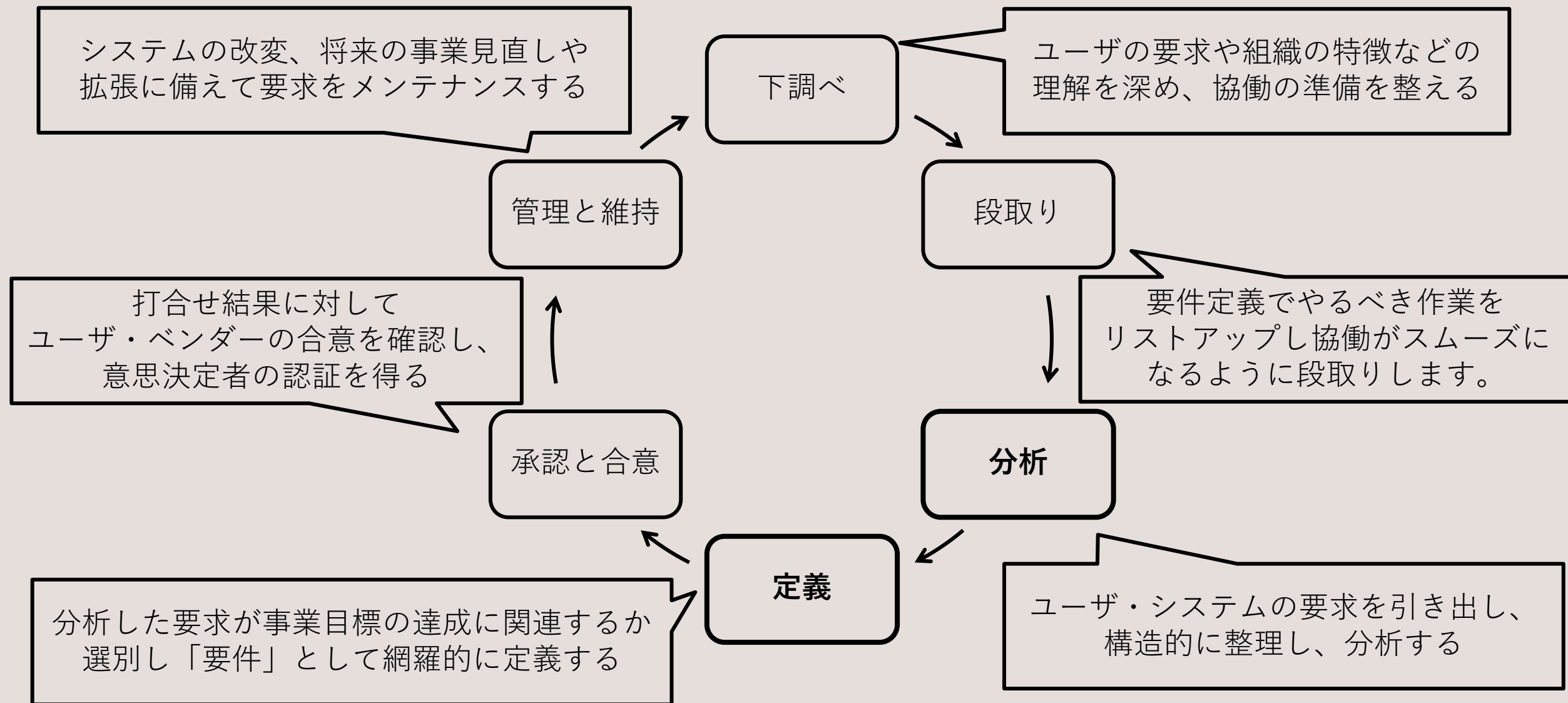




システム設計実践演習 第6回

要件定義作業の全体像





非機能要求の 分析・定義

非機能要求の分析

機能要件では、システムに求められる機能を定義しました。
それらの機能が、どの程度厳密に求められているのか、**機能が提供するサービスのレベルや品質**についても決めておくことが必要です。これを**非機能要求**といいます。

