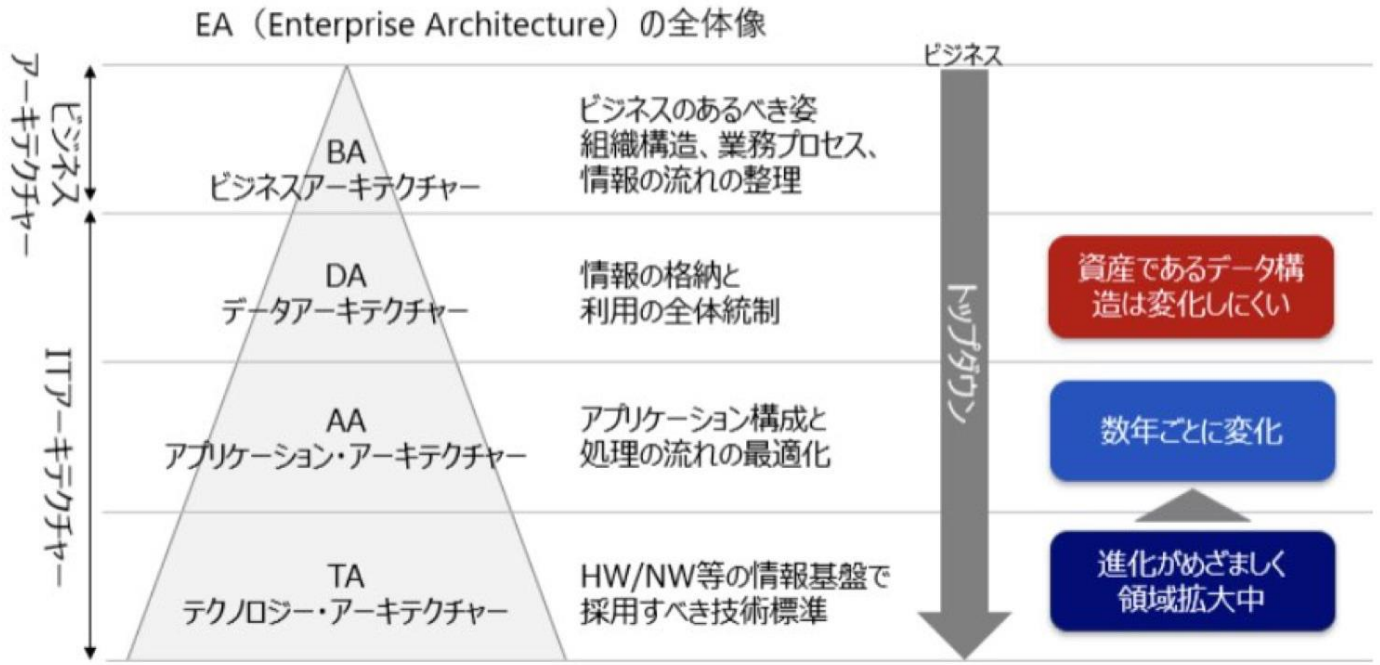
アーキテクチャとは

全体的な構造やシステムを支える構成要素を体系的に配置することを指し、機能、性能、実現可能性、コスト、美しさを最適化することを意図している

AAはEA（Enterprise Architecture)の構成要素の一つ

【キャリア実践】AE力強化研修
