

グロースエクスパートナーズ(株)  
ビジネスプラットフォーム事業ゼネラルマネージャー  
チーフITアーキテクト  
日本Javaユーザー会幹事  
日本Springframeworkユーザー会幹事

鈴木雄介

現場のITアーキテクトが知っておくべき10のこと  
- あるいは、技術が分かるPMが知っておくべき10のこと

# 自己紹介

## ■ 鈴木雄介

- エンタープライズアプリのITアーキテクト
  - 標準化支援、技術支援、新規事業企画...
- ブログ: アークランプ
  - <http://www.arclamp.jp/>
- 日本Javaユーザー会幹事
- 日本Springframeworkユーザー会幹事
- 日経SYSTEM「ITアーキテクトの視点」連載

# 狙い

- ITアーキテクトが現場で”考える”ための考え方を紹介
  - “銀の弾丸アーキテクチャ”は存在しない
  - 現場に合わせて悩み、考えることが大事。答えは自分で見つけるしかない
  - なので、Springも、DSLも、Rubyも、Cloudも、Androidも、Agileも、マインドマップも、SOAも話さない
- キーポイント:世の中にある”知識”を活用しよう

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# 知っておくべき10ぐらいのコト

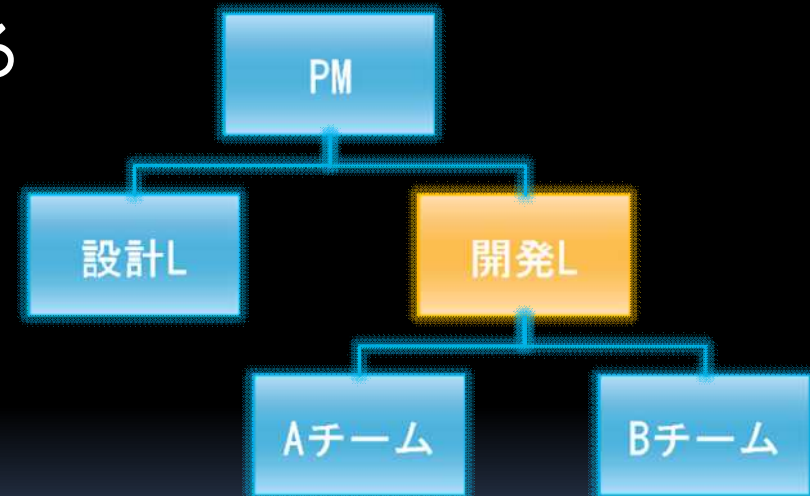
- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# ITアーキテクトとは

- いま何が起きているのか
  - 複雑化する要件
    - 変化するビジネス環境
    - アジャイル/スケーラビリティ
    - ユーザービリティ
  - 複雑化する技術
    - 技術そのものの高度化
    - ヘテロジニアス(異種混在)/レガシー
    - 今日の技術が、明日には古くなる
- この状況を整理する人が必要になる

# ITアーキテクトとは

- 開発リーダーではない
  - エンジニアを取りまとめる人
  - フレームワークを決める人
  - 作る立場のプロ
- 複雑化する要件と技術は、作る立場からだけでは整理できない



# ITアーキテクトとは

- 求められる人物像
  - 利害関係者のバランスを取る
    - 各者が求める価値(含むエンジニア) <> 品質、コスト、期日、リソース...
    - しかも、”作ること(技術)”のリスクが高い



# ITアーキテクトとは

## ■ 立ち位置

- 作る人でも、使う人でもない
  - エンドユーザーでも、システム管理者でも、マーケティング担当者でも、顧客でも、開発者でも、PMでも、保守担当者でもない
  - 中途半端
- 考えたことが誰にも理解されない
  - いくらでも論理的な理由はあるけど、最終的な理由はセンス、信念
- 自分の立ち位置を自分で見つける

# ITアーキテクトとは

- 僕が知っているITアーキテクト
  - 変人
  - 雄弁(言葉も文章も)
  - あらゆる事に興味がある(こだわりも)
    - ロボット、デザイン、建築、宗教、哲学、マンガ、アニメ、政治、経済、お酒、健康、映画、絵画、音楽、文学、ビジネス、環境問題、美、人...
  - アーキテクトは、アーキテクトが選ぶ
  - 少年の心(オタク...?)

