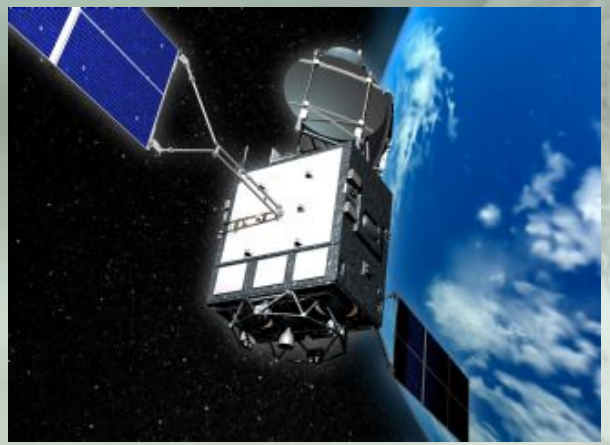
第13回衛星データ処理勉強会
WEBデータベースシステム開発における
ユーザ要求の具現化手法について



平成19年7月31日

情報・計算工学センター
プロジェクト共通情報化チーム
館下　博昭

今日の内容

1. はじめに
2. システム構築時の悩ましい問題
3. 構築手法
   - 要求仕様の話を中心に
4. 結果・効果
5. 今後の課題
6. まとめ

1. はじめに

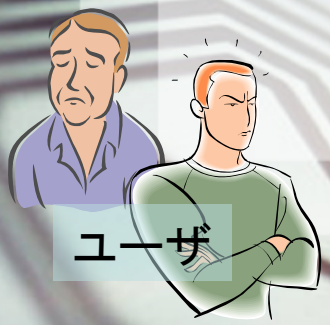
- プロジェクト共通情報化チームにて、以下のWEBデータベースシステムの開発（改修）を実施してきた
  - WINDSプロジェクト情報管理システム（**PIMS**）
  - GOSAT技術連絡書管理システム（**GOWF**）
  - 衛星技術情報共有システム（**STAR**）
- 上記の情報システムを構築する際、試行錯誤の上、ユースケース、画面イメージをベースとした手法を徐々に導入し、ユーザ要求の具現化に関する悩ましい問題が改善できた。
- ここでは、我々のチームで培ったこの手法とその結果について、苦労話も含め、紹介したい。

2. システム構築時の
よくある悩ましい問題

a. 決めることが多すぎて、長時間にわたる設計会議

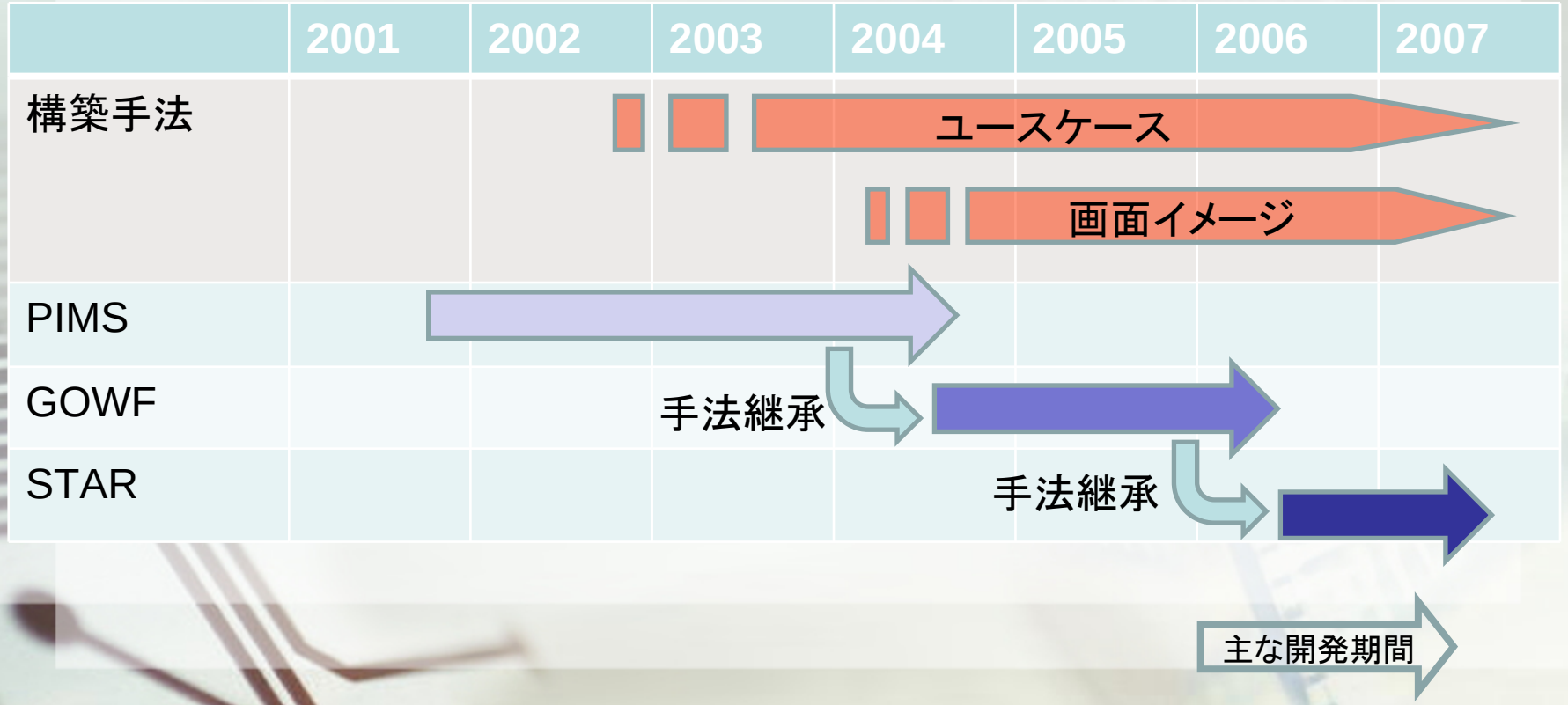
b. 要求の誤解から来る手戻り

- ユーザのシステム仕様の不理解
  - 「そういうつもりじゃなかった。。」
  - 「フロー図では、よくイメージできない」
- IT部門のユーザ要求のあいまいな理解
  - 「あの時＊＊＊って言ってて議事録にも残っているに。。」
- 開発ベンダーの要求不理解
  - 「それは、仕様と違います。追加要望ですか？」

