パターンワーキンググループセミナー

NFRのハナシ

非機能要求とアーキテクチャ

2008年8月6日

株式会社豆蔵　取締役フェロー
情報処理学会ソフトウェア工学研究会SIGSE主査・パターンWG主査
技術士（情報工学）　羽生田栄一

アーキテクチャとは

- もともとは建築の世界の用語
- 「システム」とその「アーキテクチャ」
  - そもそもの意味
    - そのシステムの構成要素とそれらの間の関係
  - ソフトウェア・アーキテクチャ
    - *ソフトウェア・コンポーネントとそれらの間の静的構造や動的関係*
- システムの階層構造
  - エコシステム＞経済システム＞・・・
    ・・・＞ある企業経営システム＞その中の情報システム
    ＞それを実現するソフトウェア群

アーキテクチャとは2

- そのシステムの設計のエッセンス
  - 詳細設計全体ではなく、しくみ・メカニズム
  - 機能とそれを実現するための構造と振る舞い
- そのシステムの時間軸でのライフサイクル全体
  - システムは運用の中で変更、拡張、保守されていくもの
  - 構造が理解しやすく、保守・移植しやすいこと
- 他のシステムとの関係性
  - プロダクトライン、シリーズ開発、企業内情報システム群
  - エンタープライズアーキテクチャ、全体最適
- そのシステムの背景にある考え方・思想
  - 誰にとっての何の目的のどんな機能を如何に提供するか
  - 設計コンセプト、フレームワーク、メタモデル

IEEE P1471 RPAD (Recommended Practice for Architectural Description)

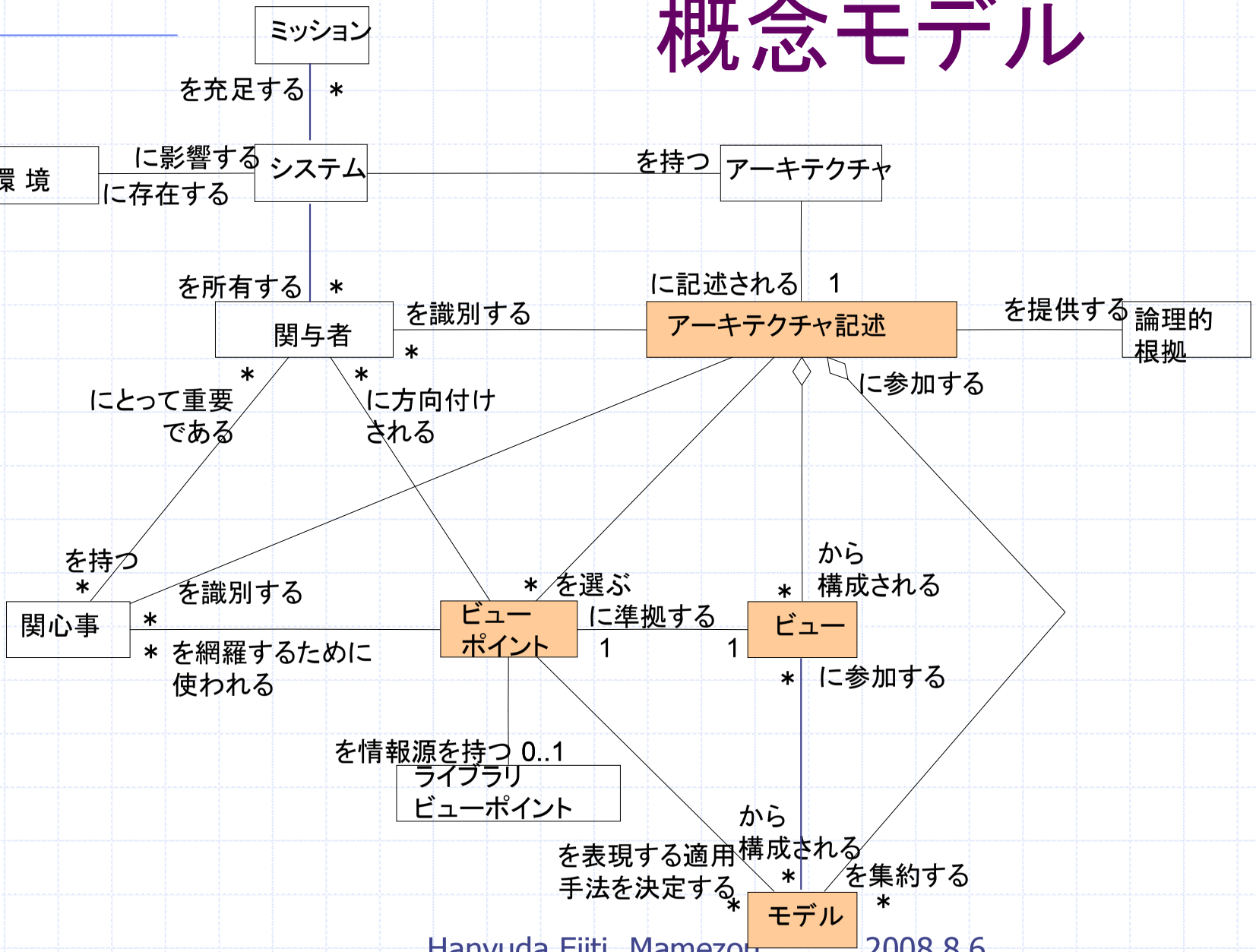
ソフトウェアアーキテクチャ記述に関する実用的な方式を推奨

アーキテクチャは、
- その環境におけるシステムの
- 最上位レベルの「概念」であり、
- 複数のビューポイントで定義される
  各ビューに基づく
- モデルによって表現される

アーキテクチャ IEEE P1471 RPAD

(Recommended Practice for Architectural Description)

概念モデル



豆蔵 mamezou

システム4+1ビュー

（注）オリジナル4＋1ビュー(Kruchten)に羽生田が5W1Hとの関係、縦・横の軸を追加.

静的構造

1．What（System）

論理ビュー

クラス，関連，属性，操作

（クラス図，シーケンス図）

アナリスト，アーキテクト

2．What-Who（Software）

コンポーネントビュー

コンポーネント，依存関係

インターフェース、開発チーム

（コンポーネント図）

開発者，構成管理者

+1．Why

ユースケースビュー

アクター，ユースケース

（ユースケース図，シナリオ）

動的構造

3．How（System）

プロセスビュー

コラボレーション

（コラボレーション図，
ステートチャート図）

デザイナ，アーキテクト

4．What@Where（Platform）

配置ビュー

ノード，ネットワーク

プロセス，コンポーネント

（配置図）

システムエンジニア，運用管理者

設計 ← → *開発*

Non-Functional Requirements (NFR)：非機能要求

- Non-functional requirements, as the name suggests, are those requirements which are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Alternatively, they may define constraints on the system such as the capabilities of I/O devices and the data representations used in system interfaces.

  (Software Engineering 6th Edition, Ian Sommerville, 2001)

- 非機能要求は，時として，制約条件(constraints)もしくは品質要求(quality requirements)として知られている。また，この要求はさらに以下のように分類できる。（例えば）性能要求，保守容易性要求，安全性要求，信頼性要求，電磁気互換性要求，その他の様々なタイプの要求である。

  （ソフトウェアエンジニアリング基礎知識体系 – SWEBOK, 2003）

ソフトウェアの品質（Bertrand Meyer）

- 工学の目的は品質
  - ソフトウェア工学の目的：品質の高いソフトウェアの生産
- 外的品質要因＜＝最終的に問題になる特性
  - ユーザが認識できるソフトウェア製品の性質
- 内的品質要因＜＝外的要因を達成する手段
  - ユーザが直接認識できないソフトウェアの性質

FURPS+ （HP Grady）

- FURPS+
  - UP（Unified Process）において、要求を定義するための指針として用いられている
- 以下の要件の頭文字＋α
  - Functionality（機能性）
  - Usability（使用性）
  - Reliability（信頼性）
  - Performance（性能）
  - Supportability（保守性）

  } 非機能要求
- ＋α（プロセス制約）

良いアーキテクチャの条件

ソフトウェア品質特性･品質副特性ISO/IEC 9126(JISX0129)