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# PMとITアーキテクト

- マネージャとリーダーシップ
  - Do
    - Good managers do the things right
    - Good leadership does the right thing
  - フォーカス
    - マネージャー: 目標と目的、どうやって? いつ?、組織と構造、リスク回避 ...
    - リーダーシップ: ビジョン、何を? なぜ?、チャレンジ、イノベーション、リスクは機会...

# PMとITアーキテクト

- マネージャとリーダー
  - 何をするのか
    - マネージャとは、複雑さに対応する人
    - リーダーとは、変革を起こす人
  - 何を考えているのか
    - 優秀なマネージャはみな教育本能を持っていた。状況がどうあれ、彼らが最初に考えるのはつねに部下一人ひとりに係わること、その部下の成功を助けるために何ができるかということだ。
    - リーダーは未来に惹かれるということだ。リーダーは変化を待ちかね、進歩を待ちわび、現状に強い不満を抱いていてこそ、初めてリーダーなのだ。

# PMとITアーキテクト

- PMとITアーキテクトの違い
  - PMはマネージャー的
  - ITアーキテクトはリーダー的
    - 作ることのリスクに対応するには必要な能力
  - PMはお母さんの
  - ITアーキテクトはお父さんの
- どちらも大事、どちらも必要

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# アーキテクチャとは何か

- アーキテクチャとは、「コンポーネント」、コンポーネント間および「環境」との「関係」、またその設計と進化の指針となる原理に体现された「システム」の基本「構造」である

# アーキテクチャとは何か

- アーキテクチャとは、ソフトウェアシステムの構造に関する「一連の重要な判断」「構造要素」の選択、そしてシステムを構成するインターフェイスに加え、これらの要素間のコラボレーションで明記されたその「動作」、徐々に大型化するサブシステムへのこれら要素の「組み込み」、そしてこの構造の指針となる「アーキテクチャスタイル」である。つまり、これらの要素とインターフェイス、そのコラボレーション、そして構成である

# アーキテクチャとは何か

- ソフトウェアの要素、外部から見えるこれらの「要素」の特性、そしてこれらの「関係」を構成するシステムの構造が、プログラムあるいは計算処理システムのソフトウェアアーキテクチャである (Bassなど)。

# アーキテクチャとは何か

- [アーキテクチャは] 組織の構成であり、システムの関連動作だ。アーキテクチャは、インターフェイス経由でやりとりするパーツ、パーツをつなぐ関係、そしてパーツを組み立てる制約に、再帰的に分解することができる。インターフェイス経由でやりとりするパーツには、クラス、コンポーネント、およびサブシステムなどがある

# アーキテクチャとは何か

- システムのソフトウェアアーキテクチャもしくはシステムの集合は、ソフトウェア構造に関する重要な設計判断すべてと、そのシステムを構成する構造間の対話によって構成されている。設計判断は、システムを成功へと導くために望ましい一連の資質をサポートする必要がある。設計判断は、システムの開発、サポート、および保守のためのコンセプト基盤を提供する

# アーキテクチャとは何か

- 内圧(作ること)と外圧(使うこと)のバランスを、360度で取った結果
  - きれいな円になっているのか？
  - 外圧が強いと潰れてしまう。内圧が強いと破裂してしまう
  - 圧力を見つけることが大事
  - チーム構成やスケジュールまで含んでいい
    - 作れないアーキテクチャに意味はない

