要求工学イブニングチュートリアル
第4回
ゴール指向要求分析法

2004年4月22日

信州大学
海谷 治彦

目次

- 要求工学におけるゴール指向分析の役割
  - ゴール分析の利点
- ゴール分析の基礎 – 典型的な記法等
- ゴール指向手法の代表例他
  - i* (I star)
  - KAOS
  - NFR framework
  - GQM
  - AGORA (時間があれば)
- まとめと論点
  - RE04の紹介を少々

講師紹介

- 1999年～ 信州大学 工学部
  - 要求工学を利害対立や妥協の観点から研究
    - そのためのツールとしてゴール分析手法を利用

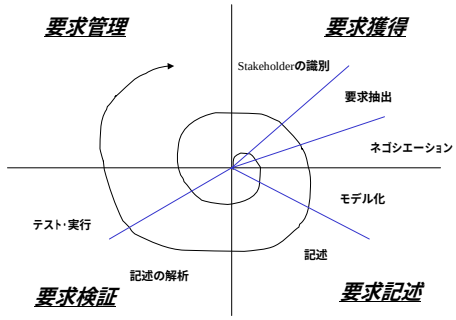
ゴール指向分析の役割

要求工学, ソフトウェア工学の
文脈において

要求工学 (復習)

- ソフトウェア工学の一部
- ソフトウェア要求を正しくまとめる(定義する)ための技術や技法の集大成
- ソフトウェア要求
  - 要求は立場によって異なる
  - 利用者は当たり前のことを言わない
  - 要求は誤っていることがある
  - 要求は変わることがある

要求定義のプロセス (復習)



各段階で必要なこと

- 獲得
  - 組織の現状(as-is)を理解する(利害関係者の識別等)
  - 現状をどのように変化させたいのかを理解する
- 記述
  - 組織の目標(ゴール)を，開発するシステムの機能や性能(非機能)と関連付ける
- 検証 (verification and/or validation)
  - 記述した(要求)仕様が利害関係者の本々の目標にあうかを確認する．

ゴールの明示的記述

- ゴールを明示的に記述することで，獲得，記述，検証で必要な活動を円滑に行うことができる．

ゴール(目標)とは

- the object of a person's ambition or effort (辞書的な意味)
- 開発するシステムが達成しなければならないこと．[Jac95,Zav97]
- プログラムが仕様を実現するのと同様に，ゴールは要求を実現する．[Kav02]
- ゴール=why, (要求)仕様=what, 実装=how

従来の分析手法

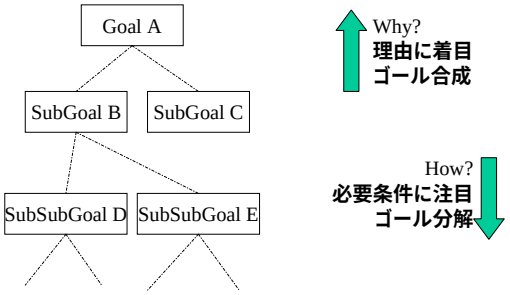
- プログラム技法をヒントにしている
  - 「理由」や「目標」を明示する部分が無い
  - 「what」に相当するものが一番抽象的．
- 構造化分析
  - データフロー図(コンテクスト図)で機能と境界を明確化
- オブジェクト指向分析
  - ユースケース図で機能と境界を明確化

ゴール分析の基礎

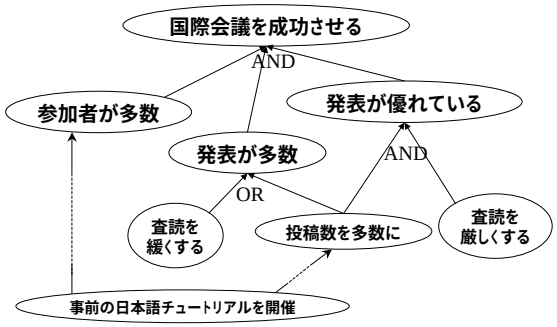
典型的なゴール構造の記述法

- ゴールを木(または非循環有向グラフ)構造で記述する．
  - 下位構造が上位構造の詳細化となっている．
- 分解をAND/ORに区別する．
  - AND 上位ゴールを達成するのに下位全ての達成が必要．
  - OR 上位ゴールを達成するのに下位どれかの達成が必要．
- 人工知能での問題分析手法の流用．