- 機能性（**functionality**）
  - ･合目的性（**suitability**）
  - ･正確性（**accuracy**）
  - ･相互運用性（**interoperability**）
  - ･標準適合性（**compliance**）
  - ･セキュリティ（**security**）
- 信頼性（**reliability**）
  - ･成熟性（**maturity**）
  - ･障害許容性（**fault tolerance**）
  - ･回復性（**recoverability**）
- 使用性（**usability**）
  - ･理解性（**understandability**）
  - ･習得性（**learnability**）
  - ･運用性（**operability**）
- 効率性（**efficiency**）
  - ･時間効率性（**time behavior**）
  - ･資源効率性（**resource behavior**）
- 保守性（**maintainability**）
  - ･解析性（**analyzability**）
  - ･変更性（**changeability**）
  - ･安定性（**stability**）
  - ･試験性（**testability**）
- 移植性（**portability**）
  - ･環境適応性（**adaptability**）
  - ･設置性（**installability**）
  - ･規格適合性（**conformance**）
  - ･置換性（**replaceability**）

非機能要求の定義はなぜ困難か

- 利害関心性（主観的）
  - 人・組織や利害・関心によって解釈・評価が異なる
- 価値相対性（相対的）
  - 何が重要か・緊急か等は，対象システムの目的や特性によって異なる
- 相互依存性（相互干渉的）
  - あるNFRのプラスは、他のNFRのマイナスの可能性

非機能要求の定義はなぜ困難か2

- 要求定義自身がそもそもむずかしい
- 機能と違って明確に定義しづらい
- 正誤ではなく善し悪しや好き嫌いの側面がある
- 当たり前品質だとして要求に挙がらない
- ユーザーや顧客は業務機能にまず目が行く
- 定性的な側面が強く、定量化しづらい
- システムやアーキテクチャのことを知らないと、思い付いたり指定したりできない部分がある

➔ *システムのゴール(目的や方針)との相対的な関係で判断すべきものである！*

非機能要求の分類

主にユーザーに重要	主に開発者に重要
可用性（Availability）	保守性（Maintainability）
効率性（Efficiency）	可搬性、移植性（Portability）
柔軟性（Flexibility）	再利用性（Reusability）
完全性（Integrity）	試験性（Testability）
相互接続性（Interoperability）	
信頼性（Reliability）	
堅牢性（Robustness）	
使用性（Usability）	

NFR の相互干渉（トレードオフ）の例

	可用性 (Availability)	効率性 (Efficiency)	柔軟性 (Flexibility)	整合性 (Integrity)	相互運用性 (Inter-Operability)	保守性 (Maintain-ability)
可用性 (Availability)						
効率性 (Efficiency)			−		−	−
柔軟性 (Flexibility)		−		−		+
整合性 (Integrity)		−			−	
相互運用性 (Inter-operability)		−	+	−		
保守性 (Maintain-ability)	+	−	+			

悪影響

好影響

出典: “Software Requirements”, Karl E. Wiegers（表の一部を抜粋）

NFRトレードオフ・マトリクス

『ソフトウェア要求』日経BPソフトプレス

	可用性	効率性	柔軟性	完全性	相互接続性	保守性	移植性	信頼性	再利用性	堅牢性	試験性	使用性
可用性								+		+		
効率性			−		−	−	−	−		−	−	−
柔軟性		−		−		+	+	+			+	
完全性		−			−				−		−	−
相互接続性		−	+				+					
保守性	+	−	+					+			+	
移植性		−	+		+				+		+	−
信頼性	+	−	+			+				+	+	+
再利用性		−	+	−	+	+	+	−			+	
堅牢性		−										+
試験性	+	−	+			+		+				+
使用性		−								+	−	

ソフトウェアにおけるフォース

- 設計空間に存在するフォース（設計上の要因）
  - コンサーン、ゴール、制約、トレードオフ、動機
  - よい設計は複数の**競合するフォース**を*うまくバランスさせる*
- フォースの例:
  - メモリの使用は最小に抑えつつ、性能は最大化せよ
  - 設計はできるだけシンプルに、一方できるだけ柔軟性は確保して欲しい
  - プログラムは既存コンポーネントと相互運用性があること
  - プログラムはリモートのサーバーと非同期に通信できること
  - プログラムは決して事故で停止したりしないこと
  - サーバーは1000クライアントの同時接続を可能にすること
  - 追加プラグインがネット経由でスムーズにダウンロード可能
  - 早い市場投入と安い初期コスト　vs　充実した機能と長期に渡る高品質

IPA/SEC要求・設計開発技術に関する部会活動

詳しくは、IPA「非機能要求記述ガイド」の公開 http://sec.ipa.go.jp/reports/20080717.html

【部会体制】

要求・設計開発技術研究部会

主査:　青山先生（南山大学）
副主査：榊原氏（IBM），羽生田氏（豆蔵）

非機能要求とアーキテクチャWG

リーダ:　榊原氏（IBM）
副リーダ：羽生田氏（豆蔵）

【活動の狙い】

- ソフトウェア開発において最初に行う要求獲得や要求定義の確実な実施と得られた要求を実現する開発技術の洗練と普及

調査期	成果作成期		成果普及期	
第0期	第1期	第2期	第3期	第4期
・産業の実態調査（シーズ・ニーズ調査） ・SECで扱う領域・テーマの集約・選定	・技術調査、成果の高度化・体系化		・企業における実証実験及び結果のフィードバック	・支援ツールの開発・提供 ・成果の本格的導入・普及

- 非機能要求をテーマに選定（2006年3月）
- 技術報告書（2007年9月初め公開）
- 記述ガイド第1版（2008年3月末公開予定）

非機能要求とアーキテクチャWG委員

項番	メンバー名		所属
1	榊原　彰	リーダ	日本アイ・ビー・エム株式会社
2	羽生田　栄一	副リーダ	株式会社豆蔵
3	青木　奈央		キャツツ株式会社
4	青山　幹雄		南山大学
5	粟津　正輝		株式会社富士通エフサス
6	石谷　靖		株式会社三菱総合研究所
7	市原　尚久		株式会社NTTデータ
8	牛島　郁子		株式会社アルゴ21
9	大嶽　隆児		日本アイ・ビー・エム・システムズ・エンジニアリング株式会社
10	岸上　信彦		日本電気株式会社
11	酒匂　寛		有限会社デザイナーズデン
12	鈴木　三紀夫		TIS株式会社
13	瀬尾　明志		日本ユニシス株式会社
14	関根　智		東芝ソリューション株式会社
15	辻　孝夫		日本電気株式会社
16	富田　哲夫		株式会社豆蔵
17	中谷　多哉子		筑波大学大学院
18	三上　理		日本電気株式会社
19	吉田　尚志		株式会社NTTデータ
20	鷲崎　弘宜		国立情報学研究所
21	塚本　英昭		IPA SEC

(2007年11月6日現在)