例）アーキテクチャ検討の進め方

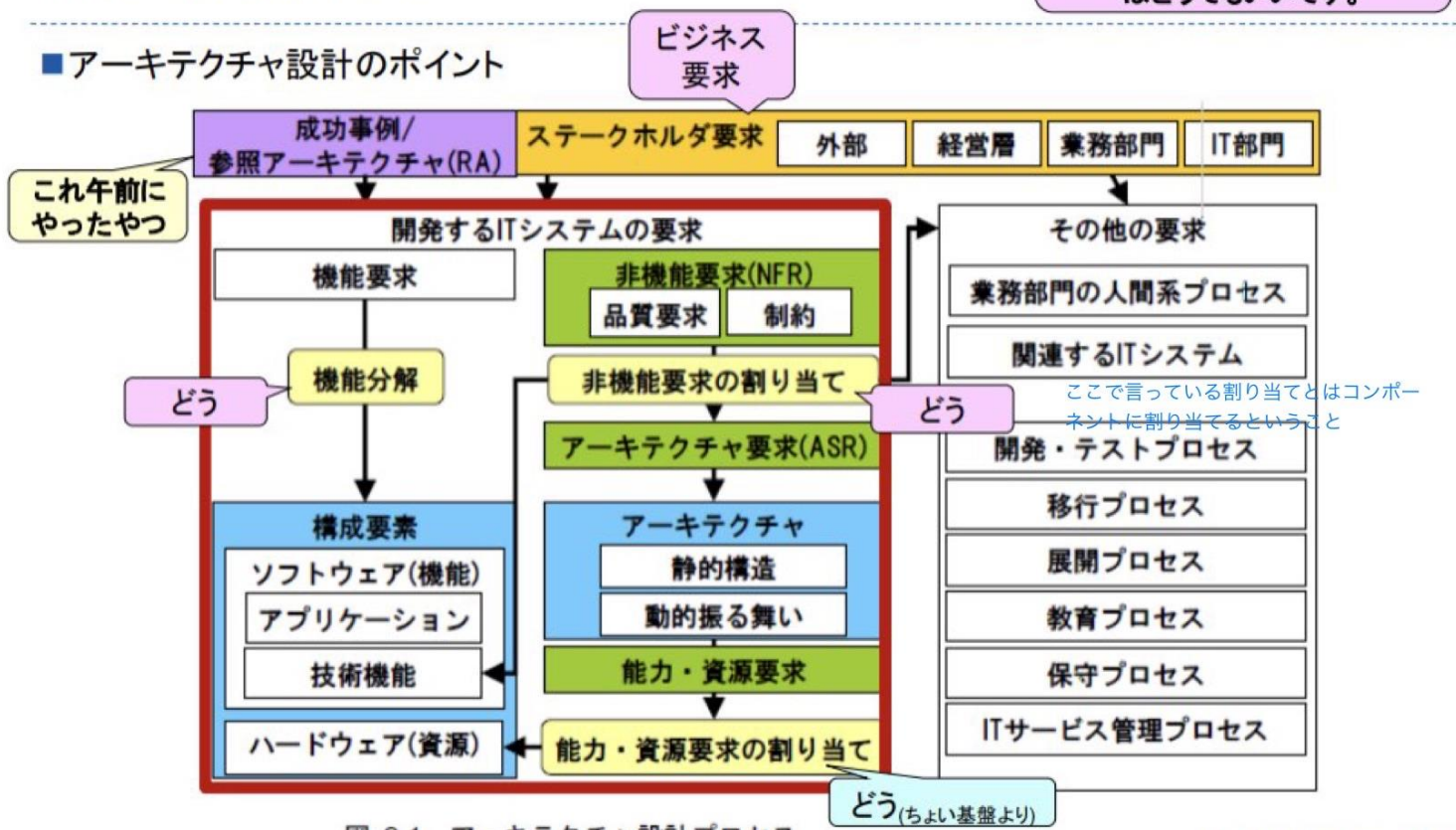
■ **アプリケーションアーキテクチャの位置づけ**



【キャリア実践】AE力強化研修
アプリアーキテクチャについて

この後の研修で「どう」を考えてもらいます。結果(アーキテクチャ)はどうでもいいです。

■アーキテクチャ設計のポイント