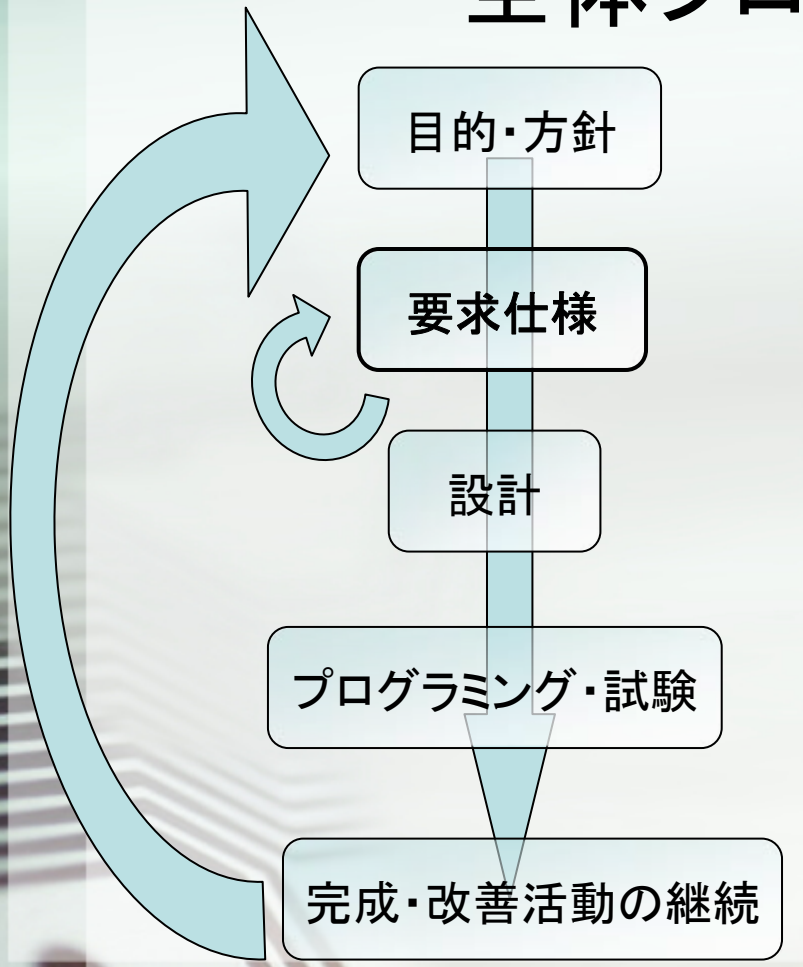


3. 構築手法

- 悩ましい問題の解決のために、試行錯誤の結果、“ユースケース”と“画面イメージ”に辿り着く



3-0. システム構築 全体フロー



- 今日の話は、主に「要求仕様」の部分
  - ユーザ要求を仕様に落としていく過程
  - その時に作成する資料
  - 資料間の整合性の反復チェック

3-1. 目的・方針

- 情報システム導入以前の話
- やはりここが一番重要だったりする

目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続

- 例（STARの場合）
  - 利用本部における基本方針：
    - 各プロジェクトは全ての技術情報を蓄積
    - 蓄積は電子ファイルが原則
    - プロジェクト進行中からの技術情報の整理及び終了後の確実な移管
    - 共通的な分類方法の適用（利用本部内の共通的な分類を参考）
    - 各プロジェクトにおいて整理された技術情報はすべて共有が原則。ただし、プロジェクト進行中については、書誌情報のみの共有も可とする。
    - 利用については利用者側が責任を持つ
  - これを調整するのに1年かかった

3-2. 要求仕様の定義 (全体像)

- ここでは、主にSTARの時の、要求仕様の定義の手順を説明する。

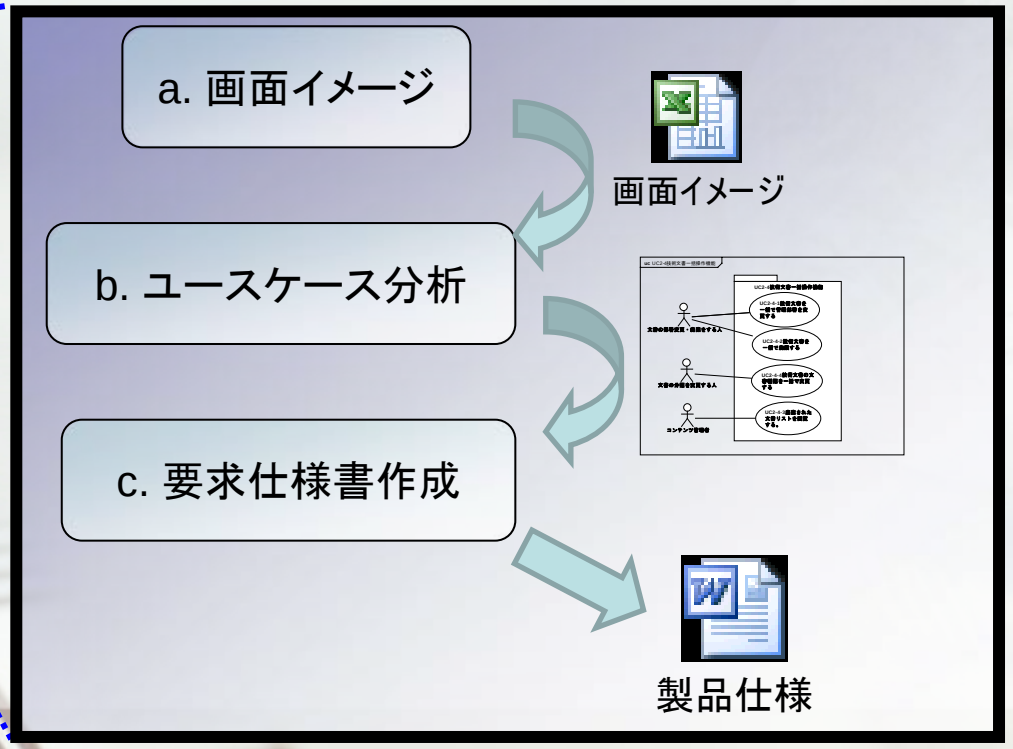
目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続



