連続チュートリアル:要求工学

2004. 1. 15
立命館大学理工学部情報学科
大西 淳

本日の構成

- 第1部 :連続チュートリアル1回目 要求工学」(大西 淳)
- 第2部 :要求工学国際会議紹介 (東京工業大学 佐伯元司教授)

今後の予定

- 第2回連続チュートリアル
  日時 :2月17日(火)18:00～20:00
  場所 :情報処理学会会議室(田町)
  講師 :Sjaak Brinkkemper博士 (オランダ)
- 第3回連続チュートリアル
  日時 :3月4日 (木)18:00～20:00
  場所 :明治大学 (御茶ノ水)
  講師 :郷 健太郎先生(山梨大学)

講師紹介

- 大西 淳 (1957年兵庫県西宮に生まれる)
- 京都大学工学部情報工学科卒業、同大学院工学研究科修士課程情報工学専攻修了、同博士課程中退
- 京都大学助手、助教授を経て1994年より立命館大学教授(理工学部情報学科)
- 情報処理学会ソフトウェア工学研究会要求工学ワーキンググループ主査
- 要求工学国際会議 (RE'04)Local Chair, PC member

第一部の内容

1. ソフトウェア要求とは
2. 要求工学とソフトウェア工学
3. 要求定義
4. 要求獲得技術
5. 要求仕様

参考書

- 大西 淳・郷健太郎 「要求工学」 共立出版 ソフトウェアテクノロジーシリーズ9

1. はじめに
2. 要求獲得
3. シナリオ
4. 要求仕様
5. 要求言語
6. 要求仕様化技法
7. 形式的仕様
8. 要求仕様とソフトウェア開発管理
9. ラピッドプロトタイピング
10. 要求定義と支援技術
11. おわりに

ソフトウェア要求

- ソフトウェア要求はユーザや顧客がソフトウェアに当然備わっていると望む特性
- ユーザは「計算機システムのサービスを利用する人々」、特に「計算機システムを直接操作したり、直接やり取りする人々」
- 顧客＝発注者であり、顧客とユーザは同じ場合もあるが、異なる場合もある。

ソフトウェア製品と要求

2つのタイプのソフトウェア製品がある

1. 開発者側がユーザのニーズを予測してそれを満足するソフトウェア製品を開発し、広く大量に販売する。
2. ユーザ側がニーズを開発者側に出して発注し、開発者側はニーズを分析して、それを満足するソフトウェア製品を開発する。単品での開発・販売となる場合が多い。

一般の工業製品とソフトウェア

- 一般の工業製品は前者（開発者側がユーザのニーズを予測してそれを満足するソフトウェア製品を開発する）がほとんど

- ソフトウェアでは後者（ユーザ側がニーズを提供し、特別仕様のソフトウェア製品を開発者側に開発してもらう）が一般に行われている

一般の工業製品とソフトウェア (2)

- 一般の工業製品は、機能の追加や変更は簡単にはできない。あらかじめオプションとして用意された範囲でできる程度。
- 一方、ソフトウェアの場合、利用者側のニーズに応じて独自の機能を持ったソフトウェアが作られる。また既存のソフトウェアへの機能の追加や変更が、場合によっては簡単に実現してしまう。

システム(機器)の規模

システム(機器)	最大構成要素数
ミシン・自転車	10^2
テレビ	10^3
自動車	10^4
ジェット機	10^5
宇宙ロケット	10^6
宇宙ステーション	10^7
都市システム	10^8
超大規模ソフトウェア	10^9

多様なニーズとソフトウェア

- このため利用者側は多様なニーズを持ち、それをソフトウェアで解決することを望む。
- 利用する組織や人が異なると同じような業務であっても、業務が異なるので、それをソフトウェア化した場合、ソフトウェアに求める解は異なる。

ソフトウェア要求の重要性

- 開発者側は、ユーザが解決したい問題やどういったことをしたいかといったニーズを的確に把握し、ソフトウェアが備えるべき機能や性能等をソフトウェア要求としてまとめなければならない。

ソフトウェア要求（例題）

「三角形の3辺の値を入力して、その面積を求めるプログラムを作れ」

ソフトウェア要求の解

「三角形の3辺の値を入力して、その面積を求めるプログラムを作れ」

⇒

3辺の値をa,b,cとし、s=1/2(a+b+c)と置くと、面積S＝√s(s-a)(s-b)(s-c)
となる。（ヘロンの公式）

プログラム

```
read(a, b,c );          % 3辺の値を入力
s=1/2(a+b+c);
S=√s(s-a)(s-b)(s-c);    %ヘロンの公式
print(S);               %出力
```