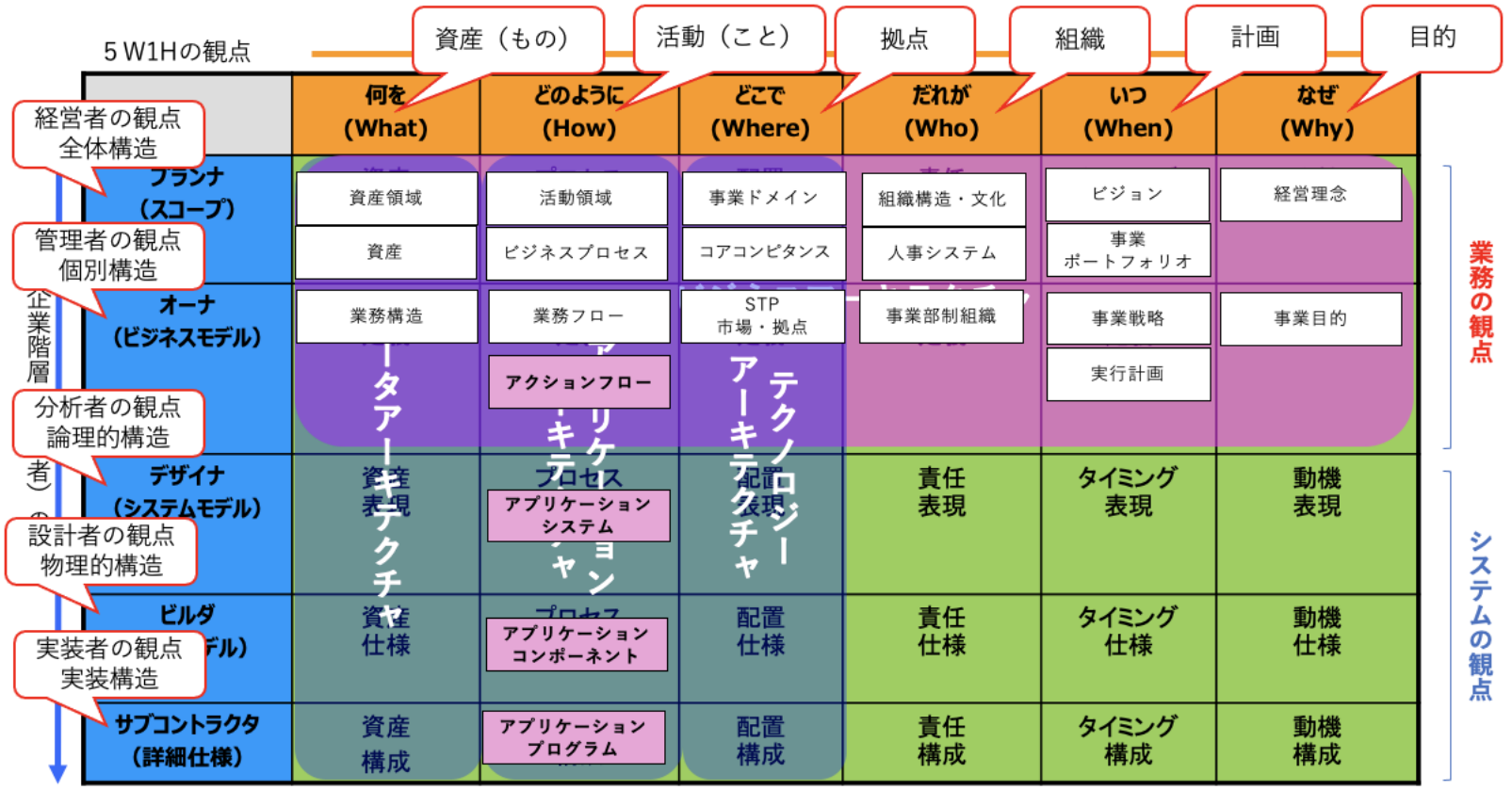
ここで言っている割り当てとはコンポーネントに割り当てるということ

図 2-1　アーキテクチャ設計プロセス

出典：独立行政法人情報処理推進機構
ソフトウェア・エンジニアリング・センター
エンタプライズ系ソフトウェア開発力強化推進委員会
非機能要求とアーキテクチャ分析WG

