ITアーキテクト概論

TIS株式会社
テクノロジー＆イノベーション本部
テクノロジー＆エンジニアリングセンター

目次

- はじめに
- ITアーキテクトの主な業務

  あるオンラインチケッティングシステムの開発
  - 提案工程
  - 要件定義工程
  - 設計工程
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

- **はじめに**
- ITアーキテクトの主な業務
  あるオンラインチケッティングシステムの開発
  - 提案工程
  - 要件定義工程
  - 設計工程
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

ITアーキテクトってどんな人？

ITアーキテクトと呼ばれる役割に対して、興味はあるけれど、あまりその役割の人に接する機会もなく、どんな仕事をしているのだろう？、という疑問を持っているのではないでしょうか。

本資料の内容

本資料では架空のプロジェクトを舞台に若手技術者の目を通して、ITアーキテクトの仕事内容や、やりがいを紹介していきます。

また最後にキャリアパスの参考として、業務領域のスペシャリストからITアーキテクトに進んだ事例を紹介します。

ITアーキテクトの専門分野

ITアーキテクトの専門分野は以下の3つに分かれています(※)。

- **アプリケーションアーキテクチャ**
  - **機能属性、仕様を明らかにし、アプリケーションアーキテクチャ（アプリケーションコンポネント構造、論理データ構造等）を設計する。**
- インテグレーションアーキテクチャ
- インフラストラクチャアーキテクチャ

本資料では、「アプリケーションアーキテクチャ」を専門分野とするITアーキテクトについてお話しします。

※ITアーキテクトの専門分野は、 IPAのITスキル標準V3「職種の概要と達成度指標（４） ＩＴアーキテクト」より抜粋。
https://www.ipa.go.jp/jinzai/skill-standard/plus-it-ui/itss/ps6vr70000004x60-att/000024927.pdf

- はじめに
- **ITアーキテクトの主な業務**

  あるオンラインチケッティングシステムの開発

  - 提案工程
  - 要件定義工程
  - 設計工程
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

あるオンラインチケッティングシステムの開発
～プロローグ～

人物設定

- 主人公
  - 入社6年目のアプリケーションスペシャリスト（APS：業務領域のスペシャリスト）
  - APSを名乗っているが、技術面への興味も強い。
  - この為、業務面だけでなく、技術面に触れることのできる役割をやってみたいと思っている。

事の始まり

月に一度の定例上司面談にて・・・

上司

ところで、今のプロジェクトの離任が近付いていますが、次はどんな事をやってみたいですか？

主人公

実は技術面に色々触れられる機会があれば、挑戦してみたいと思っています。

そうですか、ふーん・・・。

事の始まり

月に一度の定例上司面談にて・・・

上司

今、オンラインチケッティングシステムの提案案件があるんです。
そこのアーキテクチャ担当が、誰か元気な人はいないかと言っていたんですが。
やってみます？

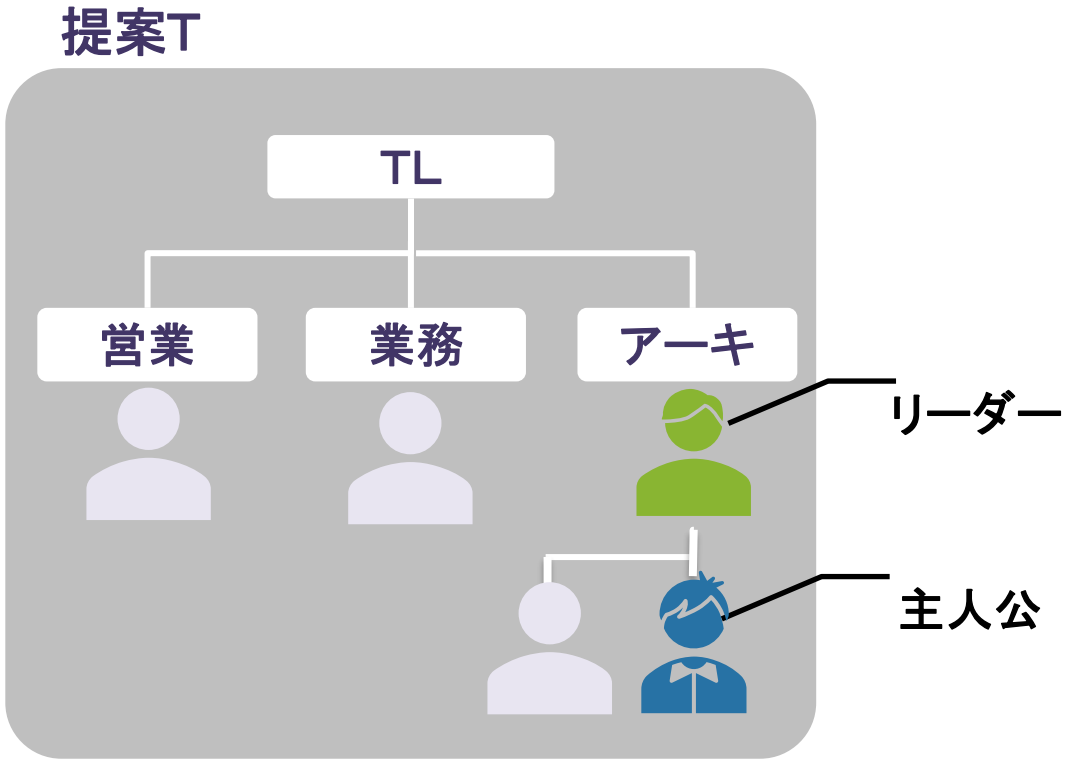
主人公

せっかくのチャンスですし、やってみます！

じゃあ、そういうことで。

チームの体制

主人公は、オンラインチケッティングシステム提案案件のアーキテクチャチーム（アーキチーム）に所属することになりました。



人物設定

- リーダー
  - 入社15年目のITアーキテクト（ITA）。
  - 業務アプリチームがスムーズに開発作業を行えるよう心がけている。

最初の挨拶

主人公はリーダーに挨拶に行きました。

リーダー

こんにちは。主人公です。
今後、よろしくお願いします。

主人公

こちらこそよろしく！
やることは色々あるからね。でも、**何でもかんでも自分たちで解決しなければいけない、とは思わないでね**。
僕らが決めたことが大丈夫か、第3者がチェックする仕組みもあるし、全社的にアーキを支援する仕組みもある。
肩の力を抜いて頑張っていこう。

アーキチームの仕事(1/4)

アーキチームの仕事には、どのようなものがあるのでしょうか？

業務アプリムやアーキ、インフラなどの各チームごとの仕事を一覧化した資料があります。

- Fintan > ウォーターフォール開発における役割分担シート
  https://fintan.jp/?p=1326

  ※本資料では、役割分担シートの「アプリ基盤チーム」を「アーキチーム」と表記しています。

アーキチームの仕事(2/4)

本資料では主に、役割分担シートの大分類レベルで、設計、製造作業以外に注目します（設計、製造作業は業務アプリチームとほぼ同じの為）。

役割分担シート			本講座	
大分類	工程	中分類	工程	表題
立ち上げ	全工程	PJ計画	提案	体制構築、 開発プロセスの定義
	要件定義	非機能要件定義	要件定義	非機能要件定義
	要件定義	評価/選定		
	要件定義	基盤機能抽出		
	要件定義	アプリケーション方式設計	要件定義	方式設計
標準化	要件定義	アプリ設計標準の策定	要件定義	設計の準備
	外部設計	アプリ開発標準の策定		
	外部設計	アプリ開発環境構築		
	-	アプリ保守環境構築		
フレームワーク設計	外部設計	Nablarchカスタマイズ		
	外部設計	共通機能		
データモデル設計	外部設計	基本設計		
	外部設計	論理設計 (アプリ基盤使用テーブル)		
	外部設計	索引設計(アプリ基盤)		
	外部設計	マスタデータ作成(アプリ基盤)		

Fintan > ウォーターフォール開発における役割分担シート
https://fintan.jp/?p=1326

アーキチームの仕事(3/4)

役割分担シート			本講座	
大分類	工程	中分類	工程	表題
サンプリングレビュー	外部設計	サンプリングレビュー	設計、製造	業務開発チームのサポート
インフラ設計	外部設計	その他ミドルウェア設計		
	外部設計	監視機能設計		
	外部設計	バッチジョブ/ジョブネット設計		
環境構築	製造	インフラ/ミドルウェア構築		
	製造	アプリケーション構築設計	製造	リリース手順の検討
	製造	アプリケーション構築		
構成管理	製造	アプリケーション構成管理		
プログラミング・単体テスト	製造	業務アプリケーション		
	製造	業務共通機能		
	製造	システム共通機能		
テスト	外部設計	テストツール		
	製造	アプリケーションフレームワーク単体テスト	-	ご参考
運用設計/構築	外部設計	セキュリティ運用		
その他	全工程	新規着任者研修(アプリ基盤)		
	結合テスト	引継ぎ		

Fintan > ウォーターフォール開発における役割分担シート
https://fintan.jp/?p=1326

アーキチームの仕事(4/4)

これらの仕事を工程ごとに整理して並べると、次のようになります。

アーキチームの仕事の流れ(1/2)

