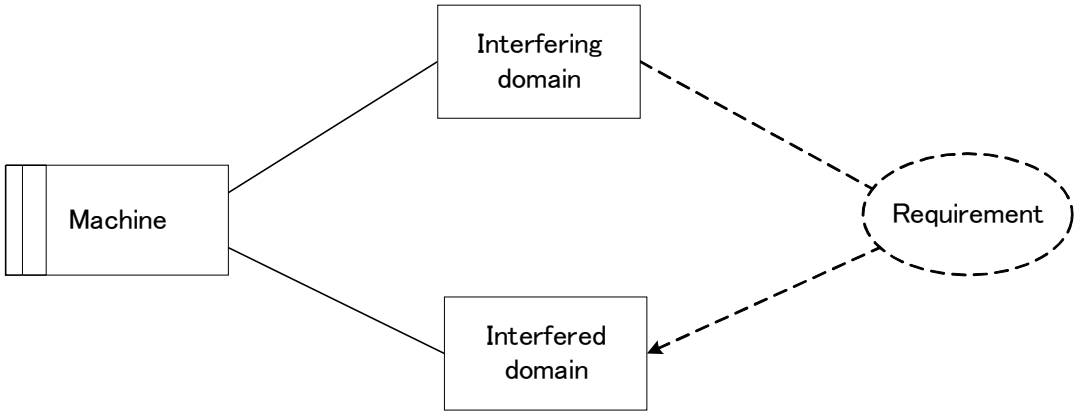


図 17　Diamond archetype

Diamond archetype は、ある問題領域をからの入力を基に別の問題領域へ何かを出力する場合の物となる。広義での Transformation frame と考える事もできる。Diamond archetype に当てはまるのは表 1 に示されている問題フレームである。Interfering domain や Interfered domain に当てはまる領域も同時に示す。

表 1　Diamond archetype と対応する問題フレーム

問題フレーム	Interfering domain に当てはまる領域	Interfered domain に当てはまる領域
Commanded behavior	Operator	Controlled domain
Information display	Real world	Display
Simple workpieces	User	Workpieces
Transformation	Inputs	Outputs
Model building subproblem	Real world	Model

また、7.2.1 にて後述する新たに定義した 2 つの問題フレームの場合も、両方とも Diamond archetype に当てはまる。表 2 に対応関係を示す。

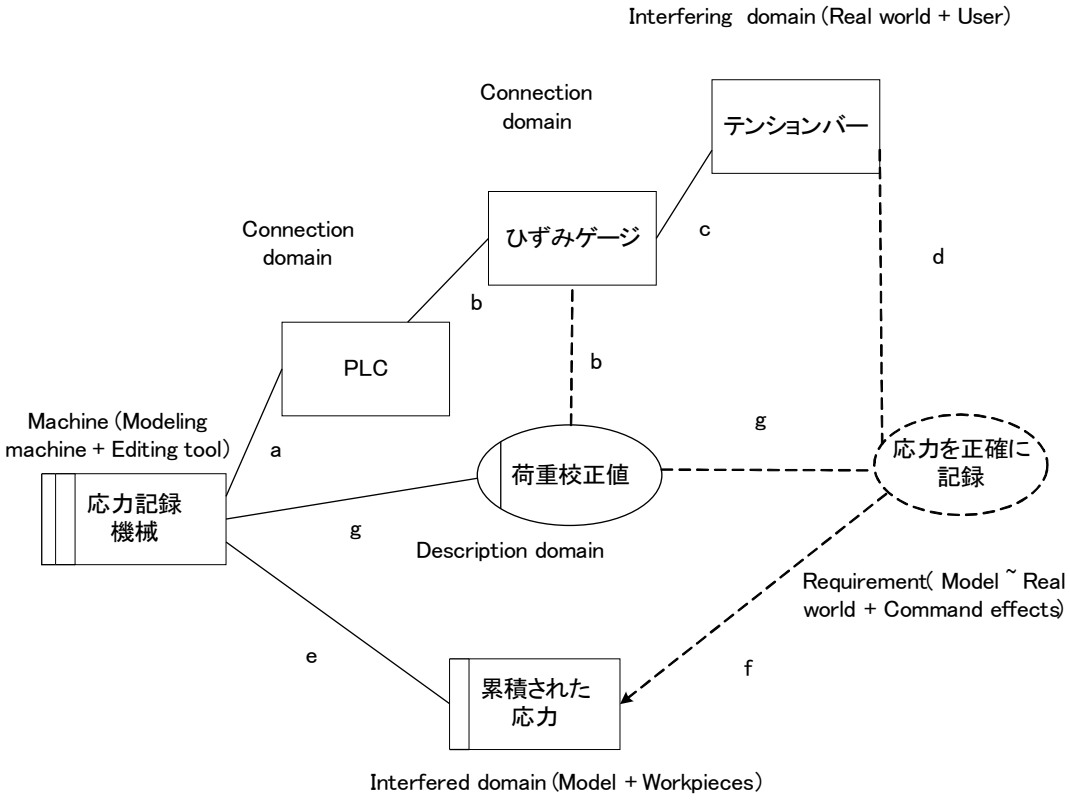
表 2　Diamond archetype と新問題フレームとの対応

問題フレーム	Interfering domain に当てはまる領域	Interfered domain に当てはまる領域
Transmission	Source	Destination
Remote required behavior	Observed domain	Controlled domain

同じアーキタイプに属する問題フレーム同士なら、一つの問題を 2 つ以上のフレームに当てはめていい事とする。ただし、作成する問題図において問題特有の問題領域をフレームの問題領域に当てはめる場合、親となるアーキタイプでの領域が一致しなければならない。そ

して、アーキタイプにおける領域と各問題フレームにおける領域を全て書き込む。

例として BE 式連続アンローダの「応力の測定」と「応力の累積的記録」を合わせた別バージョンの問題図「応力の累積的記録・別パターン」を図 18 に示す。これは Diamond archetype を基としたフレームになり、Simple workpieces frame と Model building subproblem frame の両方の性質となる。



a : PL! {DigitalizedStress}

b : ひず! {MeasuredStress}

c : テン! {Deformation}

d : テン! {Stress}

e : 応機! {StressValue}

f : 累応! {Data}

g : 荷校! {CalibrationValue}

図 18　応力の累積的記録・別パターン： Diamond archetype (model building subproblem frame + Simple workpieces frame の Control variant) の connection variant, description variant

7.1.2上位の問題図の作成

問題間の関係を理解しやすくするために、いくつかの下位の問題をまとめた上位の問題図を作る。（ヘリコプターなら速度制御＋高度誘導＋水平面誘導＋離陸＋着陸で飛行制御、等）その際、簡略化のためにConnection domainやDescription domainが多くなる場合は除外しても良いこととする。下位の問題にそういった領域が入っていれば良い。上位の問題図はあくまでも問題の位置関係を表すための物とする。もし双方の記述に不一致がある場合には、下位の問題図に従う事とする。「海へ持っていくクロノメーターは二個ではいけない。一個だけにしておくか、さもなくば三個持っていく方がいい。」といった格言のように、両方の定義がある場合には一方を標準とし、もう一方をそこから派生する記述とすべきである。[2]

この階層関係は7.1.3で後述するディクショナリにて説明可能だが、階層図も作るべきである。また、上位の問題図では、必要な場合は対応する機械領域の内部を説明する文脈図も作る。階層図のルートは常に全体文脈図とする。

同じ「上位のものを作る」アーキタイプとの違いは、アーキタイプが上位のクラスと言えるのに対し上位の問題図は上位のインスタンスということである。

薬剤散布ヘリコプターにおける上位の問題図および階層図を付録C.1に示す。バケットエレベーター式連続アンローダーにおける上位の問題図および階層図を付録C.2に示す。

上位の問題図とbasic frameの違いについて以下に述べる。

下位のフレームでは理解しやすいほどシンプルになるように細分化した各問題を記述している。これに対し、上位の問題図はある程度問題を抽象化した物を記述している。上位の問題図は下位のフレームで記述した問題をある程度まとめたものであり、あくまで問題間の関係の理解を助けるための補足的な物である。

Michael Jacksonは、問題フレームは分割し単純化された問題をパターンに当てはめ表現する物だと言っているが、上位の問題図が作成できるという事はこれに反する。違ったパターンの問題図をまとめた物が一つのパターンに当てはまってしまうからであり、問題が変質しているということになってしまう。また、まだ分割する余地が有る以上問題はまだ単純化されているとは言えない。ここにひとつ、問題フレームといった概念に曖昧さがあると考えられる。

Michael Jacksonは問題を分割し問題図を作成する際、核になる問題と補助的な問題を定義するべきだと述べている。なので、上位の問題図は「ある規模の副問題の中で核になる問題の問題図を単純化した物」とも考える事ができる。違ったフレームの問題図は変わっているのではなく、省略されているか上位の問題図の機械領域に内包されているという事である。つまり、記述した問題フレームが必ずしも問題の最小単位であるとは限らない。

また、Michael Jacksonは問題フレームにおける問題の分割は階層的な分割というより並

列な分割だと述べている。しかし完全に階層的な分割の概念を排除すると、どれとどれが関連する問題かといった事が分からなくなる。なので、並列的な分割をした後、全体文脈図をルートとした階層的な整理をすべきだと考えられる。

従来のような階層構造の場合は、階層構造の内部の各要素は兄弟やその子孫の要素との重複部分が存在しない事となっている。しかし、このルールを排除し、並列的な分割と階層的な分割を併用する事で、より自然な形で問題を理解できるようになる。事実、問題フレームにおいても文脈図では階層的な分割を行っている。

以上の事から、問題フレームを問題の最小単位のみを表現する物としておくと不都合である。粒度に関わらず、ある特定の一貫した目的に対する問題を表現するものとした方が良い。そして多くの場合は下位の問題図群ではその上位の問題図と同じ内容であるフレームの核となる問題が一つ以上あり、それを補助する別の問題が派生している、といった形になる。BE 式 CSU の場合は、「総合疲労監視システム」の下位の問題図群の核となる問題は「疲労寿命監視システム」であるし、「総合掘削力ピークカットシステム」の下位の問題図群の核は「掘削力ピークカットシステム」である。薬剤散布ヘリコプターの場合は、「薬剤の散布」の下位の問題図群の核となる問題は「散布地域の制御」および「散布精度の制御」であるし、「飛行制御」の下位の問題図群の核は「水平面誘導」、「高度誘導」、「速度制御」である。

7.1.3ディクショナリの作成

作成者以外が図を理解できるために、ディクショナリを作成する。要件の記述との対応関係もディクショナリで対応できる。

ディクショナリの形式は、以下のようにする。

- 全ての図をカバーするグローバルなディクショナリとして（領域名をキーとする）
  - 領域名→領域の意味、出てくる問題図または文脈図とその中での役割（機械領域、問題領域、記述領域）、分割前の領域、分割後の領域、領域を説明するするドキュメント名
- 一つの問題図をカバーするローカルなディクショナリとして（問題名をキーとする）
  - 問題名→要件の意味、インターフェースの意味、対応する要件を説明したドキュメント名、分割前の問題図、分割後の問題図

薬剤散布ヘリコプターのディクショナリを以下に示す。

グローバルなディクショナリ

○ 計画スタッフ

- 意味：飛行計画を設定するオペレーター
- 登場する図と役割：全体文脈図（問題領域）、飛行計画の入力（問題領域）

○ 計画編集機械
- 意味：飛行計画を編集する機械
- 登場する図と役割：飛行計画の入力（機械領域）
- 分割前の領域：ヘリコプター制御機械

○ 高度計
- 意味：高度の値を検知する装置
- 登場する図と役割：無人ヘリコプターの文脈図（問題領域）
- 分割前の領域：無人ヘリコプター

○ 高度誘導機械
- 意味：ヘリコプターの高度を制御する機械
- 登場する図と役割：高度誘導（機械領域）
- 分割前の領域：飛行制御機械

○ 散布地域制御機械
- 意味：飛行計画に基づき散布を制御する機械
- 登場する図と役割：散布地域の制御（機械領域）
- 分割前の領域：総合散布機械

○ 散布量計算機械
- 意味：適正な散布量を計算する機械
- 登場する図と役割：散布量計算（機械領域）
- 分割前の領域：総合散布機械

○ GPS
- 意味：水平面上の座標を検知する装置
- 登場する図と役割：無人ヘリコプターの文脈図（問題領域）、散布地域の制御（問題領域）
- 分割前の領域：無人ヘリコプター

○ 地面
- 意味：薬剤の散布対象となる地面
- 登場する図と役割：全体文脈図（問題領域）、薬剤の散布（問題領域）、飛行制御（問題領域）、散布精度の制御（問題領域）、散布地域の制御（問題領域）

○ 水平面誘導機械
- 意味：ヘリコプターの水平面上の位置を制御する機械
- 登場する図と役割：水平面誘導（機械領域）
- 分割前の領域：飛行制御機械

○ 精度制御機械
- 意味：適正量に基づき制度を制御する機械
- 登場する図と役割：散布精度の制御（機械領域）
- 分割前の領域：総合散布機械

○ 総合散布機械
- 意味：散布を制御する機械の全体
- 登場する図と役割：薬剤の散布（機械領域）
- 分割前の領域：ヘリコプター制御機械
- 分割後の領域：散布量計算機械、精度制御機械、散布地域制御機械

○ 速度計
- 意味：速度の値を検知する装置
- 登場する図と役割：無人ヘリコプターの文脈図（問題領域）、散布量計算（問題領域）
- 分割前の領域：無人ヘリコプター

○ 速度制御機械
- 意味：ヘリコプターの速度を制御する機械
- 登場する図と役割：速度制御（機械領域）
- 分割前の領域：飛行制御機械

○ 適正量
- 意味：速度から算出した適正な散布量
- 登場する図と役割：散布量計算（問題領域）、散布精度の制御（記述領域）

○ 着陸制御機械
- 意味：ヘリコプターの着陸を行う機械
- 登場する図と役割：着陸（機械領域）
- 分割前の領域：飛行制御機械

○ 飛行計画

- 意味：オペレーターによって設定された飛行計画
- 登場する図と役割：全体文脈図（問題領域）、薬剤の散布（記述領域）、飛行制御（記述領域）、散布地域の制御（記述領域）、飛行計画の入力（問題領域）、速度制御（記述領域）、高度誘導（記述領域）、着陸（記述領域）、水平面誘導（記述領域）

○ 飛行制御機械

- 意味：飛行を制御する機械の全体
- 登場する図と役割：飛行制御（機械領域）
- 分割前の領域：ヘリコプター制御機械
- 分割後の領域：速度制御機械、高度誘導機械、離陸制御機械、着陸制御機械、水平面誘導機械

○ 飛行制御装置

- 意味：ヘリコプターのローター等を制御し飛行を行う装置
- 登場する図と役割：無人ヘリコプターの文脈図（問題領域）
- 分割前の領域：無人ヘリコプター

○ ヘリコプター制御機械

- 意味：機械領域の全体
- 登場する図と役割：全体文脈図（機械領域）
- 分割後の領域：総合散布機械、飛行制御機械、計画編集機械

○ 無人ヘリコプター

- 意味：無人ヘリコプターの全体
- 登場する図と役割：全体文脈図（問題領域）、薬剤の散布（問題領域）、飛行制御（問題領域）、速度制御（問題領域）、高度誘導（問題領域）、離陸（問題領域）、着陸（問題領域）、水平面誘導（問題領域）
- 分割後の領域：GPS、高度計、速度計、薬散装置、飛行制御装置

○ 薬散装置

- 意味：薬剤を散布する装置
- 登場する図と役割：無人ヘリコプターの文脈図（問題領域）、散布精度の制御（問題領域）、散布地域の制御（問題領域）、

- 分割前の領域：無人ヘリコプター

○ 離陸制御機械
- 意味：ヘリコプターの離陸を行う機械
- 登場する図と役割：離陸（機械領域）
- 分割前の領域：飛行制御機械

ローカルなディクショナリ

□ 高度制御
- 要件の意味：一定高度を維持する
- インターフェースの意味
  - a：ClimbCMD：上昇のコマンド
  - a：DecendCMD：下降のコマンド
  - a：AltitudeValue：高度の値
  - b：Altitude：高度
- 分割前の問題図：飛行制御

□ 散布地域の制御
- 要件の意味：計画に基づき散布を行い、散布除外地域では散布を停止する
- インターフェースの意味
  - a：StartAppCMD：散布を開始するコマンド
  - a：StopApppCMD：散布を停止するコマンド
  - b：HorizonalPosStatus：水平面上の位置
  - a：DirectionStatus：ヘリコプターの向き
  - c：Apply：薬剤の散布
  - d：Polygon：散布除外地域の形状を表す多角形
  - d：Point：散布除外地域の位置
  - e：StartAppPoint：散布を開始する地点
  - e：StopAppPoint：散布を停止する地点
  - e：ExpectedArea：散布除外地域
  - f：Chemical：薬剤
- 分割前の問題図：薬剤の散布

□ 散布制度の制御

- 要件の意味：速度に応じた適正量に基づき、一定精度での散布を行う
- インターフェースの意味
  - a：StartAppCMD：散布を開始するコマンド
  - a：StopApppCMD：散布を停止するコマンド
  - a：IncreaceCMD：散布量を増やすコマンド
  - a：DecreaceCMD：散布量を減らすコマンド
  - b：Apply：薬剤の散布
  - c：AmountParameter：適正量の値
  - d：Amount：適正量
  - e：Chemical：薬剤
- 分割前の問題図：薬剤の散布