例えば、登録処理で「ボタン押下後、何秒以内に登録が必要か」というサービスの速さ(応答の性能)は、非機能要求のひとつです。

非機能要求グレードの利用

非機能要求は、機能と異なり定義が難しいのが実状です。なかにはシステム基盤に関することも多く、IT知識が豊富でないユーザは、要求を正しく伝えられない問題があります。

そこで、IPAが非機能要求の見える化と確認の手段を提供する「非機能要求グレード」を取りまとめて公開しています。

<https://www.ipa.go.jp/archive/digital/iot-ency/jyouryuu/hikinou/ent03-b.html>

非機能要求グレードの利用

非機能要求グレードには非機能要求を「可用性」「性能・拡張性」「運用・保守性」「移行性」「セキュリティ」「システム環境・エコロジー」の6分野に分け、ユーザとベンダーとの間で確認すべき項目が200以上リストアップされています。各項目についてレベル(0から5)とメトリクス(指標)が定義されています。

非機能要求グレードを使った要求定義の進め方

ステップ	内容
1	非機能要求グレードの「モデルシステムシート」から、 今回のシステム化案件のモデル を判断する(16項目の質問に答えると、システムの社会的影響度が3パターンに判断できる)
2	非機能要求グレードの「樹系図」に示された重要項目と、ステップ1のモデルを参照しながら、 検討すべき項目を抽出する。
3	検討すべき項目について、ユーザとベンダーが 適切なレベルを話し合う。
4	決定事項を設計フェーズに引き継ぐ。また必要に応じてステップ3以外の項目についても議論、決定し、設計に反映させる。

※あくまでIPAの推奨であり、この方法でなくとも良い

非機能要求(分析・定義)の構成要素

<p>非機能に関する分析・定義</p> <p>(可用性、性能・拡張性、運用・保守性、移行性、セキュリティ、システム環境・エコロジー)</p>	<p>非機能要求を、「非機能要求グレード」を参考に6種類に分類し、それぞれ求められること(要求)を分析して、要件を定義する</p>
<p>非機能要件の文書化</p>	<p>上記で、分析・定義した内容を文書化して、「非機能要件定義書」を作成する</p>

可用性に関する分析・定義

システム稼働時間や停止予定などの運用スケジュール、障害や災害時における稼働目標について、**システムサービスを継続的に利用可能とするための要求**です

運用スケジュールや業務の継続性、システム障害時の目標復旧水準および稼働率などを定義します。

「可用性」の非機能グレードの中分類

中項目	説明
継続性	システムが稼働している状態を定義したり、障害発生時の復旧目標を明らかにする
耐障害性	障害に対する耐性を、システムを構成する要素の単位(例：業務単位)で分割して明らかにする
災害	耐障害性のうち、特に大規模災害に対する対策の考えを明らかにする
回復性	障害発生時、システム回復やデータの復旧についての能力と、必要な労力について明らかにする

性能・拡張性に関する分析・定義

業務量の見積もり、今後の増加の見積もり、システム化対象業務の特性(ピーク時、通常時、縮退時)について、**システムの性能、および将来のシステム拡張に関する要求**です。

業務処理量、性能目標値、リソースの拡張性などを定義します。

「性能・拡張性」の非機能グレードの中分類

中項目	説明
業務処理量	通常時の業務量と、業務の追加度合いを明らかにする。 ある程度余裕を持った値を仮決める。
性能目標値	業務処理の特徴(オンライン・バッチ)やピーク特性（ピークの頻度や時間帯）、縮退を考慮し、性能目標を確認する。
リソース拡張性	システム稼働中にどれぐらい空きが必要かを明らかにし、利用率やリソース増設の可否を確認する。