工程ごとの仕事

提案

- 顧客の要求確認
- 体制構築
- 開発プロセスの定義
- フレームワークの選定

要件定義

- 方式設計
- 非機能要件定義
- 移行方式の設計
- 設計の準備

設計

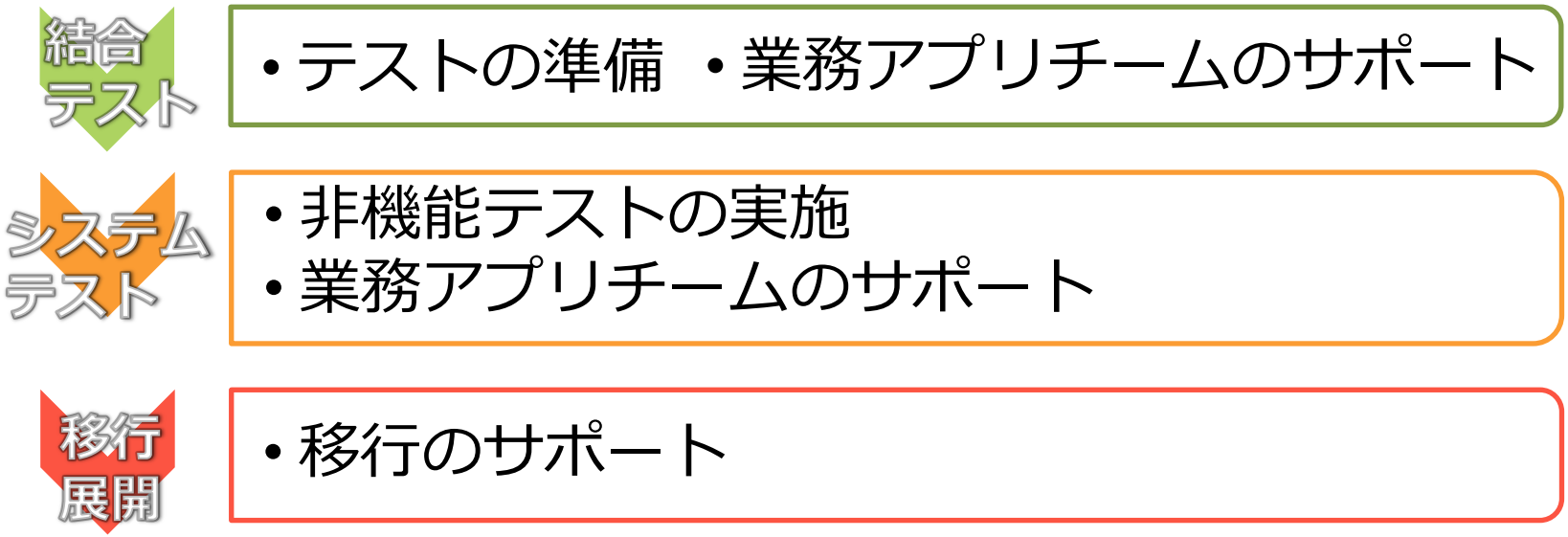
- 製造の準備
- 業務アプリチームのサポート

製造

- テストの準備
- リリース手順の検討
- 業務アプリチームのサポート

アーキチームの仕事の流れ(2/2)

工程ごとの仕事



それでは各工程ごとに、一つひとつの仕事について、主人公とリーダーの会話を通して見ていきましょう。

- はじめに
- ITアーキテクトの主な業務

  あるオンラインチケッティングシステムの開発

  - **<u>提案工程</u>**
  - 要件定義工程
  - 設計工程
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

アーキチームの仕事の流れ

工程ごとの仕事

提案

- 顧客の要求確認
- 体制構築
- 開発プロセスの定義
- フレームワークの選定

要件定義

- 方式設計
- 非機能要件定義
- 移行方式の設計
- 設計の準備

設計

- 製造の準備
- 業務アプリチームのサポート

製造

- テストの準備
- リリース手順の検討
- 業務アプリチームのサポート

顧客の要求確認

顧客の要求確認(1/8)

リーダー

まずは、RFPから読み取れるお客様の要求を確認していこう。

主人公

わかりました。まずは、要求の確認。業務チームと同じですね！

ええ。ただ、着眼点は増えるよ。

※RFP：Request For Proposal。
お客様から頂く提案依頼書。

顧客の要求確認(2/8)

RFPを確認していたリーダーは、以下の部分に深く掘り下げる必要性を感じ、お客様にヒアリングすることにしました。

ＲＦＰ

チケットは金券と同等の扱いのため、
本システムの信頼性・可用性は金融系システムと同等とする。

リーダー

たとえば、障害が起きてシステムが利用できなくなるのは、全く許容されないのかな？

顧客の要求確認(3/8)

リーダー

障害によるシステム停止は、数秒であっても許容されないでしょうか？

お客様

そうですね。数秒であっても困ります。１か所で障害が起きたくらいでは停止しないようにしてください。

そうなると、仕組みも複雑になり、費用もかなりかさんでしまいます。それでも、許容できないのでしょうか。それから、計画停止などは？

顧客の要求確認(4/8)

リーダー

お客様

数秒の間に飛んでくるチケット購入依頼数は莫大なので、その機会損失やその後の風評を考えると、システム停止は避けたいんです。
必要な投資だと思っています。
具体的には、計画停止は、月1日、それ以外に、1日1秒止まるかどうか、という感じです。

なるほど、わかりました。（実現には、かなり工夫が必要そうだけど、不可能じゃないな！）

顧客の要求確認(5/8)

リーダー

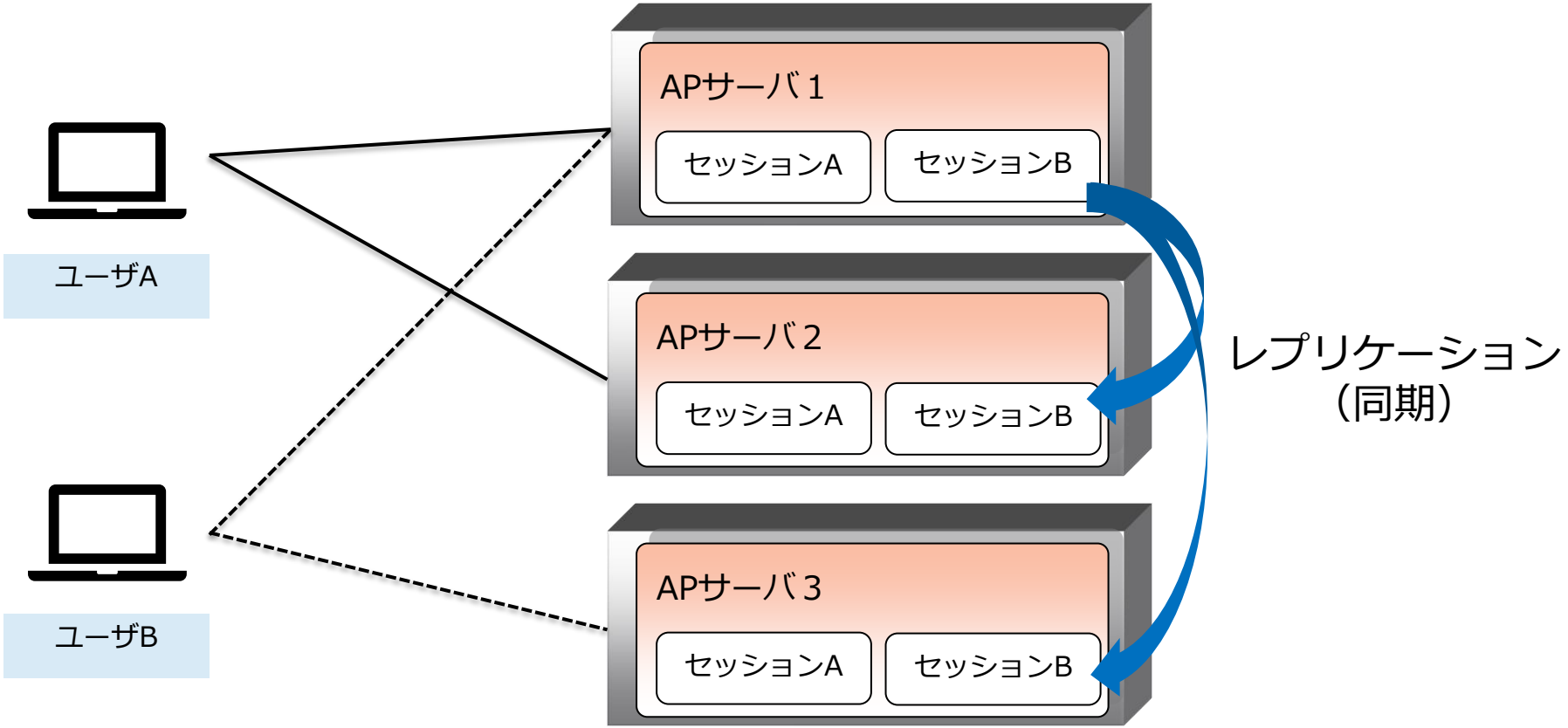
どうやって実現するんですか？

主人公

アプリケーションのサーバを冗長化して、サーバー間でセッションを複製する、などだね。色々とやることがあるよ。

顧客の要求確認(6/8)

具体的には複数のサーバを用意し、セッションをサーバ間で同期することで要求を実現しました。



※実現方法について詳しく知りたい方は以下を参照ください。
Fintan > アプリケーションアーキテクチャの学習コンテンツ > Webアプリケーションのセッション管理
https://fintan.jp/?p=8376

顧客の要求確認(7/8)

リーダー

業務アプリを作った経験しかなかったので、新鮮です！これまでの案件でも、誰かがこういったことを考えてくれていたんですね。

主人公

そうだね。システム全体を支える土台に関わることだから、とっても重要なんだ。

その分、すごくやりがいがあるよ。

顧客の要求確認(8/8)

リーダー

このように、ＩＴアーキテクトは、機能要求だけでなく、**非機能要求にも**着目する必要があるんだよ。

主人公

今回のことで、よく分かりました！

体制構築

主人公の疑問

リーダーのスケジュールを確認していた主人公は、ある疑問を持ちました。

★アーキチームリーダーのスケジュール★

	進捗					
	進捗		---			
	進捗					

○月×日 10:00～

PMと打ち合わせ

主人公

進捗以外で、技術の人間が
PMと相談する内容って何だろう？

PMとの要員の調整 (1/3)

PMと、どんな話をしたんですか？

リーダー

プロジェクト（PJ）を順調に進めるために、「必要な要員」について説明して要員調整をお願いしてきたよ。

主人公

「必要な要員」を検討・調達するのはPMの仕事ではないんでしょうか…？

そうとばかりは言えないよ。

PMとの要員の調整 (2/3)

リーダー

例えば、今回のシステムは高信頼システムとする為に、色々工夫が必要だよね。
もし、アーキチームにその知見のあるメンバーがいなかったらどうなるだろう？

主人公

検討不足で設計に抜け漏れが発生するかもしれません。

そうだね。そういったリスクを潰すために、今回のアーキチームには、高信頼システムの開発経験のあるITアーキテクトが欠かせないんだ。

PMとの要員の調整 (3/3)