アプリケーションアーキテクチャの位置づけ

ザックマンフレームワーク

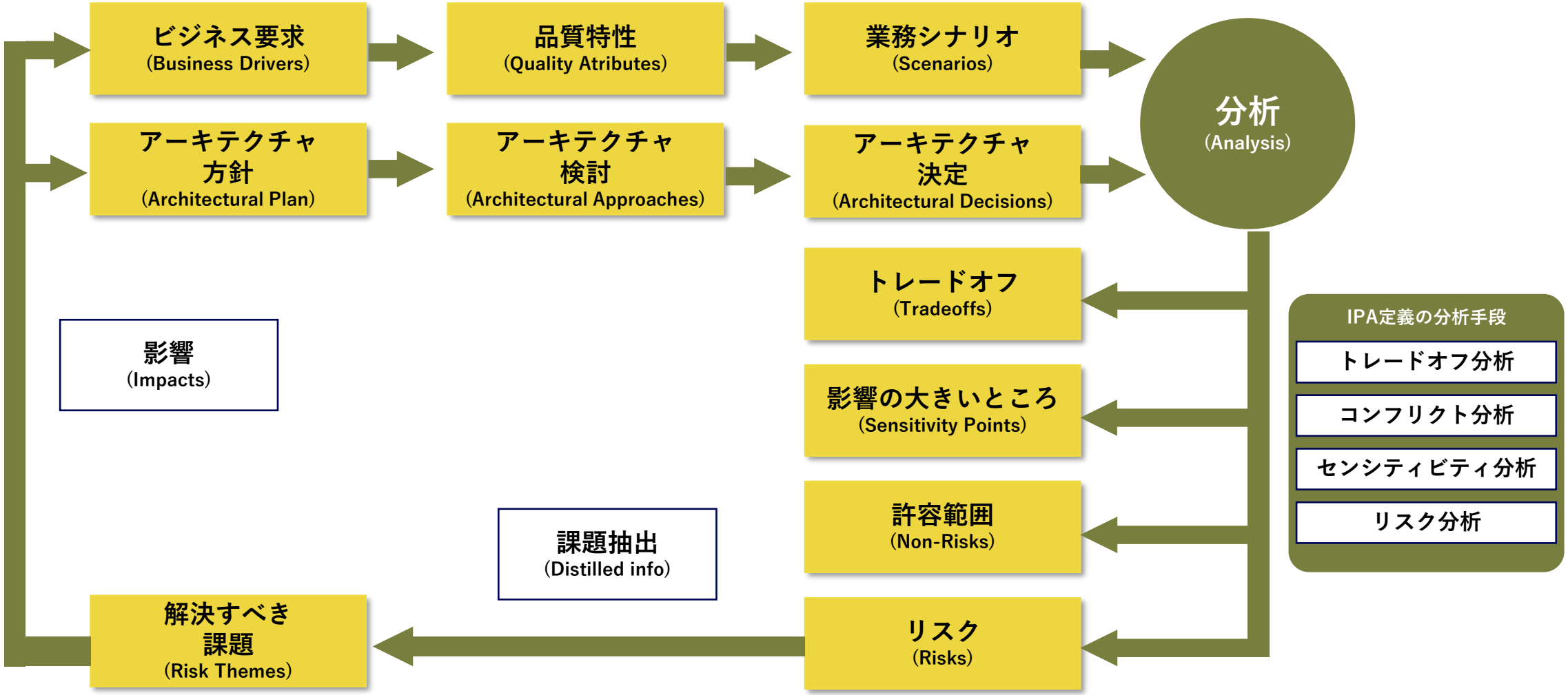


	データ （What）	機能 （How）	位置関係 （Where）	人員 （Who）	時間 （When）	動機 （Why）
範囲 （状況レベル） 計画立案者	重要事項のリスト	ビジネスプロセスのリスト	拠点のリスト	組織のリスト	出来事・周期のリスト	目標と戦略のリスト
ビジネスモデル （概念レベル） オーナー	例：意味論モデル	例：ビジネスプロセスモデル	例：ビジネスロジスティクスシステム	例：ワークフローモデル	例：マスタスケジュール	例：事業計画
システムモデル （論理レベル） 設計者	例：論理データモデル	例：アプリケーションアーキテクチャ	例：分散システムアーキテクチャ	例：ヒューマンインターフェイスアーキテクチャ	例：プロセッシング構造	例：ビジネスルールモデル
技術モデル （物理レベル） 開発者	例：物理データモデル	例：システム設計	例：技術アーキテクチャ	例：プレゼンテーションアーキテクチャ	例：コントロール構造	例：ルール設計
詳細仕様 （状況外） 実装作業者	例：データ定義	例：プログラム	例：ネットワークアーキテクチャ	例：セキュリティアーキテクチャ	例：タイミング定義	例：ルール仕様
実際の企業	例：データ	例：機能	例：ネットワーク	例：組織	例：スケジュール	例：戦略