運用・保守性に関する分析・定義

運用中に求められるシステムの稼働レベルや、問題発生時の対応レベルなど、**システムの運用と保守のサービスに関する要求**です。

通常運用、保守運用、運用の環境やサポート体制、運用管理方針などを定義します。

「運用・保守性」の非機能グレードの中分類

中項目	説明
通常運用	通常の利用時間と、通常スケジュール以外の特定日(バックアップ日、計画停止など)の有無や、その内容を確認する
保守運用	システムの品質を維持するために実施するメンテナンス作業の方針や内容を明らかにする
障害時運用	システム障害発生時の対応（復旧作業の内容、異常県知事の対応など）を明らかにする
運用環境	開発用環境、テスト用環境、リモートオペレーションなど、運用の対象やマニュアルの準備を確認する
サポート体制	運用について、ユーザとベンダー間の役割分担や、サポート体制や保守契約の内容を確認する
運用管理方針	内部統制、サービスデスク、インシデント管理など、対応方針や具体的な実現方法を確認する

移行性に関する分析・定義

新システムへの移行期間および移行方法、移行対象資産の種類および移行量など、**現行システム資産の移行に関する要求**です。

移行時期、移行方法、移行対象などを定義します。

「移行性」の非機能グレードの中分類

中項目	説明
移行時期	システム切替までの期間や、移行作業中にシステム停止が可能な 否か、並行稼働が必要かを明らかにする
移行方法	複数場所へ設置の場合、一斉か多段階移行かを明らかにする 複数業務が対象の場合、新旧システムの共存稼働の範囲や期間を 明らかにする
移行対象	移行対象の機器やデータについて、一斉か部分的な入れ替えかを 明らかにする。部分的な入れ替えの場合は、新旧の互換性を確認 する。データ移行については移行ツールを検討する。
移行計画	ユーザーとベンダーの作業分担を明らかにする。 外部連携についてリハーサル環境、回数などを検討する。 移行中トラブルが発生した場合、切り戻しや対応プランを検討す る。

セキュリティに関する分析・定義

利用制限や不正アクセスの防止など、**情報システムの安全性の確保に関する要求**です。

アクセス・利用制限、ネットワーク対策、データの秘匿、不正追跡・監視などを定義します。

「セキュリティ」の非機能グレードの中分類

中項目	説明
前提・制約事項	業界の基準や企業の方針に対して順守すべき規定、法令、ガイドラインの有無を確認し、セキュリティ対策を検討する
セキュリティ分析	システム開発に対して、脅威の洗い出しの範囲や、影響分析の実施の有無についての方針を確認する
セキュリティ診断	対象システムや、各種ドキュメントに対して、セキュリティに特化した各種試験や検査の実施の有無を確認する
セキュリティリスク分析	運用開始後に発見された脅威の洗い出しとその影響分析の対象範囲や対象方針を確認する
アクセス制限・利用制限	システムで扱う資産へのアクセスおよび利用の制限について、対象(サーバー・ストレージなど)ごとに明らかにする

「セキュリティ」の非機能グレードの中分類

中項目	説明
データの秘匿	機密性のあるデータを、伝送時や蓄積時に秘匿するための暗号化を実施するかを確認する
不正追跡・監視	システムの運用後に発生する不正行為の追跡や監視について、監視範囲や記録保存の期間などを明らかにする
ネットワーク対策	不正な通信を遮断するための制御や、システム内の不正行為や通信を検知する仕組みの要否を明らかにする
マルウェア対策	マルウェアの感染を防止する対策の実施範囲やタイミングを確認
Web対策	Webアプリ特有の脅威、脆弱性に関する対策を実施するかを確認
セキュリティインシデント対応	セキュリティインシデントが発生した時に、早期発見し、被害の最小化、復旧の支援などをするための体制について確認