リーダー

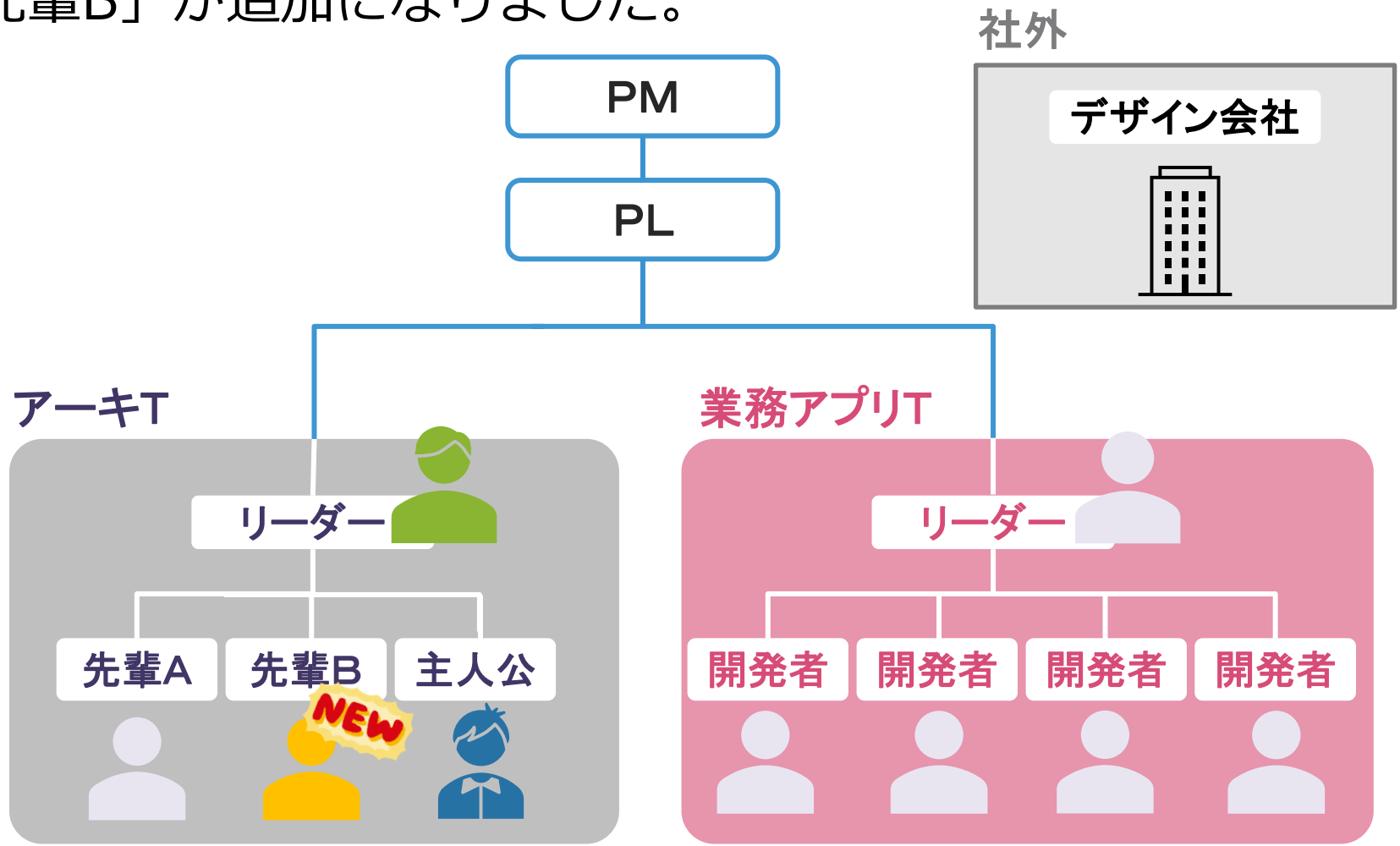
このように、**技術要件やリスクから、必要な要員を検討**し、リクエストするのは、技術の視点でPJ全体を見ているアーキならではの重要な仕事なんだ。

主人公

ITアーキテクトは、設計や技術課題の解決だけが仕事だと思っていたから意外でした。こんなこともやっていたんですね。

チームの体制（予定）

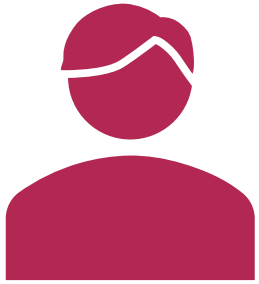
リーダーの調整により、開発時のアーキチームの体制に
「先輩B」が追加になりました。



プロジェクトからの期待(1/2)

リーダーはPMから、アーキチームに期待していることを伝えられました。

PM



アーキチームには、特にUIと性能面でPJをリードしてくれることを期待していますよ！

リーダー

わかりました。
頑張っていきましょう！

プロジェクトからの期待(2/2)

リーダー

UIと性能に関しては、十分な戦略を練って臨む必要があるね。

主人公

どういうことですか？

PJからの期待に応えるべく、場当たり的ではなく、前もって作戦を立てて行こうということだよ。
期待されているというのは、やる気の源でもあるよね。

はい！

PJからの期待（PJ全体の成否に関わる期待）は、アーキチームのやる気の源、やりがいの一つです。

開発プロセスの定義

開発プロセスの定義(1/4)

リーダー

「開発プロセス定義」と聞くと、PMが担当かなって思っちゃいます。アーキが担当するのは、体制構築と似たような理由ですか？

主人公

そうだね。ここで言う開発プロセスとは、各工程の進め方なんだ。開発プロセスは、技術要件やリスクを見越して、品質や生産性、性能を十分に仕上げるための作戦の一つだからね。

開発プロセスの定義(2/4)

リーダー

細かい進め方は直前の工程で決めるとして、大まかにどう進めるかは、今考えておかないとね。

例えば、業務アプリチームが準備してくれた設計書フォーマットにプログラミングに必要な情報が足りていなかったり、いざ設計を始めてみたら使いにくいところがあったりしたらどうなるだろう？

主人公

そのまま使い続ければ、品質や生産性が下がるし、先に修正しておかなければその工程が遅れる。いずれにしろ、PJの進行に関わる問題になります。

開発プロセスの定義(3/4)

リーダー

そうだね。だから1つ前の工程の間に、先行して次の工程を試行してみるんだ。
こういう風に、品質、生産性、性能でPJ進行に関わるような問題が生まれないように、開発プロセスを定義しておくんだ。

主人公

先の先まで考えておくんですね。

そうやってPJを成功に導く。アーキの仕事の醍醐味の一つだよ。
ちなみに、品質、生産性、性能の3つはアーキテクトが持つ着眼点の主なものだよ。

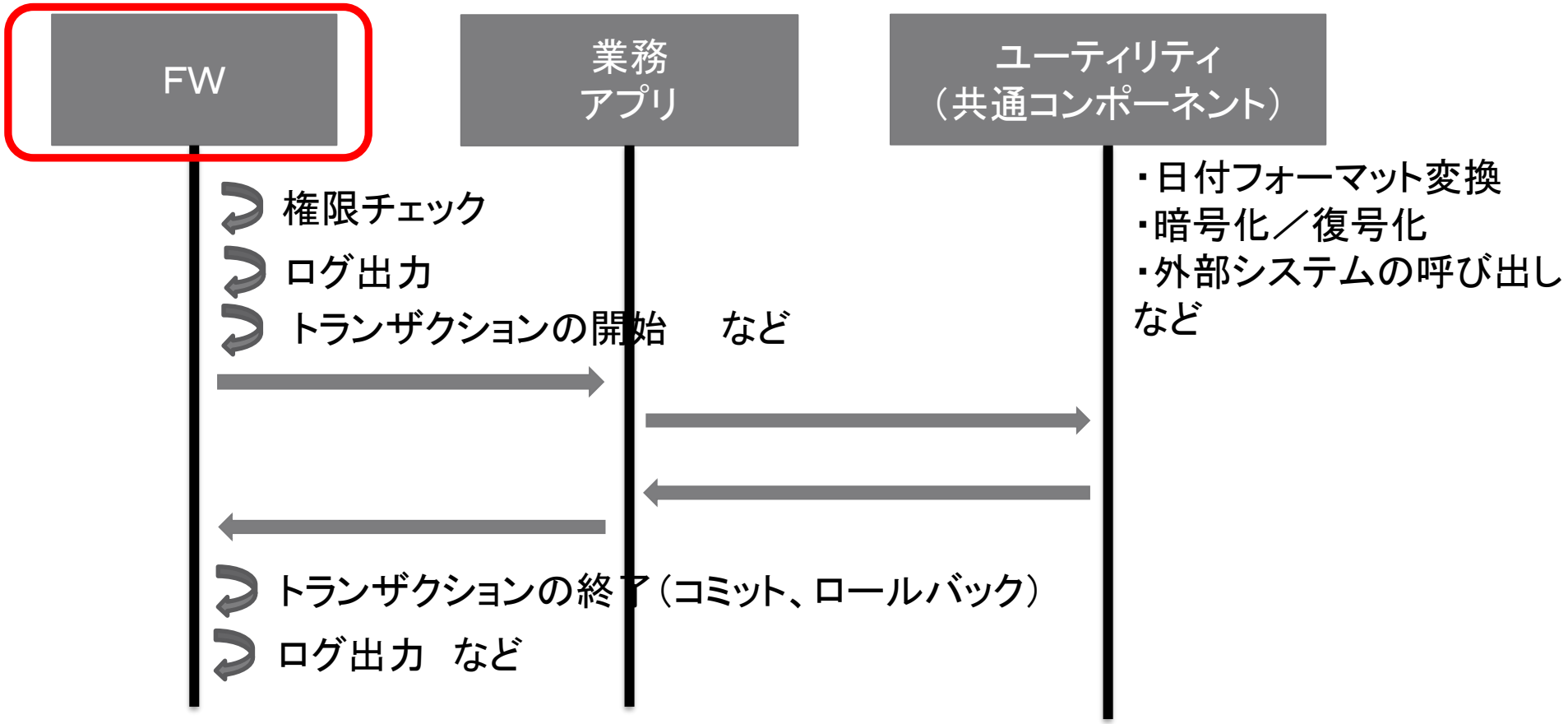
開発プロセスの定義(4/4)

検討の結果、以下のことを各工程に組み込むことにしました。

- 要件定義工程
  - 業務アプリチームと設計の先行実施
  - 高難度機能の洗い出し
- 設計工程
  - 業務アプリチームと製造の先行実施
  - 設計書のアーキレビュー（机上性能検証）
- 製造工程
  - アーキによるコードレビュー
- 結合テスト工程
  - テストデータによる性能検証
- システムテスト工程
  - 本番相当データによる性能検証

フレームワークの選定

フレームワーク（ＦＷ）とは？



フレームワークとは、どのような業務処理であっても共通となる処理を実行する、アプリケーションの土台となるソフトウェアのこと。

FW選定とカスタマイズ

FW選定の検討ポイント

- 類似システムへの導入実績があるか
- 初期学習コストや要員調達のしやすさ
- どのような機能が搭載されているか
- 調達方法（無償製品導入、有償製品導入、自前構築）
- サポート体制
- バージョンアップ頻度、EOSL（サポート期間）のサイクル