## 参考：

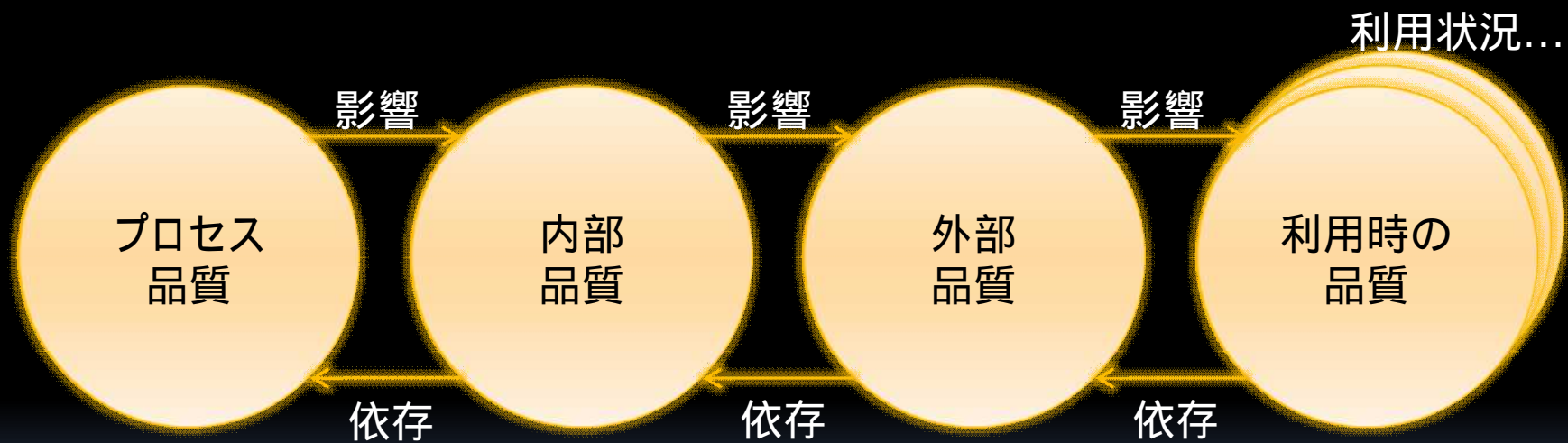
- “そのものの内側から出る適正な力の美を「張り」といい、そのものに外側から加わる圧力のことを「選択圧」という。”
- 深澤直人『デザインの輪郭』