パターン



例



典型的なゴールの記述様式

- システムによって達成されるステークホルダの望むべき状態･状況を記述する.
  - 上位のゴールはこの形態の場合が多い.
- その状態･状況にするために行うことを記述する.
  - 下位のはこの形態が多い.
  - この形態で書かれたゴールは個々の要求項目 (例えばユースケースや機能項目)とみなしてよい場合がある.

ゴール以外の要素との関連

- ほとんどのゴール分析技法はゴール以外の要素も記述する.
  - ゴールの遂行者
  - ゴール達成の利害関係者
  - ゴールを達成するための手順や作業
  - 手順や作業の実行に必要な資源

  等
- 個々の技法でこれらは解説する.

Stakeholder

- ステークホルダ, 利害関係者
- (開発されるシステムの導入によって)起こりうる変化に利害関係を持つ人, 利益や損失が生じる可能性のある人 [Mac96]
- ステークホルダは(UMLやDFDでの)アクターとは異なる場合がある.
  - 利害関係はあっても運用には関係ない人もいるため.
- 「誰の」ゴールなのか？を考えることがゴール分析では重要となる.

代表的なゴール分析手法

要求工学の文脈において

i*, KAOS, NFR, GQM, AGORA

i* (eye star)

- An agent-oriented modelling framework
- 要求獲得に役立つ.
  - 現状のビジネスを理解
  - システム導入による効果を分析
- 5種類の要素をグラフで表現し，システム要求とそれに関連する情報を表現.

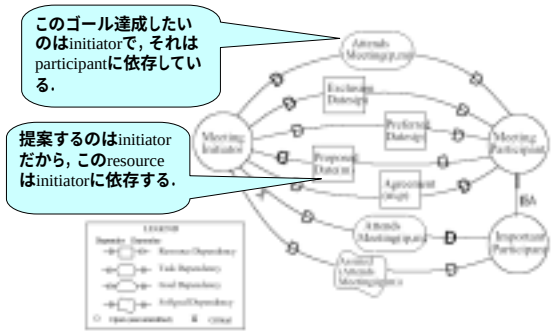
5つの要素

- **Actor**: 処理の遂行者だけでなく，目的や理由，言質を与える(コミットする)者(物)を表す．ステークホルダに近い.
- **Goal**: 遂行できるか否かを判断できる条件や状態．機能要求にほぼ対応.
- **Task**: あるgoalを達成する特定の手順.
- **Soft-goal**: 遂行の可否が明確に判断できないgoal．非機能(品質や性能)要求に対応.
- **Resource**: goal達成(task遂行)に利用できる物や情報.

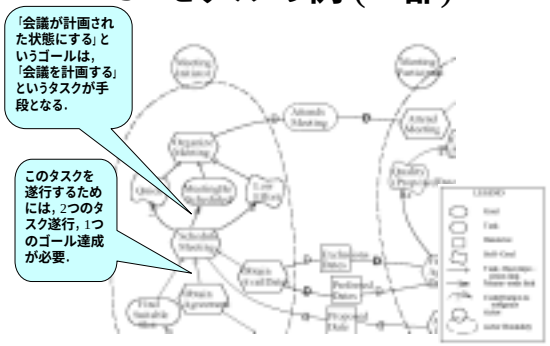
SD modelとSR model

- 現状(as-is)分析と望むべき状態(to-be)の分析をするための記述.
  - 前述の5要素を使ったグラフで記述する.
- Strategic Dependency (SD) model
  - Actor間の依存関係(dependency link)をgoal, task, resource, soft-goalそれぞれの観点から記述したグラフ.
  - それぞれの観点からの需要供給関係がわかる.
- Strategic Rationale (SR) model
  - 各actorが内部的にどんなgoalやsoft goalを持ち，どんなtaskを遂行し，どんなresourceを持つかをモデル化する.