□ 散布量計算

- 要件の意味：速度を入力とし、計算規則に基づき計算を行う
- インターフェースの意味
  - a：SpeedValue：速度の値
  - b：Amount：適正な散布量
- 分割前の問題図：薬剤の散布

□ 水平面誘導

- 要件の意味：飛行計画に基づき移動、旋回、ホバーを行う
- インターフェースの意味
  - a：AdvanceCMD：前進のコマンド
  - a：RotateCMD：旋回のコマンド
  - a：HoverCMD：ホバーのコマンド
  - a：HorizonalPosStatus：水平面上の座標
  - a：DirectionStatus：ヘリコプターの方位の値
  - b：PointParameter：目標地点のパラメーター
  - b：RDirectionParameter：旋回方向のパラメーター
  - b：HDirectionParameter：ホバー方位のパラメーター
  - b：FormParameter：飛行形式のパラメーター
  - b：RadiusParameter：旋回半径のパラメーター
  - b：TimeParameter：ホバー時間のパラメーター
  - c：HorizonalPosition：水平面上の位置
  - c：Direction：ヘリコプターの方位
  - d：FlightForm：飛行形式

- ✧ d：TargetPoint：目標地点
- ✧ d：RotationgDirection：旋回方向
- ✧ d：RotationgRadius：旋回半径
- ✧ d：HoverringTime：ホバー時間
- ✧ d：HoverringDirection：ホバー方位

➢ 分割前の問題図：飛行制御

□ 速度制御

➢ 要件の意味：飛行計画の目標速度を維持する

➢ インターフェースの意味

- ✧ a：SpeedValue：速度の値
- ✧ a：SpeedUpCMD：加速のコマンド
- ✧ a：SpeedDownCMD：減速のコマンド
- ✧ b：TargetSpeedParam：目標速度のパラメーター
- ✧ c：Speed：速度
- ✧ d：TargetSpeed：目標速度

➢ 分割前の問題図：飛行制御

□ 着陸

➢ 要件の意味：飛行計画に基づき着陸を正しく行う

➢ インターフェースの意味

- ✧ a：LandCMD：着陸のコマンド
- ✧ a：AltitudeValue：高度の値
- ✧ a：HorizonalPosStatus：水平面上の座標
- ✧ a：DirectionStatus：ヘリコプターの方位の値
- ✧ b：PointParameter：着陸地点のパラメーター
- ✧ b：DirectionParameter：着陸方位のパラメーター
- ✧ c：Landing：着陸
- ✧ d：LandingPoint：着陸地点

➢ 分割前の問題図：飛行制御

□ 飛行制御

➢ 要件の意味：計画に基づきヘリコプターの飛行を行う

➢ インターフェースの意味

- ✧ a：CurrentStatusValue：ヘリコプターの現在の状態を表す値
- ✧ a：FlightCMD：飛行を操作するコマンド

- ✧ b：TargetParam：飛行計画の目標パラメーター
- ✧ c：FlightStatus：ヘリコプターの現在の状態
- ✧ d：TargetStatus：目標となる状態

➢ 分割後の問題図：速度制御、高度誘導、離陸、着陸、水平面誘導

□ 飛行計画の入力

➢ 要件の意味：スタッフの正しい入力に対し飛行計画を作成する

➢ インターフェースの意味

- ✧ a：PlanOperations：計画を入力するコマンド
- ✧ a：PlanStatus：計画の状態
- ✧ b：EnterFlightForm：飛行形式の入力
- ✧ b：EnterStartAppPoint：散布開始地点の入力
- ✧ b：EnterStopAppPoint：散布停止地点の入力
- ✧ b：EnterLandingPoint：着陸地点の入力
- ✧ b：EnterTargetPoint：目標地点の入力
- ✧ b：EnterTargetSpeed：目標速度の入力
- ✧ b：EnterRotatingDirection：旋回方向の入力
- ✧ b：EnterRotatingRadius：旋回半径の入力
- ✧ b：EnterHoverringTime：ホバー時間の入力
- ✧ b：EnterHoverringDirection：ホバー方位の入力
- ✧ b：EnterExpectedFiled：散布除外地域の入力
- ✧ c：FlightForm：飛行形式
- ✧ c：StartAppPoint：散布開始地点
- ✧ c：StopAppPoint：散布停止地点
- ✧ c：LandingPoint：着陸地点
- ✧ c：TargetPoint：目標地点
- ✧ c：TargetSpeed：目標速度
- ✧ c：RotatingDirection：旋回方向
- ✧ c：RotatingRadius：旋回半径
- ✧ c：HoverringTime：ホバー時間
- ✧ c：HoverringDirection：ホバー方位
- ✧ c：ExpectedFiled：散布除外地域

□ 薬剤の散布

➢ 要件の意味：計画に基づき薬剤を散布する

➢ インターフェースの意味

- ✧ a：AppCMD：散布関連のコマンド
- ✧ b：CurrentStatus：現在のヘリコプターの状態
- ✧ c：Apply：薬剤の散布
- ✧ d：PlanData：飛行計画の値
- ✧ e：Plan：飛行計画
- ✧ f：Chemical：薬剤

➢ 分割後の問題図：散布量計算、散布精度の制御、散布地域の制御

□ 離陸

➢ 要件の意味：離陸を正しく行う
➢ インターフェースの意味
- ✧ a：TakeOffCMD：離陸のコマンド
- ✧ a：AltitudeValue：高度の値
- ✧ b：TakingOff：離陸

➢ 分割前の問題図：飛行制御

BE 式 CSU のディクショナリを以下に示す。

グローバルなディクショナリ

○ 運転室

➢ 意味：CSU を操作する運転室。本システムでは情報の表示のみ関知する
➢ 登場する図と役割：全体文脈図（問題領域）、総合疲労寿命監視システム（問題領域）、疲労寿命監視システム（問題領域）、軸受異常監査システム（問題領域）

○ 応力記録機械

➢ 意味：現在の応力を測定し記録する機械
➢ 登場する図と役割：応力の測定（機械領域）
➢ 分割前の領域：総合疲労寿命監視システム

○ オペレーター

➢ 意味：掘削力ピークカット制御システムをリセットするオペレーター
➢ 登場する図と役割：総合掘削力ピークカットシステム（問題領域）、掘削力ピークカット制御システム（問題領域）

○ 荷重校正値
- 意味：荷重校正のために使う、初期値と現在のテスト値の差
- 登場する図と役割：精度チェック（問題領域）、応力の測定（記述領域）、掘削力の計算（記述領域）

○ 加速度センサ
- 意味：軸受の加速度を検知するセンサ
- 登場する図と役割：CSU の文脈図（問題領域）、軸受異常監査システム（問題領域）
- 分割前の領域：BE 式 CSU

○ 脚部
- 意味：BE 式 CSU のブーム脚部部分
- 登場する図と役割：CSU の文脈図（問題領域）、掘削力ピークカットシステム（問題領域）
- 分割前の領域：BE 式 CSU

○ 掘削力計算機械
- 意味：応力と荷重校正値から掘削力を計算する機械
- 登場する図と役割：掘削力の計算（機械領域）
- 分割前の領域：総合掘削力ピークカット機械

○ 掘削力ピークカット機械
- 意味：掘削力のピークカット制御を行う機械
- 登場する図と役割：掘削力ピークカット制御システム（機械領域）
- 分割前の領域：総合掘削力ピークカット機械

○ 公衆電話回線
- 意味：システムとサービスセンターを接続する回線
- 登場する図と役割：全体文脈図（問題領域）、サービスセンターへのリモートアクセス（問題領域）

○ サービスセンター
- 意味：データの累積と評価及びトラブルの復旧を行うサービスセンター
- 登場する図と役割：CSU の文脈図（問題領域）
- 分割後の領域：メンテナンス PC、データベース

○ 軸受

- 意味：CSU の主要な部分の軸受
- 登場する図と役割：CSU の文脈図（問題領域）、軸受異常監査システム（問題領域）
- 分割前の領域：BE 式 CSU

○ 軸受監視機械

- 意味：軸受の加速度を監視し警報を発する機械
- 登場する図と役割：軸受異常監査システム（機械領域）
- 分割前の領域：予防保全システム

○ 主要部分の損傷計算機械

- 意味：テンションバーの損傷度を基に主要部分の損傷度を計算する機械
- 登場する図と役割：テンションバー以外の累積損傷度の計算（機械領域）
- 分割前の領域：総合疲労寿命監視システム

○ 主要部分の累積損傷度

- 意味：計算によって求められたテンションバー以外の主要部分の累積的損傷度
- 登場する図と役割：テンションバー以外の累積損傷度の計算（問題領域）、疲労寿命監視システム（問題領域）、累積損傷度の PC への表示（問題領域）

○ 初期値

- 意味：計測器類が初期にテストで出した値。計測器類の精度のチェックに使う
- 登場する図と役割：精度チェック（記述領域）

○ 精度チェック機械

- 意味：ひずみゲージの精度をチェックする機械
- 登場する図と役割：精度チェック（機械領域）
- 分割前の領域：予防保全システム

○ 旋回速度制御機械

- 意味：BE の旋回速度を制御する機械
- 登場する図と役割：BE 旋回速度制御システム（機械領域）
- 分割前の領域：予防保全システム

○ 総合掘削力ピークカット機械

- 意味：掘削力ピークカット制御システムに関連する機械領域の全体
- 登場する図と役割：総合掘削力ピークカットシステム（機械領域）
- 分割前の領域：予防保全システム
- 分割後の領域：掘削力計算機械、掘削力ピークカット機械

○ 総合疲労寿命監視機械

- 意味：疲労寿命監視システムに関連する機械領域の全体
- 登場する図と役割：総合疲労寿命監視システム（機械領域）
- 分割前の領域：予防保全システム
- 分割後の領域：応力記録機械、累積機械、テンションバー損傷度計算機械、主要部分の損傷計算機械、疲労寿命監視機械、PC 表示機械

○ 測定された応力

- 意味：測定された最新の応力
- 登場する図と役割：応力の測定（問題領域）、応力の累積的記録（問題領域）

○ データベース

- 意味：サービスセンターのデータベース
- 登場する図と役割：サービスセンターへのリモートアクセス（機械領域）
- 分割前の領域：サービスセンター

○ テンションバー

- 意味：BE 式 CSU のテンションバー
- 登場する図と役割：CSU の文脈図（問題領域）、応力の測定（問題領域）、掘削力の計算（問題領域）
- 分割前の領域：BE 式 CSU

○ テンションバー損傷度計算機械

- 意味：テンションバーの損傷度を計算する機械
- 登場する図と役割：テンションバーの累積損傷度の計算（機械領域）
- 分割前の領域：総合疲労寿命監視システム

○ テンションバーの累積損傷度

- 意味：計算によって求められたテンションバーの累積的損傷度
- 登場する図と役割：テンションバーの累積損傷度の計算（問題領域）、テンショ

ンバー以外の累積損傷度の計算（問題領域）、疲労寿命監視システム（問題領域）、累積損傷度の PC への表示（問題領域）

○ バケットエレベーター
- 意味：BE 式 CSU に設置されたバケットエレベーター本体
- 登場する図と役割：CSU の文脈図（問題領域）、BE 旋回速度制御システム（問題領域）、掘削力の計算（問題領域）、掘削力ピークカットシステム（問題領域）
- 分割前の領域：BE 式 CSU

○ BE 式 CSU
- 意味：無人バケットエレベーター式連続アンローダの全体
- 登場する図と役割：全体文脈図（問題領域）、総合疲労寿命監視システム（問題領域）、総合掘削力ピークカットシステム（問題領域）
- 分割後の領域：PLC、加速度センサ、軸受、ひずみゲージ、テンションバー、バケットエレベーター、ブーム、脚部、CSU 上の PC

○ PLC
- 意味：BE 式 CSU の制御装置
- 登場する図と役割：CSU の文脈図（問題領域）、精度チェック（問題領域）、応力の測定（問題領域）、軸受異常監査システム（問題領域）、BE 旋回速度制御システム（問題領域）、掘削力の計算（問題領域）、掘削力ピークカットシステム（問題領域）
- 分割前の領域：BE 式 CSU

○ PC 表示機械
- 意味：累積損傷度を監視して CSU 上の PC に表示する機械
- 登場する図と役割：累積損傷度の PC への表示（機械領域）
- 分割前の領域：総合疲労寿命監視システム

○ ひずみゲージ
- 意味：応力を計測する溶接型ひずみゲージ
- 登場する図と役割：CSU の文脈図（問題領域）、精度チェック（問題領域）、応力の測定（問題領域）、掘削力の計算（問題領域）
- 分割前の領域：BE 式 CSU

○ 疲労寿命監視機械
- 意味：疲労寿命監視システムの本体。累積損傷度を監視して警報を発する機械
- 登場する図と役割：疲労寿命監視システム（機械領域）
- 分割前の領域：総合疲労寿命監視システム

○ ブーム
- 意味：BE 式 CSU のブーム（可動式アーム）部分
- 登場する図と役割：CSU の文脈図（問題領域）、掘削力ピークカットシステム（問題領域）
- 分割前の領域：BE 式 CSU

○ メンテナンス PC
- 意味：サービスセンターの PC
- 登場する図と役割：サービスセンターへのリモートアクセス（機械領域）
- 分割前の領域：サービスセンター

○ 予防保全システム
- 意味：機械領域の全体
- 登場する図と役割：全体文脈図（機械領域）
- 分割後の領域：精度チェック機械、総合疲労寿命監視機械、軸受け監視機械、旋回速度制御機械、総合掘削力ピークカット機械、リモートアクセス機械

○ リモートアクセス機械
- 意味：サービスセンターにリモートアクセスする機械
- 登場する図と役割：サービスセンターへのリモートアクセス（機械領域）
- 分割前の領域：予防保全システム

○ 累積機械
- 意味：測定された応力を累積する機械
- 登場する図と役割：応力の累積的記録（機械領域）
- 分割前の領域：総合疲労寿命監視システム

○ 累積された応力
- 意味：累積的に記録された応力
- 登場する図と役割：応力の累積的記録（問題領域）、テンションバーの累積損傷度の計算（問題領域）

○ CSU 上の PC

- 意味：BE 式 CSU に設置された PC
- 登場する図と役割：CSU の文脈図（問題領域）、総合疲労寿命監視システム（問題領域）、累積損傷度の PC への表示（問題領域）、サービスセンターへのリモートアクセス（問題領域）
- 分割前の領域：BE 式 CSU

ローカルなディクショナリ

□ 応力の測定

- 要件の意味：定期的な応力の測定を正確に行う
- インターフェースの意味
  - a：DigitalizedStress：数値化された応力
  - b：MesuredStress：測定された応力
  - c：Deformation：ひずみ
  - d：Stress：応力
  - e：StressValue：記録された応力の値
  - f：Data：記録されたデータ
  - g：CalibrationValue：荷重校正値
- 分割前の問題図：総合疲労寿命監視システム

□ 応力の累積的記録

- 要件の意味：測定された応力のデータを累積していく
- インターフェースの意味
  - a：AddData：データの追加
  - b：ReadData：データの読み込み
  - c：StressData：応力のデータ
  - c：CumulatedData：累積されたデータ
- 分割前の問題図：総合疲労寿命監視システム

□ 掘削力の計算

- 要件の意味：応力と荷重校正値から計算規則に応じて掘削力を算出する
- インターフェースの意味
  - a：DigitalizedStress：数値化された応力

- ✧ b：MesuredStress：測定された応力
- ✧ c：Deformation：ひずみ
- ✧ d：Stress：応力
- ✧ e：PowerValue：掘削力の値
- ✧ f：Power：掘削力
- ✧ g：CalibrationValue：荷重校正値

➢ 分割前の問題図：総合掘削力ピークカットシステム

□ 掘削力ピークカット制御システム

➢ 要件の意味：掘削力に合わせ BE の動作、ブームの動作、BE 式 CSU 本体の移動を制御する

➢ インターフェースの意味

- ✧ a：Level1CMD：掘削力が設定値 1 より大きい時の各部減速コマンド
- ✧ a：Level2CMD：掘削力が設定値 2 より大きい時の各部停止コマンド
- ✧ a：Level3CMD：掘削力が設定値 3 より大きい時の全動作停止コマンド
- ✧ a：RestartCMD：全動作停止後をリセットし動作を通常に戻すコマンド
- ✧ b：PowerValue：掘削力の値
- ✧ c：ResetCMD：オペレーターからのリセットのコマンド
- ✧ d：SlwdnBacket：バケット速度の半減
- ✧ d：SlwdnBoomRot：ブーム旋回速度の半減
- ✧ d：SlwdnTravel：走行速度の半減
- ✧ d：SlwdnBERot：BE 旋回速度の半減
- ✧ d：StopBoom：ブーム旋回の停止
- ✧ d：StopTrabel：走行の停止
- ✧ d：StopUpDown：起伏の停止
- ✧ d：StopBERto：BE 旋回の停止
- ✧ d：StopSwing：スウィングの停止
- ✧ d：StopAll：全動作の停止
- ✧ d：Restart：全動作停止からのリセット
- ✧ e：Power：掘削力
- ✧ f：Movement：各部の動作