<http://images.google.co.jp/images?q=深澤直人>

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# ソフトウェア品質モデル

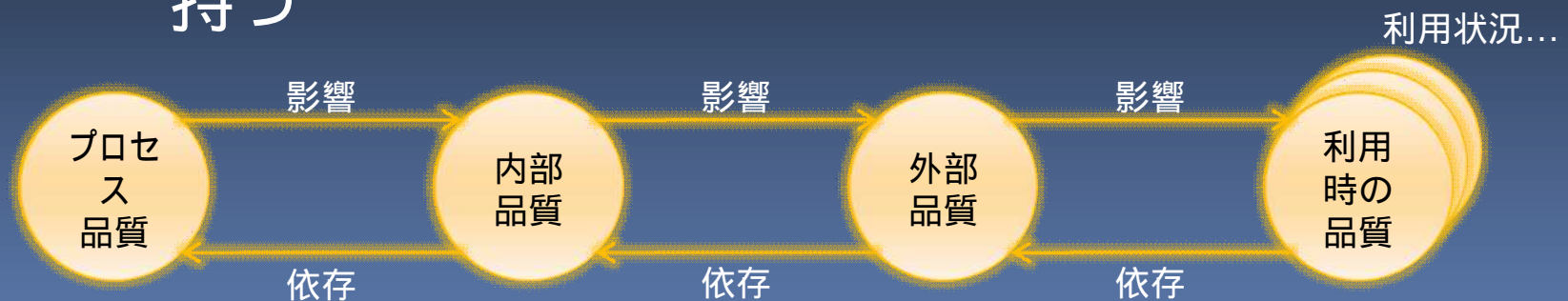


# ソフトウェア品質モデル

	特徴	例
利用時の品質	<ul style="list-style-type: none"><li>・利用状況によって評価が異なる</li></ul>	<ul style="list-style-type: none"><li>・ユーザーAさんとユーザーBさんで評価が異なる</li></ul>
外部品質	<ul style="list-style-type: none"><li>・システムの振る舞い</li><li>・誰がテストしても同じ結果</li><li>・一般的な仕様策定の対象</li></ul>	<ul style="list-style-type: none"><li>・テストケース</li><li>・外部仕様</li></ul>
内部品質	<ul style="list-style-type: none"><li>・システムを構成している要素すべて(含ドキュメント)</li><li>・後に残り、評価が可能</li><li>・エンジニアがこだわるところ</li></ul>	<ul style="list-style-type: none"><li>・クラス図</li><li>・フレームワーク</li><li>・ドキュメント</li></ul>
プロセス品質	<ul style="list-style-type: none"><li>・後に残らない</li></ul>	<ul style="list-style-type: none"><li>・コミュニケーション</li><li>・</li></ul>

# ソフトウェア品質モデル

- どれか1つの品質を向上してもダメ。すべてをバランスよく向上されることが重要
  - 設計は、利用時の品質->外部品質->内部品質->プロセス品質と考えていく
  - 実装は、プロセス品質->内部品質->外部品質->利用時の品質と作られていく
  - アーキテクトは、各品質間のバランスにも責任を持つ



# ソフトウェア品質モデル

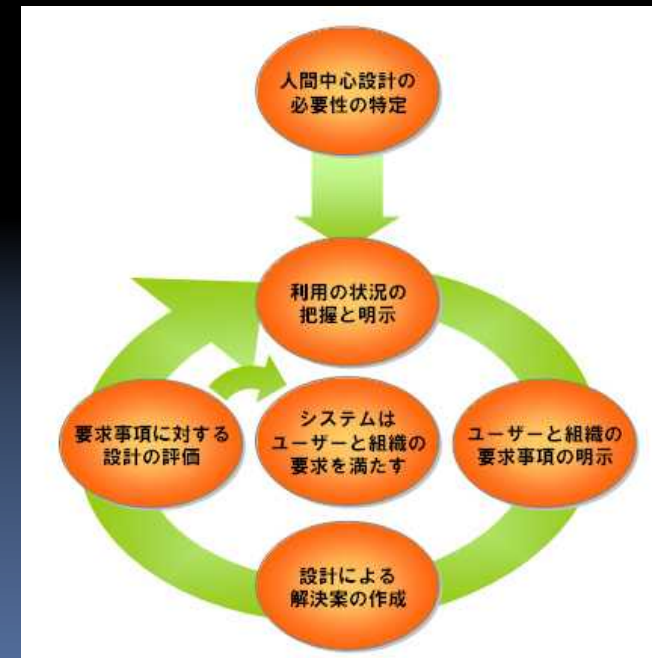
- ユーザーが真に評価するのは、“利用時の品質”だけ
  - だから、UIは重要
- データベースチューニングのコツ
  - 「クエリーの実行速度を3秒短くよりも、3秒待ってもらおうUIを作ること」  
この視点が重要
  - 利用状況を知ることも大事



# 参考: ISO13407人間中心設計

## ■ "Human-centred design processes for interactive systems" (インタラクティブシステムの人間中心設計プロセス)

- 人間中心設計の必要性の特定: 何をデザインするのか、デザインにより何を実現するかを明確にする。
- 利用の状況の把握と明示: 市場でその商品が使われてきた歴史を理解し、各ユーザーが実際どう使っているかを知る。
- ユーザーと組織の要求事項の明示: ユーザーの利用状況から要求を抽出する。デザインに求められる組織的な構造を分析、明示する。
- 設計による解決案の作成: ユーザーと組織の要求事項を元に、それを解決する具体策としてのデザインを作成する。
- 要求事項に対する設計の評価: 作成したデザインが要求事項をしているかの評価を行い、デザインの問題点を抽出する。