今回、購入は予算的に、1から作るのはスケジュール的に厳しい
－＞実績のある無償製品(OSS)をカスタマイズする。
主なカスタマイズは、製造担当者の自由度を落とす
ために部品を作り込み、性能測定ポイントを多数
埋め込む。

FW選定の検討ポイント(1/3)

FW選定の検討ポイントについて、主人公がリーダーに質問しています。

リーダー

導入実績は、お客様が気にしますよね。機能や調達方法、サポートがポイントになるのもわかります。
その他のポイントは、なぜ考慮が必要なんですか？
実績があり機能もサポートも十分、コストが見合うなら、アーキとしては採用決定で良いような気もするんですけど・・・。

主人公

FW選定の検討ポイント(2/3)

リーダー

要員調達もアーキの仕事の一つだからね。どんなによくできたFWでも、使い始めるまでに時間がかかったり、使える人を集めるのが大変だったりしたら、PJがスムーズに進まないでしょ。

主人公

FW選定の検討ポイント(3/3)

リーダー

バージョンアップ頻度やEOSLサイクルはカットオーバー後を考えての事だね。
EOSLサイクルは、お客様が計画している耐用年数に合致するか。
バージョンアップも、不具合が放置されるような低頻度では困るけど、多すぎるのも考え物だ。
製品として不安定かもしれないし、それだけで一大PJになりうるFWバージョンアップは気軽にはできない。

主人公

主人公の気付き

主人公

FW選定では、単純に機能だけでなく、要員のこと、さらにカットオーバー後のことまで考えてる。
カスタマイズは、製造工程や性能テストのことを考えての対応だよな。
考えておくことは、技術面以外にもあるし、ずっと先のことまで考えておくんだな。

- はじめに
- ITアーキテクトの主な業務

  あるオンラインチケッティングシステムの開発

  - 提案工程
  - **要件定義工程**
  - 設計工程
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

アーキテクトの視点(1/2)

リーダー

主人公

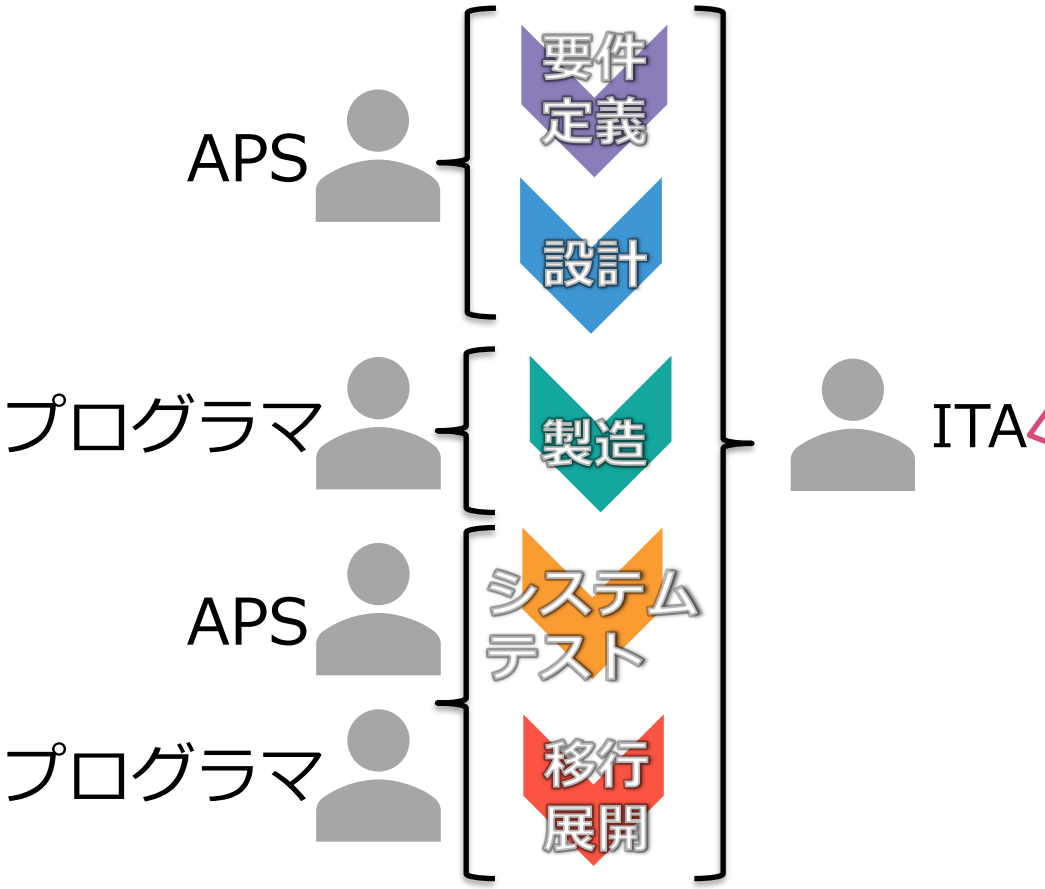
要件定義工程では何をするんですか？

要件定義の裏側で、方式設計書の作成や設計の準備などを行うよ！

それから、「アーキテクトの視点」でPJを見て仕事をする必要があるんだ。

うーん、「アーキテクトの視点」って何ですか？

アーキテクトの視点(2/2)



PJが最善の結果を出せるよう、一つの工程だけでなく、**PJの最初から最後まで**を技術の観点から**見渡す**視点のことだよ。

アーキチームの仕事の流れ

工程ごとの仕事

提案

- 顧客の要求確認
- 体制構築
- 開発プロセスの定義
- フレームワークの選定

要件定義

- 方式設計
- 非機能要件定義
- 移行方式の設計
- 設計の準備

設計

- 製造の準備
- 業務アプリチームのサポート

製造

- テストの準備
- リリース手順の検討
- 業務アプリチームのサポート

方式設計

処理方式の設計(1/4)

リーダー

方式設計ですか！何だかアーキテクトっぽい作業ですね。具体的には何をするんですか。

主人公

そうだね。色々な作業があるけど、「方式設計」というくらいだから、処理方式の設計がメインだね。
結果は「方式設計書」としてまとめるよ。

処理方式の設計(2/4)

リーダー

なるほど！ところで「処理方式の設計」ってどんなことをするのでしょう。

主人公

今までのPJでも、登録処理のようなリクエストにはPOSTを使え、検索処理ならGETでも良い、とか、ブラウザにキャッシュさせないようにこれを書くこと、とか決まっていなかったかい？

そういえば、決まっていました。

処理方式の設計(3/4)

リーダー

他にも、入力値のバリデーションのような、仕掛けは知らないけれど、必ず実施する処理などあっただろう。

簡単に言うと、そういった色々な処理の方法を決めるのが処理方式の設計なんだ。

主人公

なるほど。今まで、知らない間に決まっているなぁと思っていた処理の方法を、実際に決める作業なんですね。
すると、システムの基本動作を決めるってことですか。

処理方式の設計(4/4)

そうなるね。

リーダー

影響範囲が広くてワクワクします。
具体的にはどう進めるんですか。

主人公

例えば、GETとPOSTであれば、
GETだと情報が漏れる危険性があるけど、URLに情報を載せられる。
だから、機密情報がない場合や、
キャンペーンサイトからの遷移を特定したい時等はGET、それ以外はPOSTにする、と、これが設計。
それを方式設計書にまとめるんだ。

方式設計書の例(1/2)

3.1.3. HTTPメソッドの使い分け

3.1.3.1. HTTPメソッドの使い分け手段

方法	方針	理由
ログイン画面への遷移	GET	機密情報を含まないため。
パラメータ付きURLからの遷移	GET	キャンペーン等のURLからの遷移を特定するため。
上記以外の遷移	POST	機密情報がGETリクエストのパラメータに含まれてしまい、履歴から当該情報が漏洩することを防止するため。

3.1.4. ブラウザキャッシュ制御

3.1.4.1. ブラウザキャッシュ制御機能概要

ブラウザなどが、取得したウェブページのデータを一時的に保存する機能である。
繰り返し同じページにアクセスしたときに、インターネット上のデータではなく、保存されたデータを参照するため、すばやく表示できる。
WebサーバアプリケーションやProxyサーバにキャッシュされている場合があるため、アプリケーションのブラウザキャッシュ制御に加えて各サーバのキャッシュ制御を併用する。

3.1.4.2. ブラウザキャッシュ制御手段

対象	手段	理由
ブラウザキャッシュ制御	HTTPヘッダ	3.1.12.コンテンツ更新機能があり、キャッシュに有効期限を設ける必要があるため

3.1.4.3. ブラウザキャッシュ制御手段詳細

(1) HTTPヘッダ

画面レスポンス低下など操作性を損なわないよう、プロジェクトで定めたキャッシュ保持期間を基準として保持を許容する。

3.1.4.4. PJ固有の要件

条件付きリクエストを付与して更新を都度確認する手法もあるが、ブラウザの種類・設定等の環境固有の問題を排除できないため一定の保持期間を設ける。

パラメータ	値
キャッシュ保持期間	XX時間

Fintan > アプリケーション方式設計書サンプル
https://fintan.jp/?p=1323

方式設計書の例(2/2)

7. 処理方式共通

7.1. 入力値精査

7.1.1. 入力値精査の種類と順番

7.1.1.1. 入力値精査処理方式の選択

本システムでの選択基準を以下に示す。

精査の種類	方針	理由
Java EE7のBean Validation(JSR349)に準拠したバリデーション機能(Bean Validation)	使用しない	Bean Validationはバリデーションの実行順序を指定できないため、本プロジェクトではでこちらは選択しない。
Nablarch独自のバリデーション機能(Nablarch Validation)	使用する	上記の理由から本プロジェクトでは、こちらを選択する。

7.1.1.2. 入力値精査の種類

入力値精査には、以下の種類がある。

精査の種類	概要
単項目精査	必須、桁数など、属性値単独の精査。
項目間精査	必須、大小など、同一Formの複数の属性値の精査。
データベースを用いた精査	データベースを使用した精査。

7.1.1.3. 入力値精査対象

クライアントや外部システムから送信されるすべてのデータをサーバサイドで入力値精査する。

なお、ラジオボタンやリストボックスなどの選択項目、hidden項目も偽装可能なため対象とする。

Fintan > アプリケーション方式設計書サンプル
https://fintan.jp/?p=1323

非機能要件定義

非機能要件定義(1/2)