➢ 分割前の問題図：総合掘削力ピークカットシステム

□ 軸受異常監査システム

➢ 要件の意味：軸受の加速度を監視し、異常なデータがあれば警報を発する

- インターフェースの意味
  - a：DigAcc：数値化された加速度
  - b：MesuredAcc：測定された加速度
  - c：Acceleration：加速度
  - d：ScreenOpns：モニターへの表示を行うコマンド
  - e：Movements：軸受の動作
  - f：Warnings：警報

□ サービスセンターへのリモートアクセス

- 要件の意味：CSU 上の PC からサービスセンターのデータベースへデータの転送を行う
- インターフェースの意味
  - a：SourceData：転送元のデータ
  - b：SendData：データを送信する
  - c：ReceiveData：データを受信する
  - d：StoreData：データを保存する
  - d：DBStatus：データベースの状態
  - e：DBValue：データベース上の値

□ 精度チェック

- 要件の意味：初期値と現在のテストの値の差から荷重校正値を出す
- インターフェースの意味
  - a：DigitalizedStress：数値化された応力
  - b：MesuredStress：測定された応力
  - c：CalibrationValue：荷重校正値
  - d：Value：初期値を数値化したもの
  - e：InitialValue：初期値

□ 総合掘削力ピークカットシステム

- 要件の意味：掘削力に合わせ BE 式 CSU の動作を制御する
- インターフェースの意味
  - a：ControlCMD：BE 式 CSU の動作を制御するコマンド
  - a：Power：掘削力
  - b：Movement：BE 式 CSU の動作
  - c：ResetCMD：オペレーターによるリセット操作
- 分割後の問題図：掘削力の計算、掘削力ピークカット制御システム

□ 総合疲労寿命監視システム

- 要件の意味：BE 式 CSU の疲労を監視し必要な箇所へ情報を送る
- インターフェースの意味
  - a：Stress：応力
  - b：ScreenOpns：モニターへの表示を行うコマンド
  - c：Damages：損傷
  - d：Information：情報
- 分割後の問題図：応力の測定、応力の累積的記録、テンションバーの累積損傷度の計算、テンションバー以外の累積損傷度の計算、疲労寿命監視システム、累積損傷度の PC への表示

□ テンションバー以外の累積損傷度の計算

- 要件の意味：計算規則に従いテンションバーの累積損傷度からテンションバー以外の主要部分の累積損傷度を計算する
- インターフェースの意味
  - a：TBDamageValue：テンションバーの累積損傷度の値
  - b：MPDamageValue：テンションバー以外の主要部分の累積損傷度の値
  - c：TBDamage：テンションバーの累積損傷度
  - d：MPDamages：テンションバー以外の主要部分の累積損傷度
- 分割前の問題図：総合疲労寿命監視システム

□ テンションバーの累積損傷度の計算

- 要件の意味：計算規則に従い累積された応力からテンションバーの累積損傷度を計算する
- インターフェースの意味
  - a：StressValues：累積された応力の値
  - b：DamageValue：累積損傷度の値
  - c：StressData：応力のデータ
  - d：Damage：累積損傷度
- 分割前の問題図：総合疲労寿命監視システム

□ BE 旋回速度制御システム

- 要件の意味：バケットエレベーターの角度に応じて旋回速度を制御する
- インターフェースの意味
  - a：AngleValue：BE の角度の値

- ✧ a：SpeedUpCMD：旋回速度上昇のコマンド
- ✧ a：SpeedDownCMD：旋回速度下降のコマンド
- ✧ b：Angle：BE の角度
- ✧ b：SpeedUp：旋回速度上昇
- ✧ b：SpeedDown：旋回速度下降
- ✧ c：Speed：BE の旋回速度

□ 疲労寿命監視システム

- ➢ 要件の意味：累積損傷度を監視して運転室に警報を発する
- ➢ インターフェースの意味
  - ✧ a：DamageValues：各部分の累積損傷度の値
  - ✧ b：ScreenOpns：モニターへの表示を行うコマンド
  - ✧ c：Damages：累積損傷度
  - ✧ d：Warnings：警報
- ➢ 分割前の問題図：総合疲労寿命監視システム

□ 累積損傷度の PC への表示

- ➢ 要件の意味：累積損傷度を CSU 上の PC に表示する
- ➢ インターフェースの意味
  - ✧ a：DamageValues：各部分の累積損傷度の値
  - ✧ b：ScreenOpns：PC への表示を行うコマンド
  - ✧ c：Damages：累積損傷度
  - ✧ d：PCInformation：PC 上の情報
- ➢ 分割前の問題図：総合疲労寿命監視システム

7.2 一般的な改善案

ケーススタディで記述した図の補完以外にも、以下のような改善案を考案した。

7.2.1新しい問題フレームの定義

6.2 で述べたように新しい問題フレームが必要になるケースが考えられるので、必要な問題フレームを定義した。一つはデータの転送を扱うフレーム、もう一つは Required behavior のセンサーとアクチュエーターが分離されているバージョンである。図 19 に Transmission フレームのフレーム図を示す。図 20 に Remote required behavior フレームのフレーム図を示す。

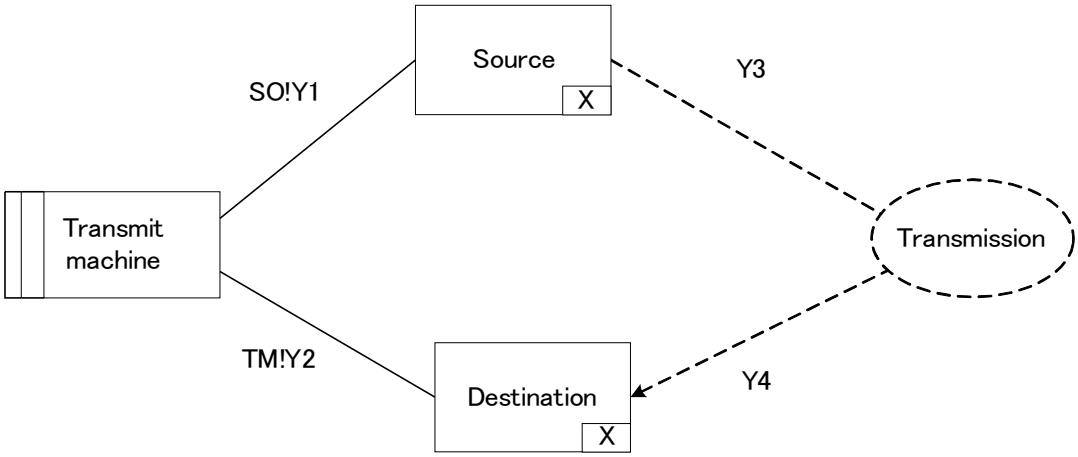


図 19　Transmission フレーム

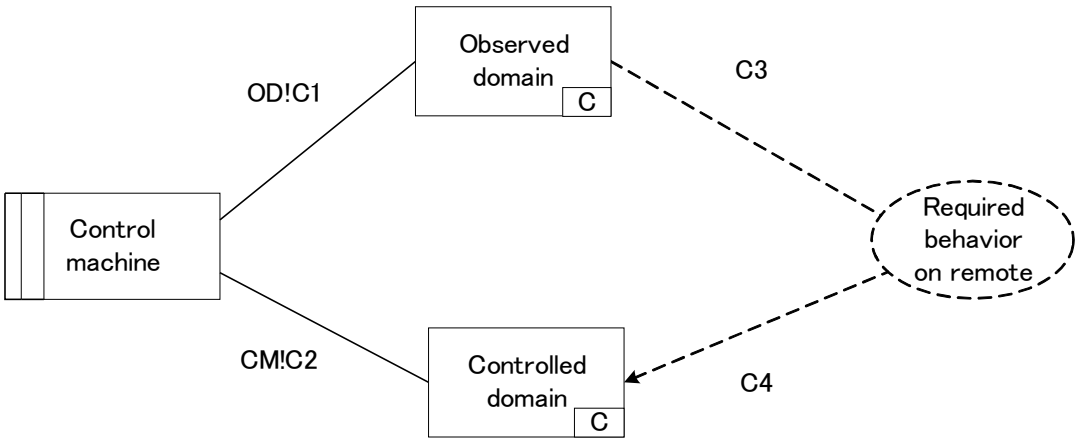


図 20　Remote required behavior フレーム

7.2.2概念的な領域について

Michael Jackson は「ローン」等の概念的な物は領域ではなくプロセスと考え、インターフェースの中の現象として表現するべきだという事を Conceptual flavors として述べているが、それだけでは不都合が出る問題の性質が分かりにくくなる場合も有りうる。もし概念的な領域が必要な場合は、図が解法の視点に近づいてしまうかもしれないが、敢えてファイルやデータベース等としてその領域を物理的な領域と同様に考えるべきである。ケーススタディでは、薬剤散布ヘリコプターの散布の「適正量」の領域をはじめ、このアプローチをしている。

これよりは狭い範囲のアプローチだが、概念的な領域のプロセスを Model building subproblem によってモデル領域にする、といった方法も Michael Jackson は紹介している

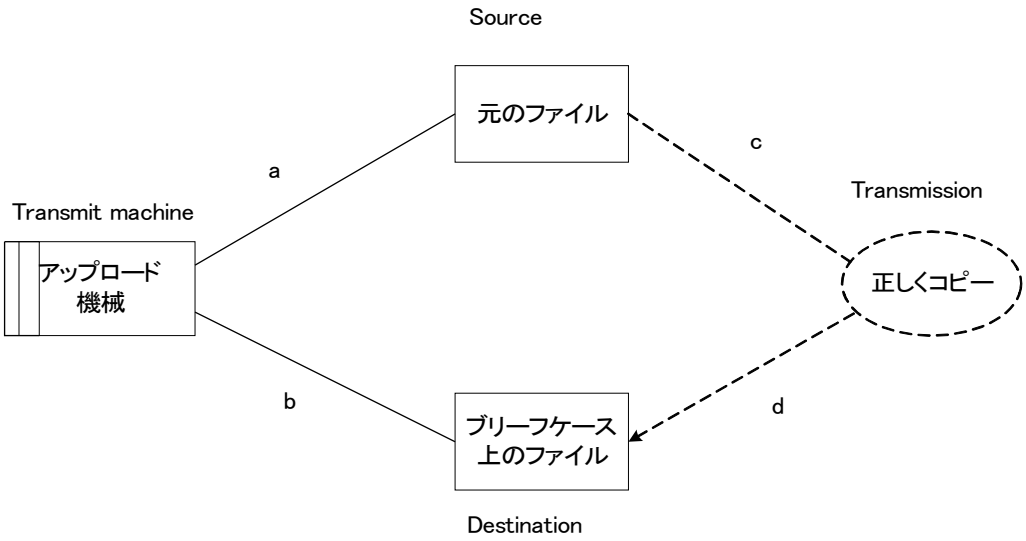
7.3 各改善案の背景および必要となるケース

各改善案の背景にある考え方を以下に示す。

- アーキタイプについて
  - 似通った形のフレームが多く、またそういったフレームの場合に複数のフレームに当てはまる問題が出てくる事から、似た形のフレームには共通する概念があると考えられる。なので、その共通の部分を抽出する事が必要と考えた。さらにオブジェクト指向における継承を参考とし、アーキタイプを考えた。
- 上位の問題図について
  - 問題の階層的な分割法から考えた。Jackson は並列的な分割のみ強調しているが、ある程度階層的な分割は必要となる。
- ディクショナリについて
  - データディクショナリを参考にした。問題図の補足的な説明を、文章表現のドキュメントを問題図に対応させる形で作成すべきと考えた。

各改善案が必要となるケースを以下に示す。

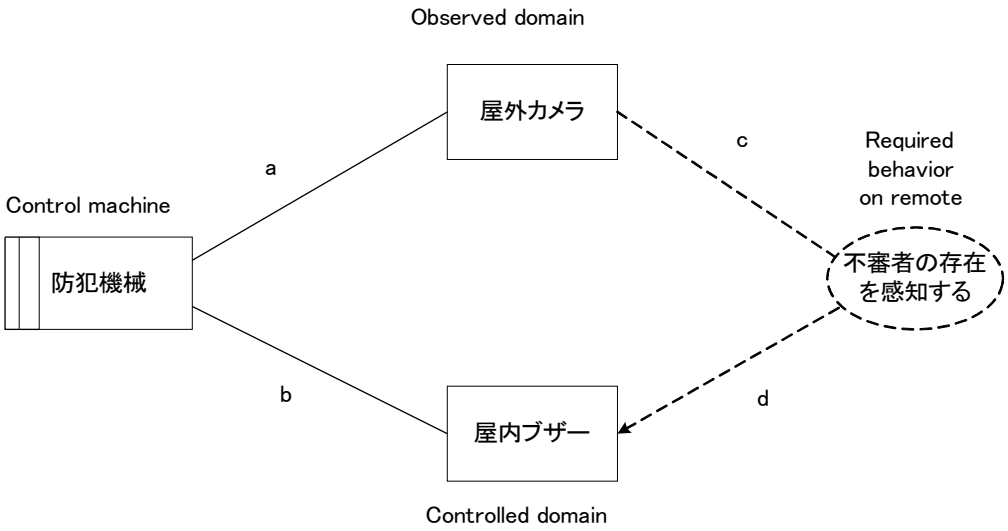
- アーキタイプについて
  - 必要になる例として 7.1.1 にて BE 式 CSU の応力の累積的記録の別パターンの図を作成した。
- 上位の問題図について
  - 大きく複雑な問題の場合、最小単位の問題図だけだと関係がわからなくなるため必要となる。
- ディクショナリについて
  - 作成した図の補足的な説明なので、常に必要となる。
- 新しい問題フレームについて
  - 例として新たな問題フレームを使った問題図を作成した。Transmission フレームが必要になる例として図 21 にオンラインブリーフケースにおけるファイルのアップロード問題を示す。また Remote required behavior フレームが必要な例として防犯システムの問題について図 22 に示す。



a : 元フ! {ReceiveData}　　c : 元フ! {SourceData}

b : ブフ! {WriteData}　　d : ブフ! {DestinationData}

図 21　オンラインブリーフケース：Transmission フレーム



a : 屋外! {Image}　　c : 屋外! {SuspiciousPSN}

b : 屋内! {BeepCMD}　　d : 屋内! {Beep}

図 22　防犯システム：Remote required behavior フレーム

8　終わりに

8.1 まとめ

この論文では、まず問題フレームを利用した要件定義のケーススタディを行った。そしてケーススタディを基に問題フレームという方法の評価と改善案の提唱を行った。改善案によってケーススタディの時に起きていた不都合が軽減されたので、より有効な方法となった。

問題フレームの特徴の一つとして Michael Jackson が主張している事に、問題に対して階層的な分割ではなく並列的な分割を行うべきといった事があるが、階層的な分割の概念を完全に排除する事は問題同士の相互関係を分かりにくくしてしまう事が分かった。なので、並列的な分割と階層的な分割を併用する事がより自然な問題の記述に繋がる。

また別の主張として、物理的な領域だけを図に記述するべきともあるが、概念的な領域を完全に排除する事は場合によっては図を難解にしてしまうので、ある程度の妥協が必要なことも分かった。

また問題フレームというアイディアそのものではなく、問題フレームの図の描き方について Michael Jackson が説明している範囲について考えると、厳密に見ると説明不足な所や曖昧な部分が残っている。

- 新しい問題フレームや新しい Variant の定義の仕方については全く触れられていないので、利用者の思い思いのやり方で定義されてしまう。
- いくつかの例ではフレーム名を明記せず問題図を描いている。当てはまるフレームが明らかだからということも考えられるが、これは問題の無い行為なのか。
- 一つの図に機械や要件、つまり問題を２つ以上描く事は意味のある行為なのか。Michael Jackson は一つの図に機械領域は一つと言っているが、複数の機械および複数の問題を一つの図に掲載している箇所もある。

これらの部分の明確な定義が求められる。

8.2 研究の将来の展望

本研究の将来の展望としては、問題フレームを利用した CASE ツールの開発が考えられる。作成したメタモデルを基に図の作成規則を設定し、問題フレームの図の記述を行うツールが実装できる。またツールでは本研究における改善案もサポートし、以下の機能を盛り込むことが考えられる。

- 領域や問題図からディクショナリの項目や文章表現のドキュメントへの対応付け
- ディクショナリの作成
- 問題図の階層化のサポート
- アーキタイプの設定
- 新たな問題フレームの定義のサポート

参考文献

[1] Robert L. Glass, *Facts and Fallacies of Software Engineering*, Addison-Wesley, 2003.

[2] Frederick P. Brooks Jr【著】, 滝沢徹, 牧野祐子, 富澤昇【訳】, *人月の神話　新装版　狼人間を撃つ銀の弾はない*, ピアソン・エデュケーション, 2002.

[3] Michael Jackson, *Problem Frames: Analyzing and structuring software development problems*, Addison-Wesley, 2001.

[4] Michael Jackson【著】, 玉井哲雄, 酒匂寛【訳】, *ソフトウェア要求と仕様: 実線、原理、偏見の辞典*, 新紀元社, 2004.

[5] Laura Scharer, "Pinpointing Requirements", *Datamation*, 1981.

[6] Robin Laney, Leonor Barroca, Michael Jackson and Bashar Nuseibeh, "Composing Requirements Using Problem Frames", 12th IEEE International Requirements Engineering Conference, 2004.

[7] Shari Lawrence Pfleeger【著】, 堀内泰輔【訳】,*Software Engineering: Theory and Practice*, ピアソン・エデュケーション, 2001.