SDモデルの例



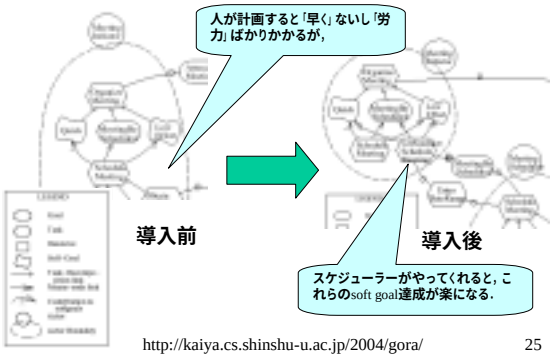
SRモデルの例 (一部)



i*による要求獲得戦略

- 現状の業務(as-is)をSD, SRモデルで記述.
  - 矛盾や不具合，人への仕事の負荷がある.
- 作成されたシステムが導入された場合(to-be)を同様にモデル化する.
  - 前述の，矛盾や不具合，負荷が軽減されていればシステムの価値はある.
  - 価値のあるようなシステムの要求を構築しなければならない.

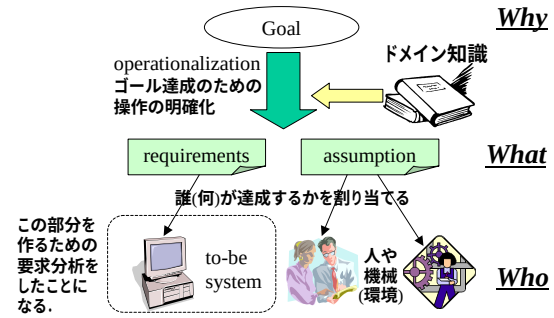
例: soft goal達成の改善



KAOS

- Knowledge Acquisition in autOmated Specification [Vl91]
- 識別したゴールを満足する要求項目を系統的に導出できる．
- 導出された要求項目によってゴールが達成されることを形式的に検証(verify)できる．
- 形式的: 数理論理の利用
  - 個々のゴール記述に様相論理を利用．
  - ゴール分解に矛盾が無いことに数学的な証明を利用．

KAOSでの要求分析の考え方



KAOSで利用する概念

- Goal: 目標とする状態．KAOSでは最終的にそのような状態を論理式で形式化する．
- Object: UML等で言うObjectにほぼ同じ．
- Operation: Goalを達成するための操作．
- Agent 個々のゴール達成に責任を持つ存在．当然，ある程度，詳細化･具体化しないと誰が責任を持つか解らない．
- Requirements 開発するソフトウェア(to-be software)が達成責任を持つゴール
- Assumption それ以外(環境)が達成責任を持つゴール

KAOSでの要求分析の手順

1. 初期文書からのゴールの識別
2. ゴールの形式記述とオブジェクトの識別
3. Why質問で上位のゴールを識別
4. How質問で具体的なゴールを識別
5. Agentのゴール達成責任を識別
6. Agentが観測･制御可能な変数を識別
7. Operationを識別

その他，障害予測や対立解消も可能．

1. ゴール識別

初期文書等からキーワードをもとにゴールを識別．

ほとんどはsoft goalなので，それを詳細化することで，形式化可能なゴール(formalizable goal)に分解してゆく．

ここでの例はBART(サンフランシスコにある鉄道)の分析をしている．

形式化されたゴールを分類するキーワード

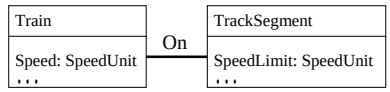
- Maintain
  - □ (P → Q)
  - – (状態が変化しても)真偽値Pが成り立てば，Qも成り立つ．
  - – 「→」は普通の論理的含意
- Avoid
  - □ (P → ¬ Q)
  - – いつでも，Pがなりたっていればなりたたない．
- Achieve
  - □ (P → ◇ Q)
  - – いつでも，Pが成り立っていれば，将来Qが成り立つ可能性がある．