リーダー

非機能要件定義というと、応答時間や、可用性などたくさんのことを、細かに決めていくイメージで、知識の幅も深さも持っていなくてはいけない感じがします。

主人公

そうだね。でも、全てをアーキチームだけで解決しようとする必要はないよ。IPAの「非機能要求グレード」というチェックリストを使ったり、インフラと協力したりしながら進めるんだ。

非機能要件定義(2/2)

非機能/機能要件定義について詳しく知りたい方は
以下のコンテンツもご参照ください。

非機能要件定義

- Fintan > アプリケーションアーキテクチャの学習コンテンツ > 非機能要件定義
  https://fintan.jp/?p=8376

要件定義

- Fintan > 要件定義基礎研修テキスト
  https://fintan.jp/?p=1389

IPAの「非機能要求グレード」
https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html

移行方式の設計

移行方式の設計(1/2)

リーダー

移行方式の設計って、本番カットオーバーの準備ってことですよね。随分早くないですか？

主人公

テスト環境から本番環境へ移行するための準備だね。
例えば、手作業でマスタデータの移行をすると、作業ミスの原因を作ってしまう。だから、ツールを用意したりするのだけど、そうなると、今から準備しておかないと間に合わないんだよ。

移行方式の設計(2/2)

リーダー

なるほど。実施は先の事だけど、
準備は違うんですね。
どんなことをするんですか。

主人公

そもそも、どういった方法で移行を実施するのか、という移行の方式設計だね。
例えば、カットオーバー前までは、サイトにアクセスしても繋がらないけど、カットオーバーしたら繋がるようになる。この切り替えをどういう仕組みで実現するかとかね。
他にも、移行ツールをどうするか、といったことも検討するよ。

設計の準備

設計の準備

リーダー

設計の準備で主にやることは、
- 設計書フォーマット類の策定
- 設計標準の策定と顧客合意
- 設計工程の作業プロセスの定義

だね。

主人公

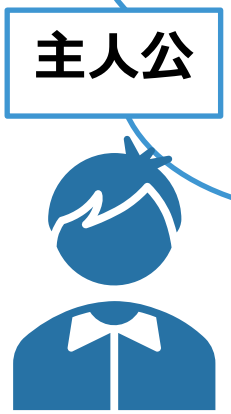
設計書フォーマット類の策定

設計書フォーマット類の策定では、以下のような作業を行います。

- 設計書フォーマットの策定
- サンプル設計書の作成
  - サンプルコードの作成時にも使える機能を対象とする。
- 設計書類についての、業務チーム向けの説明会
  - 作成すべき成果物の一覧と書き方の説明。
  - 設計書間の繋がりを説明し、後工程で「作成した設計書がどのように使われるか」を業務チームが理解できるように説明する。

一例として、Fintanで公開しているNablarchの設計書フォーマットとサンプルをご参照いただけます。
Fintan > Nablarch開発標準 > 設計書フォーマット＆サンプル
https://fintan.jp/?p=1145

主人公の気づき



今まで、設計工程で設計書フォーマットがある事なんて当然だった。でもそれは、業務アプリチームが困らないように、アーキチームが**先回りして**フォーマットなんかを用意してくれていたんだな。
アーキチームは、縁の下の力持ちって感じだな。

設計標準の顧客合意

リーダーは設計標準の中でも、特にユーザインターフェース標準をお客様に詳細に説明する事にしました。

- 設計標準
  - **ユーザインターフェース標準**
  - DB設計標準
  - アプリケーション設計ガイド
  - ・・・

リーダー

全ての設計標準を、お客様に細かく説明するわけではありません。
お客様のこだわりのある部分を重点的に説明します。
また、今回はPJからの期待も考慮しました。

UI標準の顧客合意(1/4)

お客様

今日はユーザインターフェース、つまり、今回のシステムの見た目や使い勝手の説明に参りました。まず、対応ブラウザはEdge、Chrome、Safari、FireFoxとします。バージョンは、テスト実施時の最新バージョンとします。

対応ブラウザを限定するのは百歩譲って、バージョンは何とかならない？

過去バージョンのサポートはセキュリティ上のリスクになりえるのでお勧めできません。期間と費用も必要になりますし。

リーダー

UI標準の顧客合意(2/4)

次に、画面に使う部品をご説明します。

お客様

適当にお任せするよ。画面イメージで見てもいいし。

リーダー

RFPに、「UIにこだわる」とありましたし、全体の統一感や、今後の作業をスムーズに進めるためにも、ぜひ確認いただきたいです。

そうなんだ。じゃあ、確認しようか。

あ、これは変えてほしいな・・・

さて、続いて・・・

・・・

・・・

UI標準の顧客合意(3/4)

お客様

以上となります。細かい内容は割愛していますので、実際に読んでいただき、結果をお知らせください。

リーダー

了解です。

エンドユーザーの目に直接触れる部分の決めごとで、本システムの評判を大きく左右する部分でもあります。お手数をおかけしますが、よろしくお願いします。

UI標準の顧客合意(4/4)

UI標準、合意してくれるといいですね。それにしても説明がとても丁寧というか、しつこかった気がするのですが。

リーダー

最後にお客様に言った通りだよ。それに、「お任せするよ」と言われても、実物を見ると、「これじゃない」なんてこともよくあるし。そうなると、**大変な手戻りになるからね。それを防ぎたい**から、ちょっとしつこいくらいにきちんと説明するんだ。

主人公

この後、数度のやり取りの後、UI標準を合意することができました。

設計工程の作業プロセスの定義(1/5)

設計工程の作業プロセス定義について、主人公がリーダーに質問しています。

主人公

設計工程のプロセスって、各業務チームで設計書を作成してレビューするだけじゃないんですか。

リーダー

ああ、そんな単純ではないよ。
確かに、「作ってレビュー」が中心だ。でも、

- 要件の抜け漏れを防ぐ
- 製造に十分な記述か確認

など、**先々の工程**で課題が発生しないように、**今から手を打って**おかないとね。

設計工程の作業プロセスの定義(2/5)

どんな手を打っておくんですか。

リーダー

簡単なところだと、チェックリストを作って、設計者に自己チェックしてもらう、というのがあるね。これは基本。

顧客に設計内容の合意を取る時は、文章ではなくモックで確認してもらう、というのもある。設計書の文章は、技術者以外にはわかりにくいし、モックは直感的だから。

これらは要件の抜け漏れを防ぐ工夫だね。

主人公

設計工程の作業プロセスの定義(3/5)