例えば、３，４，５を3辺の値とする直角三角形では
s＝6、面積S＝√6×3×2×1＝6
と正しい値が求まる。

Q:例題プログラムにユーザは満足するか？

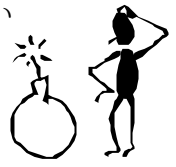
1. 三角形を構成しない場合（6, 4, 4)でも
   s=1, S=√7×(-3)×(-3)=3√7と正の答が求まる
2. 三角形を構成しない場合のソフトウェアの振舞いが未定義
3. 面積の精度が未定義(3辺が1,2,√5で面積は1となるか？)
4. 3辺の入力値に対する規定（全角数字、漢数字は？小数、無理数は？一括でファイルから入力する、対話式に入力？単位？)が不明確

A:例題プログラムは要求を満たさない

- 元の要求には誤り（抜け、不完全）があるが、ユーザにとって当り前のことはわざわざ要求として明示しない

⇒

ユーザにとって満足できないプログラムになってしまう



ソフトウェア要求

- ユーザがソフトウェアに当然備わっていると望む特性
- 特性は機能に関する要求（機能要求）と非機能要求に分かれる
- ユーザにとって当たり前の特性は開発者に逐一伝えない場合が多い

要求はユーザによって異なる

- 使う人の立場によってソフトウェアに求める特性（要求）は異なる

例えばワープロソフトに対して

ワープロソフトに対する要求（1）

- Aさんは英語の論文をまとめたい

- スペルチェック、文法チェック
- 複雑な数式の記述
- 学会指定の段組
- 学会指定のフォント

ワープロソフトに対する要求（2）

- Bさんは年賀状を作りたい

- 干支のイラスト
- デジカメ画像の貼り付け
- フルカラー印刷
- はがき印刷
- 住所管理

ワープロソフトに対する要求（3）

- Cさんは会議資料を作成し、高速に印刷したい

- 簡単に資料が作れる
- 高速に印刷
- 大量に印刷する
- 仕分け機能

ワープロソフトに対する要求（4）

- Dさんは初めてなので、一度ワープロを使ってみたい

- どんな機能があるかも分からないけれど、使ってみたい

要求は立場によって異なる

- ユーザと顧客の立場によってソフトウェアに求める特性（要求）は異なる（矛盾する）

ユーザは使い勝手の良さを
顧客は投資効果の大きいものを

要求が誤っていることがある

- 要求は不完全であったり（抜け）
- 互いに矛盾していたり
- あいまいであったり（解釈が何通りもできる）
- ユーザのニーズと整合していない
  （ユーザが解決したい問題を解決できない）

要求は変わることがある

- 利用者が正確に伝えなかった要求を開発者側が誤解すると、誤解された要求は修正されなければならない
- 新技術の開発や環境の変化に伴って利用者のニーズ自体が変わることがある。時間が経つと上記の変化は起こりやすい。当初正確にニーズを開発者側が引き出していても、ニーズが変わることによって要求が変わる

「ソフトウェア要求」のまとめ

- 要求は利用者側がソフトウェアに備わるべきと思う特性
- 利用者側は様々なニーズを持つ
- 開発者はニーズを的確に捉える、それを実現すべくソフトウェアを開発する
- でも、要求は立場によって異なる
- でも、利用者側は当たり前のことは言わない
- でも、要求は誤っていることがある
- でも、要求は変わることがある

本日の内容

1. ソフトウェア要求とは
2. 要求工学とソフトウェア工学
3. 要求定義
4. 要求獲得技術
5. 要求仕様

要求工学

- 要求工学は「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」
- ソフトウェア要求を正しくまとめられないと、

1. 開発コストの増加や開発スケジュールの遅延
2. 開発のやり直し、
3. 開発プロジェクトの失敗

ソフトウェア工学と要求工学

- 要求工学はソフトウェア工学の1分野
- ソフトウェア工学は「高機能高品質なソフトウェアを効率よく作成したり、利用するための方法論の集大成」
- 発端は1968年NATO主催の会議でソフトウェア危機の提案 -認識とその打開から
- 当時は巨大システムの開発が相次いだが、軍用システムの多くはソフトウェアトラブルが続出

システム工学が母体となってスタート

- それまでの職人芸的なソフトウェア開発を改め、工場で高品質な自動車が高い生産性で製造されるのと同じようにソフトウェアを開発したい
- ６０年代はアポロ計画を始めとして、システム工学が成果をあげていた
- システム化の対象をソフトウェアに特化することでシステム工学をカスタマイズすることからソフトウェア工学は始まった

ソフトウェア工学の初期の成果

- ソフトウェアライフサイクルモデル
  ⇒開発の分業化と各工程での成果物による工程管理
- 構造化プログラミング
- データ中心設計
- ….