非機能要求とアーキテクチャWGの目的

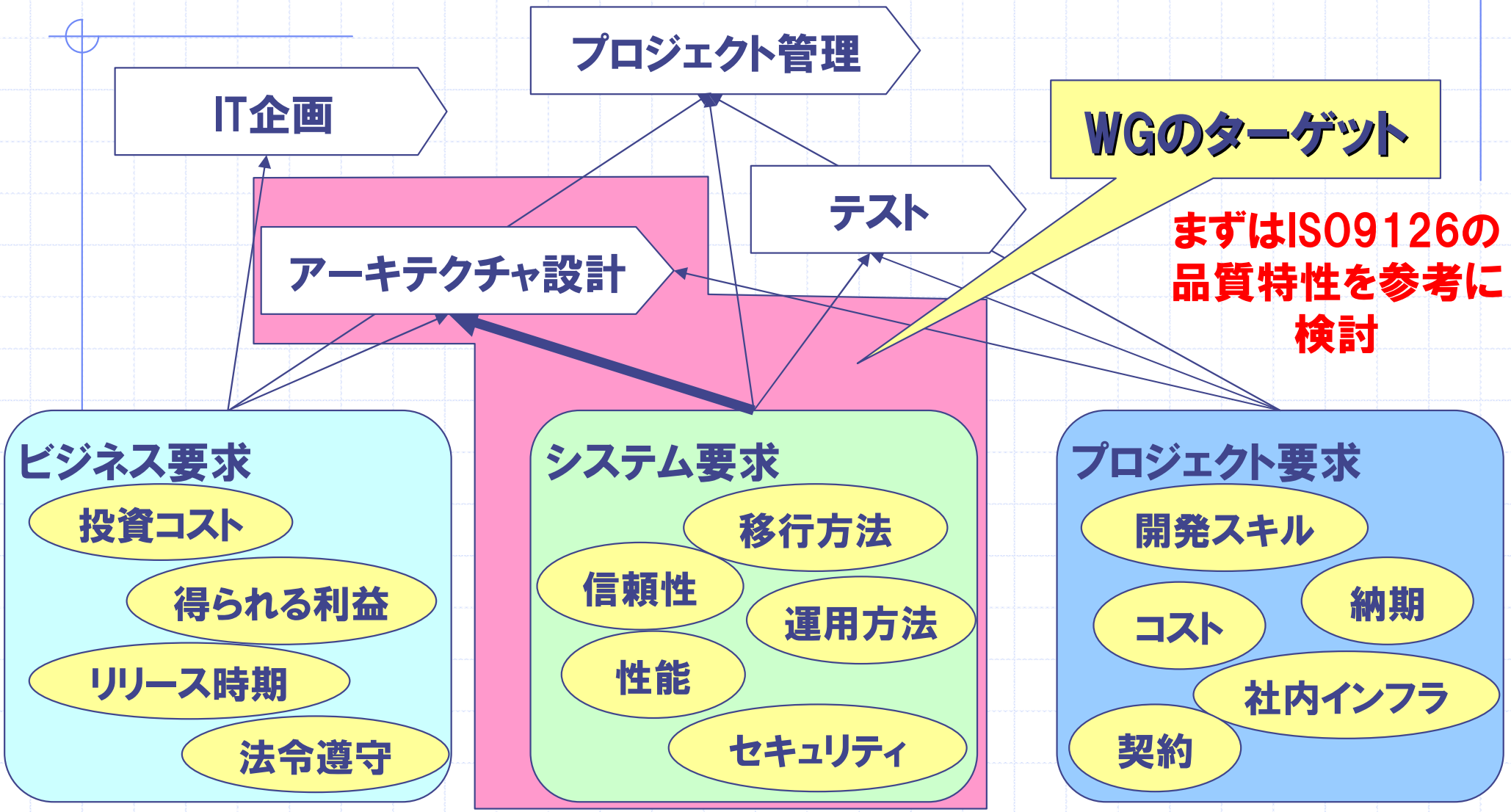
（1）非機能要求を正確に要求仕様書に書くこと

（2）非機能要求を記述するにあたって明確化できる部分と不明瞭なままの部分が切り分けられること

（3）非機能要求の検討にあたって利用できる参照モデルがあること

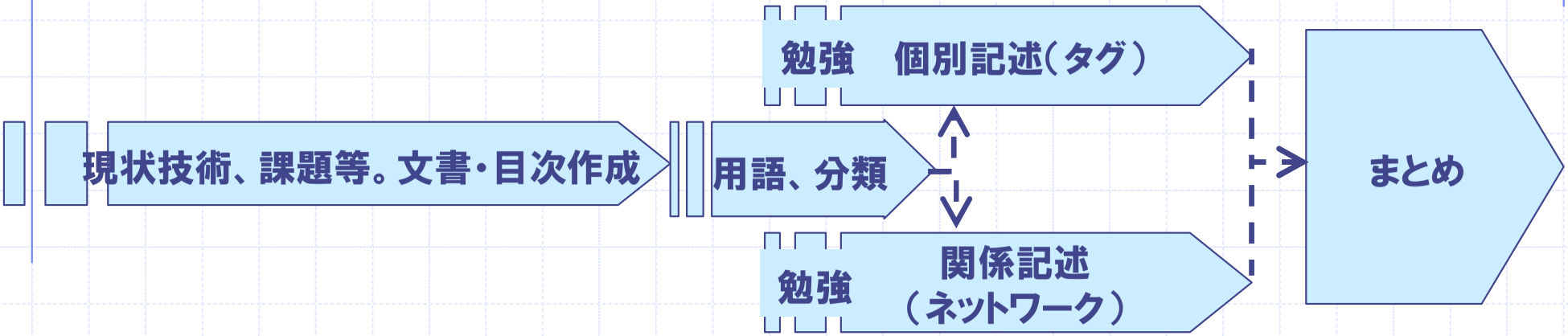
（4）記述する内容の網羅性を判断しやすくすること

非機能要求検討の対象範囲



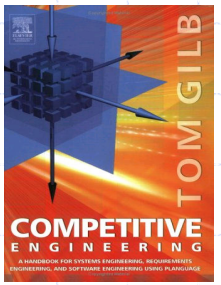
活動の進め方

ステップ①	ステップ②	ステップ③	ステップ④
ゴール設定、対象とする問題についての議論		まとめ成果物への記述内容や進め方の意識あわせ	品質特性の記述方法の議論・検討(グループ化)
記述やNFRに関するテーマの勉強(各委員発表)			



個別記述(タグ)

- Planguage(要求，設計およびプロジェクト計画を記述するための仕様言語)をベースに検討



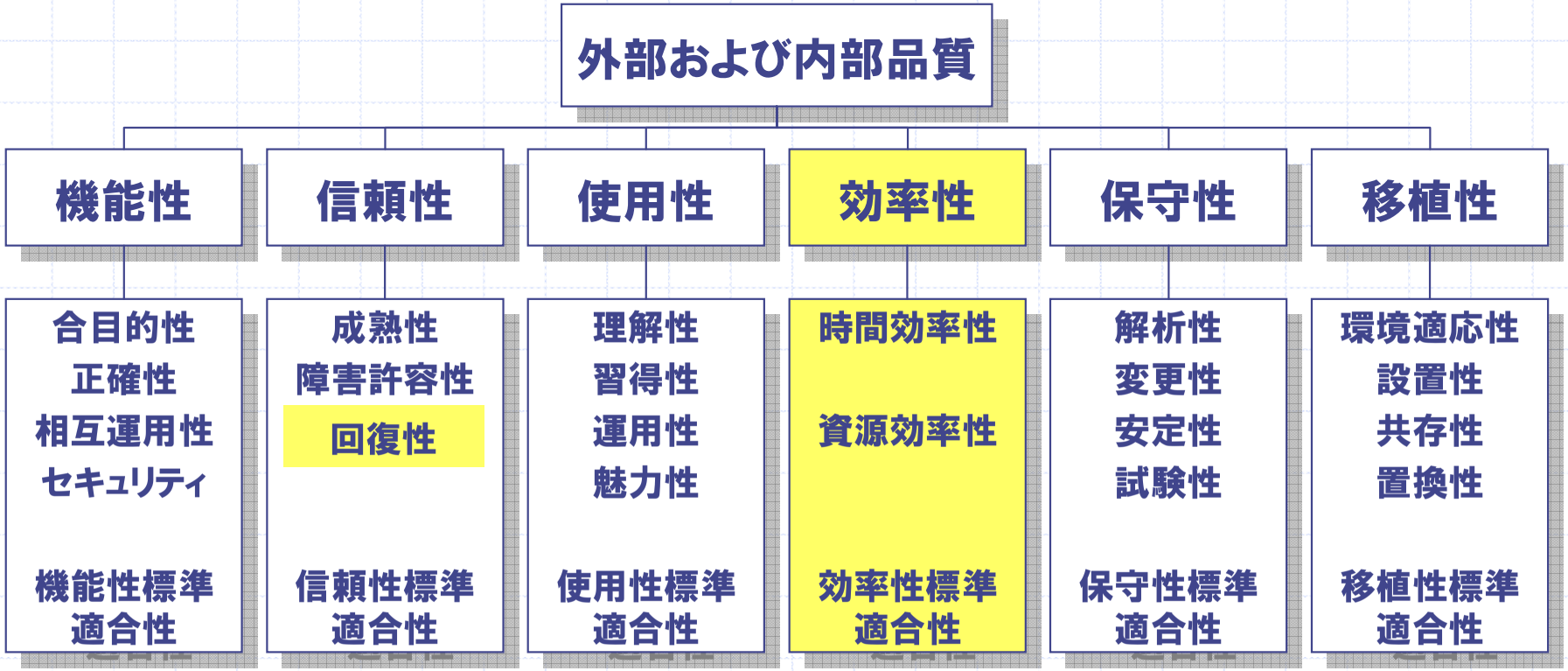
関係記述(ネットワーク)

- NFRフレームワーク(NFRを明示的にソフトゴールとして表現する技法)をアーキテクチャ設計につながる実現手段も含めて検討

非機能要件対象範囲（2007年度）

まずはISO9126の品質特性を参考に検討

2007年度は品質特性を“回復性”（信頼性の副品質特性）と“効率性”（時間効率性と資源効率性）に絞込み



出典：JIS X 0129-1: 2003（ISO/IEC 9126-1: 2001）

目指している「非機能要件記述」のイメージ

個別の品質要件記述仕様

コントロールケース品質要件記述

コントロールケース品質要件記述

```
。。。。。
<Reliability>
   <Recoverability>
      <Scope>SeatReservation
      <Scope/>
      <MTTR>less than 30min
      <MTTR/>
      <Recoverability/>
<Reliability/>
。。。。。。。。
```