設計書の品質を守るためにやる事は、やっぱりアーキレビューかな。

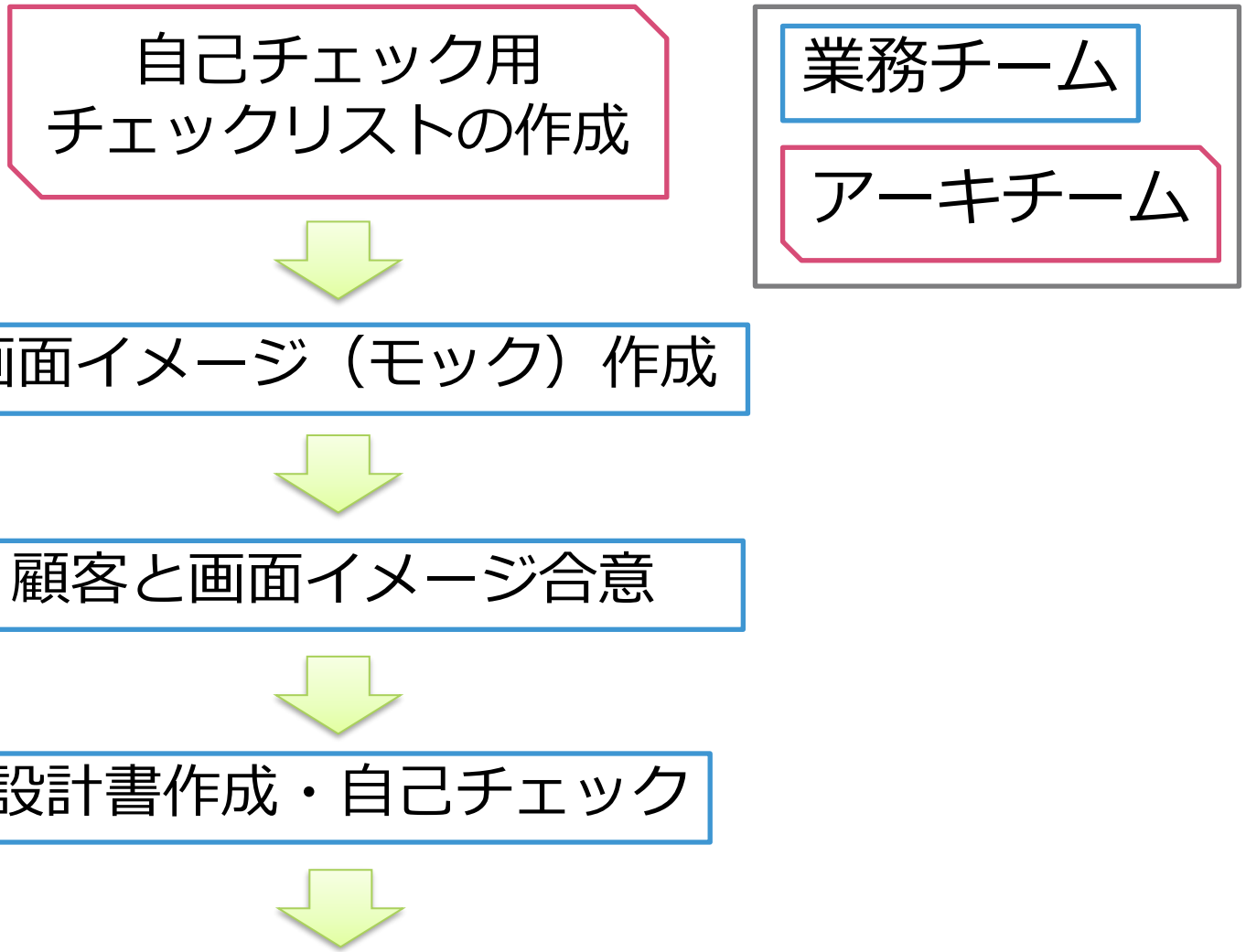
リーダー

アーキチームが直接設計書をレビューするんですよね。業務チームの時に受けました。

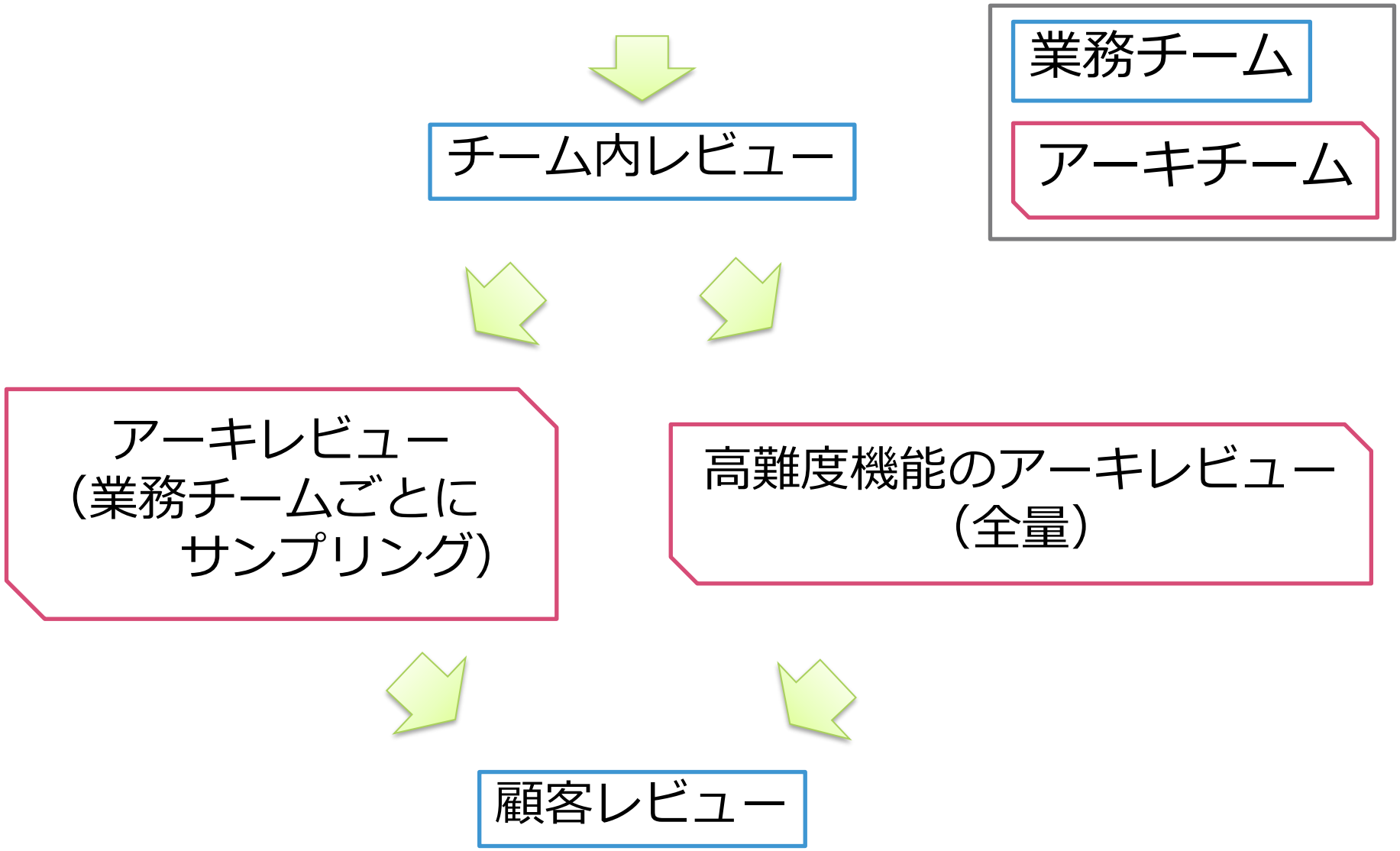
主人公

うん。業務チームが「完成」としたものを、アーキが直接確認するんだ。
これらを並べると、次のようなプロセスになるよ。

設計工程の作業プロセスの定義(4/5)



設計工程の作業プロセスの定義(5/5)



高難度機能の洗い出し(1/4)

業務アプリチームが進めている要件定義情報を元に以下のような機能を洗い出しておく。

- 業務的に難易度が高いもの（業務要件が複雑）
- 技術的に難易度が高いもの
（画面遷移・DBアクセスなどが複雑）

こういった高難度機能は、後々品質や性能、実装方法が問題になる可能性が高い。また、後々問題に気付いたのでは、対応にとても手間がかかる。
後工程で問題が発生しないよう、早い段階で見極め、対応を準備する必要がある！

高難度機能の洗い出し(2/4)

リーダー

今回のシステムだと、どの要件の難易度が高いのでしょうか？

例えばこの要件とか難しいね。

- 指定席のチケットは、単に売れたチケット数とその残数を管理するだけではなく、在庫管理を行う

どういうことかと言うと・・・

主人公

高難度機能の洗い出し(3/4)

在庫管理は「仮押さえ→購入(在庫から席を減らす)」処理が必要。座席指定から決済するまでの間も、座席をロック(仮押さえ)しなければならない。つまり、排他制御が複雑。

放っておくと

排他制御の処理が正しく設計されず、以下の問題が発生
- 早いもの勝ちされて仮押さえしていたのに席が取れない。
- ロックを取りすぎてしまい顧客の期待する性能が出ない。
- ロック(仮押さえ)の解放漏れが起きる。

どんな先手を打つか

方式設計で排他制御を細かに定義する。
設計レビューで排他制御に関して詳細に確認する。
コードレビューだけでなくテストケースもレビューする。

高難度機能の洗い出し(4/4)

リーダー

言われてみれば確かにそうですね・・。何だか軍師のようです。

主人公

後手に回ると大変だからね。

みんなが平和に暮らせるように**先手を打っていく**のが、アーキテクトの仕事なんだよ。

- はじめに
- ITアーキテクトの主な業務

  あるオンラインチケッティングシステムの開発

  - 提案工程
  - 要件定義工程
  - **設計工程**
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

作業内容

リーダー

常に先の工程を考えるアーキチームが設計工程でやることは、製造工程の準備ですね。

そうだね。それから、業務アプリチームが設計を始めるから、そのサポートも始まるよ。

主人公

アーキチームの仕事の流れ

工程ごとの仕事

提案

- 顧客の要求確認
- 体制構築
- 開発プロセスの定義
- フレームワークの選定

要件定義

- 方式設計
- 非機能要件定義
- 移行方式の設計
- 設計の準備

設計

- 製造の準備
- 業務アプリチームのサポート

製造

- テストの準備
- リリース手順の検討
- 業務アプリチームのサポート

製造の準備

製造の準備

リーダー

製造の準備で主にやることは、
- 製造工程の作業プロセスの定義
- サンプルプログラムの作成
- 開発標準の策定
- ガイドの作成

だね。

主人公

製造工程の作業プロセスの定義(1/6)

リーダーが主人公に、製造工程の作業プロセス定義で気にすべきことは何か質問しています。

リーダー

製造工程の作業プロセスの定義で気をつけることは、設計工程の時と基本的に同じだよ。
さて、何に気をつけると思う？

主人公

製造工程の作業プロセスの定義(2/6)

リーダー

品質は当然として、後は・・・。
あまりに当たり前だけれど、設計書通りに作られてることですか？

主人公

そうだね。当たり前のことが当たり前にできてるって、意外に難しいし、ケアレスミスもあるよね。
設計書に書いてある処理が丸々抜けていたり、書いていない処理をプログラミングしていたりする人もいるからね。

製造工程の作業プロセスの定義(3/6)

リーダー

その為にチェックリストの活用やレビューを行うんですね。

主人公

うん。でも、人手にも限界があるから、自動化できるところは自動化する。Checkstyleなどの静的解析ツールの利用が代表的だね。
ルールはもちろんアーキが定義する。
そうすると、どういうプロセスになるかな？

※Checkstyle（静的コード解析ツール）
http://checkstyle.sourceforge.net/

製造工程の作業プロセスの定義(4/6)