ソフトウェアライフサイクル

- ソフトウェアの開発や運用・保守がどのように進んでいくかをモデル化したもの

要求定義 → 設計 → 実装 → テスト

→ 導入と設置 → 運用と保守 → 廃棄

ウォータフォール型ライフサイクルモデル

ウォータフォール型ライフサイクルモデルの特徴

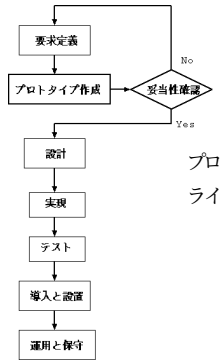
- 各工程でドキュメントが作成され、それが次の工程の入力となる。 例えば要求定義工程ではプロダクトとして要求仕様が作成され、それが設計工程の入力となる）
- 各工程でのプロダクトを検査すればその工程が正しく実施されたかどうかを判断可
- 各工程ごとに専門家チームを用意できる

ウォータフォール型モデルの問題点

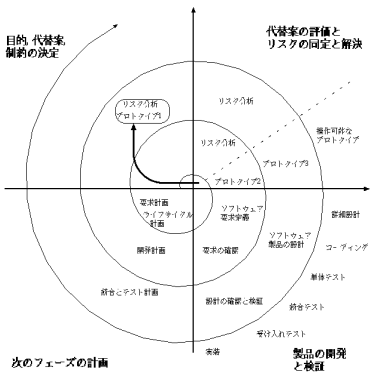
- ソフトウェア開発はスムーズに進むとは限らず、後戻りが生じることがある。
- 特に要求定義を誤ると、最終製品をユーザが使うまで、誤りが検出できない
- この結果、開発コストの超過やスケジュールの遅延が生じる

改善されたライフサイクルモデル

- プロトタイピング型ライフサイクルモデル
  開発の初期段階でプロトタイプを作成し、利用者に妥当性を確認してもらう
- スパイラル型ライフサイクルモデル
  ソフトウェア開発を「目的や代替案、制約の決定」、「代替案の評価やリスク分析」、「次レベルの製品の開発と検証」、「次フェーズの計画」の４段階を繰り返しと捉え、繰り返しを経るに従い、プロトタイプが成長して、最終製品に至る



プロトタイピング型
ライフサイクルモデル



スパイラル型
ライフサイクル
モデル

要求工学とソフトウェア工学」のまとめ

- 要求工学はソフトウェア工学の1分野
- システム工学の開発対象をシステムに特化することでソフトウェア工学が始まった
- 初期のソフトウェア工学の成果の一つにウォータフォール型ライフサイクルモデル
- 改良されたライフサイクルモデルがいくつか提案されている

本日の内容

1. ソフトウェア要求とは
2. 要求工学とソフトウェア工学
3. 要求定義
4. 要求獲得技術
5. 要求仕様

要求定義は必要か？

- ユーザ要求は時間が経つと変化する
- 従って、要求を正確に定義することは不可能

変化する要求と変化しない要求がある。

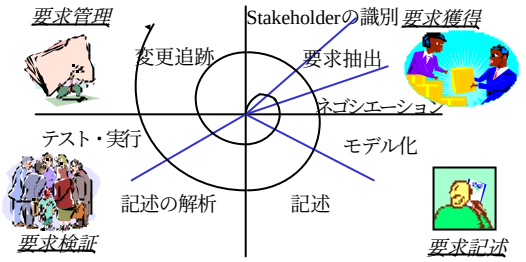
それらを切り分けて、変化しないコアとなる要求を正確に定義することは必要。

要求定義のプロセス

1. 要求の獲得（開発者がユーザから獲得）
2. 要求の仕様化（獲得した要求の文書化）
3. 要求仕様の検証（要求仕様の品質チェック）
4. 要求の管理（版管理、再利用）

これらの4フェーズを繰り返すことによって、要求が成長する

要求定義プロセスサイクル



要求定義の前提

- 一般に解決すべき問題あるいはニーズが存在し、問題を解決するため、またはニーズを実現するためにソフトウェアが作られる
- 問題(ニーズ)は複雑であったり、不完全であったり、あいまいであったり、矛盾している場合が多い
- 特に前例がないような新規システムの開発の場合は問題(ニーズ）を理解することすら難しい