これらはゴールの傾向を示すマークに過ぎない．

2. ゴールの形式記述･オブジェクト識別

Goal Maintain[TrackSegmentSpeedLimit]
InformalDef Trainは各TrackSegmentで決められたSpeedLimit以下を維持しないといけない．
FormalDef ∀tr: Train, s: TrackSegment:
□(On(tr,s) → tr.Speed ≦ s.SpeedLimit)

上記のように形式化されたゴールを記述する．
記述にオブジェクトとその属性が必要なのでクラス図を書く．



5. Agentの責任識別

- 形式化されたゴールの達成責任があるAgentを決める．
- 責任が決められない場合は，さらにゴール分解する必要がある．
- 基本的には1つのゴールを1つのAgentが責任を持つ必要がある．

左記は表記法の例

6. Agentが観測･制御可能な変数

- 形式化ゴールは論理式で記述されている．
- その式はオブジェクトの属性で構成されている．
- そのゴールに責任を持ったAgentは，
  - – 達成のため属性を制御できないといけない．
  - – 必要に応じて属性を観測できないといけない．

Goal Maintain[TrackSegmentSpeedLimit]
InformalDef Trainは各TrackSegmentで決められたSpeedLimit以下を維持しないといけない．
FormalDef ∀tr: Train, s: TrackSegment:
□(On(tr,s) → tr.Speed ≦ s.SpeedLimit)

例えば，左記に責任を持つAgentは，tr.Speedを制御でき，s.SpeedLimitを観測できないといけない．

7. Operationの識別

- ゴールはオブジェクトの属性を状態変数とした状態遷移機械とみなせる．
- ゴールを特徴付ける論理式が真となるような(状態を変化させる)Operationを識別する．
- 開発するソフトウェアが責任を持つゴールに対応するOperationをRequirementsとする．

支援ツール Objectiver

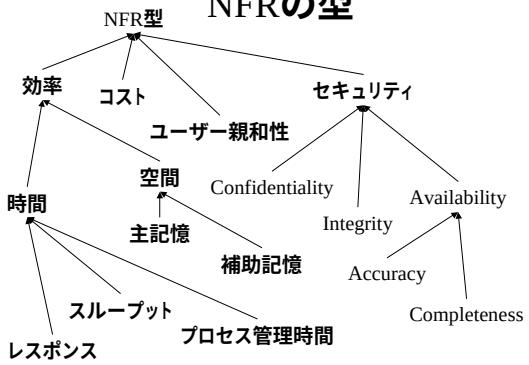
- KAOSをベースにしたREツール
- 昔はGRAILと呼ばれていた．
- CEDITI社で作成
  - – Axel先生の教え子の会社らしい

http://www.cediti.be/EN/
http://www.cediti.be/EN/Solutions_Services/
http://www.objectiver.com/

NFR framework

- NFR=Non-Functional-Requirements=**非機能要求**
- **機能ではなく性能等に関するゴール**(NFR)**の雛形をあらかじめ与えて，**
- NFR**の扱いを見落とさないようにする支援とする．**

NFR**の型**



Security NFR

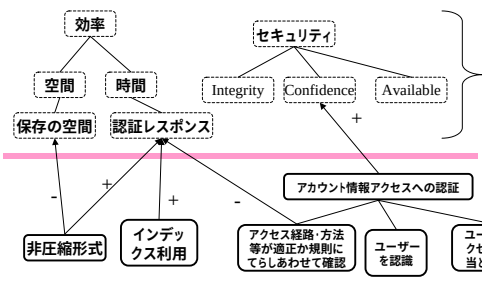
時節柄，注意しなければいけないゴール

- Confidentiality **許可されていないアクセスからデータを守ること．**
  - **一番，よく使われるセキュリティの意味．**
- Integrity **不正改竄されてないこと．**
  - Accuracy **正確である．**
  - Completeness **完全である．**
- Availability **不正なサービス割り込みを抑制すること．**