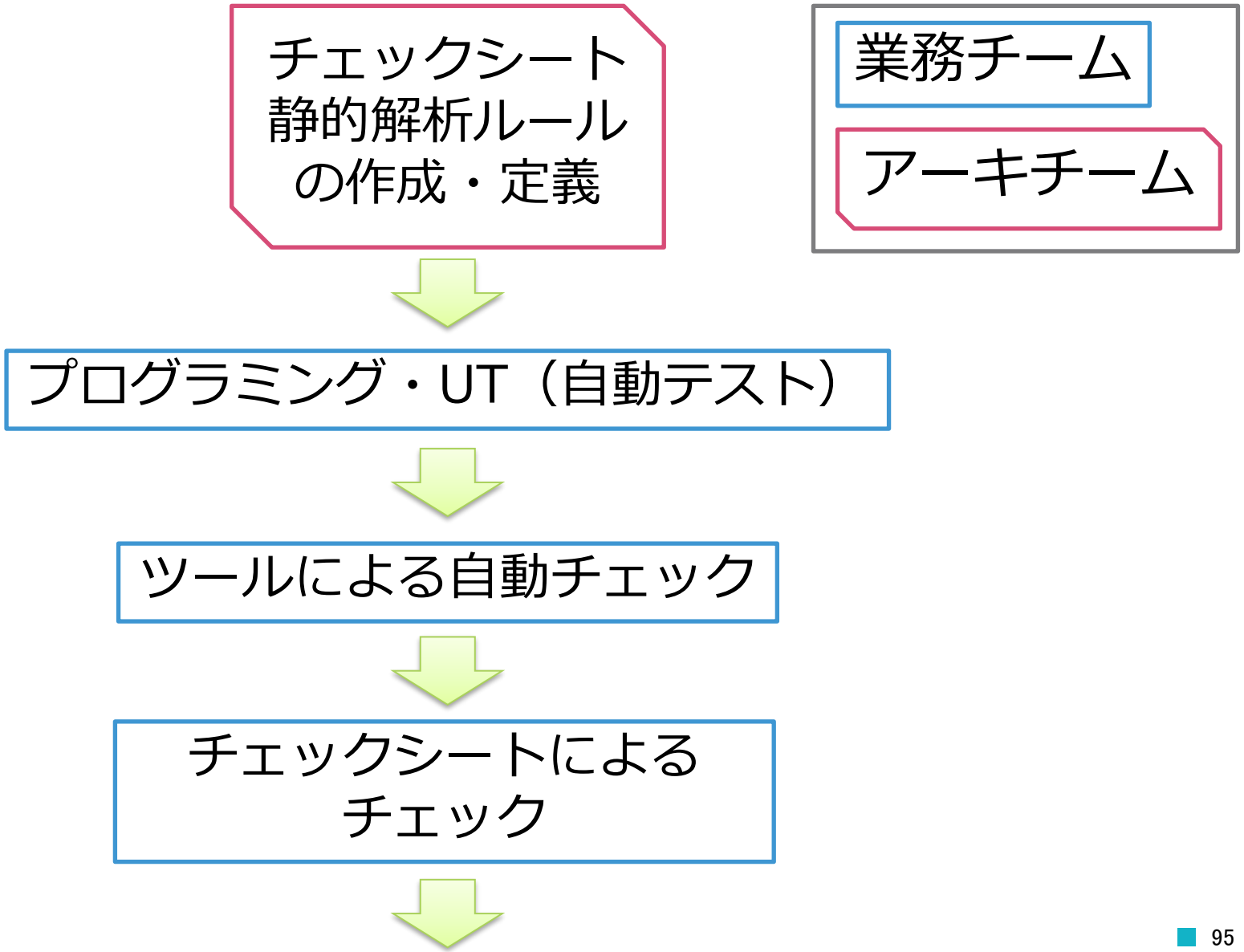
リーダー

こんな感じですか？
1. プログラミング・UT
2. ツールによるチェック
3. チェックリストによるチェック
4. レビュー

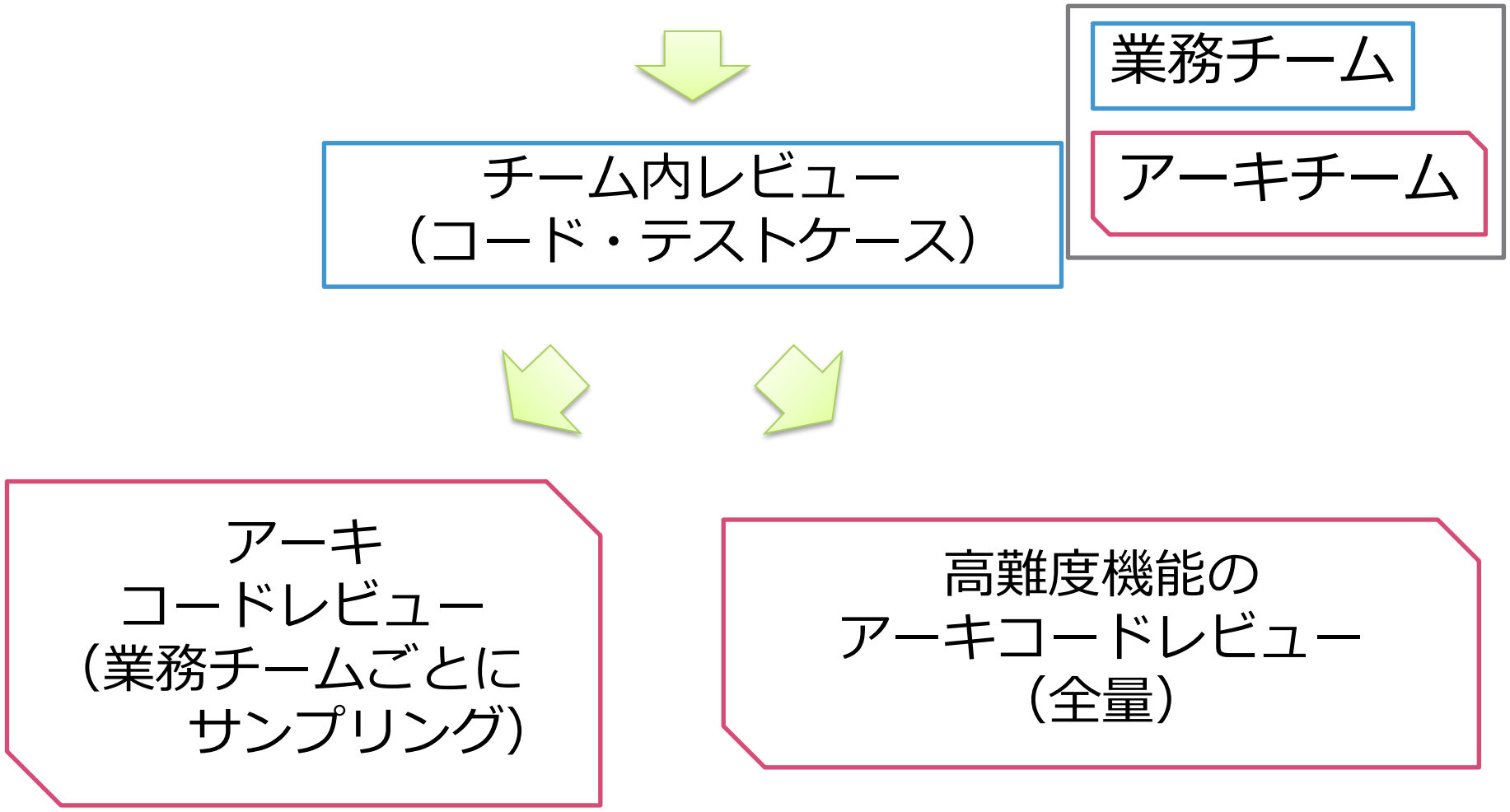
主人公

レビューは業務チーム内のレビューの事かな。設計工程同様、アーキレビューも実施するよ。
だから、次のようになる。

製造工程の作業プロセスの定義(5/6)



製造工程の作業プロセスの定義(6/6)



リーダー

ちなみに、チェックシートや静的解析ツールを使うのは、品質の担保だけが目的ではない。
我々が策定した**ルール**に皆がきちんと**従ってくれているかを確認**するための手段でもあるんだよ。
ルールは作るだけでなく、従ってくれているか確認することも重要なんだ。

主人公

チェックシートの例(1/2)

PG・UT作業の完了条件チェックリスト

取引ID：
取引名：

1. 目視によるチェック

No	チェック項目	チェック内容	チェック方法	評価指標	指標値
1	コーディング・単体テスト	開発で起きやすいコーディングミスがないこと。	「プログラマー向け成果物セルフチェックリスト」を使用してチェックする。	NG件数	0件
2	実装漏れ	外部設計書と内部設計書のすべての記載が漏れなく実装できていること。	外部設計書と内部設計書に対して、ソースコードに実装済みの箇所を塗りつぶすことでチェックする。	塗りつぶされていない箇所	0個

2. ツールによるチェック

No	チェック項目	チェック内容	チェック方法	評価指標	指標値
1	TODOコメント	ソースコードと設定ファイルにTODOコメントが残っていないこと。	ソースコードと設定ファイルにTODOコメントが残っていないことをチェックする。 ・Java ・jsp, sql, xml, config, properties, js	残件数	0件
2	Javaソースコード	一般的に推奨されるJavaソースコードの記述方法に違反していないこと。	静的解析ツールやIDE（統合開発環境）の機能でチェックする。	エラー数・警告数	0件
3	Javaコンパイル	Javaソースコードがコンパイルできること。	JDKのコンパイル機能やIDEの機能を使用してチェックする。	エラー数・警告数	0件
4	@Ignoreアノテーション （@Ignoreは単体テスト実行を無視するJUnitのアノテーション）	@Ignoreアノテーションを使用していないこと。	@Ignoreの付与されたテストが残っていないことをチェックする。	使用箇所	0件

Fintan > PG・UT作業の完了条件チェックリスト
https://fintan.jp/?p=1367

チェックシートの例(2/2)

プログラマー向け成果物セルフチェックリスト

No	大分類	小分類	チェック項目	チェック内容	Nablarch依存有無	補足説明	処理方式毎のチェック有無		
							画面	REST	バッチ
1	共通	全般	ファイルの改行コード	・ファイルの改行コードがPJ指定の改行コードに統一されていること。		■例 Java、SQL、JSP、Shell、configなどのファイルすべてを共通でチェックする。	●	●	●
2	共通	全般	ファイルの文字コード	・ファイルの文字コードがPJ指定の文字コードに統一されていること。		■例 Java、SQL、JSP、Shell、configなどのファイルすべてを共通でチェックする。	●	●	●

3	Java	全般	"" or Nullの判定	・"" or Nullの判定には、Nablarchが提供する以下のAPIを使うこと。
				・値が設定されているかの確認
				⇒StringUtil#hasValue
				・値が未設定かの確認
				⇒StringUtil#isNullOrEmpty
				・無用な否定演算子を使わないこと。
				NG例：if(!StringUtil.isNullOrEmpty(hoge))
				この場合はStringUtil#hasValueを使うこと。
					●	■値が設定されていることを確認する場合
						誤：
						if (!"".equeal(value)) {}
						if (value != null) {}
						if (!StringUtil.isNullOrEmpty(value)) {}
						
						正：
						if (StringUtil.hasValue()) {}
						
						■値が未設定であることを確認する場合
						誤：
						if ("".equeal(value)) {}
						if (value == null) {}
						if (!StringUtil.hasValue(value)) {}
						
						正：
						if (StringUtil.isNullOrEmpty(value)) {}
							●	●	●

Fintan > プログラマー向け成果物セルフチェックリスト
https://fintan.jp/?p=1369

サンプルプログラムの作成(1/3)

リーダーはサンプルプログラム作成の役割分担を伝えましたが、主人公は納得がいかない様子です。

リーダー

選定したFWでシステムを構築できるかの確認と、業務アプリチームに配る参考用にサンプルプログラムを作成しよう。

業務アプリチームに、サンプル設計書や、ガイド類に従って、製造してもらおうと思ってるんだ。

主人公

アーキチームは何をするんですか。

QA対応や、コードレビューだね。

サンプルプログラムの作成(2/3)

リーダー

なぜ、規約やガイドを知らない業務アプリチームにサンプルプログラムの作成を依頼するのですか？アーキチームで作ってしまえば、QA対応やレビューの手間もかからないし、方式の実現性も確認できるから、効率的のように思うのですが・・・

主人公

そうしてしまうと、次のようなことが確認できないんだよ

サンプルプログラムの作成(3/3)

- 製造工程に向けて作成した成果物に過不足はないか
  - FWを知らなくても、ガイドを頼りに製造できるか。
  - 設計書の記載内容は足りているか。
- 実装上の高難度箇所はどこか

リーダー

製造に向けて我々が準備した成果物の品質を、製造工程で確認する（初めて使う）のでは遅すぎるんだ。
先行して**「普通の」**業務チームに**「実際の」**製造と同じように作ってもらうことで、アーキチームが作成した各種成果物の**品質を確認**できるんだ。
それに、我々では気付けない実装上の高難度箇所もわかるしね。