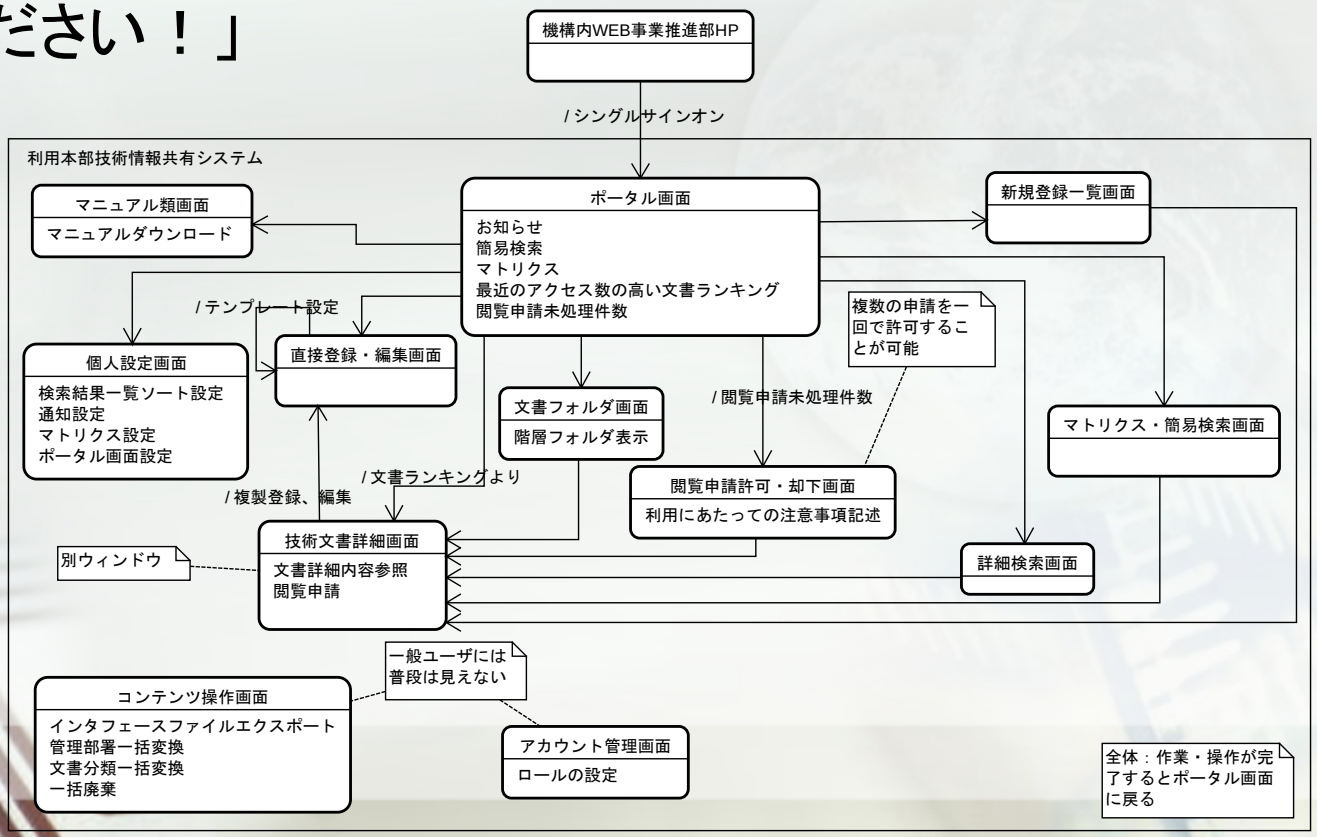
3-2a. 画面イメージ

1. 画面遷移をまず考える
   - UMLの状態遷移図が使いやすい
2. 上記で登場する画面についてそれぞれEXCELでイメージを作成
3. 上記で作成した画面遷移と画面イメージを使ってユーザヒアリング
   - システムのイメージをつかんでもらい、可能な限り細かい機能要求を聞きだす。
   - STARの場合、全プロジェクトから、1,2人ずつ参加
     - GOWFの場合も、プロジェクトメンバーの大半が参加し、大幅な修正が入った。