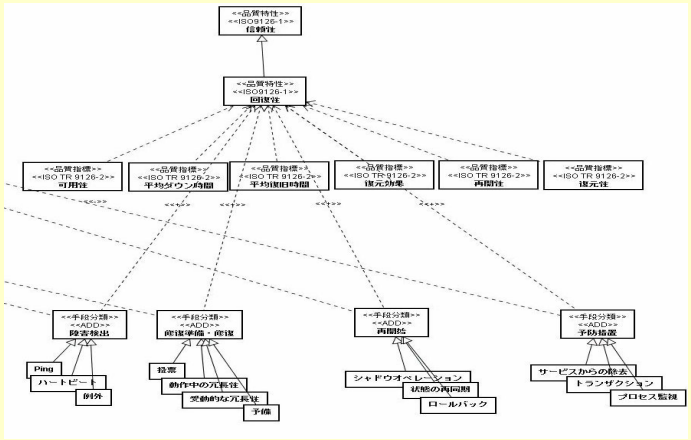
注意:中のキーワード(タグ)はWGで検討中。

ユースケースとコントロールケースの関係を明確化

個別非機能要件間の関連性および依存性のチェック

品質要件の関係性記述

クラス図形式



表形式

実現手段		ソフトゴール	効率性				信頼性		
			時間効率性		資源効率性		回復性		
			TAT、応答時間(を減らす)	スループット	利用量(を減らす)	利用率?	RPO 復旧時点指標	RTO 復旧時間指標	RLO 復旧状態指標
	負荷量削減	スケジューリングによる負荷分散、ユーザ制限	+	+	+				
	性能チューニング		+	+	+				
	保守・点検	より詳細な記述が必要?					+	+	+
ソフトウェア	圧縮		-	-	+				
	索引		+	+	-				
	バックアップ		-	-	-		+	+	+
	監視		-	-	-		+	+	+
	再起動								
	ミドルウェア適用		-	-	-				
	リトライ						?	?	?
	故障診断		-	-	-		?	?	?
ハードウェア	並列処理	マルチプロセッサ、マルチコアプロセッサ	+	+					
	分散処理	クラスタコンピューティング、ロードバランサ	+	+					
	待機予備						+	+	+
	多重決						+	+	+
	ディスクアレイ	RAID(ミラーリング、パリ	+	+	-		+		+
		RAID(ストライピング)	+	+	-		-	-	-

豆蔵 mamezou

個別記述イメージ

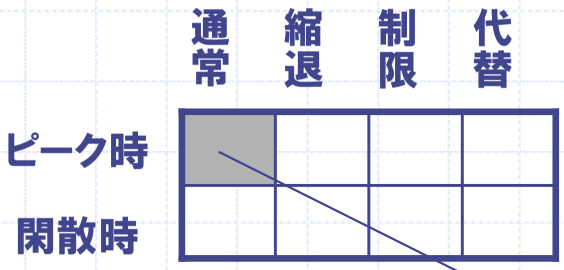
コントロールケース「レスポンスタイム」

ControlCase id:CC-001
Operating Condition: 複数ユーザが同時アクセスした場合の負荷
Description: エンドユーザまで届かないシステムのレスポンスタイムはビジネス顧客を失うリスク
をもたらす。コントロールケースはこのようなリスクを回避するトランザクションに対す
るレスポンスタイムの最大値を定義する。
Constraint: それぞれのトランザクションに対して証跡を取得・保存し、また取り出せること
Associated Use Cases: UC-001, UC-002
Impact (if not met) : 20%の顧客を失う
FirstMode: 通常
SecondMode: ピーク時
TAT
Response: 90%のユーザに対するレスポンスタイムは5秒以内。
95%のユーザに対するレスポンスタイムは10秒以内。
ThinkTime: ….
・・・

個別記述の基本的考え

- コントロールケース(Control Case)
  - システムが達成しなければならない品質を記述
  - 稼動条件(Operating Condition)、ステークホルダと共有すべきリスク記述(Description)、稼動モード(Mode of Operation)から構成

稼動モード(多次元、デフォルトは2次元)



コントロールケーステンプレート

```
ControlCase id:
  Operating Condition:
  Constraint:
  Description: 稼動条件によって発生するリスクとそのリスクを減らすために定義すべきコントロールケースの説明
  (Associated Use Cases:) ※この場合は、関連するUSE CASEとの紐つけが必要。
  Impact (if not met) : 要求が達成されないときの影響を記述
  FirstMode (列挙型:通常/縮退/制限/代替) ※追加も自由
     SecondMode (列挙型:ピーク時/閑散時)
        TAT
          Response
          ThinkTime
  ...
```

コントロールケースとは

- システムの非機能的側面にかかわる「リスク」を書き下したもの
  - 例えば、「システム性能が不十分だと、利用者の要求の処理が遅れ、顧客を失ってビジネスに損害を与える恐れがある」
  - 詳細コントロールケースでは、記述を詳細化
    - 想定される損害を回避(あるいは軽減)するためにシステムとして何をすべきか個々の状況を規定した
- リスクの3要素を明記：

①外部環境条件の内の何が変化（悪化）するのか。

- これは”Operating Condition：”に記述する。

②障害・損害の具体的な内容.