ソフトゴール ⊃ NFR

- **充足されるか否かを判断する明確な定義や基準がないようなゴールのこと．**
  - **定性的ゴールに似ているが，数値化できなくても，判断できるゴールはある．**
- **このようなゴールを無理に明確化せずに，それらの間の因果関係を記録(記述)しようというのが**NFR**フレームワークの重要な考え方．**

例: e-**アカウントの分析**



この辺から上は，事前に気を付けないといけないNFRを知識ベース化しておけば半自動的に導ける．

下層の太字枠のゴールは，「操作的ソフトゴール」．機能ゴールと考えてよい．

NFR**の**GORA**への貢献**

- **非機能的要求は忘れやすく，また，扱いにくいので，それを扱いやすくするという点で貢献がある．**
- NFR**のパターンはアプリケーション分野等を絞れば，さらに使い手のあるパターンをそろえることができる．**

Goal-Question-Metrics法 (GQM)

- あるシステムが満たすべきゴールを充足しているか否かのデータ項目(metrics)を識別するための手法.
  - G ⇒ Q ⇒ M の順に識別.
- ゴールが検証可能であることと, 検証に必要な測定対象を明確化する手段.
  - 検証自体は実現後でないとできない場合もある.
- ソフトウェア自体だけでなく開発プロセスの評価にも利用できる.
  - むしろプロセスの評価に使うほうが多いかもしれない.

GQMの背景

- 計測はトップダウンに行なわれるべきである.
  - まずは計測目標(Goal)があって, その目標を遂行するために, 尺度が定義され, 計測がおこなわれなければならない.
- データ分析は何らかの目的や仮説に基づいて行われるべきである.
  - 例えば, コスト予測の改善をたてるとか, コストを評価するのか, その目的を明確にした上で分析が行われなければならない.
- 検証(測定)項目が明確化できないような要求仕様は不備があるといってよい. それをGQMで確かめることができる.

G, Q, Mそれぞれの定義

- Goal(目標: 概念レベル) 計測の目標を, 計測対象, 計測理由, 品質モデル, 視点, および環境に基づいて明確にしたもの.
- Question(質問: 操作レベル) 特定のGoalの評価, あるいは, 達成する方法を明確にしたものである. これにより, 計測対象(プロダクト, プロセス, 資源)の特性を品質の観点から明らかにすることができる.
- Metrics(尺度: 量レベル) Questionに定量的に答えるための, データの集合である. データは客観的データ, 主観的データに分かれる.

GQMの評価ステップ

1. Goalの設定
   - 何のために評価を行うのか, その目標をGoalとして記述する. 後述のGoal記述のためのテンプレートが役立つ.
2. Questionの生成
   - Goalを量的に表現したQuestionとして記述する. 後述の3つの典型パターンがある.
3. Metricの明確化
   - Questionに答え, プロセスやプロダクトがGoalと一致しているかどうかを調べる必要なMetricを列挙する.
4. データ収集法の開発, 収集
   - ツールによる自動化などは有効.
5. データの妥当性の確認, 分析
6. データの事後分析
   - Goalが達成されたかどうかを調べる.

下線の部分がパラメタとなっています.

Goal作成用のテンプレート A

- 目的
  - 計測対象(プロセス, プロダクト, 資源)の,
    - 例: 設計ドキュメント, テスト工程, 保守担当者
  - 理由を行うために,
    - 例: 特徴づけ, 評価, 予測, 動機付け, 改善
- 観点
  - その焦点を,
    - 例: コスト, 正しさ, バグ率, 変更回数, 信頼性, 使いやすさ, 適時性
  - 視点の立場から,
    - 例: ユーザー, 管理者, 開発者, 開発組織
- 環境
  - コンテクストにおいて分析する.
    - 例: 実験プロジェクト, 実プロジェクト, 開発現場, 実験室, 大学

Goal 作成用のテンプレート B

- 対象
  - 対象の,
    - 例: ソフトウェアの修正依頼処理, システムテスト, 最終プロダクト, 設計者
- 論点
  - 焦点を,
    - 例: コスト, 正しさ, バグ率, 変更回数, 信頼性, 使いやすさ, 適時性
- 視点
  - 視点の立場から,
    - 例: ユーザー, 管理者, 開発者, 開発組織
- 目的
  - 目的する.
    - 例: 特徴づけ, 評価, 予測, 動機付け, 改善