システム環境・エコロジーに関する分析・定義

システムをどこに設置するかなど環境への配慮も含んだ要求です。

システム制約・前提条件、システム特性、適合企画、機材委設置環境条件などを定義します。

「システム環境・エコロジー」の 非機能グレードの中分類

中項目	説明
システム制約/ 前提条件	社内基準や法令、各地方自治体の条例などの制約の存在を確認する
システム特性	システムの規模や特性を決定付ける諸項目について確認する
適合規格	使用製品についての製品安全規格、機器自身が放射する電磁波について、規格取得の要否を確認する
機材設置 環境条件	耐震・免振、スペース、重量、電気設備の適合性など機材設置の環境条件の要否を確認する
環境 マネジメント	消費電力、発熱量やデータセンターのエネルギー効率などを確認する

対応は過剰も過少もだめ

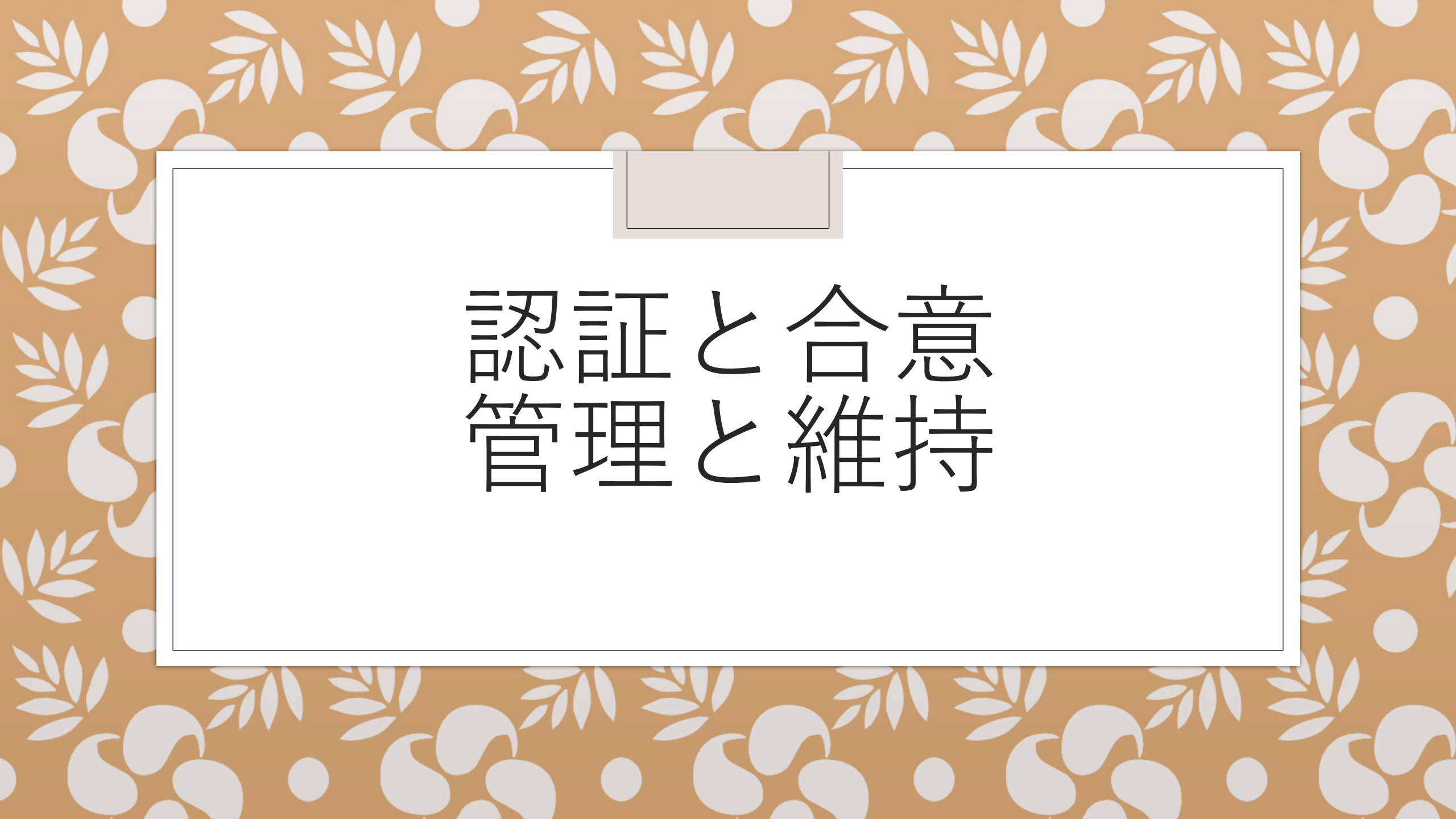
非機能要求の実現は、**コストとのバランス**を考えなければなりません。例えば、可用性を高めるために、冗長化にお金をかけすぎるとは、機材の購入および維持するための人件費なども多くかかり、利益を圧迫する要因となります。