(a) Quality of Service（QoS）の面でシステムにどのような障害(不都合) が生じ、

(b) その結果としてビジネスの面でどのような損害が生じるか。

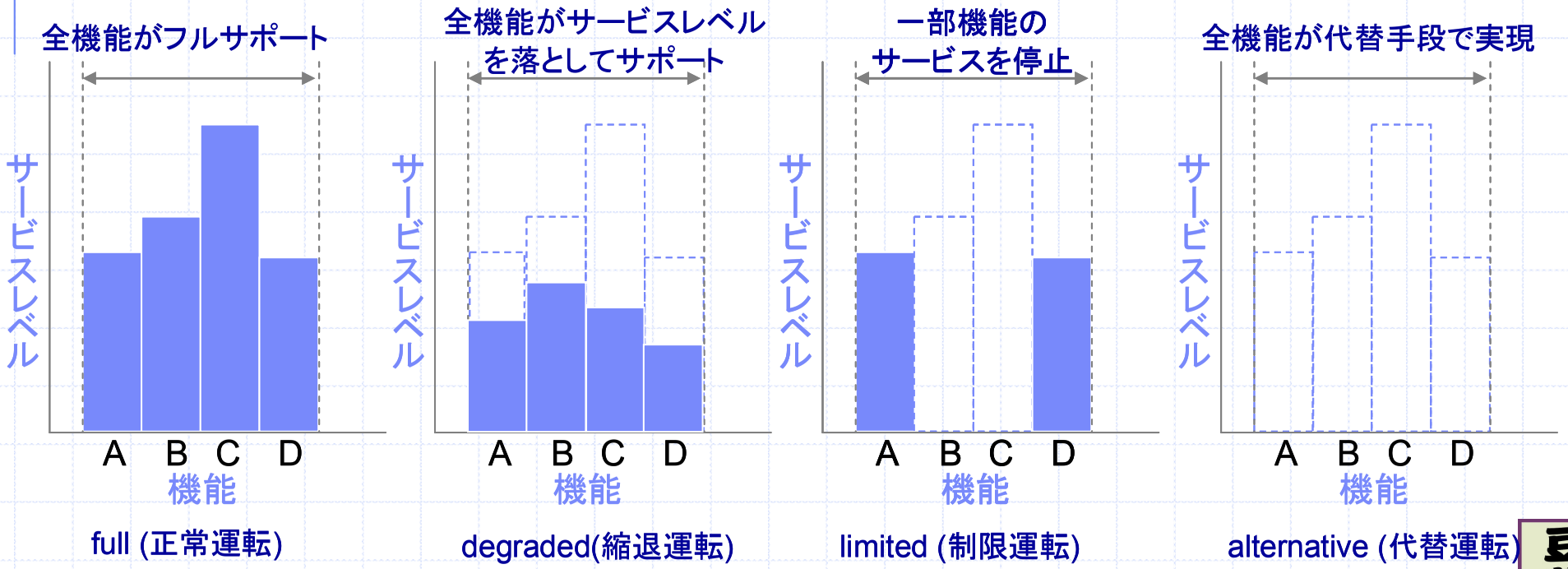
- これらを”Description：”に記述する。

③上記②(a) のQoS の種別

- これは”NFR Category”に記述する。

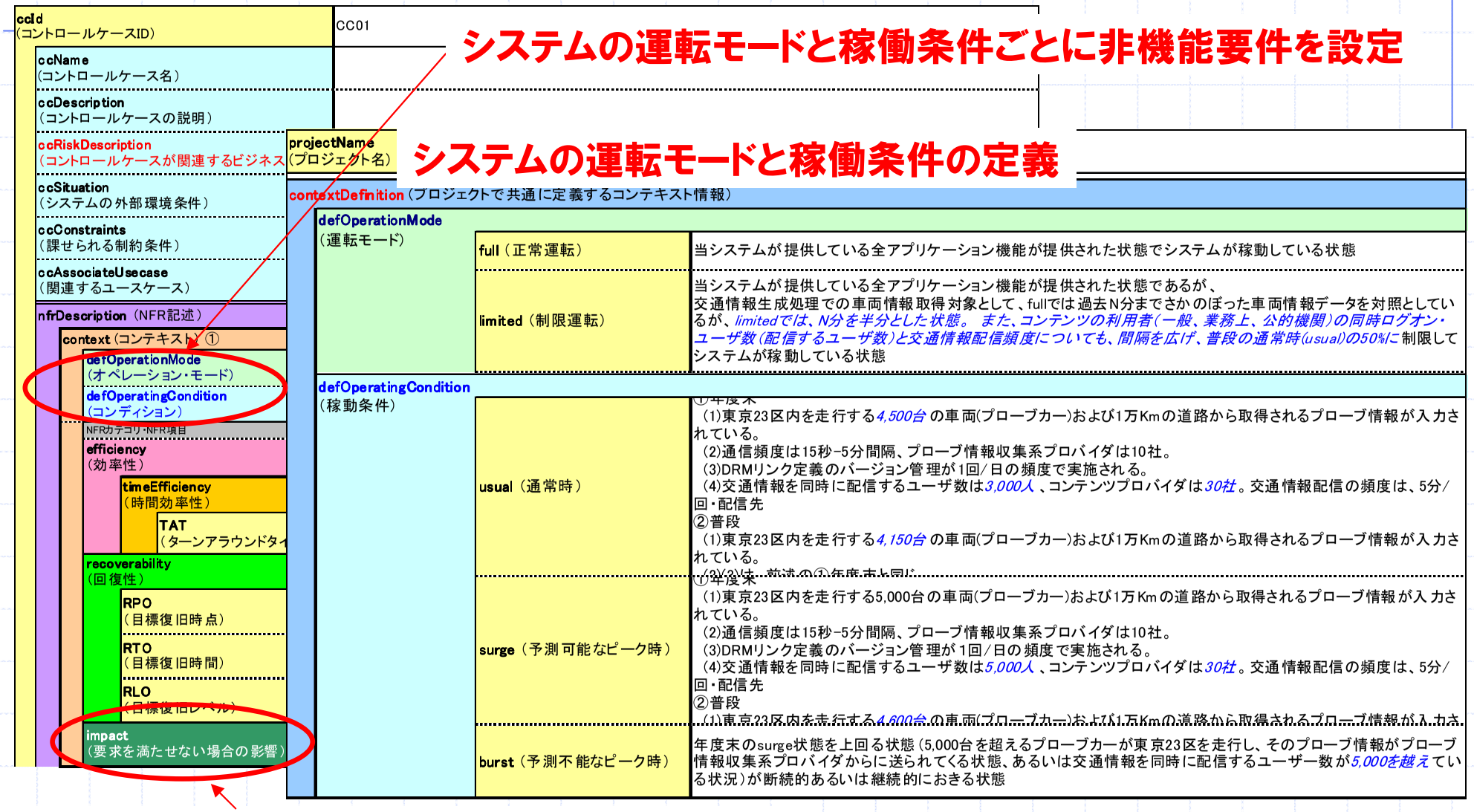
運転モード(defOperationMode)の分類

- **full(正常運転)**は、全機能がフルサービスで提供している状況
- **degraded(縮退運転)**は、全機能がサービスレベル(処理能力)を下げてサービスを提供している状況
- **limited(制限運転)**は、一部機能のサービスを停止している状態
- **alternative(代替運転)**は、全機能を代替手段で実現している状況