[8] 小柳容子, 山根章弘, 白石剛之, 岸昌俊, 金子弘行, "無人ヘリコプタ RPH2 による全自動薬散システム", *SUBARU TECHNICAL REVIEW 2005 No.32*, 富士重工業, 2005.

[9] 伊藤義和, 大北義明, "バケットエレベータ式連続アンローダの予防保全システム", *住友重機械技報 No.159 2005*, 住友重機械工業, 2005.

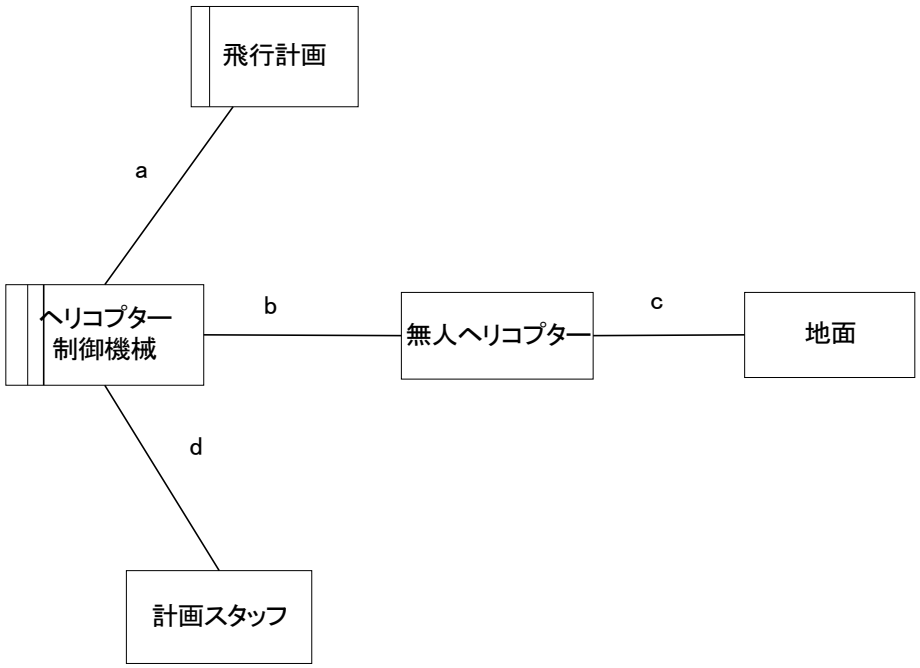
謝辞

本研究の機会を下さりご指導頂いた法政大学情報科学研究科の溝口徹夫教授に厚く御礼申し上げます。本研究は情報科学研究科における研究としては独特の切り口であり、私にとって大変貴重な経験となりました。

付録

A 問題フレームの記述

A.1 薬剤散布ヘリコプターの図



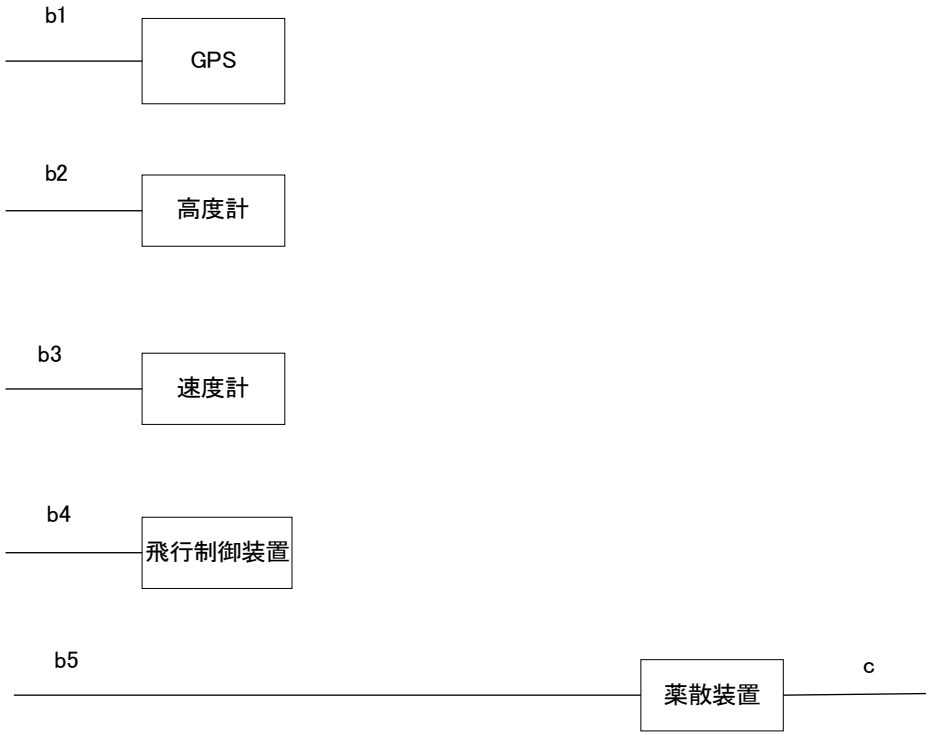
a : FlightPlanParam, ExceptedArea, etc

b : Commands, Sensors

c : Apply

d : EnterFlightPlan,
EnterExpectedFiled

全体文脈図



b1 : HorizontalPosStatus,
DirectionStatus

b2 : AltitudeValue

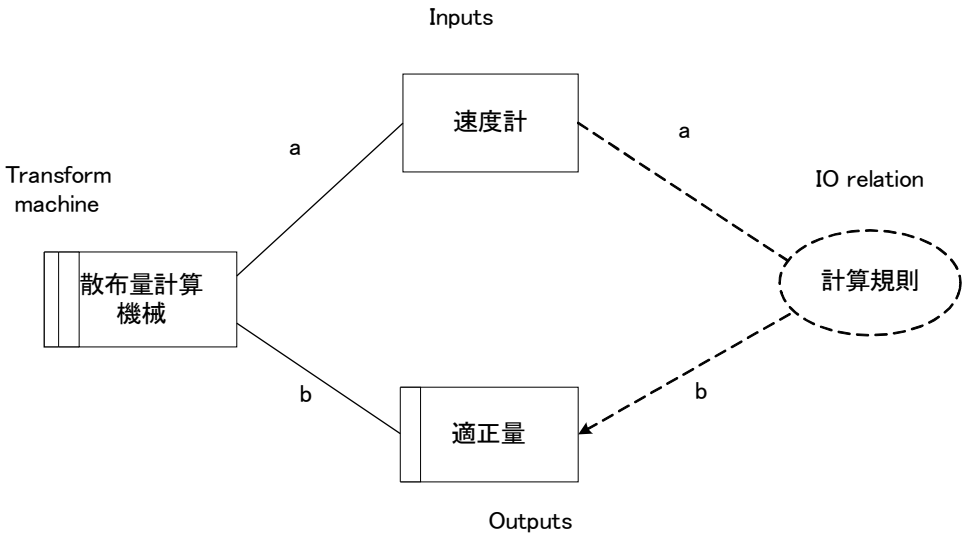
b3 : SpeedValue

b4 : AdvanceCMD, RotateCMD,
HoverCMD, TakeOffCMD, LandCMD,
SpeedUpCMD, SpeedDownCMD
RiseCMD, FallCMD

b5 : StrtAppCMD, StpAppCMD,
IncreaceCMD, DecreaceCMD

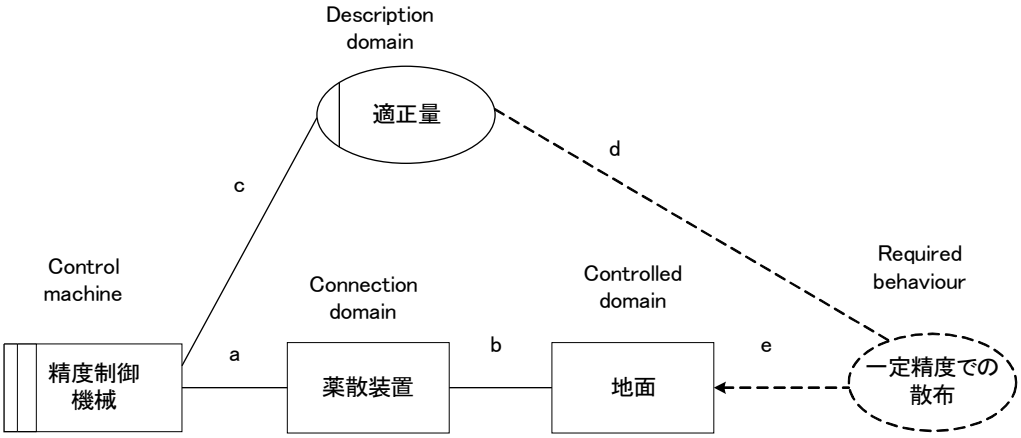
c : Apply

Unmanned Helicopter領域の文脈図



a : 速計! {SpeedValue}　　b : 散機! {Amount}

散布量計算: Transformation frame



a : 精機! {StrtAppCMD,
StpAppCMD,
IncreaceCMD, DecreaceCMD}

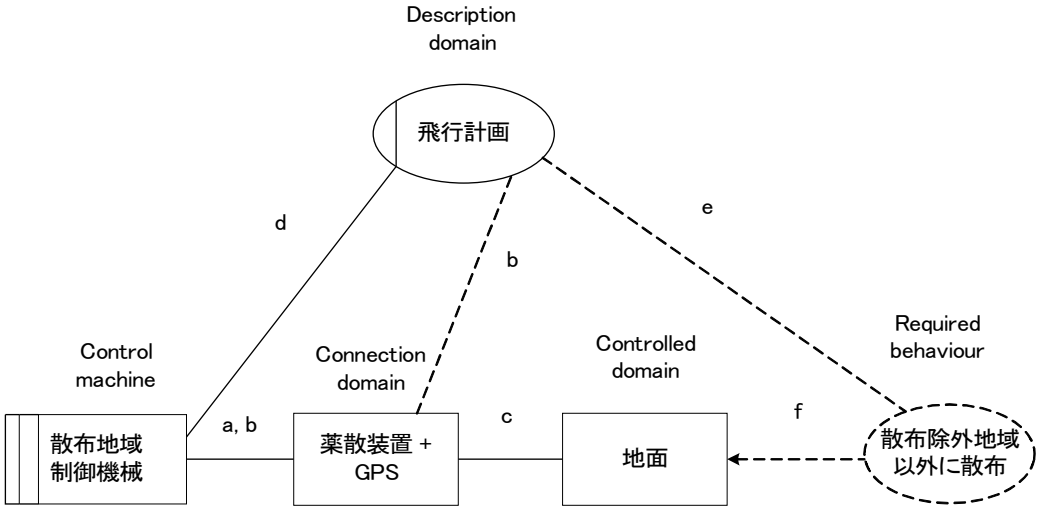
b : 薬散! {Apply}

c : 適量! {AmountParameter}

d : 適量!{Amount}

e : 地面!{Chemical}

散布精度の制御: required behavior frameのdescription variant,
connection variant



a : 散機! {StrtAppCMD, StpAppCMD}

d : 飛計! {Polygon, Point}

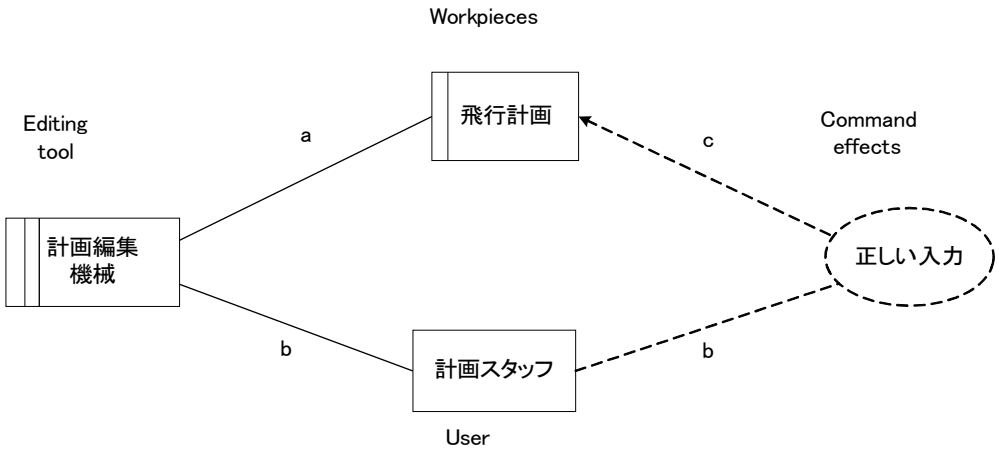
b : 薬G! {HorizontalPosStatus, DirectionStatus}

e : 飛計! {StartAppPoint, StopAppPoint, ExceptedArea}

c : 薬G! {Apply}

f : 地面! {Chemical}

散布地域の制御: required behavior frameのdescription variant, connection variant

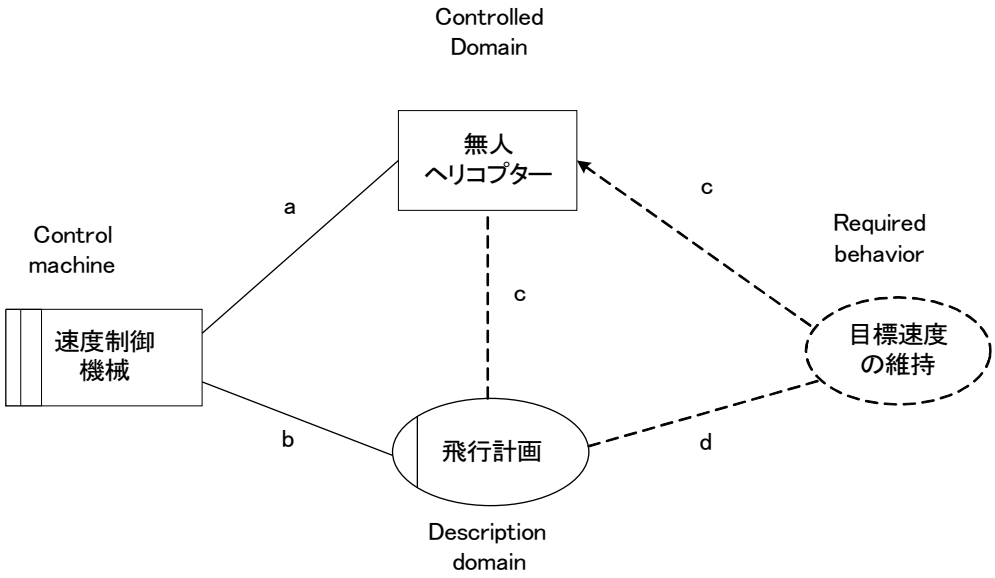


a : 計機! {PlanOperations}
飛計! {PlanStatus}

c: 飛計! {FlightForm, StartAppPoint, StopAppPoint, LandingPoint, TargetPoint, TargetSpeed, RotatingDirection, RotatingRadius, HoverringTime, HoverringDirection, ExpectedFiled}

b : 計ス! {EnterFlightForm, EnterStartAppPoint, EnterStopAppPoint, EnterLandingPoint, EnterTargetPoint, EnterTargetSpeed, EnterRotatingDirection, EnterRotatingRadius, EnterHoverringTime, EnterHoverringDirection, EnterExpectedFiled}