ザックマンフレームワーク（2007年現在） 出典：http://www.zifa.com/

例）アーキテクチャ検討の進め方

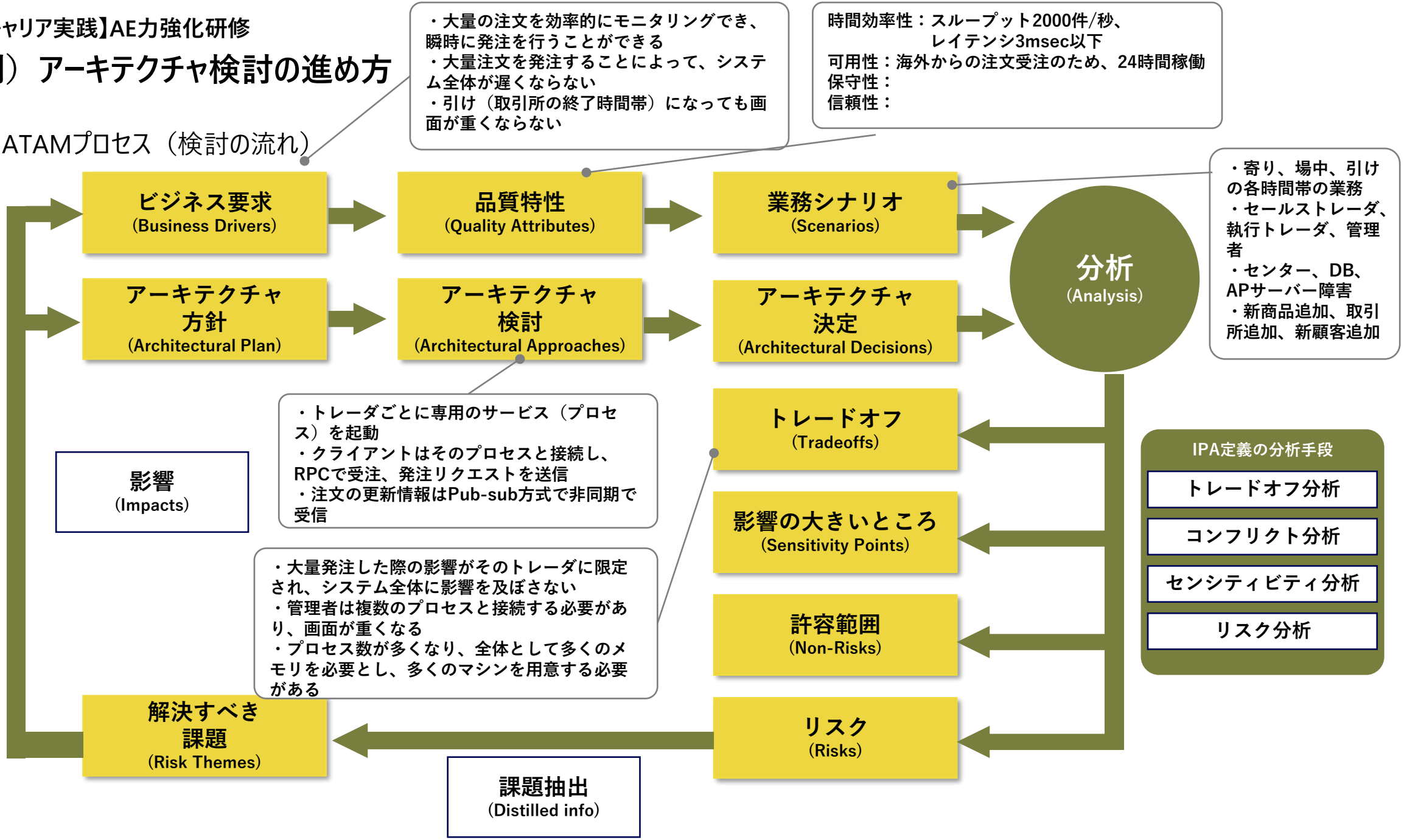
■ ATAMプロセス（検討の流れ）



【キャリア実践】AE力強化研修

例）アーキテクチャ検討の進め方

■ ATAMプロセス（検討の流れ）



【キャリア実践】AE力強化研修
例）要求に潜むリスクの切り口

リスクの高い要求はどううまくやるかだけでなく、影響範囲をどう限定するか、リスクの発生をどう回避するか

■アプリケーションとしての複雑さ（難しさ）

- 業務/アプリ要求の複雑さ(難しさ)
  - □複雑な業務要件（幾つかの特殊な業務要件が含まれる）
  - □品質に関する要求が極めて高い（ミッションクリティカル）
- システム構造/アプリ処理要求の複雑さ(難しさ)
  - □複数のシステム形態が共存
  - □コンポネント間のインタフェースの数が多くシステム構造が複雑
  - □複雑な処理ロジック、タイミング、ファイル整合性などの特性を保持
- 非機能要求の複雑さ(難しさ)
  - □24時間365日の連続稼動で変更、保守、障害回復に高度な設計が必要
  - □機能性,信頼性,効率性,保守性などの要求に対して高度な設計が必要
  - □トラフィック量またはデータ量が多くデータの制御および管理技術が高度
- その他要求(技術要求、プロジェクト要求)の複雑さ(難しさ)
  - □新技術かつ大手企業で使用実績の少ないテクノロジを使用
  - □クロスプラットフォームでのアプリケーション
  - □複数のプラットフォームや開発言語にまたがる複雑な共通基盤の開発

例)要求に潜むリスクの切り口

■アプリケーションとしての複雑さ(難しさ)

- 業務/アプリ要求の複雑さ(難しさ)
  - 複雑な業務要件(幾つかの特殊な業務要件が含まれる)