非機能要件記述のフォーマット

新しい非機能要件記述フォーマット



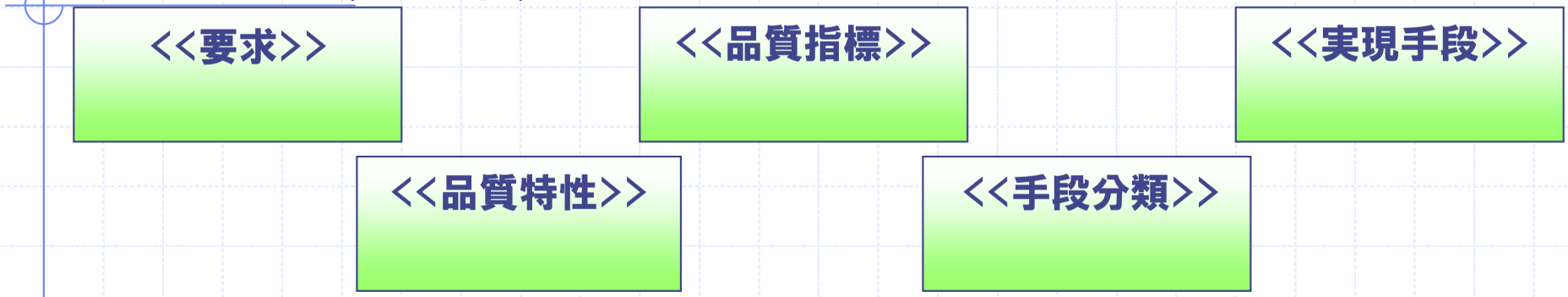
システムの運転モードと稼働条件ごとに非機能要件を設定

システムの運転モードと稼働条件の定義

非機能要件を満足しない場合のビジネス影響度

関係記述の基本モデル要素：
要求、品質特性、品質指標、手段分類、実現手段

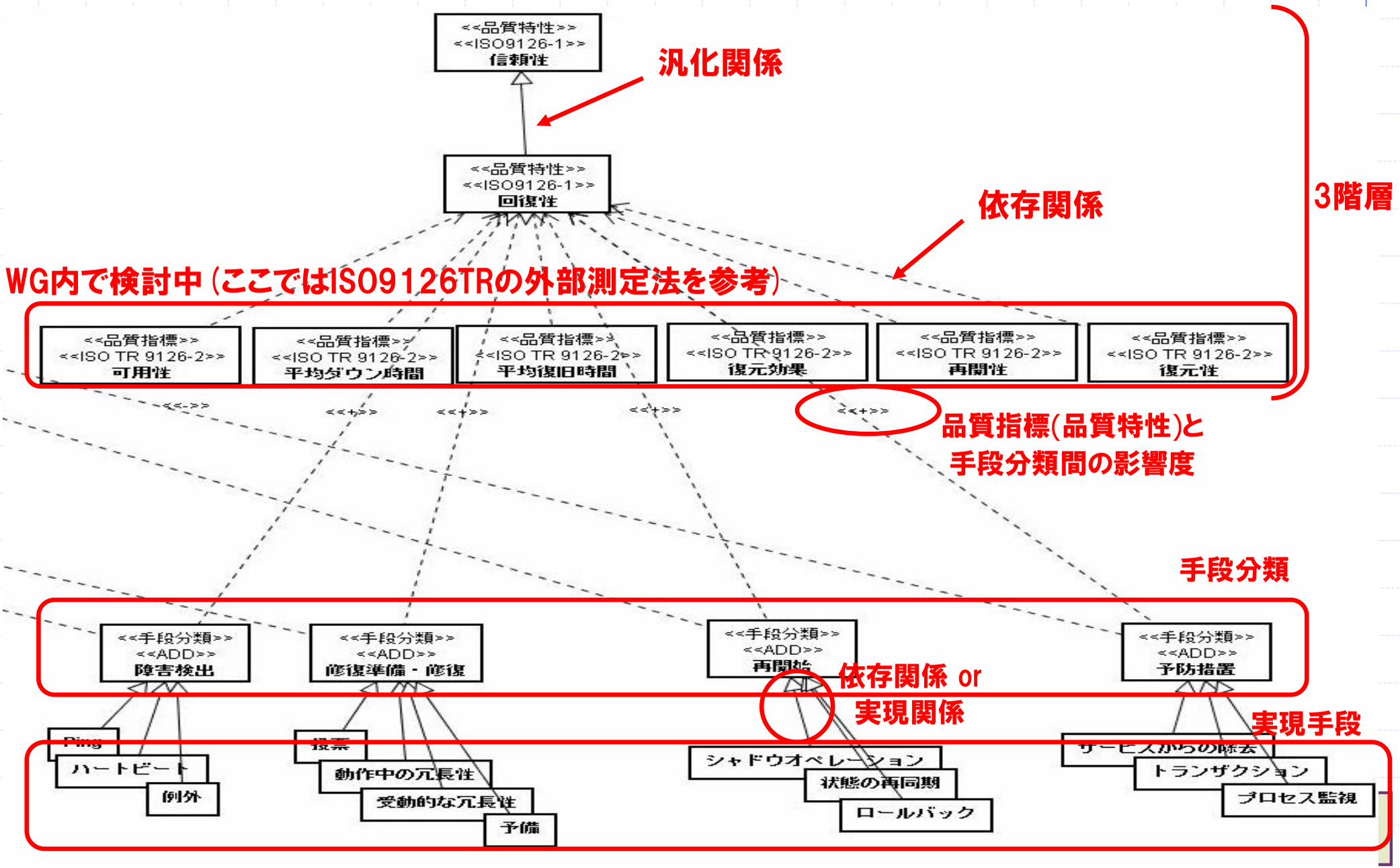
- 5つのステレオタイプ



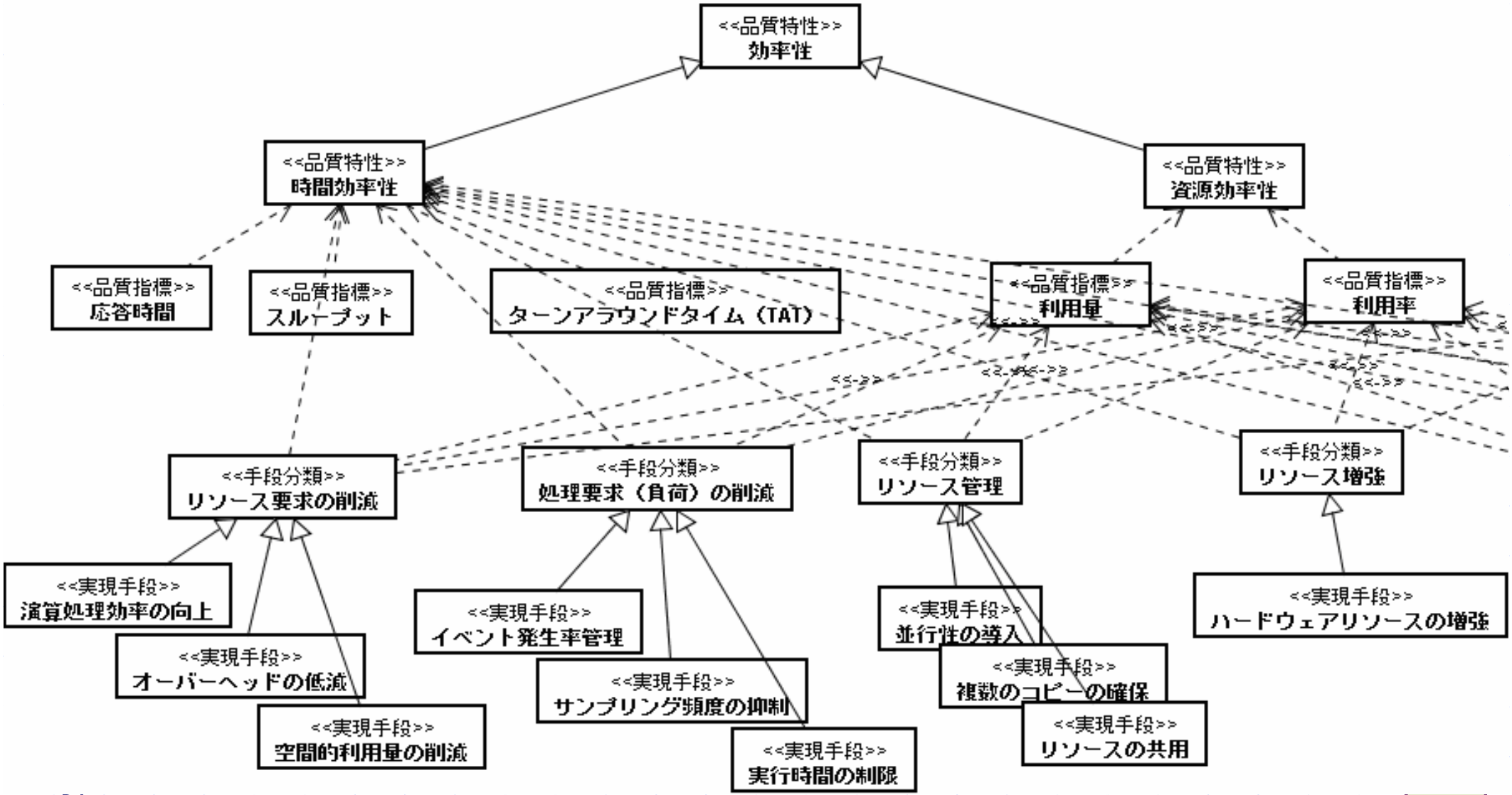
- サンプル

<<品質特性>> 効率性	<<品質指標>> スループット	<<手段分類>> 処理の並列化	<<実現手段>> ストライピング
<<品質特性>> 時間効率性	<<品質指標>> RTO (Recovery Time Objective)	<<手段分類>> 資源の共用	<<実現手段>> ストレージ仮想化

クラス図形式のNFR関係性表現



効率性に関する品質特性・品質指標と実現手段の関係記述



表形式のNFR関係性表現

クラス図形式との整合

<<手段分類>>	<<実現手段>>	具体例	<<品質特性>>	効率性				信頼性		
			<<品質特性>>	時間効率性		資源効率性		回復性		
			<<品質指標>>	TAT、応答時間（を減らす）	スループット	利用量(を減らす)	投資対効果（利用率?）	RTO 復旧時間指標	RLO 復旧状態指標	RPO 復旧時点指標
リソース要求	演算処理効率の向上	アルゴリズム改良、データ構造改良、データの圧縮								
	オーバーヘッドの低減	通信路使用の回避								
	イベント発生率管理	利用者スケジューリング、同時利用ユーザの制限								
	サンプリング頻度の抑制	定期起動プロセスの頻度を制限								
	実行時間の制限	反復回数の制限、一定時間後の処理打ち切り								
	キューサイズの制限									
リソース管理	並行性の導入	マルチタスク処理、マルチスレッド処理								
	複数のコピーを確保	データのキャッシング、クライアント/サーバ構成								
	利用可能なリソースを増強	CPUスペック、ネットワーク帯域、メモリ容量、ディスク容量、サーバ並列化、ストライピング(RAID)								
リソースの調停	スケジューリング方針	FIFO、固定優先順位、動的優先順位								
障害検出	生存確認	ピン(Ping)/エコー、ハートビート								
	例外									