飛行計画の入力: Simple workpieces frame



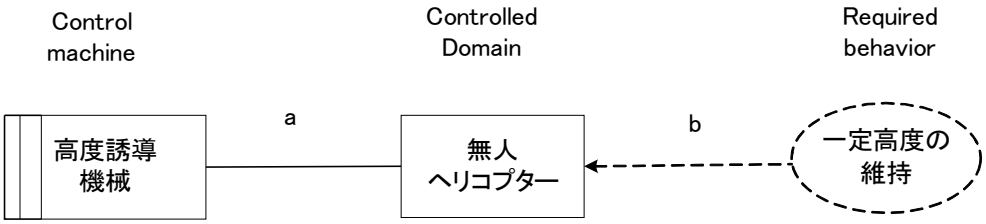
a : 無へ! {SpeedValue}
速機! {SpeedUpCMD,
SpeedDownCMD}

c : 無へ! {Speed}

b: 飛計! {TargetSpeedParam}

d : 飛計!{TargetSpeed}

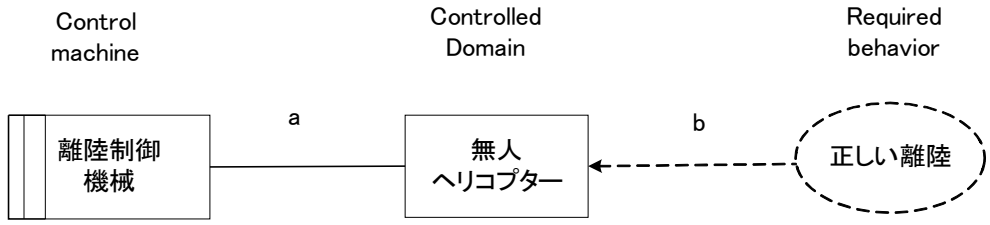
速度制御: required behavior frameのdescription variant



a : 高機! {ClimbCMD, DecendCMD}
無へ! {AltitudeValue}

b : 無へ! {Altitude}

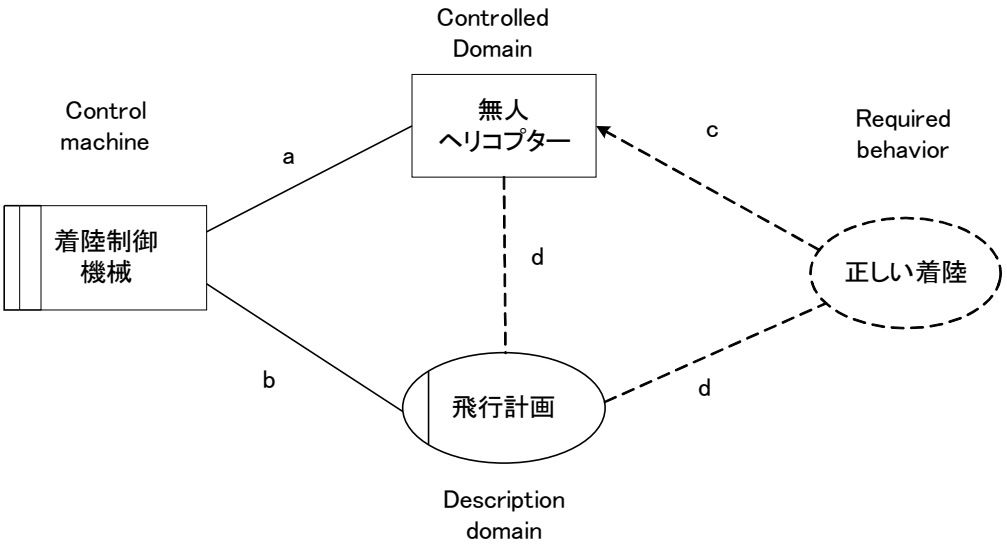
高度誘導: required behavior frame



a : 離機! {TakeOffCMD}
無へ! {AltitudeValue}

b : 無へ! {TakingOff}

離陸: required behavior frame



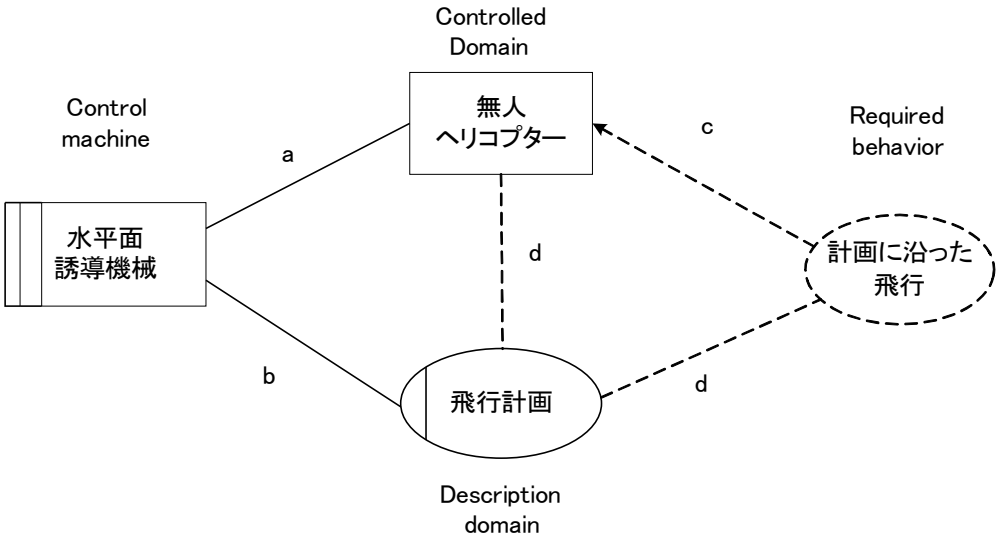
a : 着機! {LandCMD}
無へ! {AltitudeValue, HorizontalPosStatus, DirectionStatus}

c : 無へ! {Landing}

b : 飛計!{PointParameter, DirectionParameter}

d: 飛計!{LandingPoint}

着陸: required behavior frameの description variant



a : 水機! {AdvanceCMD, RotateCMD, HoverCMD}
無へ! {HorizontalPosStatus, DirectionStatus}

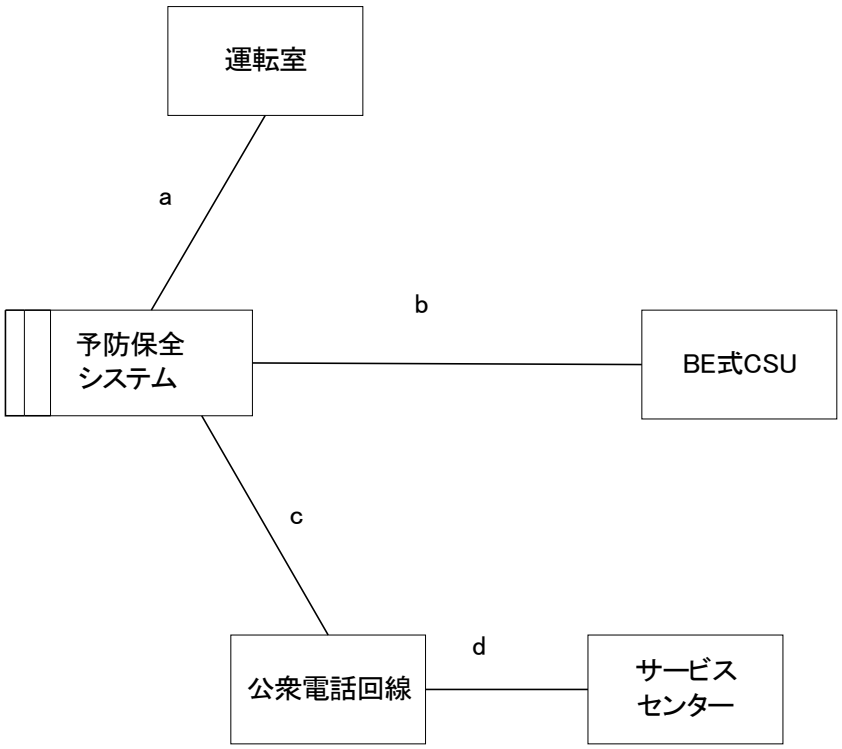
c : 無へ! {HorizonalPosition, Direction}

b : 飛計!{PointParameter, RDirectionParameter, HDirectionParameter, FormParameter, RadiusParameter, TimeParameter}

d: 飛計!{FlightForm, TargetPoint, RotatingDirection, RotatingRadius, HoverringTime, HoverringDirection}

水平面誘導: required behavior frameのdescription variant

A.2 バケットエレベーター式連続アンローダーの図



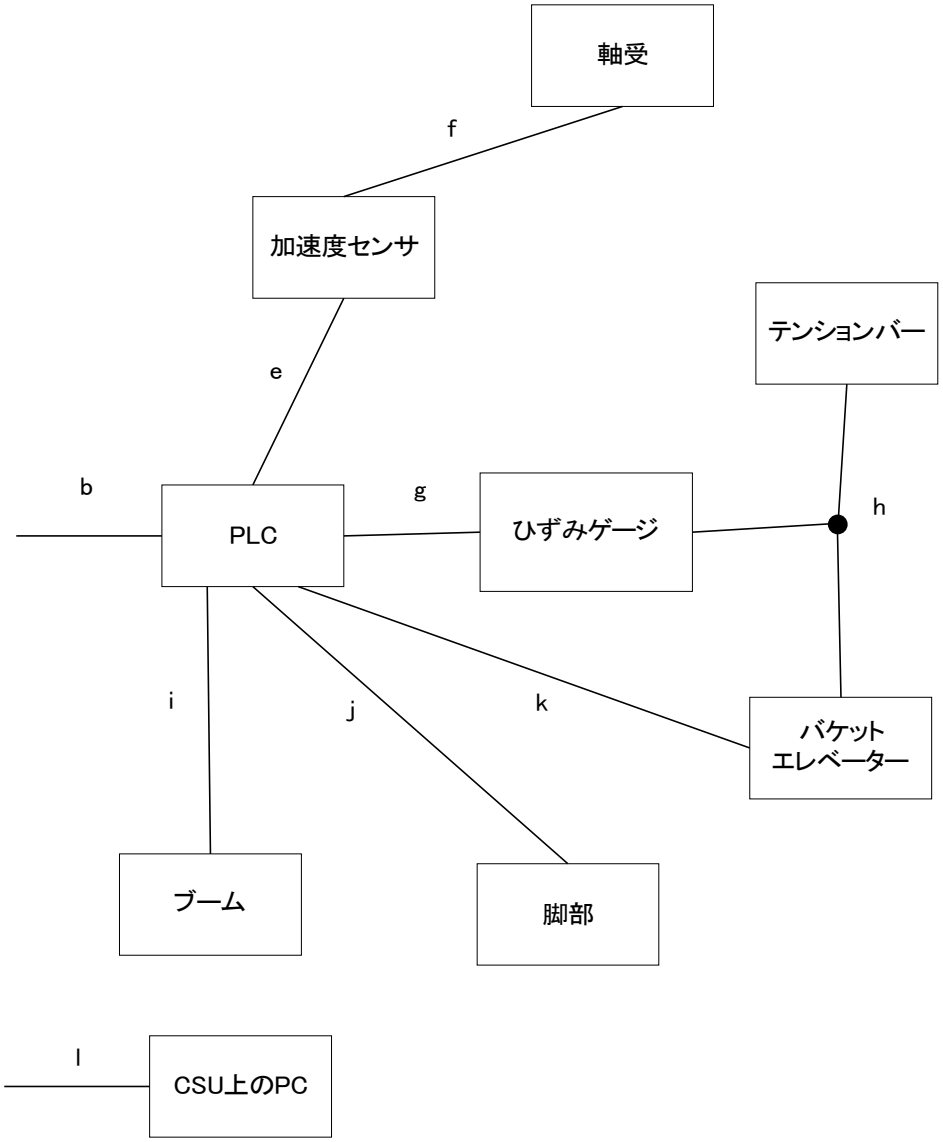
a : Warning

b : SensorValue, Command

c : SendData

d : ReceiveData

全体文脈図



b : SensorValue, Command

e : AccelerationValue

f : Acceleration

g : MeasuredStess

h : Deformation

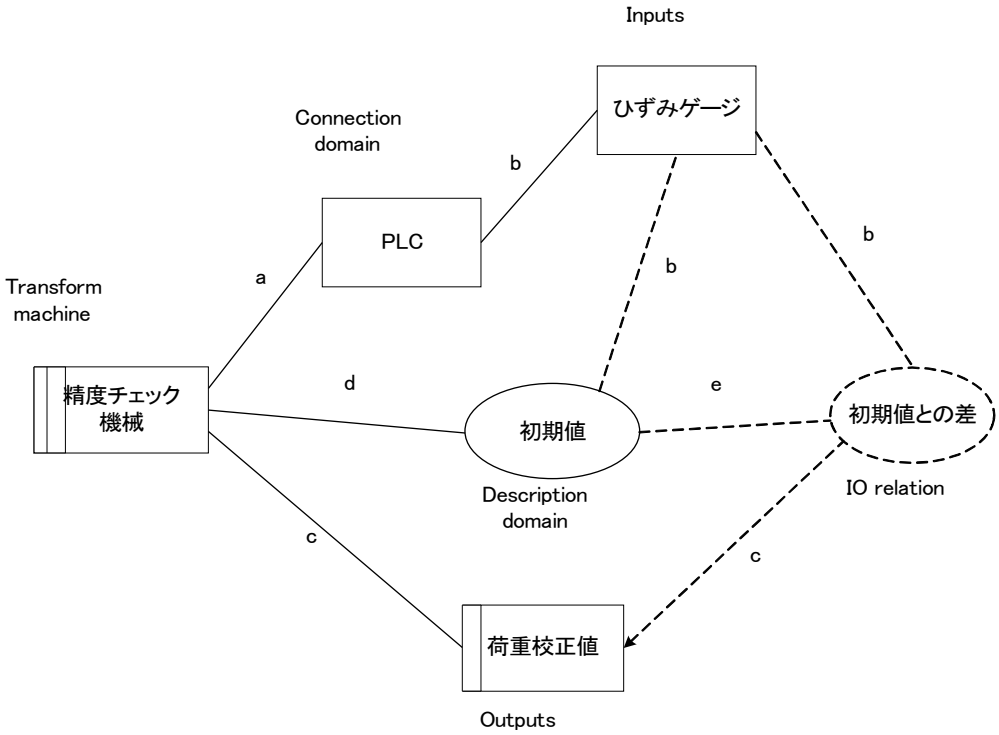
i : ControlBoom

j : ControlCruising

k : ControlBE

l : ScreenOpns, Data

CSUの文脈図



a : PL! {DigitalizedStress}

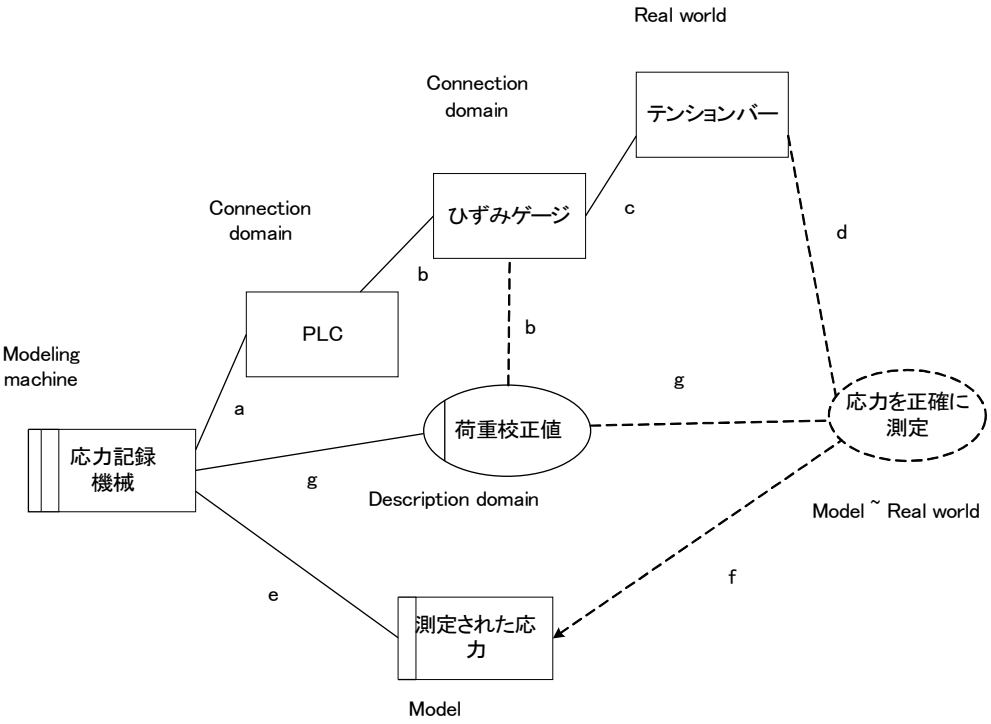
b : ひず![MesuredStress}

c : 精機! {CalibrationValue}

d : 初期! {Value}

e : 初期! {InitialValue}

精度チェック: transformation frameの connection variant, description variant



a : PL! {DigitalizedStress}

b : ひず! {MeasuredStress}

c : テン! {Deformation}

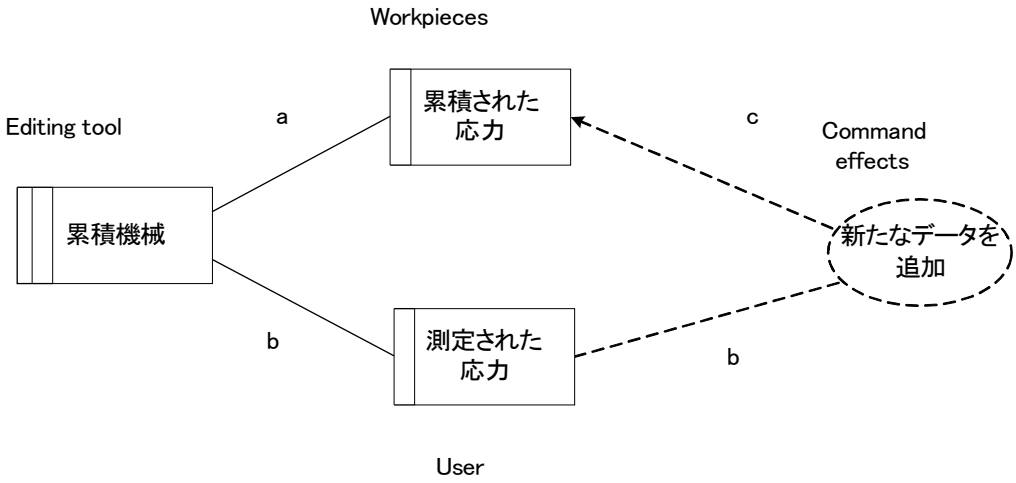
d : テン! {Stress}

e : 測機! {StressValue}

f : 累応! {Data}

g : 荷校! {CalibrationValue}

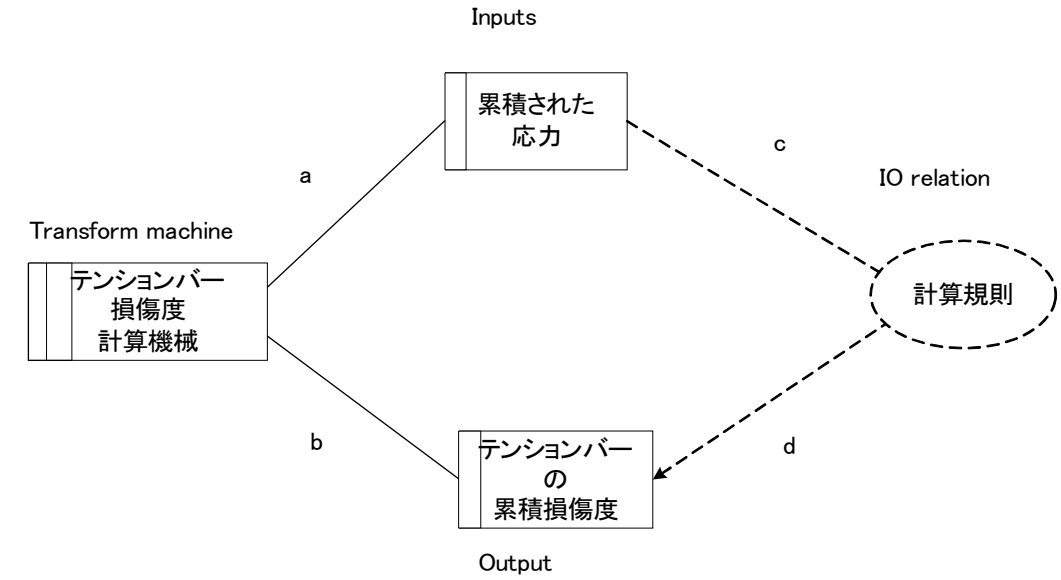
応力の累積的記録: model building subproblem frame のconnection variant, description variant