- ⇒次ページにSTARの画面遷移図と画面イメージのサンプル

3-2a. 画面遷移図サンプル

- ユーザへの説明時、できるだけ使っているところをイメージしてもらうようにした:「みなさん、想像してみてください！」



目的･方針

要求仕様

設計

プログラミング･試験

完成･改善活動の継続

3-2a. 画面イメージ（サンプル）

- 説明時は、使い方を想像してもらいながら:「この画面は、△ △ △ 時に使う画面です」
- 多少ExcelVBAでボタンの動作を模擬

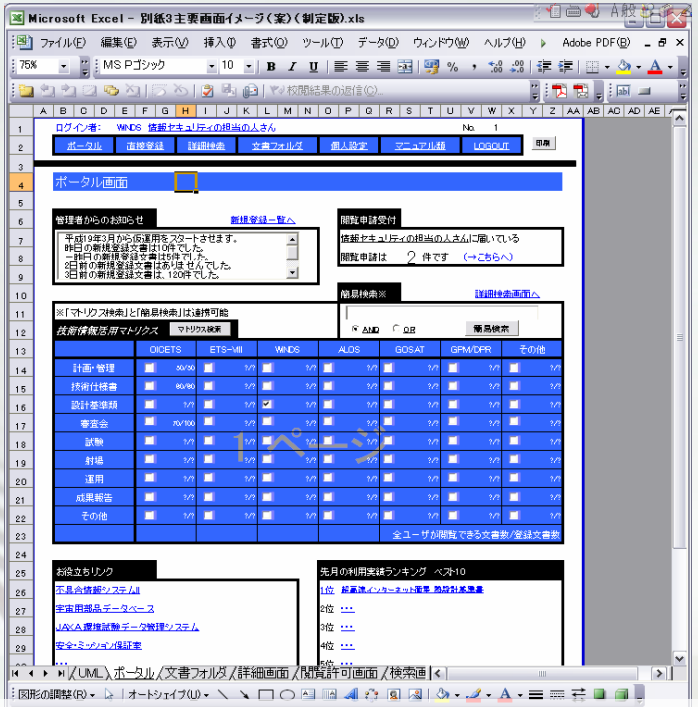
目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続



JAXA側で作った画面イメージ（EXCEL）

開発ベンダーが作った実際の画面

3-2b. ユースケース分析1

1. 画面イメージから、ユースケースを抽出していく

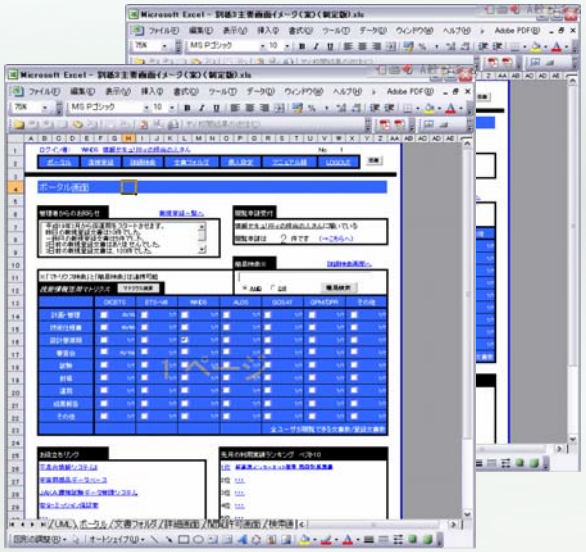
目的・方針

要求仕様

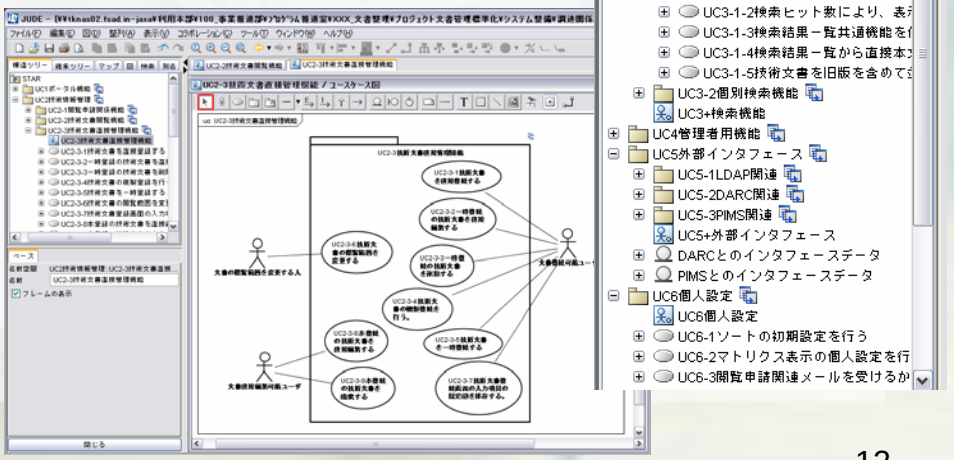
設計

プログラミング・試験

完成・改善活動の継続



•画面イメージを見ながら検討。最初は思い付きを逃さず書く
•必要に応じ、階層的に分類
  •粒度のレベルを合わせていく
  •1画面最大10個程度



•気がついた改善点があれば、画面イメージも修正する

3-2b. ユースケース分析2

- ユースケースシナリオ
  - 楕円一つ一つについて、シナリオを作成。
  - 拡張シナリオ、例外シナリオも含める
  - シナリオが長くなりすぎた場合は、ユースケースを分割する
- アクタ分析
  - そのユースケースは、誰が主で実施するかを考える
  - 登場したアクタに必要な権限（ロールと呼んでいる）を比較・分析し、統合する、もしくは継承関係を結ぶ
  - ここで分析した内容は、システムに必要な権限を考える際にも役に立った
- ユースケース分析のポイント
  - ユーザのユースケースだけでなく、システム運用の管理者用のユースケース、システム利用実績評価のユースケースなども含める
  - 開発ベンダーに頼らず、システムを発注する側で作ることが重要

3-2b. ユースケース (サンプル)

- ツールは、JUDEとExcelを使用
- ユースケース数(システムの規模による):
  - GOWF:約330個(→細かすぎたと思っている)
  - STAR:約110個