要求定義の目標

- 何が問題であるかを認識してニーズを引き出す必要がある。
- 要求定義の目標は

1. 問題の本質を明らかにして
2. 問題の解決策を策定し
3. 解決案を実現するためのソフトウェアの特性を定める

本日の内容

1. ソフトウェア要求とは
2. 要求工学とソフトウェア工学
3. 要求定義
4. 要求獲得技術
5. 要求仕様

要求獲得

- 問題の本質を明らかにする段階
- 要求獲得では

1. 現状のニーズや問題点を収集し、
2. 収集した結果を分析・整理し、
3. 解決すべき真の問題を明らかにする
4. この結果、ニーズが明らかになる

要求獲得のための技法

- 問題分析法
- 開発者とユーザ間のインタラクション分析法
- 合意形成と意思決定法
- ソフトウェア開発のために特化された要求獲得技法

問題分析法

- 創造工学
- ブレーンストーミング
- 水平思考
- KJ法

KJ法

- 川喜多二郎氏考案
- 特別な道具や知識は不要
- １人でも４～５人のグループでも可能
- 要求獲得だけでなく一般の問題解決に有効
- 参考文献：川喜多二郎「発想法」中央公論社

KJ法の手順（1）

1. 参加者に目的意識を徹底
2. 情報を広く収集
3. 情報を一つごとに１枚のカードに記入し、見出しを付ける
4. カードを壁に貼ったり机に広げて全体が見渡せるようにし、親近性のあるカード群をまとめて小グループとする
5. 小グループに名前を付ける

KJ法の手順（2）

6．４と５を繰り返す。これにより中グループ、大グループができる

7．グループ間の類似・対立・従属・因果等を表せるようカードを再配置する

8．以上の結果をまとめる

インタラクション

- 開発者とユーザや顧客とのやりとり（インタラクション）を支援すれば要求獲得は円滑に進む
- インタラクションには
  1. アンケート
  2. インタビュー
  3. 会議

などがある

インタラクション分析

- インタラクションを記録し、その記録を分析することをインタラクション分析という
- インタラクションに現れた要求と要求に関連した情報を抽出
- インタラクションの改良や支援

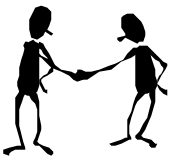
ビューポイント（視点）

- ビューポイントとは「要求を捉える際の立場や見方」
- 複数のユーザ、顧客、開発者が要求定義に関わる。 関わる人々を総称してStakeholder)
- ビューポイントが異なると、お互いに矛盾する要求が得られることが多い

合意形成

- 異なる人々から出された要求が互いに矛盾する場合、合意形成によって矛盾を解消する必要がある

1. デルファイ法
2. AHP法



デルファイ法

- あるグループの人々に対して、同じ質問を繰り返し、その結果をフィードバックさせながら収束させ、最終的に１つの意見にまとめる方法
- 単なるアンケートと異なり、同じ質問に対する全体の回答傾向と少数意見には根拠を付けてフィードバックすることにより、再評価した結果が収斂していく

デルファイ法の手順（1）

1. 幅広い分野から回答者を選定する
2. アンケートやインタビューにより数値で答えられる質問をし、結果を集める。
3. 結果をソートし、第一四分位数、中央値、第三四分位数を求める
4. 上記の３つの数値を回答者に返し、再度同じ質問をする。中央値から極端にはずれる結果を返す場合は理由を明記させる

デルファイ法の手順（2）

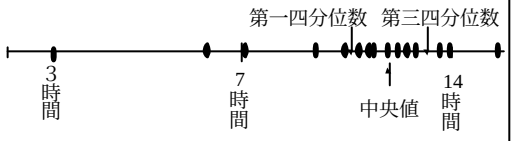
5．新しい結果について３と４のステップを繰り返す。理由がある場合は３つの数値と共に理由も全員に返す

6．ある程度、繰り返すと、結果が収束するので適当なところで中央値を結果として採用する

回答に自身の無い人は多数意見（中央値）に近づく。一方、少数意見でも回答に自信があればそれを根拠として示し、全体がその根拠に納得すれば結果が修正される。

デルファイ法の例

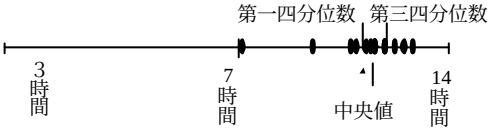
- 質問：東京とニューヨークの所要時間は2030年には何時間になるか？」
- 回答をソート



- 回答傾向を付けて再質問し、少数意見には根拠も返してもらう

デルファイ法の例 (2)