a : 累機! {AddData}　　c : 累応! {CumulatedData}

b : 累機! {ReadData}
測応! {StressData}

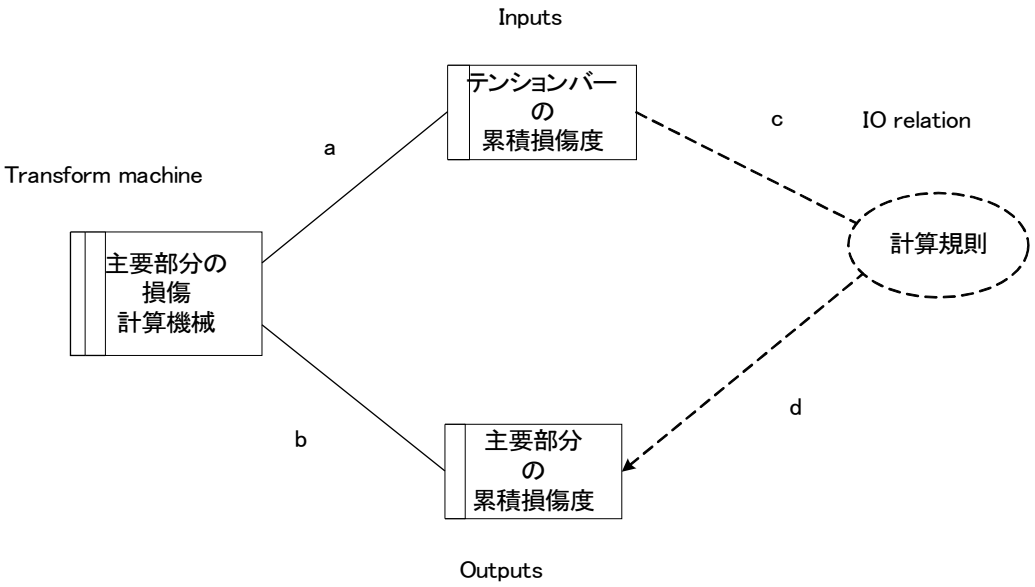
応力の累積的記録: simple workpieces frameのcontrol variant



a : 累応! {StressValues}　　c : 累応! {StressData}

b : テ機! {DamageValue}　　d : テ損! {Damage}

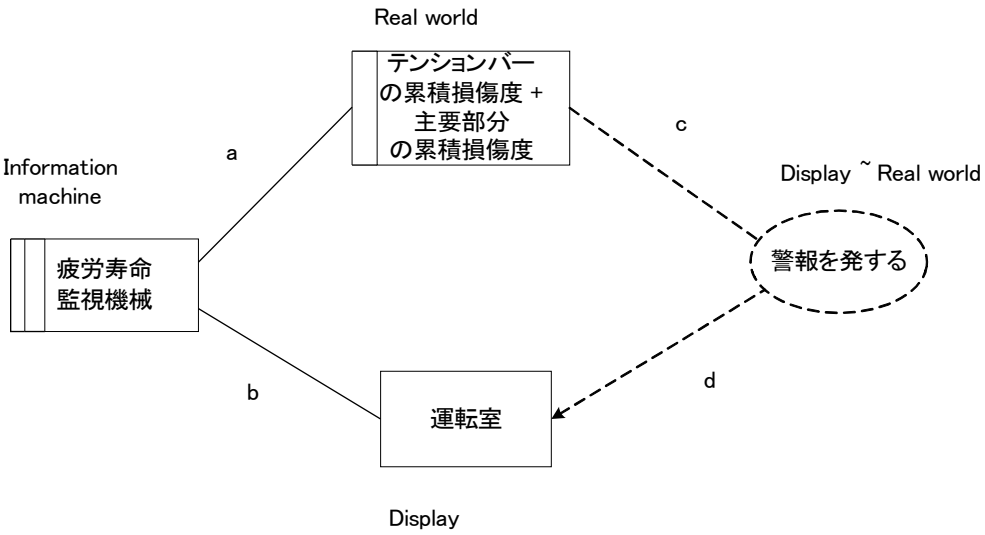
テンションバーの累積損傷度の計算 : transformation frame



a : テ損! {TBDamageValue}　　c : テ損! {TBDamage}

b : 主機! {MPDamageValues}　　d : 主損! {MPDamages}

テンションバー以外の累積損傷度の計算：transformation frame



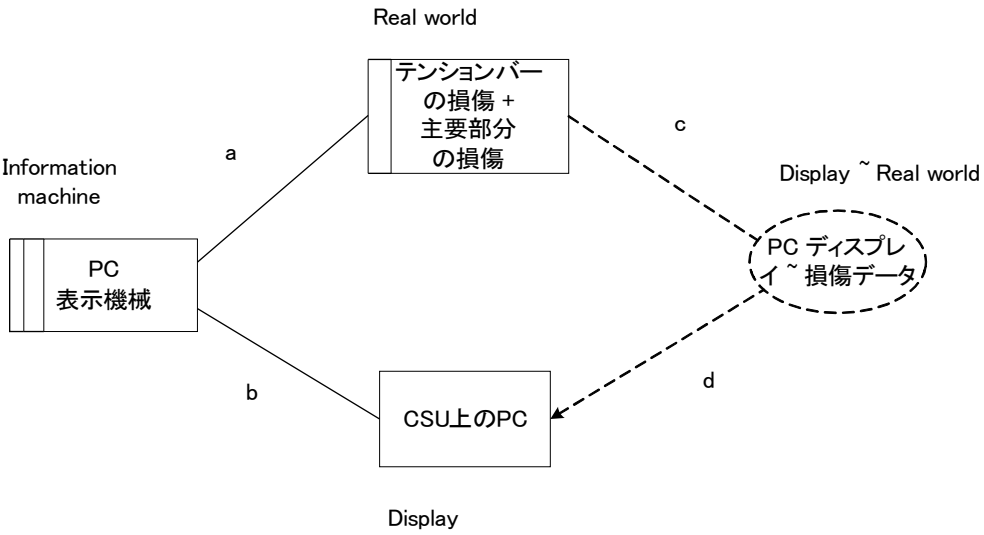
a : テ主! {DamageValues}

b : 疲寿! {ScreenOpns}

c : テ主 {Damages}

d : 運転! {Warnings}

疲労寿命監視システム : information display frame



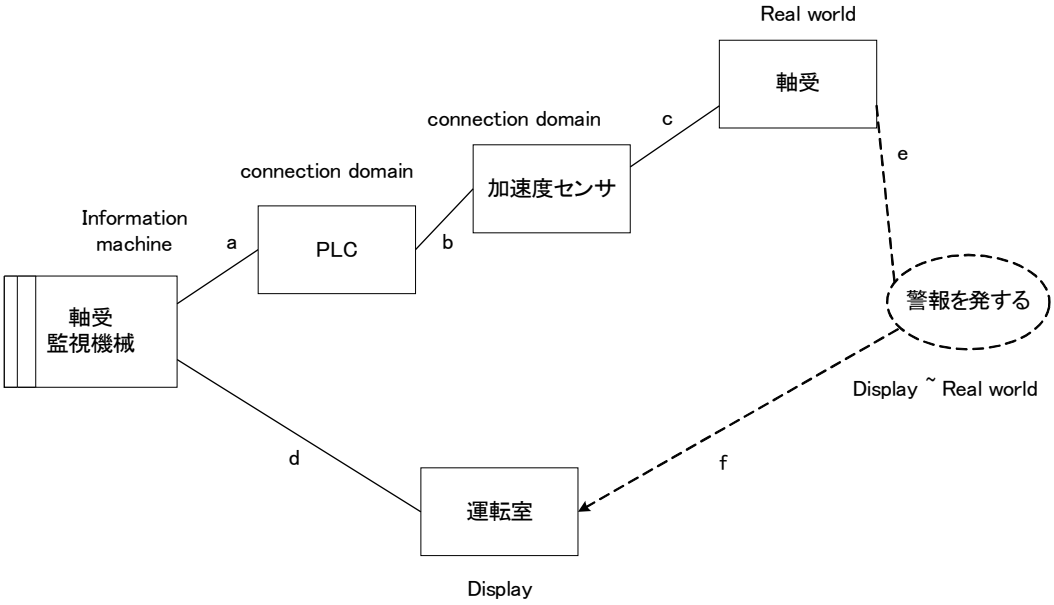
a : テ主! {DamageValues}

b : P表! {ScreenOpns}

c : テ主 {Damages}

d : CS! {PCInformation}

累積損傷度のPCへの表示: information display frame



a : PL! {DigAcc}

b : 加速! {MesuredAcc}

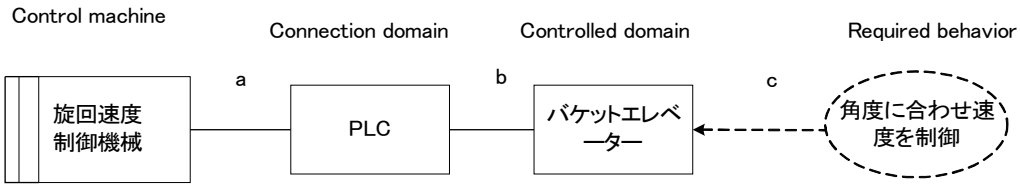
c : 軸受! {Acceleration}

d : 軸機! {ScreenOpns}

e : 軸受! {Movements}

f : 運転! {Warnings}

軸受異常監査システム : information display frameのconnection variant

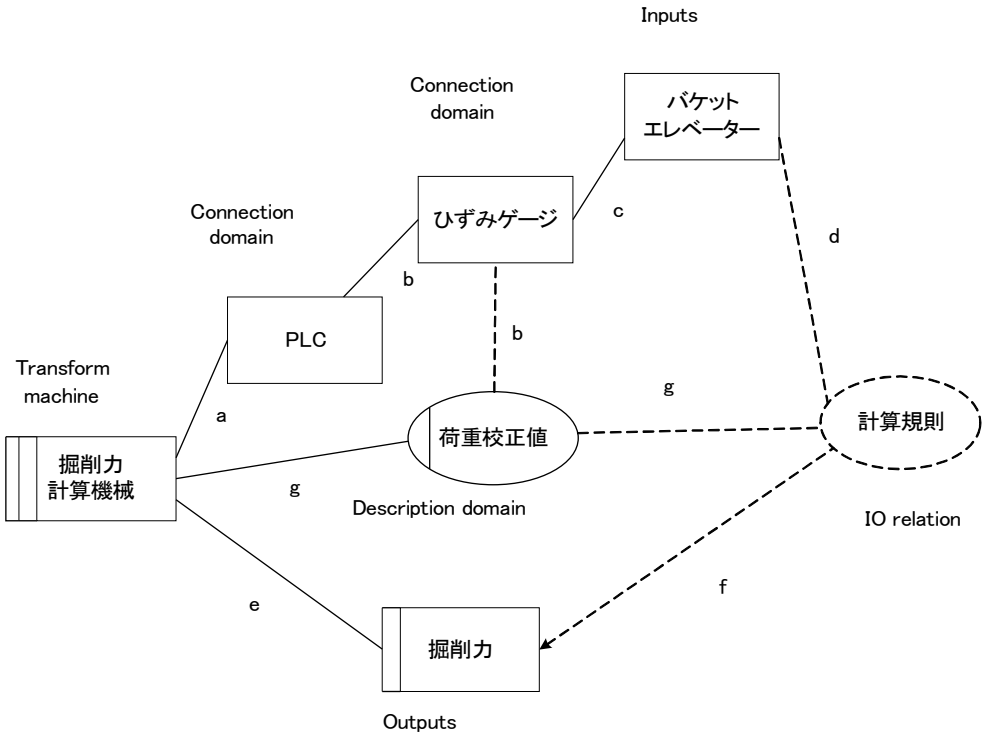


a : PL! {Anglevalue}
旋機! {SpeedUpCMD,
SpeedDownCMD}

b : バケ! {Angle}
PL! {SpeedUp, SpeedDown}

c : バケ! {Speed}

BE旋回速度制御システム : required behavior frameのconnection variant



a : PL! {DigitalizedStress}

b : ひず! {MeasuredStress}

c : バケ! {Deformation}

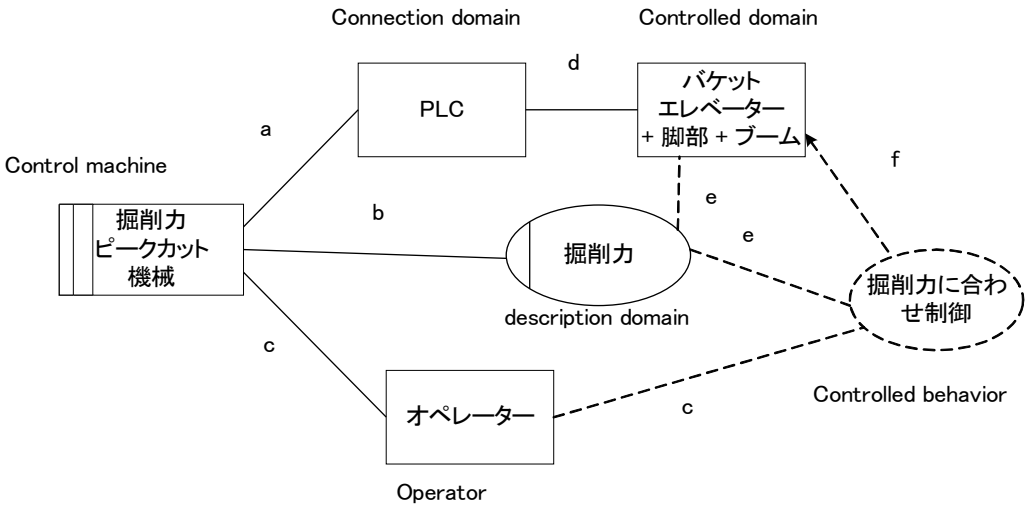
d : バケ! {Stress}

e : 掘機! {PowerValue}

f : 掘力! {Power}

g : 荷重! {CalibrationValue}

掘削力の計算: transformation frameのconnection variant, description variant



a : 掘機! {Level1CMD, Level2CMD, Level3CMD, RestartCMD}

b : 掘削! {PowerValue}

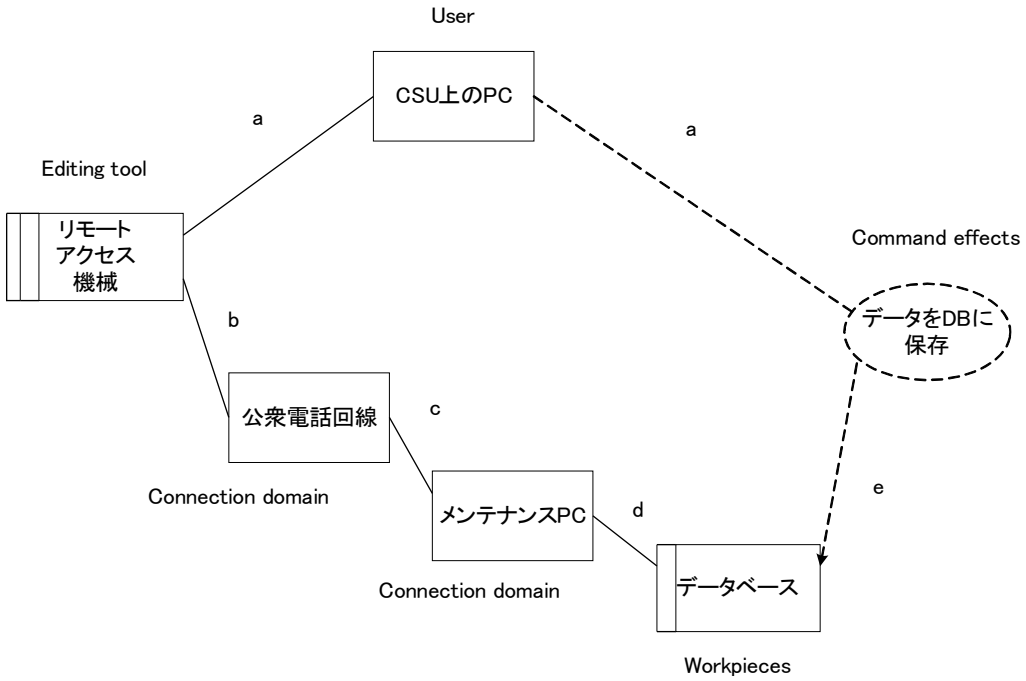
c : オペ! {ResetCMD}

d : PL! {SlwdnBacket, SlwdnBoomRot, SlwdnTravel, SlwdnBERot, StopBoom, StopTrabel, StopUpDown, StopBERto, StopSwing, StopAll, Restart}