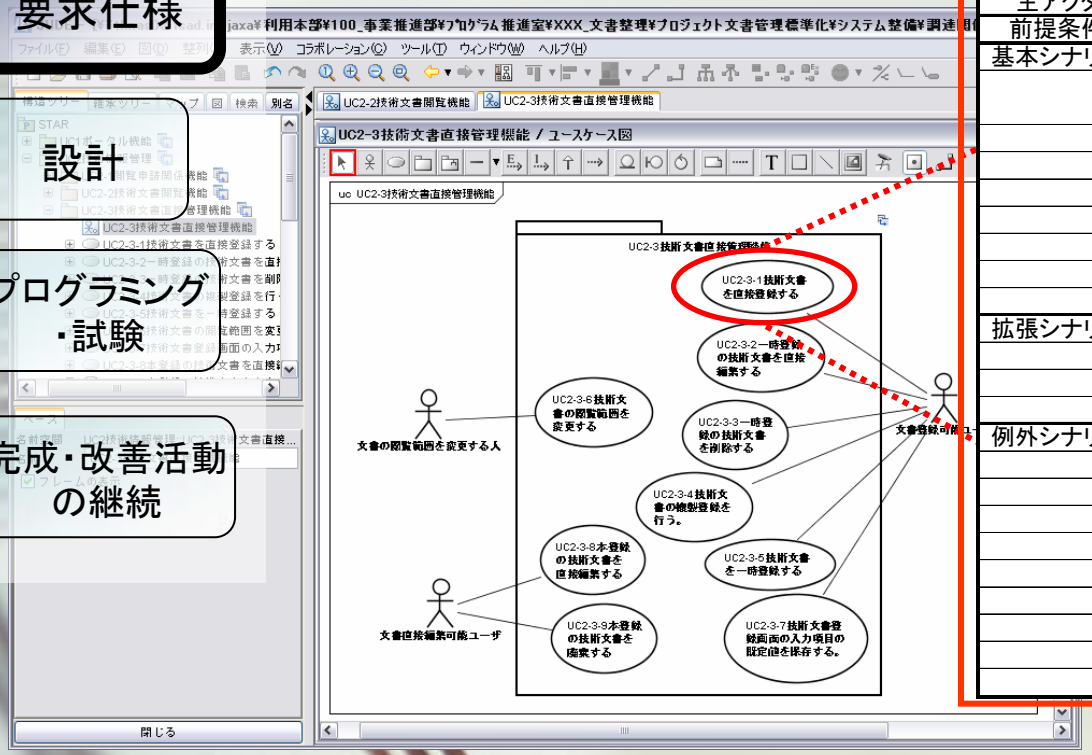
目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続



ユースケース	技術文書を登録する
整理番号	UC2-3-1
主アクタ	技術文書一時登録可能な人
前提条件	直接登録・編集画面が表示されている
基本シナリオ	
1	STARは、デフォルトで入力可能な項目を入力する。それ以外にデフォルト設定されている項目があれば入力する。
2	技術文書一時登録可能な人は、必須項目を入力する
3	技術文書一時登録可能な人は、必須ではない項目を入力する
4	技術文書一時登録可能な人は、本登録ボタンを押す
5	STARは、確認画面を出す。
6	技術文書一時登録可能な人は、OKを押す
7	STARは、その文書を本登録し、記録をログに残す。
拡張シナリオ	
4.1	技術文書一時登録可能な人は、テンプレート設定ボタンを押す(***UC2-3-7***)
4.2	技術文書一時登録可能な人は、文書の一時登録をする(***UC2-3-5***)
例外シナリオ	
4a	必須項目に抜けがあった場合
4a.1	STARは、画面上にエラーメッセージを出し、1に戻る
4.b	必須項目に不正な入力があった場合
4b.1	STARは、画面上にエラーメッセージを出し、1に戻る
4c	必須でない項目に不正な入力があった場合
4c.1	STARは、画面上にエラーメッセージを出し、1に戻る
6a	Noを押した場合
6a.1	1に戻る

3-2b. ユースケースの適度な粒度とは？

•個人的な感触ではあるが。

目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続

- 試験しやすい大きさ、1つの動作が完了する大きさ
- 例：
  - 大きすぎる：「技術文書を管理する」
    - 具体的に何をするのかわからず、試験もできない
  - 小さすぎる：「技術文書の件名を指定する」「技術文書に添付ファイルを追加する」
    - これだけでは、技術文書を登録・編集するという動作が終わらない。このレベルまで細かくすると、ものすごい数になる。1つの機能で数百ユースケースになることも。。。
  - 丁度いい：「技術文書を新規作成する」「技術文書を変更する」
    - 細かい動作、拡張、例外は、シナリオに記述すればよい。

3-2c. 機能要求の記述

- ユースケースを実現するための機能要求を羅列
  - 試験しやすいように1項目1要求を心掛けた(が。。。)
- 製品仕様書の機能要求の一つ一つの要求事項とユースケースの紐付けし、双方をブラッシュアップ。

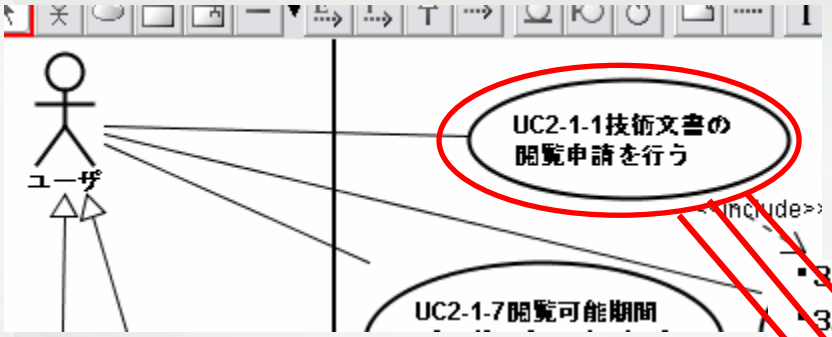
目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続



•必要な機能要求の記述
•不要もしくは冗長な機能要求の削除

3.2.2.2.技術情報管理機能

3.2.2.2.1.閲覧申請関係機能

(1) 本文データの閲覧が制限された技術文書について、閲覧申請を
(2) 閲覧申請は、閲覧を許可するユーザ(およびグループ)へのメー
(3) 閲覧申請には、申請理由(用途、利用範囲)の記載ができること。
(4) 閲覧申請を受けて、申請者にのみ本文データを開示する機能を
(5) 閲覧申請通知メールには、申請内容(対象技術文書の文書番