# ソフトウェア品質モデル

品質特性	品質副特性	主な内容
機能性	合目的性 正確性 相互運用性 セキュリティ	ソフトウェアを指定された条件の下で利用するとき、明示的および暗示的必要性に合致する機能を提供するソフトウェア製品の能力のこと
信頼性	成熟性 障害許容性 回復性	ソフトウェアを指定された条件の下で利用するとき、指定された達成水準を維持するソフトウェア製品の能力のこと
使用性	理解性 習得性 運用性 魅力性	ソフトウェアを指定された条件の下で利用するとき、理解、習得、利用でき、利用者にとって魅力的であるソフトウェア製品の能力のこと
効率性	時間効率性 資源効率性	明示的な条件の下で、使用する資源の量に対比して適切な性能を提供するソフトウェア製品の能力のこと
保守性	解析性 変更性 安定性 試験性	修正のしやすさに関するソフトウェア製品の能力のこと
移植性	環境適応性 設置性 共存性 置換性	ある環境から他の環境に移すためのソフトウェア製品の能力のこと

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# PMとアーキテクチャ

- PMBOK (A Guide to the Project Management Body of Knowledge)
  - スコープ (目的と範囲)
  - 時間 (期間)
  - コスト (予算)
  - 品質
  - 人的資源 (組織)
  - コミュニケーション
  - リスク
  - 調達
  - 統合 (上記8つの統合)

# PMとアーキテクチャ

- PM: 一般的なスケジュール管理
  - WBSの作成
    - 成果物の構造 内部品質
    - 作業の構造 プロセス品質
  - 見積もり
  - リソース割り当て/期間
  - 進捗管理
    - 実績把握と状態管理
  - 調整
    - 正しい状態 (= WBS) に戻すため

# PMとアーキテクチャ

- トヨタのマネジメントは、なぜうまくいくのか
  - 自動車の基本的な”アーキテクチャ”は100年間変わっていない
  - おなじアーキテクチャでの積み重ねの歴史
  - 最も正確な見積もりは類推。類推するためには「同じものを比べる」ことが重要
- PMの精度を高めるためにアーキテクチャの概念は必須

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# 環境としてのアーキテクチャ

- 人は指示されても動けない
  - 守ってもらうためには監視が必要。「監視されているかも知れない」という意識により自律的に守らせる
  - だけど「規約を守らない空気」には勝てない
- 環境を変質させることで、ヒトの行動は変わる
  - 環境型管理：環境のデザインにより、行動の傾向性を変えていく

# 環境としてのアーキテクチャ

- 割れ窓理論
  - 軽微な犯罪も徹底的に取り締まることで凶悪犯罪を含めた犯罪を抑止できるとする環境犯罪学上の理論。アメリカで考案された。「建物の窓が壊れているのを放置すると、誰も注意を払っていないという象徴になり、やがて他の窓もまもなく全て壊される」との考え方からこの名がある。
- マクドナルドの椅子と音楽

# 環境としてのアーキテクチャ

- アーキテクチャはプロジェクトの環境として機能する
  - 「機能してしまう」
  - だから、正しさを考える必要がある
- アーキテクチャには知性がある
  - アーキテクトの意図とは関係なく、アーキテクチャは、なんらかの振る舞いを産み出してしまう
  - 人が環境に従ってしまうことの重みを考える

# 知っておくべき10ぐらいのコト

- ITアーキテクトとは
  - プロジェクトマネージャーとITアーキテクト
- アーキテクチャとは
  - ソフトウェア品質モデル
  - プロジェクトマネジメントとアーキテクチャ
  - 環境としてのアーキテクチャ
- その他のコト
- まとめ

# その他のコト

- IT以外から学ぶことも大事
  - ファシリテーション/コーチング
    - チャンクアップ/チャンクダウン/スライドアウト
    - アクティブリスニング/承認
  - ロジカルシンキング
    - MESE
    - トップダウン/ボトムアップ
  - マインドマップ
  - フォアキャスト/バックキャスト

# その他のコト

- IT以外から学ぶことも大事 cont.
  - デザイン
    - インフォメーション・アーキテクチャ
    - アフォーダンス
  - 建築
    - 建築家(意匠設計)と構造設計士
    - ランドスケープ
  - ビジネス
    - ドラッカー
    - イノベーションのジレンマ

## さいごに

- 現場に合わせて悩み、考えることが大事。答えは自分で見つけるしかない
- 「自分の考えの考え方」を身につけよう
  - 世の中の知識やコミュニティが助けてくれる
- ITやテクノロジーじゃないことにも、たくさんのヒントがある