- 再質問：東京とニューヨークの所要時間は2030年には何時間になるか？」
- 回答結果は自信の無い人は中央値に近づく



- 再質問を繰り返し、ある程度収束した時点の中央値を合意結果として返す

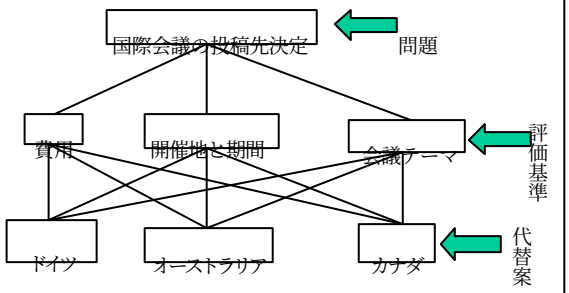
AHP法 階層化意思決定法）

- 代替案が複数ある場合に、代替案を評価することによって、最適な案を選択する。

1. 意思決定を行う問題を最上位に、代替案を最下位に、案を選ぶ評価基準を中間に置いた階層を定義する
2. 各層の要素間を一対比較し、結果を数値で表す
3. 一対比較行列を正規化し、各層の重みを算出
4. 各層の重みを合成し、代替案の重みを求める

一対比較の数量化

要素a_iとa_jを比較すると	一対比較のa_{ij}の値
同程度に重要	1
やや重要	3
重要	5
かなり重要	7
圧倒的に重要	9



評価基準の一対比較行列

	開催地と期間	会議テーマ	費用
開催地と期間	1	1	7
会議テーマ	1	1	7
費用	1/7	1/7	1

この行列の固有ベクトル ω を求める。

$A_{ij} \times \omega = \lambda_{MAX} \times \omega$　（λ_{MAX}は最大固有値）

評価基準の重みベクトル

ω＝7/15
7/15
1/15

これは評価基準として 開催地と期間」、 会議テーマ」は同程度に重要であり、 費用」はそれほど重要でないことを示している。

開催地と期間」による代替案の評価

	ドイツ	オーストラリア	カナダ
ドイツ	1	3	3
オーストラリア	1/3	1	2
カナダ	1/3	1/2	1

開催地と期間」による代替案の重み

ω=0.59361
0.15706
0.24930

開催地と期間」ではドイツ、カナダ、オーストラリアの順に好ましい

会議テーマ」による代替案の重み

ω=0.74183
0.18295
0.075200

会議テーマ」ではドイツがかなり好ましい

費用」による代替案の重み

ω=0.087615
0.61994
0.29244

費用」ではオーストラリア、ドイツ、カナダの順に好ましい

3つの重みベクトルを合わせると

	開催地と期間	会議テーマ	費用
ドイツ	0.59361	0.74183	0.08756
オーストラリア	0.15706	0.18295	0.61994
カナダ	0.24930	0.07520	0.29244

これに評価基準の重みベクトルを掛けると

ドイツ	0．62905
オーストラリア	0．20000
カナダ	0．17093

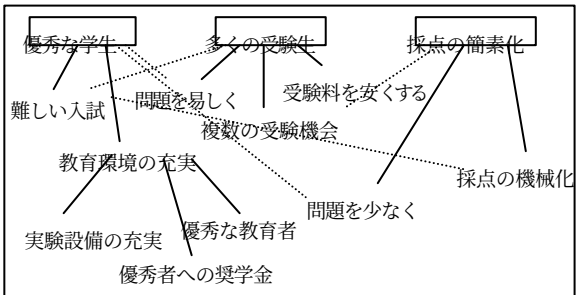
この結果、ドイツの国際会議が最も好ましいことが分かる

ゴール指向分析

- 最初に達成したいゴールを明確にする
- 次にゴールをサブゴールに分割して行く。分割にはAND分割とOR分割を用いる。OR分割によるサブゴールは代替案となり、他のゴールやサブゴールも含めて代替案同士を評価する。
- 分割によって抽象的だったゴールが詳細化され、システム化可能なサブゴールにまで詳細化された時点でそのサブゴールを要求とする

○○大学入試政策

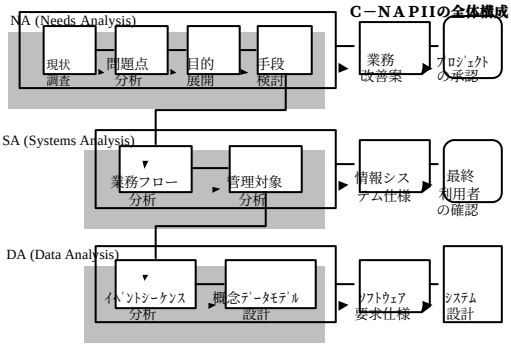
- ゴール
1. 「優秀な学生を入学させる」
2. 「受験生をたくさん集める」
3. 「入試の採点業務は簡素化したい」