Questionのパターン

- 「対象」そのものを「目的」から見て明確にするための質問.
  - 例:「修正依頼」に対する「現在の処理速度」は？
  - 例:「修正依頼処理」は,「本当に行われいる」か？
- 「対象」の属性を「観点（視点）」から見て明確にするための質問.
  - 例: 修正依頼の「実際の処理時間」は,「予測とどれくらい食い違って」いますか？
  - 例: 修正依頼の「処理能力」は,「向上」していますか？
- 「対象」の特徴を「観点（視点）」から見て評価するための質問
  - 例: 修正依頼の「処理能力」は,「プロジェクトマネージャー」から見て, 満足のゆくものですか？
  - 例: 修正依頼の「処理能力」は,「目に見えて向上」していますか？

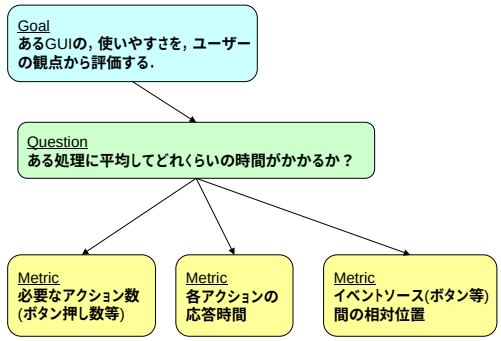
Metrics 設定に関する留意点

- 既存データの量と質: 信頼できる既存データが存在する場合は, できるだけ利用する.
- 計測対象の成熟度: 形式的に定義できる計測方法もある程度確立されている対象に対しては, 客観的尺度を用いる. そうでない場合は主観的尺度を用いる.
- 学習: GQMモデルは常に洗練し, さまざまな場合に適用させてゆく必要がある.

Metricの例

- 修正依頼の処理時間の平均
- 上記の分散
- 処理時間があらかじめ定められた上限を超えた件数
- 修正依頼処理が決められたとおりに行われいるかどうかの, プロジェクトマネージャーの主観的評価
- レビューによって明らかになった, 修正依頼処理が決められた通りに行われていない件数
- 対象プロジェクトにおける $\frac{\text{修正依頼の平均時間}}{\text{修正依頼の標準的な平均処理時間}}$

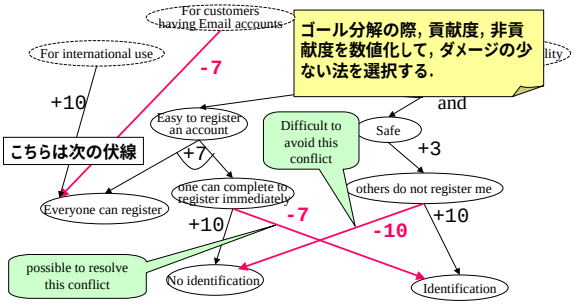
簡単な例



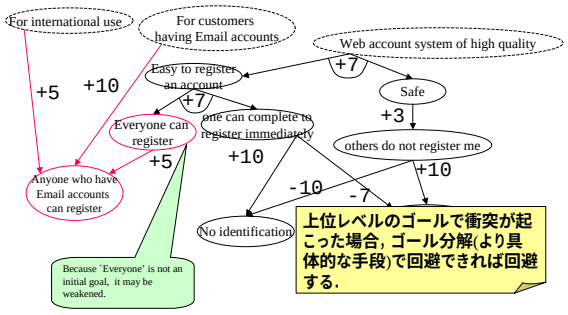
AGORA: 属性付ゴール分析法

- Attributed Goal-Oriented Requirements Analysis = AGORA
- 目的
  - ステークホルダ間の対立, 誤解等を検知を支援.
  - 対立するゴールの取捨選択を支援.
- ゴールグラフの枝と節に属性値を付加
  - 枝: 貢献度: サブゴールの親ゴールへの貢献の度合い.
  - 節: 好感度: stakeholder毎の節ゴールに対する満足度.
  - 属性付けの根拠を付記 (Rationale)