逆に、必要な準備を行った場合は、トラブルが頻発し顧客や取引先からの信用を失うこととなります。

非機能要件の文書化

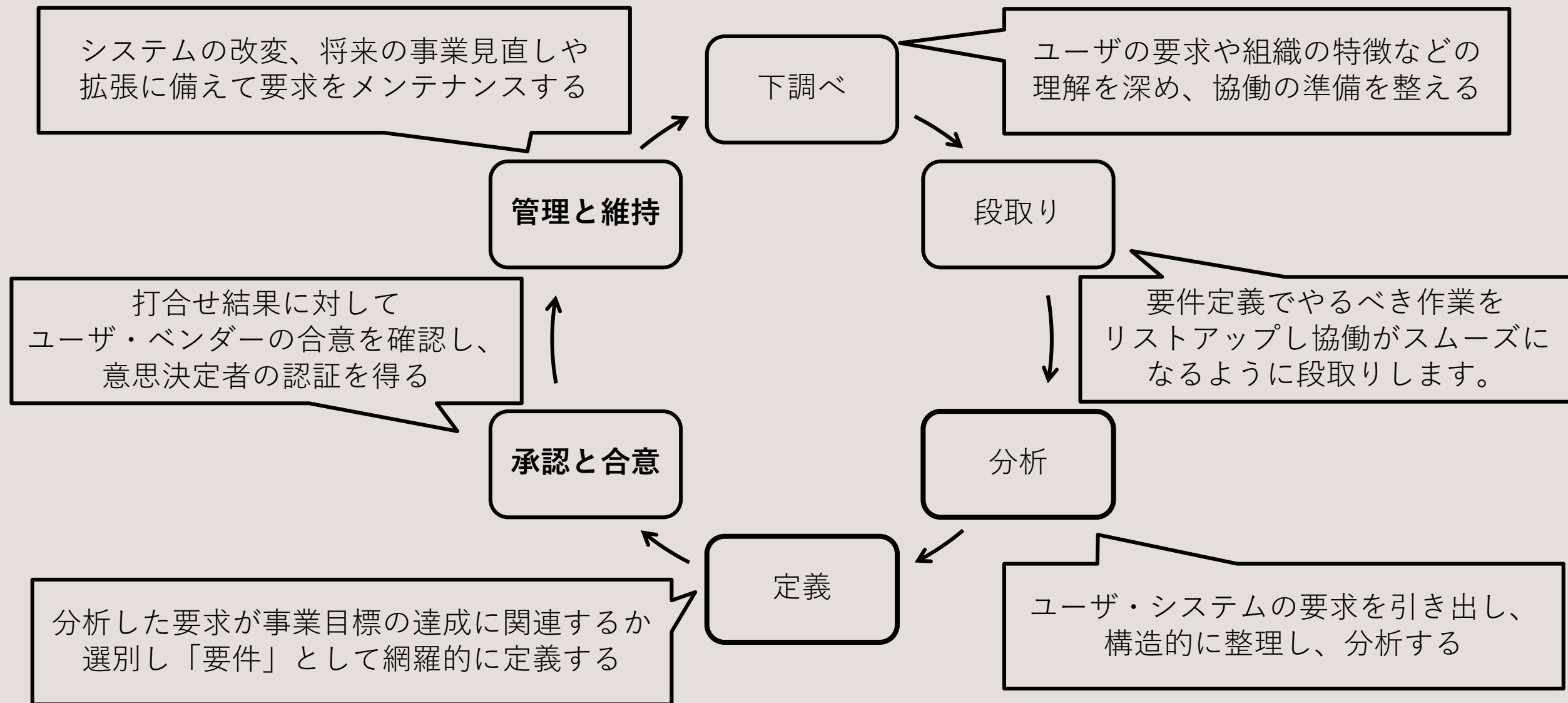
個別で分析を進めてきた、それぞれの非機能要件を取りまとめて、新業務の実現に必要な非機能要件のレベルを文書にまとめます。