  - 品質に関する要求が極めて高い(ミッションクリティカル)
- システム構造/アプリ処理要求の複雑さ(難しさ)
  - 複数のシステム形態が共存
  - コンポネント間のインタフェースの数が多くシステム構造が複雑
  - 複雑な処理ロジック、タイミング、ファイル整合性などの特性を保持
- 非機能要求の複雑さ(難しさ)
  - 24時間365日の連続稼動で変更、保守、障害回復に高度な設計が必要
  - 機能性,信頼性,効率性,保守性などの要求に対して高度な設計が必要
  - トラフィック量またはデータ量が多くデータの制御および管理技術が高度
- その他要求(技術要求、プロジェクト要求)の複雑さ(難しさ)
  - 新技術かつ大手企業で使用実績の少ないテクノロジを使用
  - クロスプラットフォームでのアプリケーション
  - 複数のプラットフォームや開発言語にまたがる複雑な共通基盤の開発

リスク回避のためにできること

■事前準備

- プロジェクトが開始されてから検討を開始するのでは遅い
  - プロジェクト失敗の大きな要因の一つは時間が足りないことからくる設計の甘さ
- たいていのシステムは既存システム＋α

■現行アーキテクチャの理解

- 現行システムのアーキテクチャを語れるようになること
  - どういう技術を使っているのか、なぜその技術である必要があるのか
  - 性能面のボトルネックは何か
  - コンポーネントの配置は最適になっているか
  - 将来的な拡張性はあるか

■新技術の検証

- 自ら手を動かして検証（パートナーに依頼する場合でも、自分の頭で理解）
- その技術の正しい使い方を理解（とりあえず動いただけで満足しないように）

【キャリア実践】AE力強化研修
例）機能・非機能の品質モデル

アーキテクチャ検討するに当たって一通り考慮しておく(明示的な要件に含まれてないこともすくなくない)