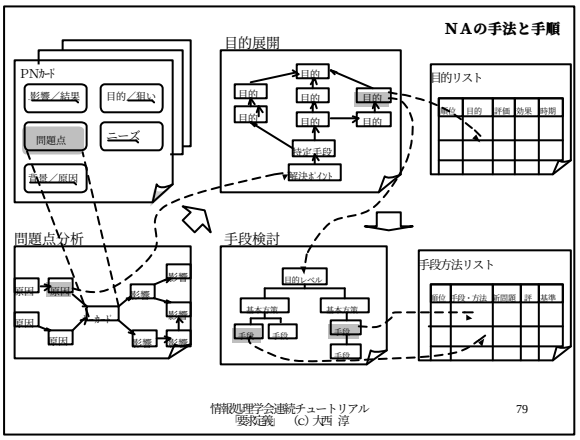


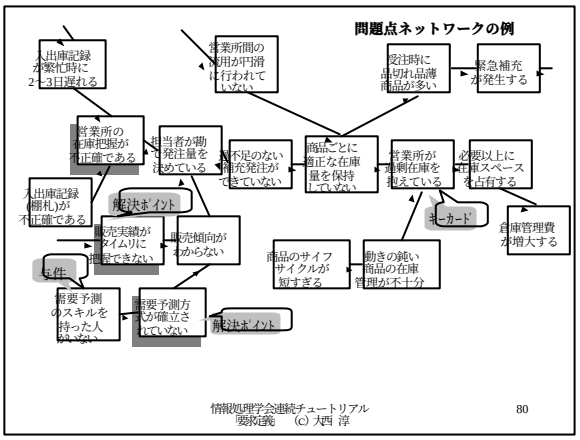
教育環境の充実によって、「優秀な学生」を確保し、
受験料の引き下げによって「多くの受験生」を集め、
採点の一部機械化によって採点業務の簡素化を図る

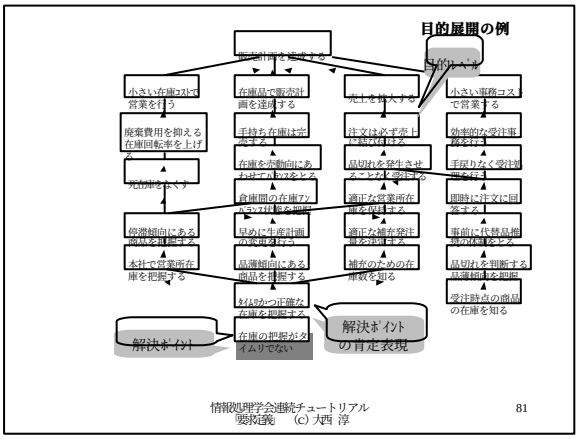
C― NAP法

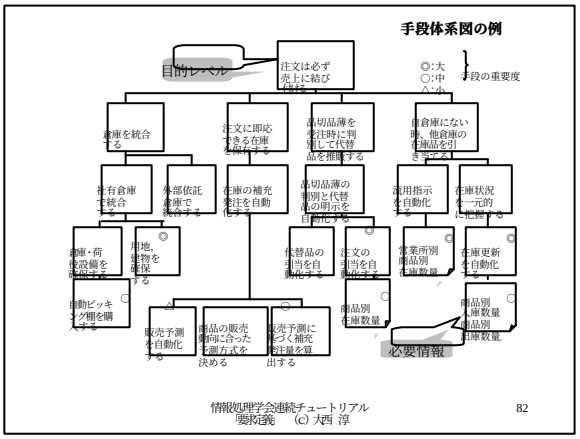
- 富士通（株）が開発したソフトウェア要求獲得のために特化された手法。
1. PNカードによる情報収集
2. 問題点ネットワークによる問題分析
3. 目的展開による目的レベルの設定
4. 目的を達成する手段の検討











「要求獲得技術」のまとめ

要求獲得には以下のものがある

- 問題そのものを分析する
- 開発者と利用者側のインタラクションを支援する
- 立場の違いから生じる矛盾を解消する
- これらを含み、かつソフトウェア開発に特化した手法

本日の内容

1. ソフトウェア要求とは
2. 要求工学とソフトウェア工学
3. 要求定義
4. 要求獲得技術
5. 要求仕様

要求仕様

- 要求仕様は要求定義段階の最終成果物
- 目的は

1. 顧客や利用者による仕様の確認
2. 開発者による仕様の検査
3. 設計に必要な情報の提供
4. 開発スケジュールやコストの見積もり
5. ソフトウェア開発の契約書の一部

要求仕様は誰が書くか？

- 開発者側の人間ではユーザ側の問題やニーズを明確に理解できない
- ユーザ側の人間に設計知識が無い場合は、設計段階で設計者に十分な情報を伝えられない

⇒

ユーザ側と開発者側の双方によって書かれるのが望ましい

要求言語

- 要求を記述するための言語
  1. 自然言語（日本語、英語など）
  2. 形式言語
  3. 制限言語（文法や語彙を制限した日本語）
  4. 図式言語（データフロー図、UMLなど）