非機能要件定義書の各事項は、確定の結果を記すだけでなく、そのように決めた**根拠や事情も併せて記載しておく**と、設計工程で役立ちます。



認証と合意 管理と維持

要件定義作業の全体像



認証と合意、管理と維持について

「認証と合意」と「管理と維持」の工程についてはこの授業の範囲外の内容なので割愛します。



ガントチャートと WBS

WBS(Work Breakdown Structure)

作業分解構成図とも言われ、プロジェクトの遂行する際の管理手法の1つ。

WBSとは**作業(Work)**を**分解(Breakdown)**して**構造化(Structure)**する。

プロジェクトの目標を決め、メンバーに作業を割り当てる。

プロジェクトを小さな作業に分解し、**プロジェクトの見える化**する。
また、進捗管理にも役に立つ手法。

ガントチャートなどのワークフロー管理ソフトを利用して作成する。

WBSの目的

- スケジュールの作成

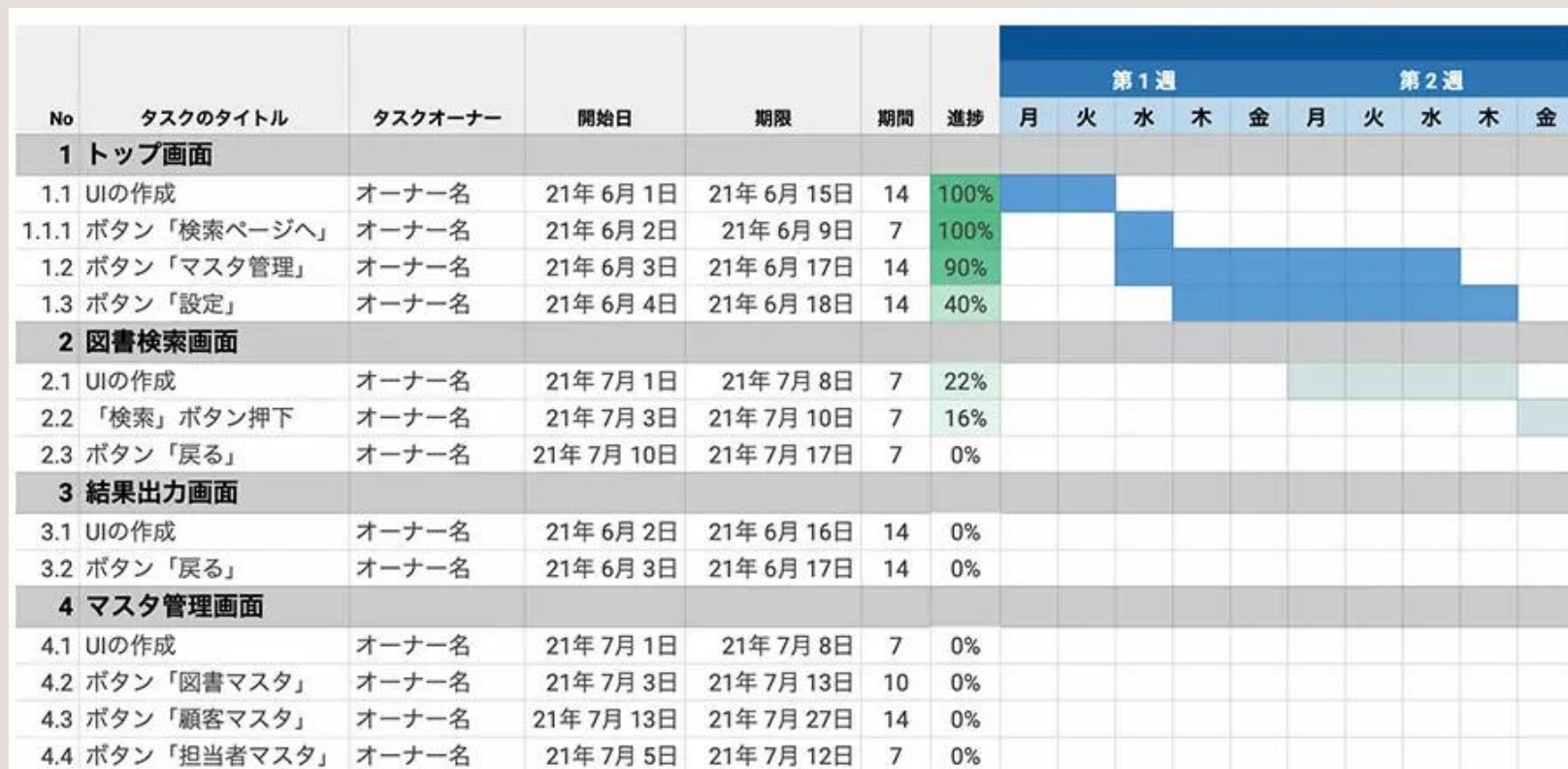
細分化されるのでやるべきことが明確化される
正確なスケジュールを立てることが出来る