■機能面の品質モデル

品質特性	特性の概要	副品質特性	概要
機能適合性	実装された機能がニーズを満たす度合	完全性	ニーズを機能がユーザの目的やタスクを包含している度合
		正確性	必要な精度で正確な結果を与える度合
		適切性	機能が定められたタスクや目的を円滑に遂行する度合
性能効率性	システムの実行時の性能や資源効率の度合	時間効率性	実行時のシステムの応答時間、処理時間などの処理能力の度合
		資源利用性	実行時に使用する資源量や種類
		キャパシティ	要求を満たすための製品やシステムのパラメータの最大許容値
互換性	他製品やシステムと機能や情報を共有、変換できる度合	共存性	他製品へ負の影響を与えず、共通の環境や資源を共有して効果的に実行する度合
		相互運用性	2つ以上の製品やコンポーネント間で情報を交換、利用できる度合
使用性	効果的、効率的に利用できる度合	適切度認識性	ニーズに適した利用かどうか認識できる度合
		習得性	システムの使い方の学習ができる度合
		運用性	運用や管理のしやすさの度合
		ユーザエラー防止性	誤操作しないように保護する度合
		ユーザインタフェースの快美性	ユーザインタフェースが親しみがあり満足感のある応答ができる度合
		アクセシビリティ	幅広い層の特徴や能力を持つ人々が利用できる度合

出典：平成23年度 経済産業省 ソフトウェアメトリクス高度化プロジェクト
「情報システム/ソフトウェアの品質メトリクスセット 利用ガイド」
「株式会社三菱総合研究所」公開資料より抜粋

■ 機能面の品質モデル

品質特性	特性の概要	副品質特性	概要
機能適合性	実装された機能がニーズを満たす度合	完全性	ニーズを機能がユーザの目的やタスクを包含している度合
		正確性	必要な精度で正確な結果を与える度合
		適切性	機能が定められたタスクや目的を円滑に遂行する度合
性能効率性	システムの実行時の性能や資源効率の度合	時間率性	実行時のシステムの応答時間、処理時間などの処理能力の度合
		資源利用性	実行時に使用する資源量や種類
		キャパシティ	要求を満たすための製品やシステムのパラメータの最大許容値
互換性	他製品やシステムと機能や情報を共有、変換できる度合	共存性	他製品へ負の影響を与えず、共通の環境や資源を共有して効果的に実行する度合
		相互運用性	2つ以上の製品やコンポーネント間で情報を交換、利用できる度合
使用性	効果的、効率的に利用できる度合	適切度認識性	ニーズに適した利用かどうか認識できる度合
		習得性	システムの使い方の学習ができる度合
		運用性	運用や管理のしやすさの度合
		ユーザエラー防止性	誤操作しないように保護する度合
		ユーザインタフェースの快美性	ユーザインタフェースが親しみがあり満足感のある応答ができる
		アクセシビリティ	度合 幅広い層の特徴や能力を持つ人々が利用できる度合

信頼性	必要時に実行することができる度合	成熟性	通常時に信頼性のニーズを満たす度合
		可用性	必要時に運用、接続できる度合
		障害許容性	障害時に運用できる度合
		回復性	障害時にデータやシステムが回復したり再構築できる度合
セキュリティ	不正にアクセスがされることなく、情報やデータが保護される度合	機密保持性	許可された者のみがアクセスできるようデータを保証する度合
		インテグリティ	プログラムやデータへの変更において未許可なアクセスを防止する度合
		否認防止性	イベントやアクションの発生を証明する度合
		責任追跡性	エンティティの実行が唯一であることを証明する度合
		真正性	リソースや物事の身元が要求されたものであることを証明できる度合
保守性	効果的、効率的に保守や修正ができる度合	モジュール性	変更による他コンポーネントへの影響が最少で済むよう、独立したコンポーネントで構成される度合い
		再利用性	他のシステムや資産を構築する際に利用できる度合
		解析性	変更部分や障害原因の特定のために診断したり、変更による影響を評価する際の効果性、効率性の度合
		変更性	欠陥や品質の低下なく変更が効果的、効率的にできる度合
		試験性	テスト基準を確立し、評価するために実行する際の効果性、効率性の度合
移植性	効果的、効率的に他のハードウェアや実行環境に移植できる度合	順応性	別のもしくは進化したハードウェアやソフトウェアや他の運用環境に効果的、効率的に順応できる度合
		設置性	正しくインストール、もしくはアンインストールする際の効果性、効率性の度
		置換性	同一の目的、環境下で他のソフトウェア製品に置換(リプレース)できる度合