影響度（+,−）の記述

具体例を記述

依存性確認表

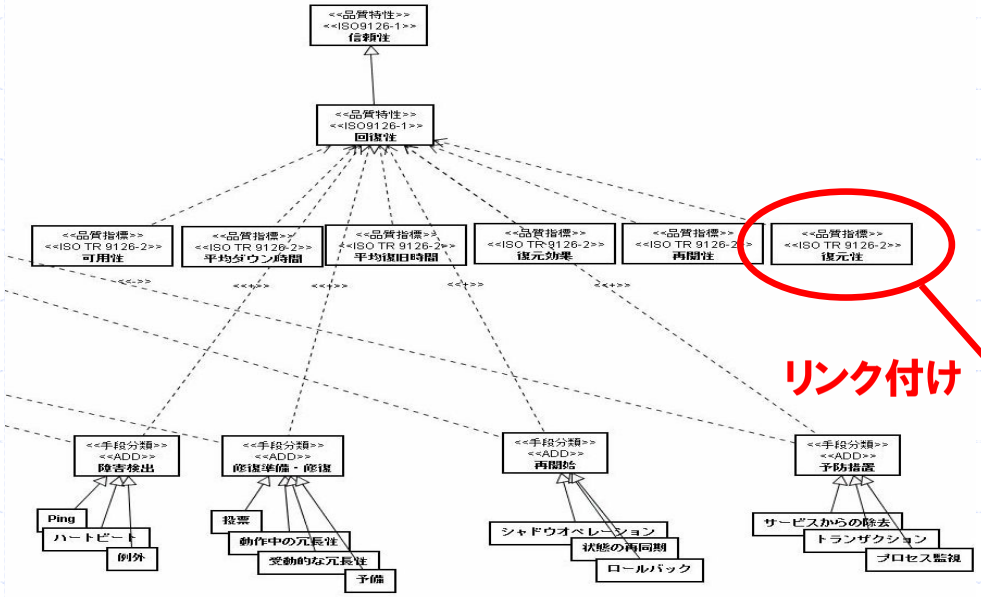
◆表形式のNFR関係性表現

手段＼品質			品質特性	効率性				信頼性		
			品質副特性	時間効率性		資源効率性		回復性		
			品質指標	TAT 応答時間	スループット	利用量	利用率	RTO 復旧時間 指標	RLO 復旧状態 指標	RPO 復旧時点 指標
手段分類	実現手段	具体例								
リソース要求の削減	演算処理効率の向上	アルゴリズム改良、データ構造改良、インデックス付与		＋	＋	＋	－	＋		
	オーバーヘッドの低減	通信路使用の回避、ディクス上のデータを主メモリ上にキャッシング		＋	＋			＋		
	空間的利用量の削減	データの圧縮				＋	－			
処理要求(負荷)の削減	イベント発生率管理	利用者スケジューリング、同時利用ユーザの制限		＋		＋	－			
	サンプリング頻度の抑制	定期起動プロセスの頻度を制限		＋		＋	－			
	実行時間の制限	反復回数の制限、一定時間後の処理打ち切り、通信におけるタイムアウト設定				＋	－			
	キューサイズの制限	プロセス間メッセージキューの最大記憶領域指定パラメータを調整				＋	－			
リソース管理	並行性の導入	マルチタスク処理、マルチスレッド処理		＋	＋	－	＋	＋		
	複数のコピーの確保	データのキャッシング、クライアント/サーバ構成におけるクライアント		＋	＋	－	＋	－	－	－
	リソースの共用	サーバの仮想化、ストレージの仮想化		＋	＋	－	＋			
リソース増強	ハードウェアリソースの増強	CPUスペック、ネットワーク帯域、メモリ容量、ディスク容量、サーバ並列化、ストライピング(RAID)		＋	＋		－	＋	＋	＋
障害検出	生存確認	HTTP,SQLなどによるピン(Ping)/エコー、ハートビート		－	－	－	＋	＋		
	例外捕捉	HDDドライブのハードウェア障害検出から例外を発生させ、原因を特定できるログなどを残す		－	－	－	＋	＋		
修復準備・修復	ホットスタンバイ	ロードバランサを介したAPサーバクラスタ。各サーバは常時稼動して内部状態を同期させる		－	－	－	＋	＋＋＋	＋	
	コールドスタンバイ	共有ディスクに接続されたDBサーバクラスタ。障害発生時にデータ同期を実施して待機系を起動する。		－	－	－	＋	＋＋	＋＋	
	バックアップ	通常は予備サーバを他のサービスに活用し、常用サーバに障害が発生したときに代替サーバとして運用する					－	＋	＋＋	
再開始	状態の再同期	データ同期、DBにおけるデータレプリケーション		－	－	－	＋	－		＋
	ロールバック	整合性が担保されたチェックポイントまで遡った状態の復元、トランザクション記録からメッセージキュー内の状態を復元		－	－	－	＋	－		＋
予防措置	サービスからの除去	サーバの定期点検、定期的なHDD交換、メモリリークを防止するための定期的なサーバ、プロセスの再起動				＋				
	トランザクション	障害時のデータへの影響範囲を限定し、スレッド間のリソース排他を実現するため処理をアトミックに実行		－	－	－	＋	－		＋
	監視	温度監視、プロセス監視、メモリ・ディスクの残量監視		－	－	－	＋			

個別記述と関係記述統合のポイント

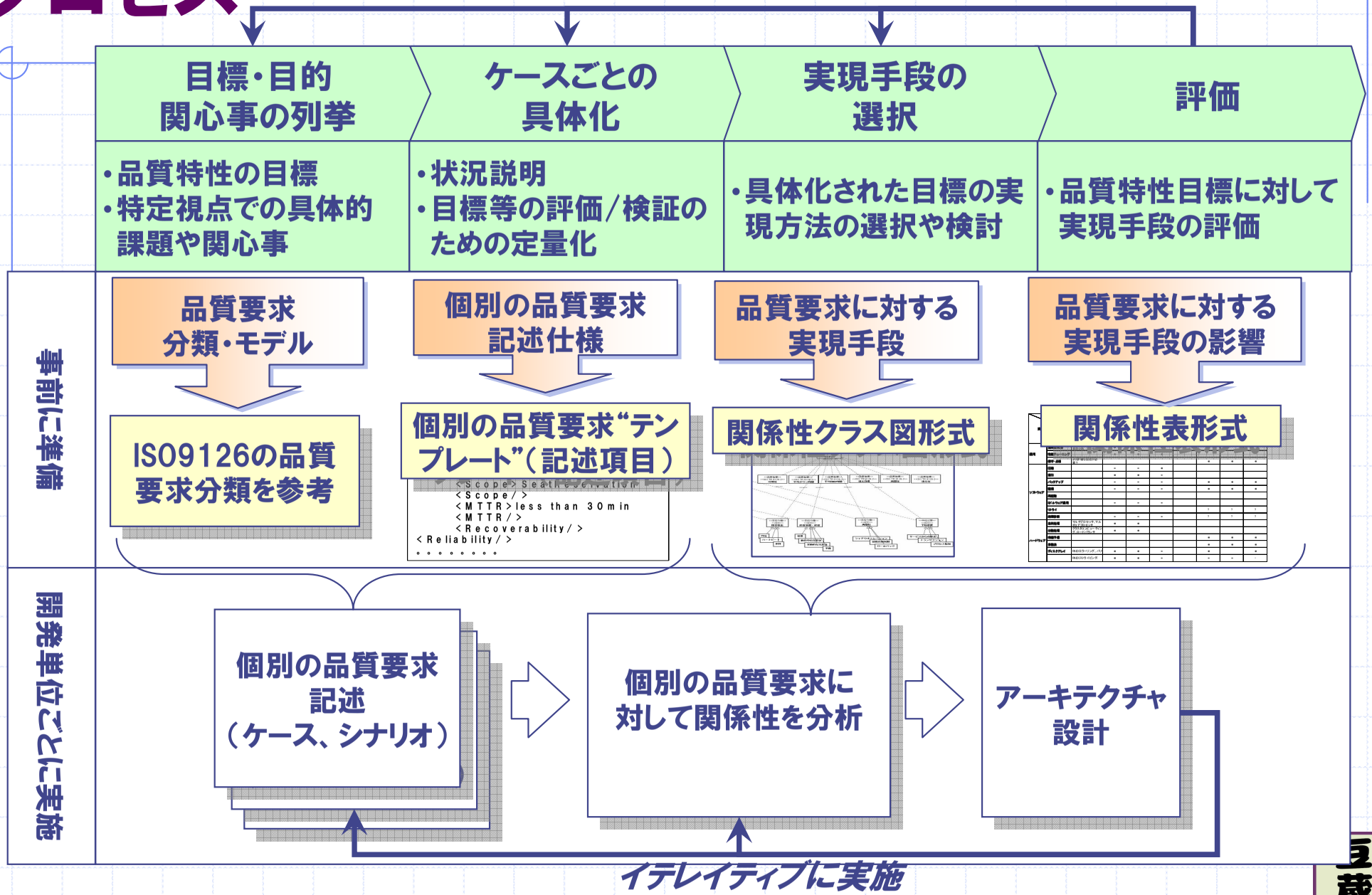
個別記述と関係記述のマッピング

- 関係記述のステレオタイプ”品質指標”のクラス名と個別記述のタグ名を対応



ControlCase id:
Operating Condition:
Constraint:
Description: 稼動条件によって発生するリスクとそのリスクを減らすために定義すべきコントロールケースの説明
(Associated Use Cases:) ※この場合は、関連するUSE CASEとの紐つけが必要。
Impact (if not met) : 要求が達成されないときの影響を記述
FirstMode (列挙型:通常/縮退/制限/代替) ※追加も自由
SecondMode (列挙型:ピーク時/閑散時)
TAT
Response
ThinkTime
・・・