•必要なユースケースの追加
•不要もしくは冗長なユースケースの削除

3-2c. その他製品仕様書について

- 機能要求以外に書いた要求（一部）
  - 一般的な要求
    - 日本での稼動実績、導入実績の要求
    - オープンソースの活用
  - 開発要求
    - J2EEテクノロジーの使用
  - システム全体要求
    - 文字コード
      - 日本語（2バイト文字）がシステムで扱えるように（扱えなくて揉めたことがあった）
    - レスポンス要求
      - 想定ハードウェアを提示した上での、検索結果表示へのレスポンス要求（要求しないと、開発メーカが手を抜く可能性あり）
    - 画面設計要求
      - 戻るボタンの動作（検索結果一覧から詳細画面に移り再度戻る時に結果一覧に戻れない場合があった）、ブラウザのアドレス欄の非表示、右クリックの禁止など
    - セキュリティ要求
      - 添付ファイルの保管を別ドライブとすることなど（直接アクセスできるシステムがあり、問題となった）

3-3. 開発ベンダーとの設計会議

目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続

- 会議で何を決めたか（STARの時）
  - 具体的なシステム（ハード）構成の決定
  - 画面の決定←開発メーカによる、よりよい画面構成の提案
  - システム間インタフェースの具体的な実現方法
  - 扱うデータ項目の決定
  - データへのアクセスコントロールの詳細
  - インタフェース試験に関するコンフィギュレーション
  - 検索時の細かい動き

  - 詳細設計結果と機能要求との整合性（勘違いがないか）を人手で確認
    - もちろん誤解が全く無かったわけではない。

3-4. プログラミング

- 実装方法は、お任せ
  - Java Studio Creator、EclipseといったIDE（Integrated Development Environment）を使用
  - STARでは、FindBugというEclipseのプラグインを用いて、ソースコードの質をアップ
  - Struts、JSFなどのフレームワークを使い、オープンソースとして公開されている部品を再利用
- 命名規約、コード規約なども任せたが。。
  - Javaの場合、パッケージ名の大まかな規約をこちらから提示しても良いかもしれない。
    - “jp.jaxa.???.???”など
  - クラス名やデータ項目名を“ローマ字表記”とするか、“英語標記”とするかも少し揉めた。
    - ローマ字はかっこ悪くないか？いや読みやすさが一番？など

3-4. 試験

- 開発ベンダー内
  - 単体試験は、開発ベンダーの社内にて実施。
    - STARでは、JUnitという試験フレームワークを使用
- JAXAにて
  - ユースケースシナリオベースの試験手順書にて、要求した機能を全て満たしているかを確認
    - STARでは、統合ディレクトリシステム、他文書管理システムとのインタフェース試験も実施（環境準備・データ準備に苦労した）
  - エラー等を発見した場合は、指摘票を起票し、すべての指摘票のクローズを求める
    - GOWF、STARでは、3,4人の職員でそれぞれ数十件の不具合が発生し、指摘票を起票した。（単純なバグ、環境設定のミスに起因するものが多かった）

3-5. 継続した改善活動

- 運用の定着化に向け（STARの話）
  - 要望どおり作ったとしても、使ってみて初めてわかる問題などが発見される場合が多い
    - 利用者の声に耳を傾け、可能な限り継続的な改善活動が必要
  - 最初のうちはシステムの存在すら知らない人も多い
    - 継続的なアピールが必要
    - 業務フローへの組み込み
    - ここにしかないコンテンツ
  - STARの実情
    - 4月運用スタート、現時点でシステム名は浸透してきたが、利用数は伸び悩んでいる
    - 利用者からの希望・クレーム・要望を集め、またログから利用傾向を分析し、可能な限り改善に向けた対応していく予定
      - 要望コンテンツの掲載
      - 今年度システム改修を実施する

目的・方針

要求仕様

設計

プログラミング・試験

完成・改善活動の継続

4. 効果・結果

画面イメージ・ユースケースを使った
結果・効果は！？

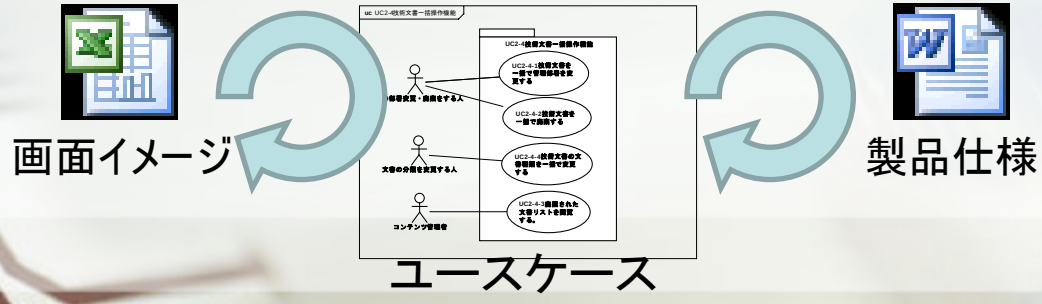
4-1. 結果・効果（良かった点）1

•改めて、見た目、操作性がユーザのシステム利用に重要かを再認識した。

- 画面イメージを使った手法
  - ユーザから具体的で主体的な要望が聞こえてくるようになった。
    - 「再点検は、点検済みのチェックをはずすイメージの方がいい」
    - 「見ただけで何をすればいいかわかるように、文字の色を変えたほうがいい。色の意味の説明も付け加えて。」
  - 発注者側も要求を理解できていた
    - 自信を持って要求機能を正確にベンダーに説明できた:「そのボタンを押すと、こういう動作をしてほしい」「そのリンクをクリックすると別画面が立ち上がる。」など
  - 開発ベンダーによるソフトウェア製造のスピードが上がった。
    - 画面作成にすぐ着手できていた
    - 画面に関しての意識の齟齬は、ほぼなかった
    - 「より機能的な画面」の提案もあった