- 仕事の効率化

作業をリストアップするので担当者全員が仕事の全体を理解し、
タスク同士の関係性を把握しやすくなる。
そのため、どの作業を優先すべきかわかる。

WBSとガントチャート

WBSはプロジェクトにおけるタスクを細分化したリストであり、ガントチャートはそれに基づいて作られる横向きの棒グラフ表を指します。



WBS

ガントチャート

WBSの作成方法

1. タスクの洗い出し
2. タスク間の依存関係を見出す
3. 各タスクの所要時間を見積もる
4. タスクを担当者に割り振る

WBSの作成方法

。タスクの洗い出し

プロジェクトからどのようなタスク(仕事)があるか洗い出す
まずは大きなタスクから書き出して、細分化していく

必要な機能は何か？

その機能を実装するためには何画面必要か？

画面間の遷移をどうするか？

その機能・画面ではどのような情報を管理しないといけないか？

WBSの作成方法

- タスク間の依存関係を見出す

タスクには、他のタスクに関係なく独立して開始できるものとタスク同士に依存関係があるものがあります。

例えば、ユーザ登録機能を実装しようとしてもデータベースがまだ設計・構築されていなければ作業を開始できないなどが、これに当てはまります。

この場合、タスクはデータベースの構築から進める必要があります。
※画面デザイン・レイアウトなどは平行して進めれると思います

WBSの作成方法

- 。各タスクの所要時間を見積もる

タスクに費やす所要時間を見積もり、各タスクの開始日・終了日を決定します。そうすることで、全体を通してかかる期間が把握できます。

この時、作業感はギリギリではなく多少余裕を持たせた時間を設定することが重要です。バッファを持たせることでスケジュールに遅れが出て調整できる余地が生まれます。

WBSの作成方法

- 。タスクを担当者に割り振る

タスクの作業時間とタスク間の依存関係に気を付けながら各タスクの担当者を決めましょう。

WBSは進捗管理にも使います。各チーム進捗状況を把握しながら作業をすることを忘れないでください。

作業が遅れそうなら、早めにチーム相談

タスクが早く終わったなら、率先して仕事がないかを探ること



工数の考え方

システム開発にかかる費用

モノを作るとき、一般的には材料と機械、工程を管理する人などが必要です。しかし、システム開発の中心はソフトウェア開発であり、材料は微々たるものしか必要ありません。

したがって、システム開発にかかる主な費用は人件費です。プロジェクトに関わる人数とそれぞれの給料が分かればおおよその費用を計算することが可能です。

人月と工数

。人月

誰が担当しても一定時間に同じ作業量をこなせると仮定し、
その前提のもとで必要な作業量を表す単位

例えば **1 人が 1 か月(1日8時間×20日)でできる作業量は 1 人月** となる。

。工数

人月などの単位を使ってシステム開発にかかる費用を見積もるために
用いられる考え方。工数は「人数×時間」で作業の量を表します。

例えば、 **6 人月の作業を 2 人で行う工数は 3 人月** となる。

見積もりについて

ソフトウェアの開発を依頼されると、開発会社はまずその開発費用を見積もります。見積もりは「概算見積」と「詳細見積」に分けられます。

概算見積は、開発してほしいシステムの内容を聞いて、その開発にかかる費用をざっくりと見積もるものです。