非機能要求をアーキテクチャ設計にしていくプロセス



要求開発宣言（エッセンスを要約）

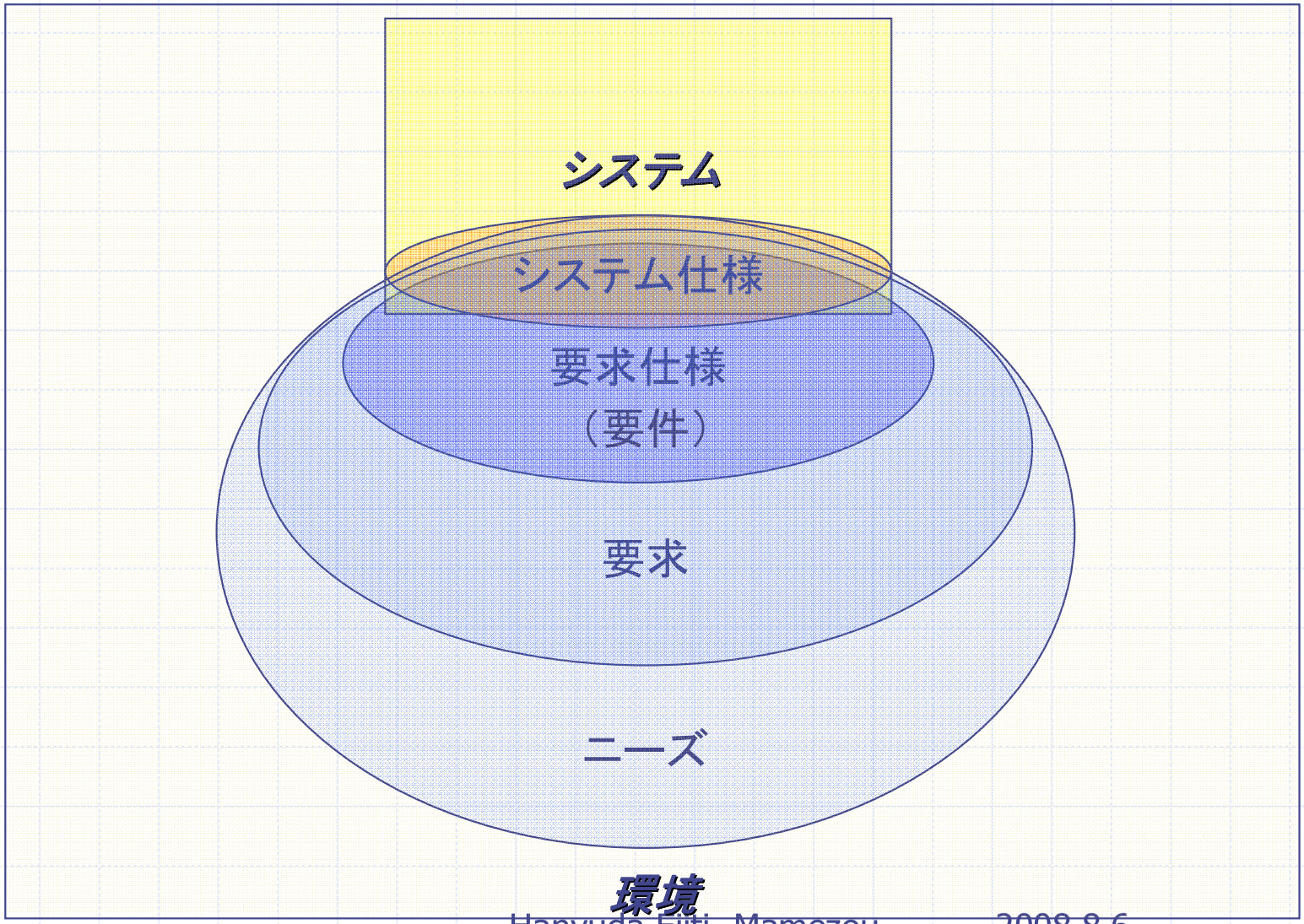
http://www.openthology.org/

要求はあるのではない。全関係者で開発するもの

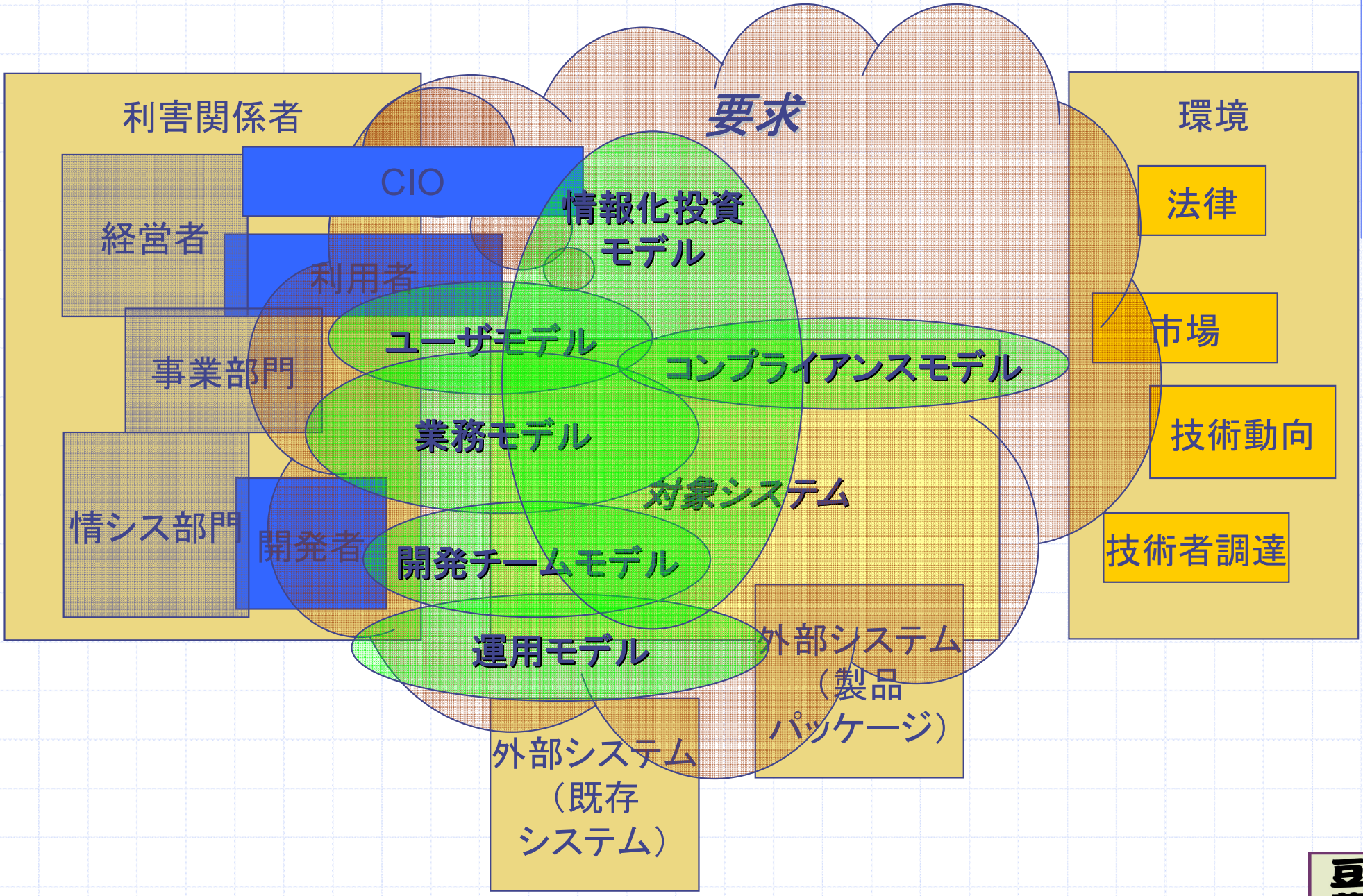
1. 要求はビジネスから開発するべき
2. システムは組織の価値を高めるべき
3. システム価値はビジネス価値へのトレーサビリティで判断
4. システム要求はステークホルダー間の合意形成プロセスの結果
5. システム開発は、参加協調にもとづく継続的改善プロセス
6. ビジネスや開発の「見える化」が上記目標の達成に不可欠

豆蔵 mamezou

要求のスペクトル



要求の環境駆動モデル


利害関係者
CIO
経営者
利用者
事業部門
情シス部門
開発者
要求
情報化投資モデル
ユーザモデル
コンプライアンスモデル
業務モデル
対象システム
開発チームモデル
運用モデル
外部システム（製品パッケージ）
外部システム（既存システム）
環境
法律
市場
技術動向
技術者調達