【キャリア実践】AE力強化研修
例）機能・非機能の品質モデル

■システム／ソフトウェア製品の標準品質モデル

品質特性	特性の概要	副品質特性	概要
信頼性	必要時に実行することができる度合	成熟性	通常時に信頼性のニーズを満たす度合
		可用性	必要時に運用、接続できる度合
		障害許容性	障害時に運用できる度合
		回復性	障害時にデータやシステムが回復したり再構築できる度合
セキュリティ	不正にアクセスがされることなく、情報やデータが保護される度合	機密保持性	許可された者のみがアクセスできるようデータを保証する度合
		インテグリティ	プログラムやデータへの変更において未許可なアクセスを防止する度合
		否認防止性	イベントやアクションの発生を証明する度合
		責任追跡性	エンティティの実行が唯一であることを証明する度合
		真正性	リソースや物事の身元が要求されたものであることを証明できる度合
保守性	効果的、効率的に保守や修正ができる度合	モジュール性	変更による他コンポーネントへの影響が最少で済むよう、独立したコンポーネントで構成される度合い
		再利用性	他のシステムや資産を構築する際に利用できる度合
		解析性	変更部分や障害原因の特定のために診断したり、変更による影響を評価する際の効果性、効率性の度合
		変更性	欠陥や品質の低下なく変更が効果的、効率的にできる度合
		試験性	テスト基準を確立し、評価するために実行する際の効果性、効率性の度合
移植性	効果的、効率的に他のハードウェアや実行環境に移植できる度合	順応性	別のもしくは進化したハードウェアやソフトウェアや他の運用環境に効果的、効率的に順応できる度合
		設置性	正しくインストール、もしくはアンインストールする際の効果性、効率性の度合
		置換性	同一の目的、環境下で他のソフトウェア製品に置換（リプレース）できる度合

出典：平成23年度 経済産業省 ソフトウェアメトリクス高度化プロジェクト
「情報システム/ソフトウェアの品質メトリクスセット 利用ガイド」
「株式会社三菱総合研究所」公開資料より抜粋

【キャリア実践】AE力強化研修
例）品質のトレードオフのイメージ

ワークシートの作り方として面白いのでイメージだけ。時間やお金の影響の方が大きいような気はしますが、、、

■品質特性間のトレードオフ(IPA経験則)

インテグリティ

情報が常に完全であることを保証する。許可されていない利用者によってデータを改ざんまたは破壊されるのを防ぐこと。

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
①可用性（Availability）								＋		＋	
②性能効率性（Efficency）			－		－	－	－	－		－	－
③柔軟性（Flexibility）		－		－		＋	＋	＋			
④インテグリティ（Integrity）		－			－				－		－
⑤相互運用性（Interoperablity）		－	＋	－			＋				
⑥保守性（Maintainability）	＋	－	＋					＋			
⑦移植性（Portability）		－	＋		＋				＋		－
⑧信頼性（Reliability）	＋	－	＋			＋				＋	＋
⑨再利用性（Reusability）		－	＋	－	＋	＋	＋	－			
⑩堅牢性（Robustness）	＋	－						＋			＋
⑪使用性（Usability）		－								＋	

（判例）＋　：対応する行の特性を上げると列の特性にプラスの影響を与える。
　　－　：対応する行の特性を上げると列の特性にマイナスの影響を与える。
　　空白：ほとんど影響を与えない。

出典：独立行政法人情報処理推進機構
要求工学・設計技術研究部会
非機能要求とアーキテクチャWG 2006 年度活動報告書