詳細見積は、要件定義などからシステム化する機能を明らかにして細かな見積もりを提示します。

FP法(フアンクシヨンプォイント法)

システムがどのような機能を持つかに注目して、その機能の開発における難易度などを加味した点数を積み上げて見積もりを計算する方法。

コンピューターの「入力」「出力」「記憶」に着目し、画面の入出力やデータの保存、他のシステムのやりとりなどの項目を整理し、その数に応じて点数化する。

FP法のフ ァ ン ク シ ョ ン タ イ プ

タイプ	内容	例
外部入力	画面からの入力、他のアプリケーションからの取り込みによって、データ更新するもの	ログイン ユーザ登録
外部出力	画面・帳票の出力、他のアプリケーションへの転送などで、計算やグラフの作成などを含むもの	ダウンロード 集計
外部照合	画面・帳票の出力、他のアプリケーションへの転送などで、計算やグラフの作成などを含むまないもの	検索 ログイン履歴
内部論理 ファイル	システム内部にあるデータで、追加や更新、削除などの対象になるもの	ユーザ情報 商品情報
外部 インターフェース	システムの外部にあるデータで、その保守が外部のシステムに任せられるもの	位置情報 気象情報

機能ごとの点数化の係数

タイプ	容易	普通	複雑
外部入力	3	4	6
外部出力	4	5	7
外部照合	3	4	6
内部論理ファイル	7	10	15
外部インターフェース	5	7	10

ファンクションポイントの算出方法

前述した、ファンクションタイプごとの機能数と難易度による係数から**基準値**をまずは算出します。そこから開発するシステムの一般システム特性から影響度を考慮した**調整値**を算出してファンクションポイントを求めます。

【ファンクションポイント計算式】

$$FP = \text{基準値} \times (0.65 + \text{調整値} / 100)$$

※調整値によりFPが65%～135%に調整される。

一般システム特性

※影響度は0~5で設定する

No.	項目	No.	項目
1	データ通信 (Data Communications)	8	オンライン更新 (Online Update)
2	分散データ処理 (Distributed Data Processing)	9	複雑な処理 (Complex Processing)
3	性能 (Performance)	10	再利用可能性 (Reusability)
4	高負荷構成 (Heavily Used Configuration)	11	インストール容易性 (Installation Ease)
5	トランザクション量 (Transaction Rate)	12	運用性 (Operational)
6	オンライン入力 (Online Data Entry)	13	複数サイト (Multiple Site)
7	エンドユーザ効率 (End-User Efficiency)	14	変更容易性 (Facilitate Change)

ファンクションポイントの工数化

ファンクションポイント法による工数化は、1人月で開発できる作業量に対応FPとして表すことで工数化が可能である。

例えばファンクションポイントが「121」のプロジェクトがあったとき、1人月で開発できるファンクションポイントが「15」であれば、 $121 / 15 = 8.066\cdots$ つまり約8人月のプロジェクトだと判断できます。