これらの組み合わせによって仕様化

要求仕様の妥当性確認

- 利用者や顧客の要求が正しく開発者側に伝わっていないと、場合によってはソフトウェア製品の納入時に利用者や顧客の考えていたソフトウェアとは異なることが判明し、再開発を余儀なくさせられる
- 開発の早い段階で利用者や顧客に妥当性を確認する手間は、納入後に修正する手間やコストに比べはるかに少ない

→

妥当性確認にはプロトタイピングが有効

要求仕様の品質特性(Std.830-1998)

- 妥当性（正当性）
- 非あいまい性
- 完全性
- 無矛盾性
- 重要度のランク付け
- 検証可能性
- 変更可能性
- 追跡可能性

仕様の妥当性

- 要求仕様中の全ての要求は、開発されているソフトウェアが満たすべき要求であること
- ある要求が開発するソフトウェアに必須か否かは利用者や顧客でないと判断できない場合が多いため、一般に妥当性の確認は利用者や顧客が行う。

→ プロトタイピングが有効

仕様の非あいまい性

- 仕様中の全ての要求が一意に解釈できるとき、仕様はあいまいでない
- 要求があいまいだと、開発者は誤って解釈する恐れがあり、結果として誤ったソフトウェアが開発されることになる。
- あいまいさの原因としては、自然言語のもつあいまいさ（文法上、指示代名詞、語句の省略）や多義語、定性的な表現がある

仕様の完全性

- 機能、性能、設計制約、属性、外部インタフェースに関する要求が記述されており
- あらゆる状態での入力に対する振舞い、特に正当な入力と不当な入力に対する振舞いの定義と
- 仕様中の図や表に対するラベル付けと参照、語句の定義、単位の定義

冗長な要求

- 完全性の逆で、同じ要求が２箇所以上に現れており、一部を削除しても正しく抜けとならない場合は「冗長」な要求という。
- 冗長な要求は誤りではないが、修正時に片方だけを修正し、残りを修正しないと矛盾となるので、できるだけ冗長な要求は避ける

仕様の無矛盾性

- 他の仕様書との矛盾ではなく、1つの要求仕様書の中で一貫していることを意味する
- 一貫しているとは個々の要求が互いに矛盾しないことを意味する

重要度のランク付け

- 要求はすべて同じ優先順位が付けられるとは限らない。個々の要求は以下のような相違を区別できないといけない。

1. 必須な要求
2. あった方が望ましい要求
3. あってもなくてもよい要求

仕様の検証可能性

- ソフトウェア製品がその要求を満たしていることを計算機や人手によってチェックできるプロセスが存在することを指す
- 定性的な表現の要求は検証不能

⇒

定量的・具体的な表現にする

仕様の変更可能性

- 仕様の構造を保持したまま、容易に、完全に、矛盾無くしようを変更できる場合、変更可能という。このためには、
- クロスリファレンスや索引・目次があって、仕様が分かりやすい構成になっていること
- 冗長な要求が無いこと
- 個々の要求を別々に表現すること

が必要

仕様の追跡可能性

- 後方追跡可能性 :各要求から、要求仕様に先立って作成された文書中の各要求の起源について参照できること
- 前方追跡可能性 :要求仕様を元にして作成された全ての文書（例えば設計仕様書やソースコード）から各要求を参照できること

後方追跡可能性

- 各要求の出された背景や理由を知りたい場合に有用。特に新規システムの要求仕様を既存システムの要求仕様から導く場合に要求の背景や理由が分かるので、新規システムに取り込むかどうかを判断しやすくなる

前方追跡可能性

- 運用段階や保守時に、ソースコードや設計文書を変更する場合に、その変更によって影響を受ける要求を同定したい場合に有用

要求仕様の章構成(Std.830-1998)

1. はじめに
2. 要求仕様の全体説明
3. 詳細な要求仕様
4. 付録
5. 索引

要求仕様の「はじめに」

- 要求仕様の目的と対象とする読者
- ソフトウェアの名称、何をするか、対象
- 用語の定義と略語の説明
- 参考文献リストと入手できる場所
- 要求仕様の概要と構成

要求仕様の「全体説明」

- ソフトウェアの全体像（関連製品も含む）
- ソフトウェアの主要な機能
- 利用者の一般的な特性
- 制約事項（法的、ハード上、セキュリティ上、他のアプリとのインタフェース上等）
- 要求に影響を与える仮定と依存事項