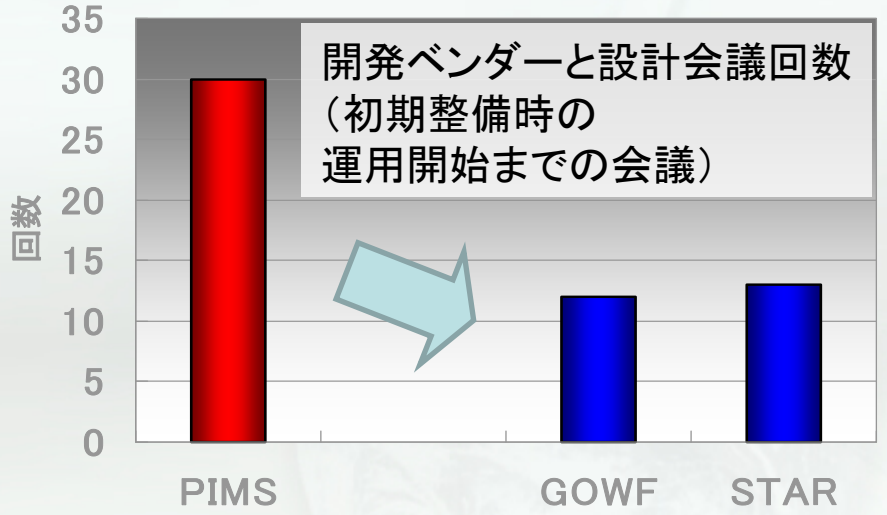
4-2. 結果・効果（良かった点）2

- ユースケースを使った手法
  - 画面イメージが洗練された
    - 画面イメージを見ながら、ユースケースを想像していくことで、細かい画面イメージの修正も（フィードバック的に）行えた。
  - 製品仕様書が洗練された
    - 仕様書上の要求事項がそもそもどうして必要になったかがわかりやすくなった。
    - 開発ベンダーの要求仕様の誤解が減った
  - 例外シナリオにより、エラーケースが明確になった
    - 例外シナリオをベースにエラーチェックを実装できた

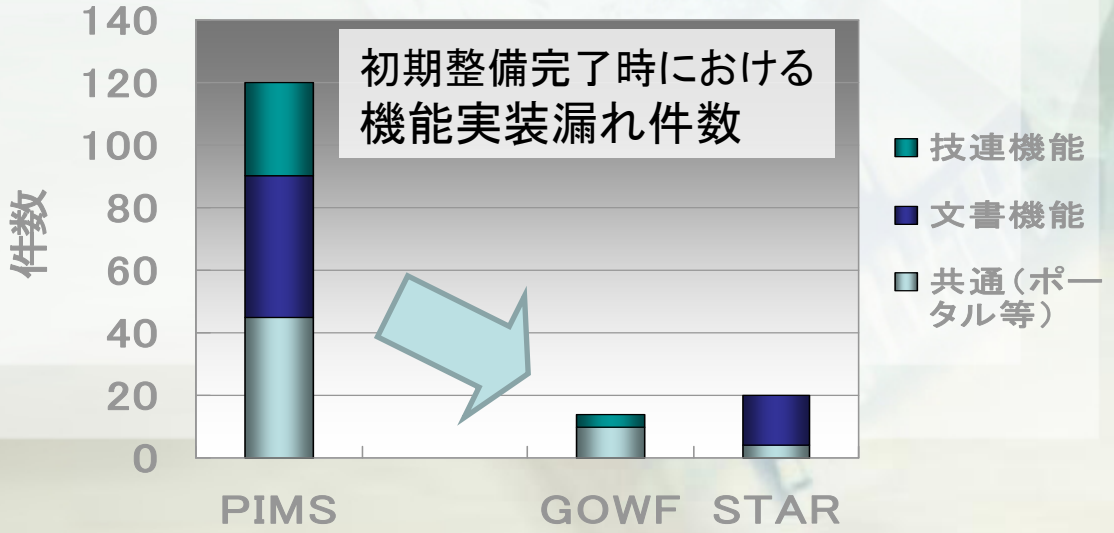


4-3. 結果・効果（悩ましい問題は!）1

a. 長時間の設計会議
   - 設計会議の回数が半分以下に減
   - 1つ1つの会議時間も短くなった（深夜に及ぶ会議がなくなった）



b. 要求の誤解から来る手戻り
   - 初期整備完了時における機能実装漏れ件数が大幅に減



4-4. 結果・効果
（悩ましい問題は!）2

•前頁補足

- ただし、状況が異なるので、単純な比較はできないが。。
  - PIMSは、要求機能の数が他と比べて多かった
  - GOWFは、PIMSの設計思想をベースに議論できた。

- 実感として、開発ベンダーとの意思疎通がスムーズになった
  - 競争入札の手続き時の開発ベンダーとの質疑応答もシステムの内容に深く踏み込む具体的なものが多かった
  - GOWFは、契約期間4ヶ月という短い期間で開発完了し、運用開始できた。
  - STARを落札した業者は、JAXA利用本部の業務フローの知識について0からのスタートであったが、特に意思疎通で苦労はしなかった。

5. 今後の課題

画面イメージ・ユースケースを使った際
デメリットは！？未解決の問題は！？

5-1. 今後の課題1

- 要求管理ツールの活用
  - 作成した資料の整合性（コンプライアンスマトリクス）など、維持管理するのが非常に面倒
    - 実際、JAXA側作成資料間でさえ、結局やりきれなかった
    - JAXA要求とメーカ設計結果の間は、なおさら
  - 作成した資料の項目間のリンクを維持できるようなツールの活用が必要
    - Doors＋TAU、RequisitePro＋ROSEなど
- 要求実現性の事前検証
  - 実現できない要求を書いてしまう危険性がある。
    - 例：レスポンスを要求するのはいいが、本当に実現可能か？
    - 今回は、別システムと比較しつつ、「おそらくできるだろう」とした
  - 経験、調査、シミュレーションなどで解決できる！？