e : 掘削! {Power}

f : バケ! {Movement}

掘削力ピーククカットシステム : commanded behaviorのconnection variant, description variant



a : CS! {SourceData}

b : リモ! {SendData}

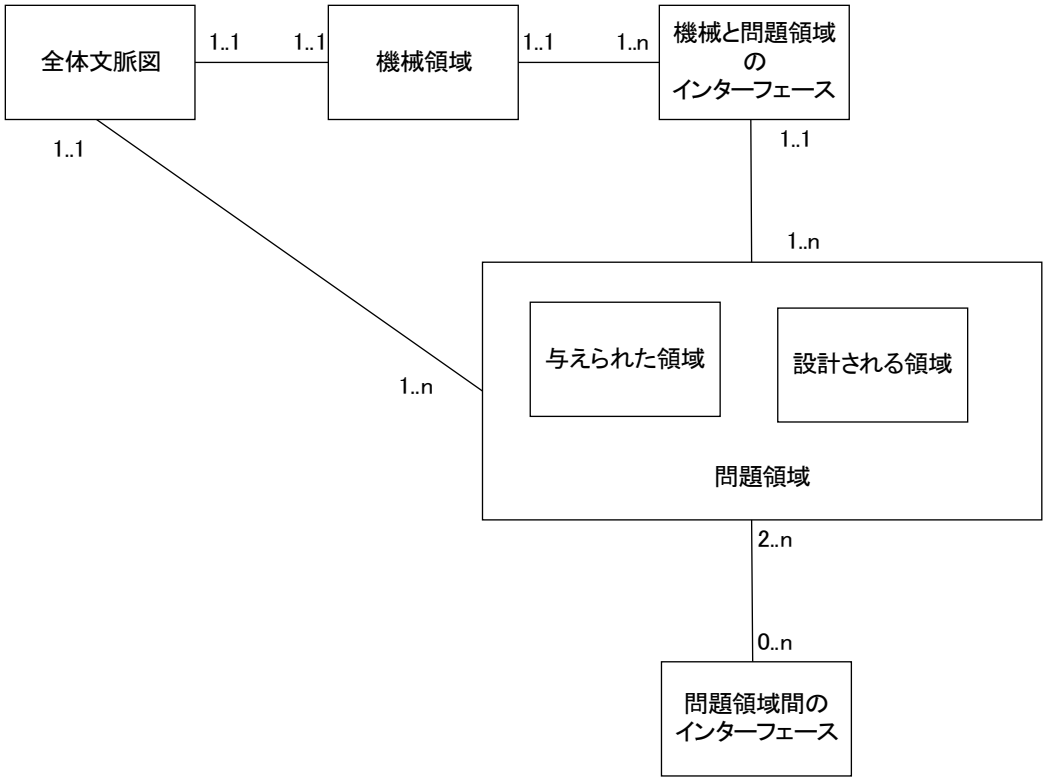
c : 公衆! {ReciveData}

d : メン! {StoreData}
デー! {DBStatus}

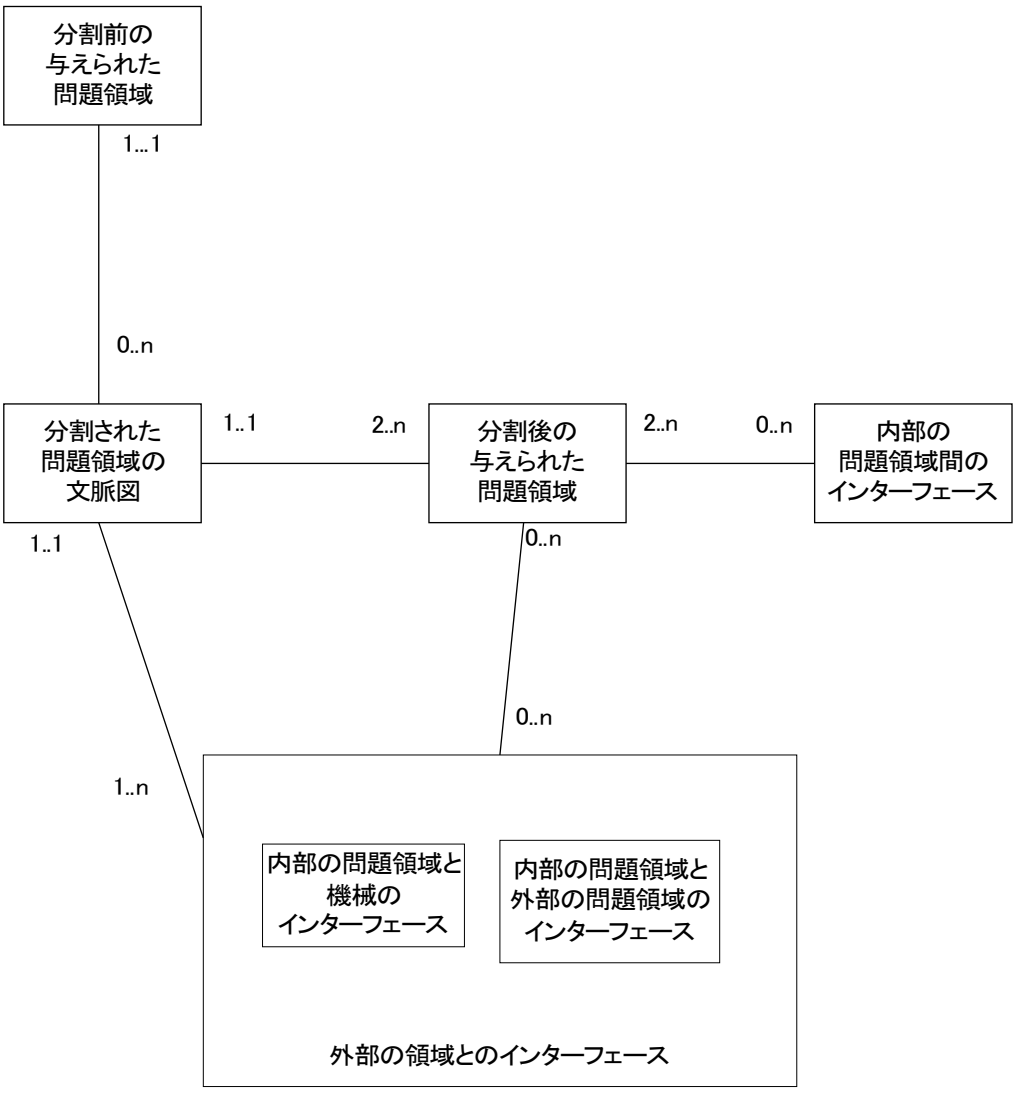
e : デー! {DBValue}

サービスセンターへのリモートアクセス : simple workpieces frameのconnection variant

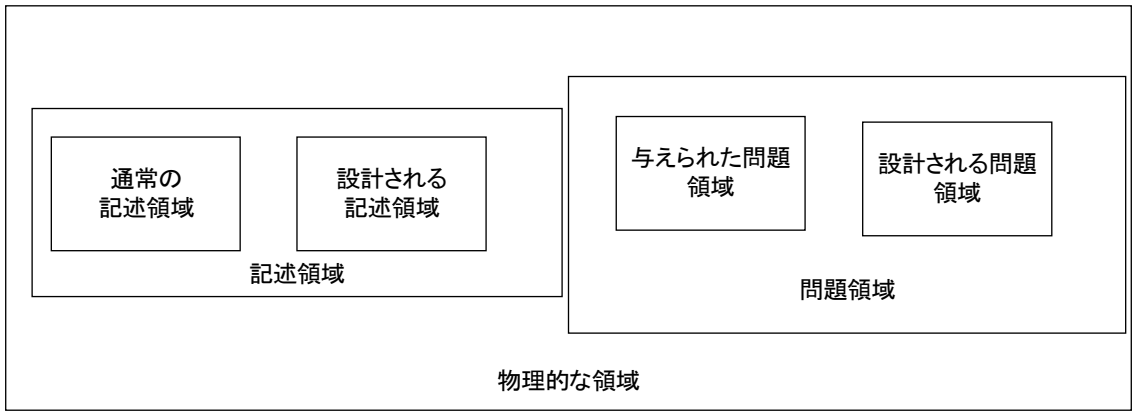
B 問題フレームのメタモデル



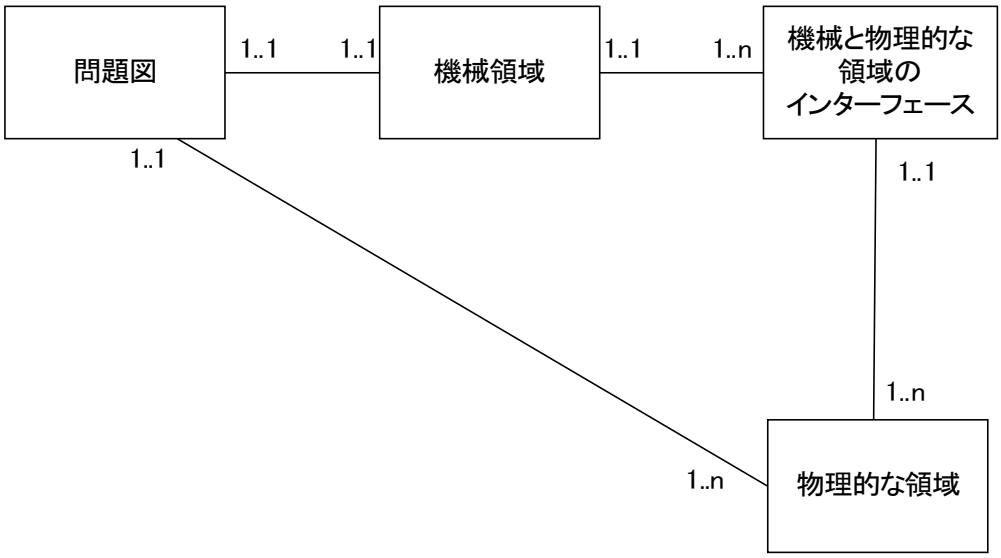
全体文脈図のメタモデル



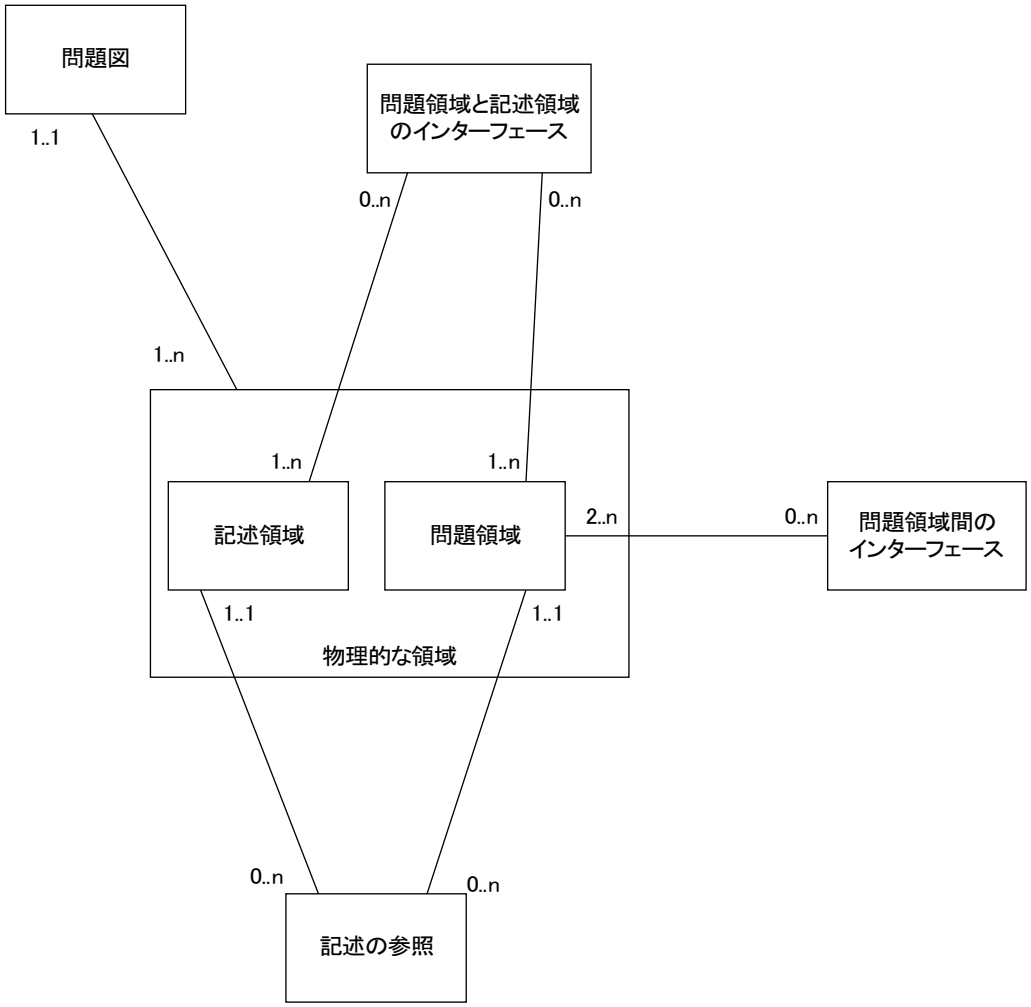
分割された問題領域の文脈図のメタモデル



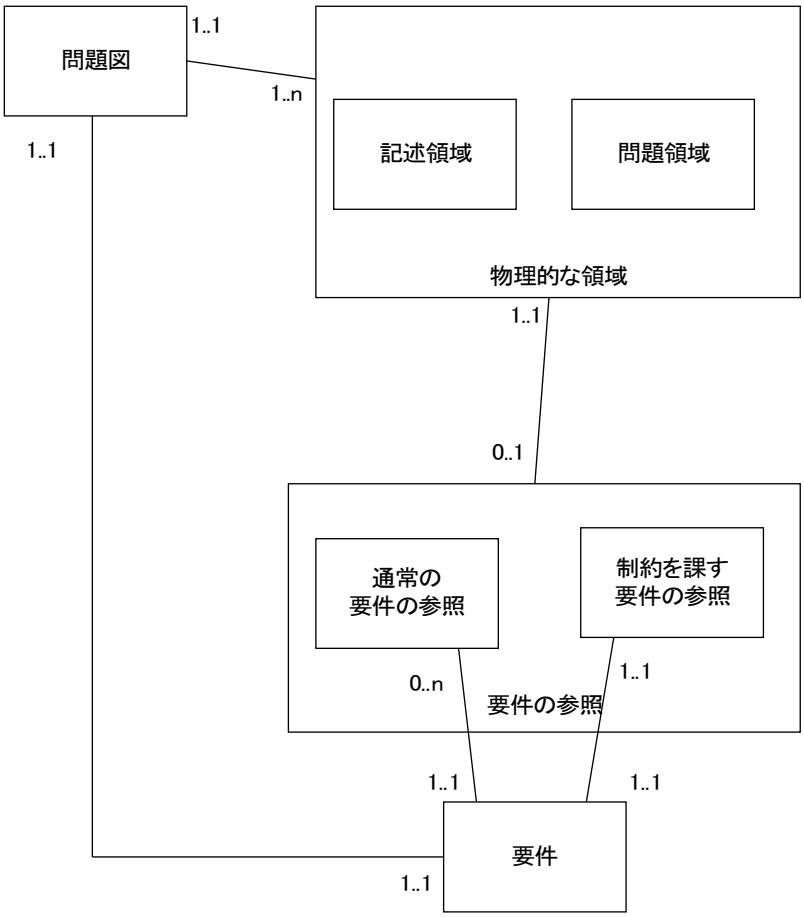
物理的な領域のサブクラス



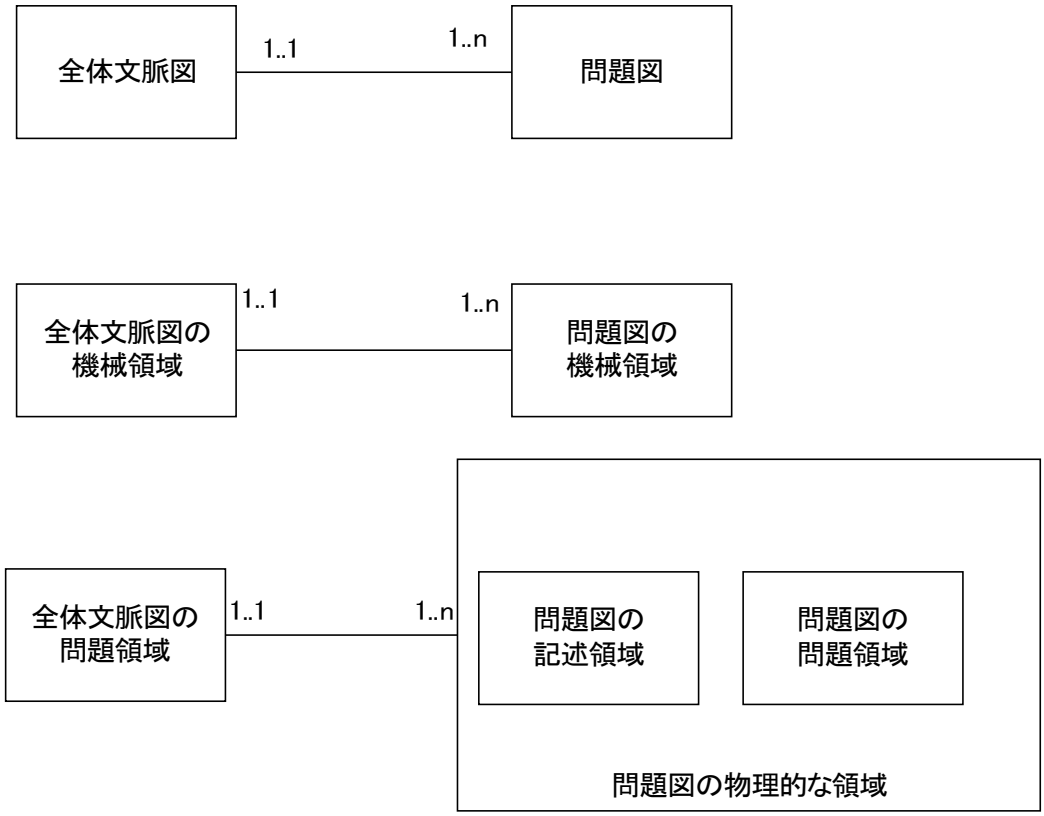
問題図のメタモデル
機械領域と物理的な領域について



問題図のメタモデル
物理的な領域の間のインターフェースについて



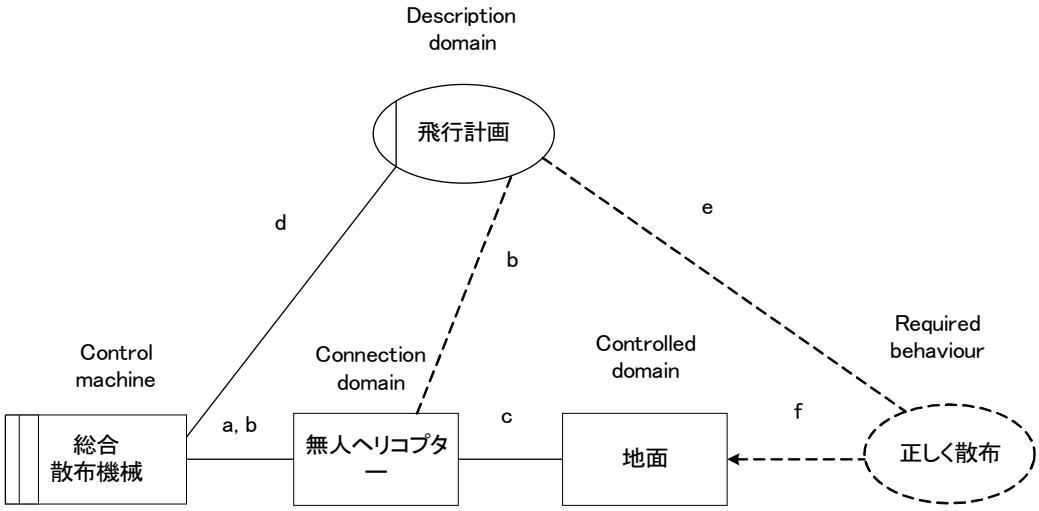
問題図のメタモデル
要件と物理的な領域について



全体文脈図と問題図の関係

C 問題図の階層化

C.1 薬剤散布ヘリコプターの図の階層化



a : 総散! {AppCMD}

d : 飛計! {PlanData}

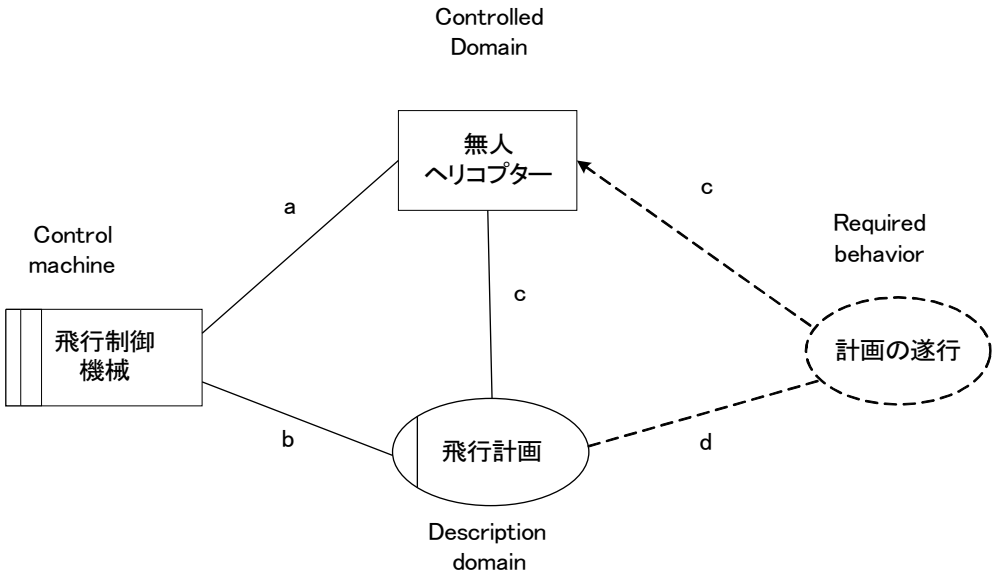
b : 無へ! {CurrentStatus}

e : 飛計! {Plan}

c : 薬G! {Apply}