例: ダメージ比較での衝突の回避



例: 詳細化による衝突の回避



Preference Matrix

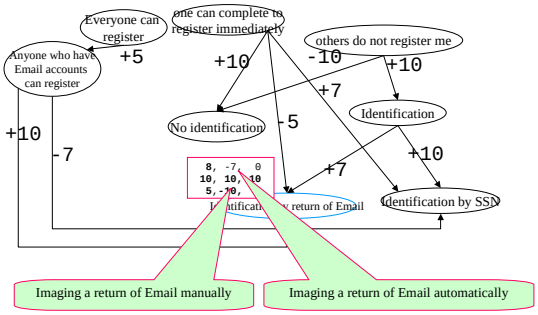
- あるゴールをステークホルダが快く思っているか否か(preference)を示す行列．
- 自分自身だけでなく他のステークホルダのpreferenceも予想してもらうことで互いの衝突や誤解を検知する．

Evaluatee

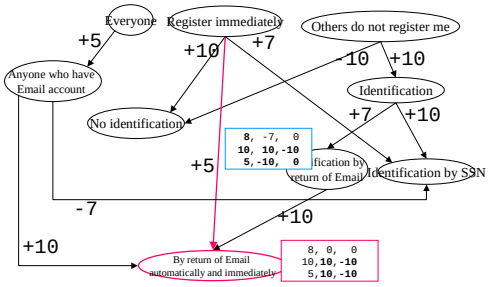
Evaluator	C	A	D
C	8	-7	0
A	10	10	-10
D	5	-10	0

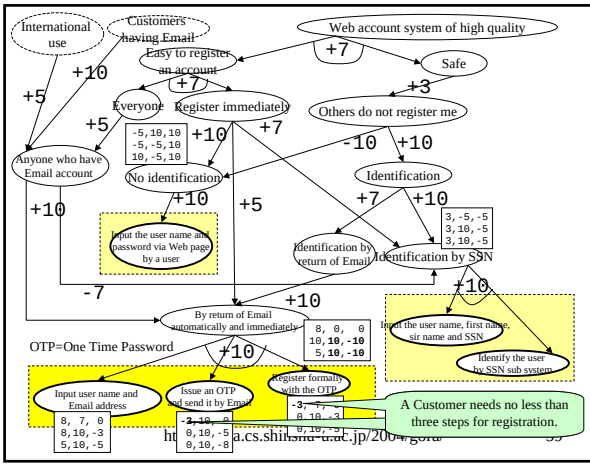
C = Customer
A = Administrator
D = Developer

属性値(優先度)決定と誤解分析

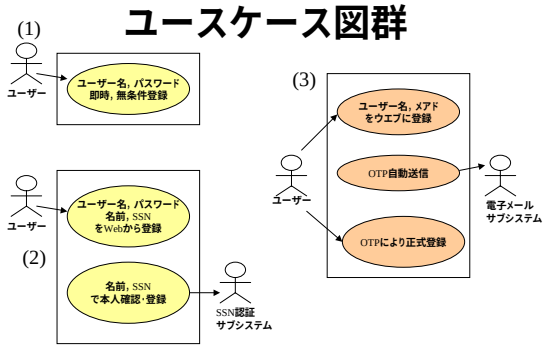


詳細化による誤解の解消





AGORAに連結されるユースケース図群



ゴール階層から要求仕様書の質を見積る

メトリクス
= ゴール階層から系統的に計算できる数値

特性	Sat	Pos	Cup	Vdv	Cov	Hdv	Tre	Con	Rat
Correctness	0.5	0.3	0.2						
Unambiguity				1					
Completeness					1				
Inconsistency		0.6			0.4				
Modifiability							1		
Traceability								0.7	0.3

IEEE830等で定義されている要求仕様書が持つべき特性

行方向の和が1となるように正規化している.
係数自体の配分は主観的かつ経験的.