5-2. 今後の課題2

- 要求と設計の分離
  - 要求の中に設計的要素（システムの作り方の話）が排除しきれなかった
    - 例：「登録時、本文データのサーバ計算機のOS上の実体ファイルは、部署毎のフォルダをOS上に作成し、そこに格納されること。」
  - 実現方法の制約となってしまう
    - より良い実装方法があるのに
    - COTSを使えばもっと安くできるのに
  - 実装方法は、プロに任せ、我々は、「如何にITを使い、目的（効率化など）を達成できるか」に力を入れたいところ

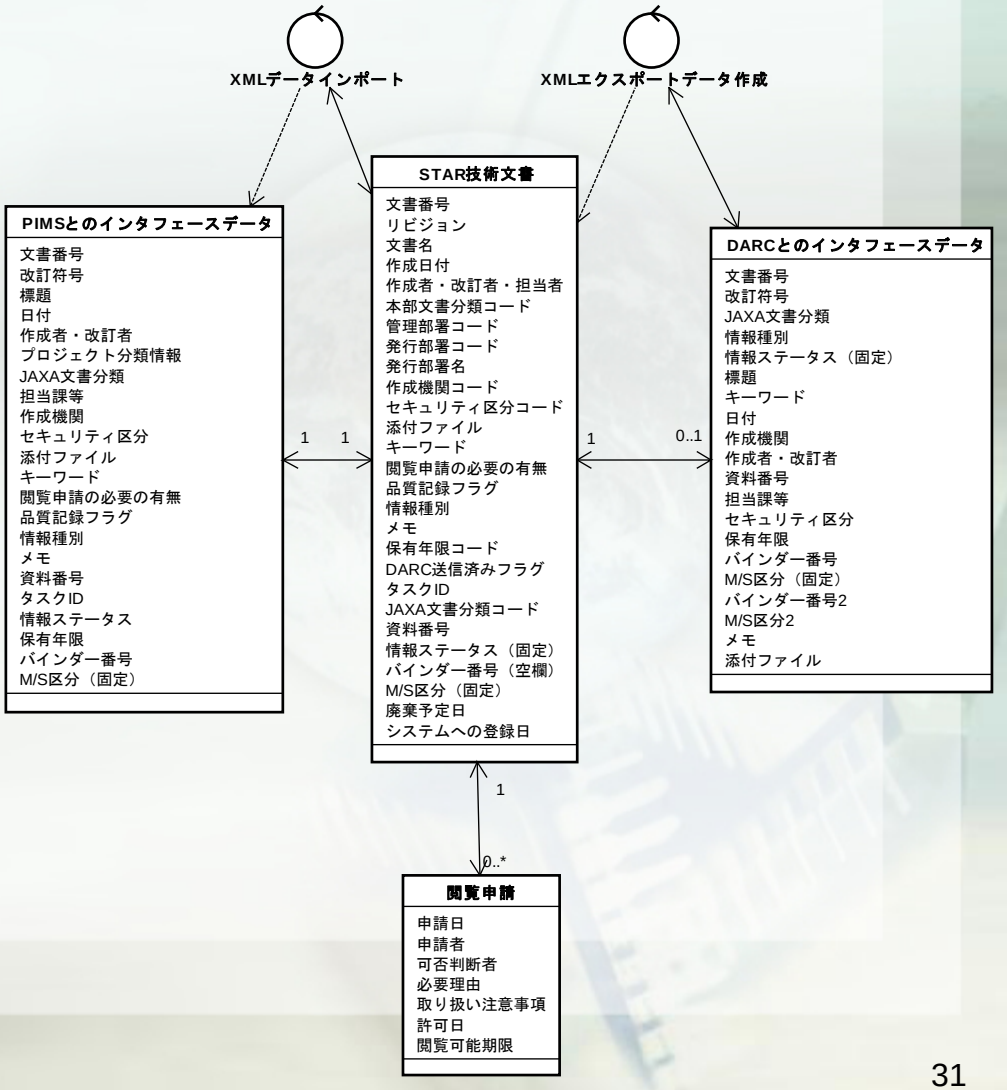
5-3. 今後の課題3

- 更なる要求の明確化
  - エンティティ分析
    - このシステムで、どのようなデータを扱うかを可能な限り分析しておく
    - クラス図にまとめるのが良いと思う（次ページにサンプル）

    - 実際は、「詳細設計のときに決めればいいか」と要求仕様作成時に決め切れなかったため、後悔した。
    - ここをもっと整理しておけば、更に設計会議にかかる時間が短縮できたと思う

5-4. エンティティ分析 (サンプル)

- 例（STARの場合）:
  - 右の図は下記の「情報」と「情報」間の関係
    - 文書の書誌情報
    - 閲覧申請情報（誰がいつ何を申請し、誰がいつ許可・却下したかなど）
    - システム間インタフェースデータ情報
  - その他、以下も同じようにまとめておくことが可能
    - 部署、グループ、ユーザ情報



6. まとめ

- PIMS、GOWF、STARのシステム構築時、画面イメージ・ユースケースを活用したユーザ要求の具現化の手法を徐々に導入した。
- その結果、ユーザとのコミュニケーション、開発ベンダーとのコミュニケーションが円滑になり、要求仕様の定義および設計作業がスムーズになった。
  - 設計会議にかかる時間が減った！
  - 要求の誤解から来る手戻りが減った！
- 今後の課題
  - 要求管理ツールの活用
  - 要求実現性の事前検証
  - 要求と設計の分離
  - 更なる要求の明確化

ありがとうございました

情報システム以外のシステム（衛星を使った地球観測システムなど）への応用は可能だと思いますか！？