要求仕様の「詳細な要求」

- 設計やテストに十分なまで詳細化された要求
- 要求仕様の品質特性を満たすこと
- 関連する文書と相互参照可
- 要求は個々に識別可能
- 読みやすく構成

「詳細な要求」の構成

- 外部とのインタフェース
- 機能要求
- 性能要求
- 論理データベース要求
- 設計制約
- ソフトウェアシステムの属性
- 要求仕様の構成
- コメント

「詳細な要求」の構成法

- システムのモードに基づいた構成
- オブジェクトに基づいた構成
- 機能階層に基づいた構成
- サービスに基づいた構成
- 利用者のクラスに基づいた構成
- 入力に基づいた構成
- 応答に基づいた構成

IEEE Std 830-1998による要求仕様第３章の構成例(オブジェクトを中心)

3. 詳細な要求仕様
 3.1 外部インタフェース要求
  3.1.1 ユーザインタフェース
  3.1.2 ハードウェアインタフェース
  3.1.3 ソフトウェアインタフェース
  3.1.4 通信インタフェース
 3.2 クラス/オブジェクト
  3.2.1 クラス/オブジェクト１
   3.2.1.1 属性（直接または継承）
    3.2.1.1.1 属性１
    .
    3.2.1.1.n 属性ｎ
   3.2.1.2 機能(サービス、メソッド、直接または継承)
    3.2.1.2.1 機能要求1.1
    .
    3.2.1.2. m 機能要求1.m
  3.2.2 クラス/オブジェクト２
  .
  .
  3.2.p クラス/オブジェクトｐ
 3.3 性能要求
 3.4 設計制約
 3.5 ソフトウェアシステムの属性
 3.6 その他の要求

機能階層を中心にしたシステムの第３章の構成例

3．詳細な要求仕様
 3.1 外部インタフェース要求
 3.2 機能要求
  3.2.1 情報フロー
   3.2.1.1 データフロー図１
    3.2.1.1.1 データ実体
    3.2.1.1.2 関連するプロセス
    3.2.1.1.3 トポロジー
   3.2.1.2 データフロー図２
   .
   3.2.1.n データフロー図ｎ
  3.2.2 プロセス記述
   3.2.2.1 プロセス１
    3.2.2.1.1 入力データ実体
    3.2.2.1.2 アルゴリズムまたは処理の式
    3.2.2.1.3 影響するデータ実体
   3.2.2.2 プロセス２
   .
   3.2.2.m プロセスm
  3.2.3 データ構成
   3.2.3.1 構成１
    3.2.3.1.1 レコード型
    3.2.3.1.2 構成フィールド
     3.2.3.2 構成２
     .
     3.2.3.p 構成ｐ
   3.2.4 データ辞書
     3.2.4.1 データ要素１
      3.2.4.1.1 名前
      3.2.4.1.2 表現
      3.2.4.1.3 構成単位/形式
      3.2.4.1.4 精度
      3.2.4.1.5 範囲
     3.2.4.2 データ要素２
     .
     3.2.4.q データ要素ｑ
 3.3 性能要求
 …

良い要求仕様とは？

1. 8つの品質特性を満たしていること
2. 要求仕様の構成上の内容を全て含んでいること

これら２つの観点から要求仕様を見れば良い。

ソフトウェア要求（例題）

*「三角形の3辺の値を入力して、その面積を求めるプログラムを作れ」*という仕様を再度考察する。

１．誤りに関する品質特性から
２．要求仕様の「詳細な要求」の構成から

例題とした要求の品質上の問題

- 妥当性：利用者側のニーズが明らかでないので判断できない
- 非あいまい性：値は整数値なのか？
- 完全性：
  - 三角形をなさない際のふるまい
  - 入出力データの書式、単位、入出力方法
  - プログラム言語・動作環境
- 無矛盾性：特に矛盾は無い

例題とした要求の構成上の問題

- 外部とのインタフェース要求：抜けている
- 機能要求：一部あるが、不正な入力のチェックや三角形を構成しない入力値の場合の処理が抜けている
- 性能要求：抜けている
- 論理データベース要求：ない
- 設計制約：プログラム言語や動作環境といった制約が抜けている
- ソフトウェアシステムの属性：ない

要求仕様」のまとめ

- 要求仕様は満たすべき特性がある
- 要求仕様に書かれるべき項目と章構成の規格がある
- こららの特性や構成を満足するように仕様をまとめる

おわりに

1. ソフトウェア要求とは
2. 要求工学とソフトウェア工学
3. 要求定義
4. 要求獲得技術
5. 要求仕様

参考書

- 大西　淳・郷健太郎　要求工学」
  共立出版
  ソフトウェアテクノロジーシリーズ9
  ISBN:4-320-02782-5
  3,600円＋税

お疲れ様でした

引き続き第2部

- 要求工学国際会議の紹介
- 論文投稿案内