f : 地面! {Chemical}

薬剤の散布: required behavior frameのdescription variant, connection variant



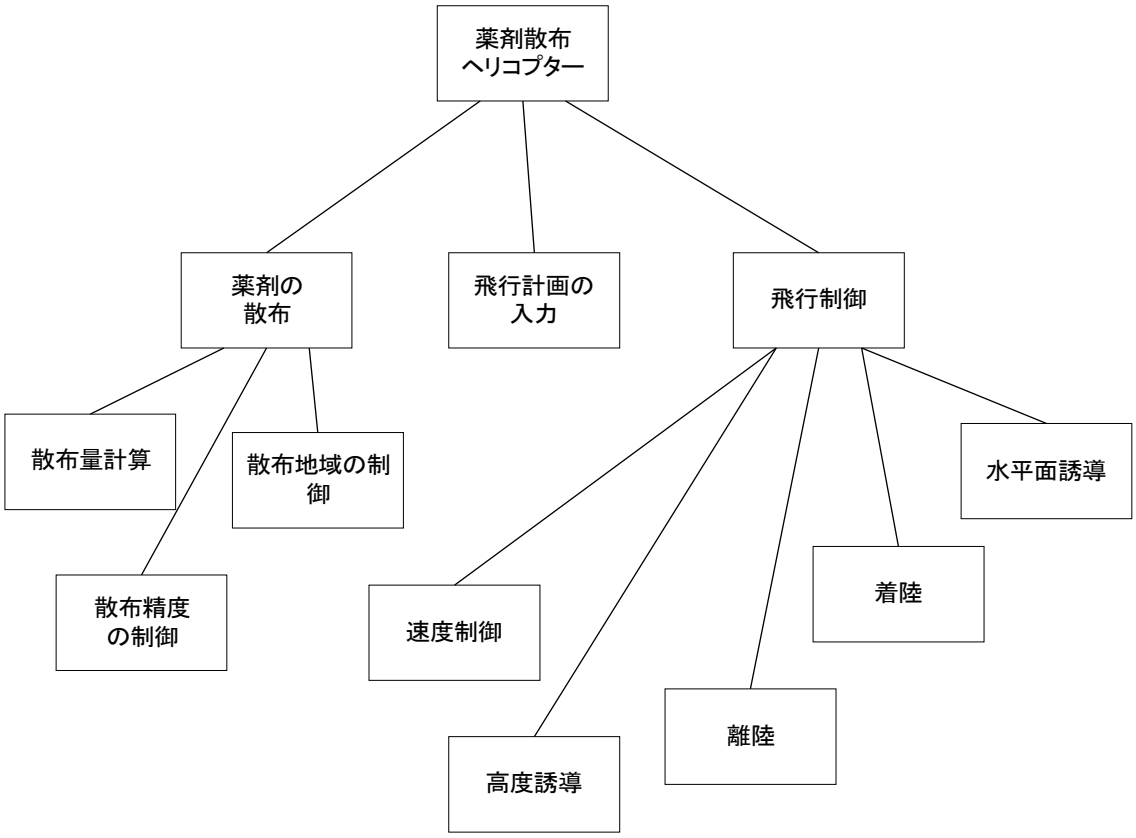
a : 無へ! {CurrentStatusValue}
飛機! {FlightCMD}

c : 無へ! {FlightStatus}

b: 飛計! {TargetParam}

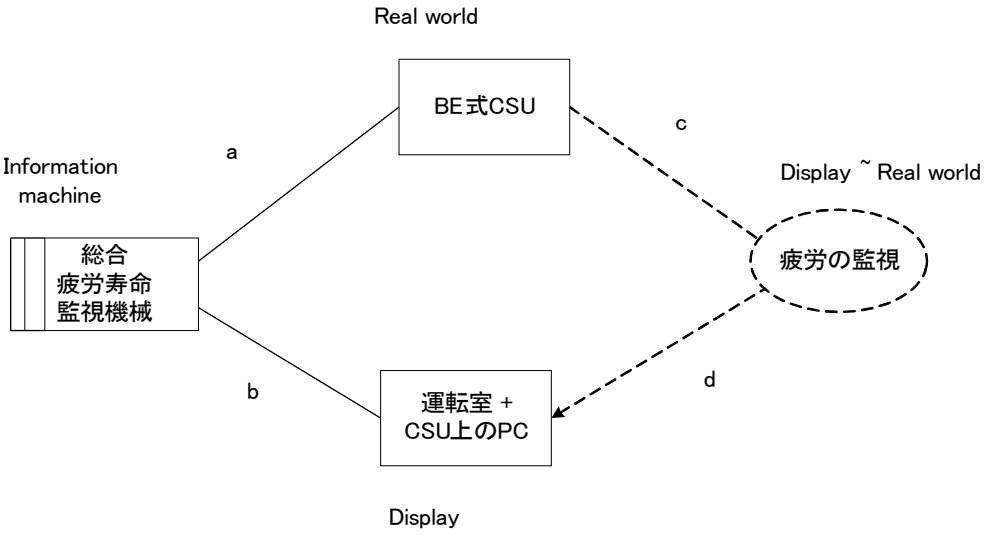
d : 飛計!{TargetStatus}

飛行制御: required behavior frameのdescription variant


薬剤散布
ヘリコプター
薬剤の
散布
飛行計画の
入力
飛行制御
散布量計算
散布地域の制
御
散布精度
の制御
速度制御
高度誘導
離陸
着陸
水平面誘導

問題の階層図

C.2 バケットエレベーター式連続アンローダーの図の階層化



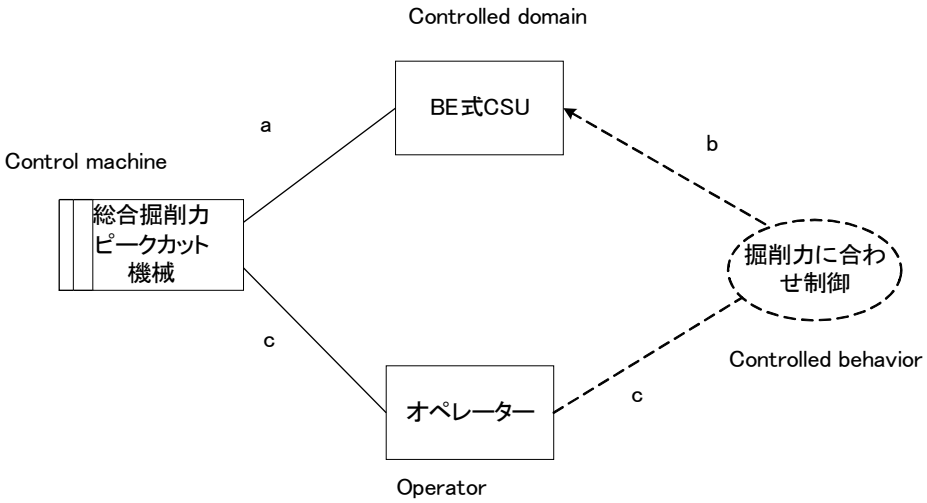
a : テ主! {Stress}

b : 総疲! {ScreenOpns}

c : テ主 {Damages}

d : 運C! {Information}

総合的疲労寿命監視システム : information display frame

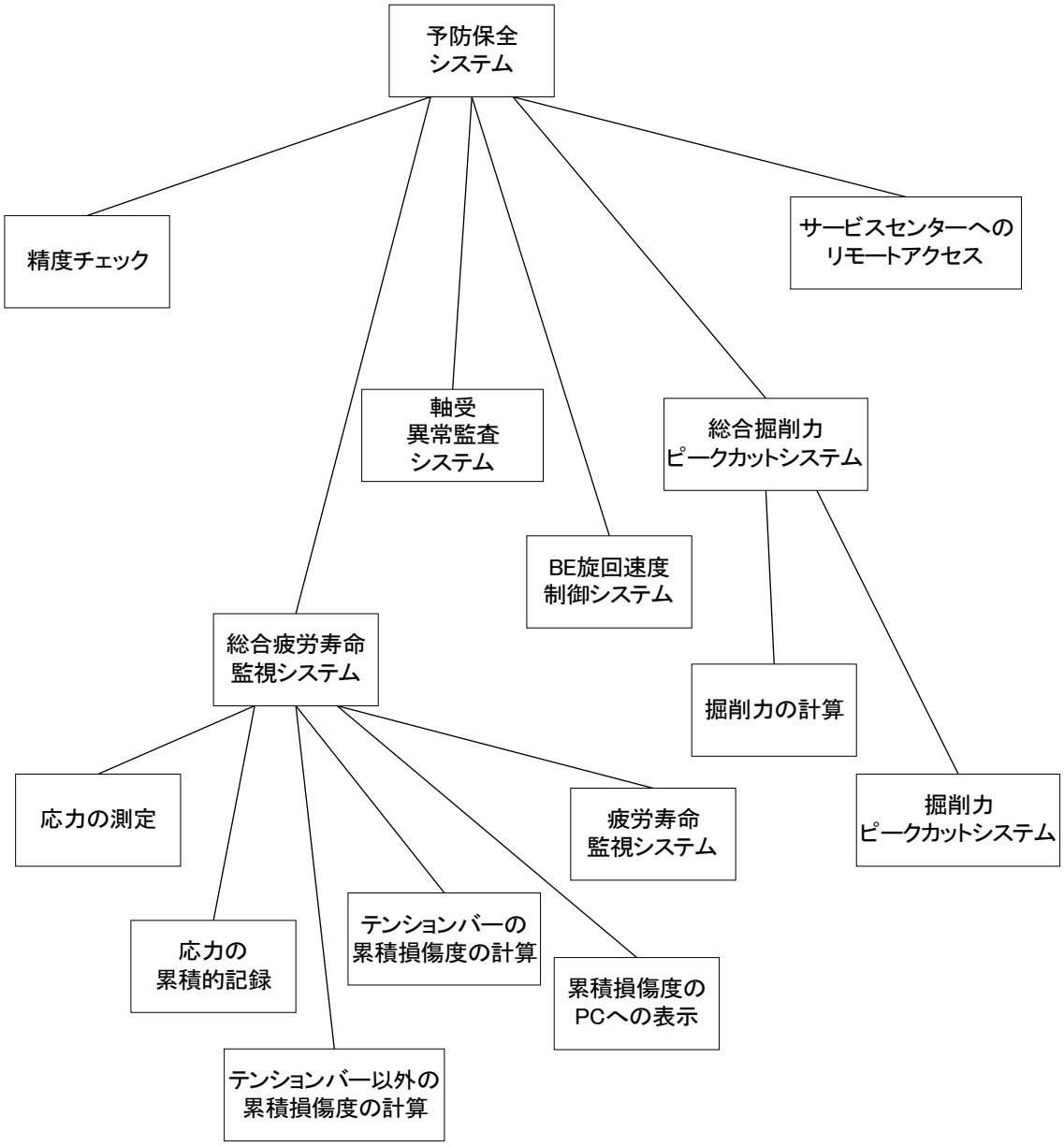


a : 総掘! {ControlCMD}
バケ! {Power}

c : オペ! {ResetCMD}

b : バケ! {Movement}

総合掘削力ピークカットシステム : comannded behavior


予防保全
システム
精度チェック
軸受
異常監査
システム
総合掘削力
ピークカットシステム
サービスセンターへの
リモートアクセス
BE旋回速度
制御システム
総合疲労寿命
監視システム
掘削力の計算
応力の測定
疲労寿命
監視システム
掘削力
ピークカットシステム
テンションバーの
累積損傷度の計算
応力の
累積的記録
累積損傷度の
PCへの表示
テンションバー以外の
累積損傷度の計算

問題の階層図