メトリクス

- Vdv: 優先度行列の縦方向の分散の平均
- 本稿の要因は「大きいほど良い」ようにしているので，実際のVdvは，
  - **Vdv = 1 – 分散の平均**

  としている.

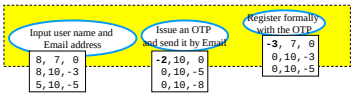
Evaluatee

Evaluator	C	A	D
C	8	-7	0
A	10	10	-10
D	5	10	0

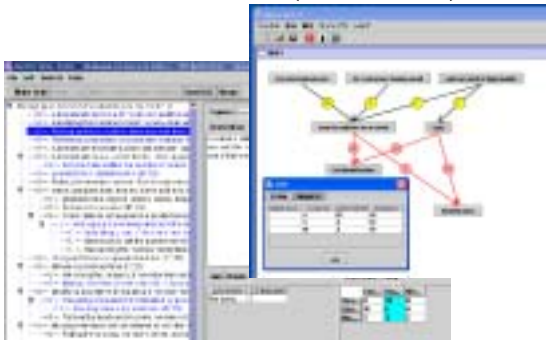
Stakeholder間の共通認識ができれいれば小さいし，そうでなければ大きい.

品質要因: Unambiguity

- **定義**: 仕様書が唯一の解釈をもつこと.
- Vdvからこの要因を計算してよいと思われる.
- 以下のVdv= 0.14 となる.



支援ツール (試作段階)



エピローグ

まとめ

- ゴール指向分析は以下のような活動に役立つ.
  - 要求獲得 (i*, AGORA 等)
  - 要求記述 (KAOS 等)
  - 要求検証 (KAOS GQM 等)
- (手続き型)プログラム技術と対応する分析技法には無い視点である「why」を重視.
- いくつかの手法やツールが提案されている.
- シナリオ等，他の技法と組合わせた試みもいくつか見られる.

皆さんの現状とゴール指向分析

- 本チュートリアルシリーズ第1回(本年1月15日)で参加者の皆さんに簡単なアンケートに答えていただきました.
- その結果,
  - ◆ビジネスモデリング, 問題領域分析, 要求分析が不十分.
  - ◆ビジネスルールや制約の獲得が不十分.
  - ◆ユーザーが要求をわかっていないため要求定義がうまくできない.
  - ◆適切な技法がないため要求定義がうまくゆかない.

  のような点を皆さんは要求工学での問題点と感じているようです. (IBM 鎌田氏らの調査による)
- ゴール指向分析はこれらの一部の解決に貢献するものと考えられます.

アンケートに協力して頂いた皆様に感謝致します

問題点・懸念

- ゴール分析により「また」書き物が増える.
  - 追加的な記述に見合う見返りがあるのか？
    - 追跡管理の充実.
- 類似技法が死滅した理由
  - 過去, IBISやQOC等, 設計理由の記録技法が流行ったが, ほとんど死滅している.
    - 目標や理由はそれほど普遍でないからこそ記述すべきなのか？記述しても無駄なのか？
- そもそもゴール識別自体, 難しい.

RE'04京都でのキーノート

ゴール指向要求分析で著名なAxel先生がキーノートをされます.

RE'04 私も発表します！

引用文献

- [Jac95] M. Jackson, Software Req. & Spec. – A Lexicon of Practice, Principles and Pejudices, Addison, 1995.
- [Zav97] P. Zave **他**, Four Dark Corners of Req. Eng. ACM TOSEM, 1997, 1-30.
- [Kav02] E. Kavakli, Goal-Oriented Req. Engineering: A Unified Framework, RE journal, 2002, 6: pp.237-251.
- [Mac96] L. Macaulay, Requirements Engineering, Springer, 1996.
- [Vl91] A. Lamsweerde**他**, The KAOS Project: Knowledge Acquisition in Automated Specification of Software, Proc. of AAAI Spring Symposium Series, Track: "Design of Composite Systems", Mar. 1991, pp.59-62.

他, 下記ウエブページをご参照ください.

以上です

御質問・御意見等をお願いします