アーキテクトの視点

リーダー

効率だけ考えていてはいけないんですね。

主人公

正確には**目先の**効率だけを考えていてはいけない、だね。
今、少し手間をかけておくことで、将来の手間を削ってPJがスムーズに進むようにするんだ。

なるほど。
「アーキテクトの視点」で考えるって、こういうことなんですね。

業務アプリチームのサポート

レビュー

リーダー

業務アプリチームにいたときに気になっていたんですけどなぜアーキレビューをやるんですか？

主人公

アーキチームは**技術的な品質を担保することも責務**だからね。品質を担保するための手段としてアーキレビューを行うんだ。

基本は業務アプリチーム毎のサンプリングレビューだ。全量レビューするのは現実的でないからね。
でも、特に品質に気を使う高難度の機能は全量みるよ。PMから言われた性能も、重要なポイントだ。

QA対応

- 多くは業務チームとの「質問票」（チケット）のやり取り。
- しかし、対面で打ち合わせした方が良い場合は、そうする。
- QA対応の結果、ガイドや標準を修正すべき場合もある。
  - その時は、修正して公開し、PJに報知する。

アーキテクトの視点

リーダー

QA対応は覚悟していましたが、それ以外にもやることがありますね。

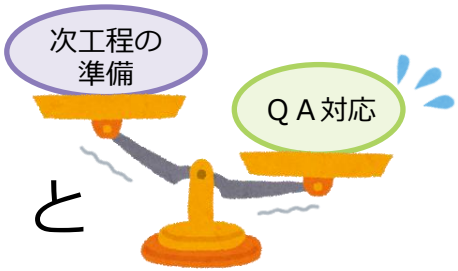
主人公

要件定義段階で実現性や性能の確保が難しいと判断した機能について設計・製造で**どうすれば品質を確保できる**か仕掛けを考えておくのがアーキテクトの視点なんだ。

ＩＴアーキテクト経験者談

苦労と喜び

- 苦労
  - 次工程の準備とＱＡ対応は、どちらも実施しないとＰＪに遅延が発生する可能性がありますが両立するのは結構大変。
  - 設計書の書き方のQAなどを見ていると「前工程でもう少し練っておくのだった」と反省することがある。



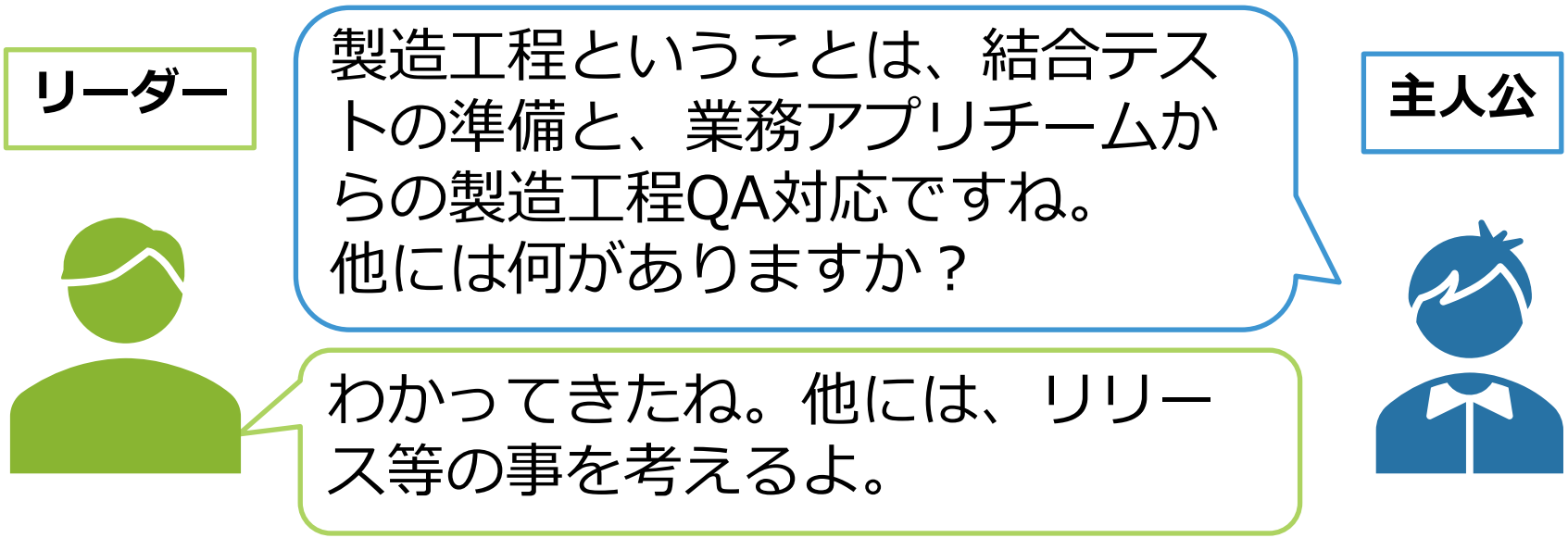
- 喜び
  - 自分が考えた方式によって、PJ全体の生産性や品質を大きく向上させることができたときに達成感を感じた。
  - 自分の知識やノウハウ等を用いて、相手の課題を解決できた時や自分が考えたプロセス等の方向に従って、PJメンバの皆が動き始めた時に面白みを感じた。

- はじめに
- ITアーキテクトの主な業務

  あるオンラインチケッティングシステムの開発

  - 提案工程
  - 要件定義工程
  - 設計工程
  - **製造工程**
  - 結合テスト工程、システムテスト工程、移行展開
- ITアーキテクトへのキャリアパス

作業内容



アーキチームの仕事の流れ

工程ごとの仕事

提案

- 顧客の要求確認
- 体制構築
- 開発プロセスの定義
- フレームワークの選定

要件定義

- 方式設計
- 非機能要件定義
- 移行方式の設計
- 設計の準備

設計

- 製造の準備
- 業務アプリチームのサポート

製造

- テストの準備
- リリース手順の検討
- 業務アプリチームのサポート

テストの準備

結合テストの準備 (1/4)

リーダー

結合テストの準備というと、作業プロセスの定義やガイドの作成は当然として、他には何があるんでしょうか。

テスト環境構築にも関わるんだ。

えっ、全てインフラチームの作業ではないんですか？

主人公

結合テストの準備 (2/4)

リーダー

例えば、準備するテスト環境の数を決めたりする。
業務要件や採用する処理方式の数で、必要な環境数も変わってくる。
それらを両方押さえているのはアーキチームだからね。

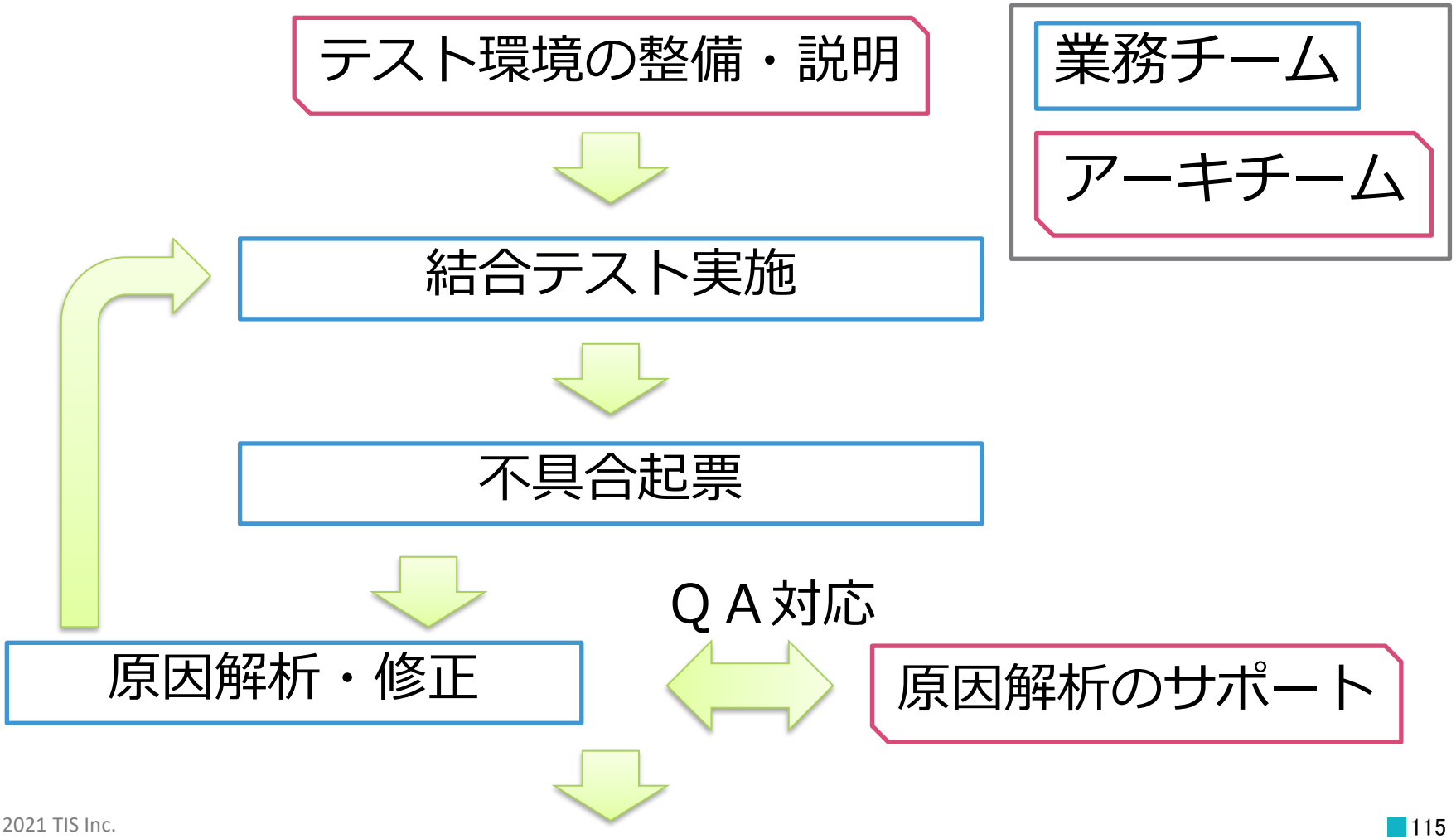
主人公

アプリを見ているアーキチームだから、インフラに関わることはないと思っていました。

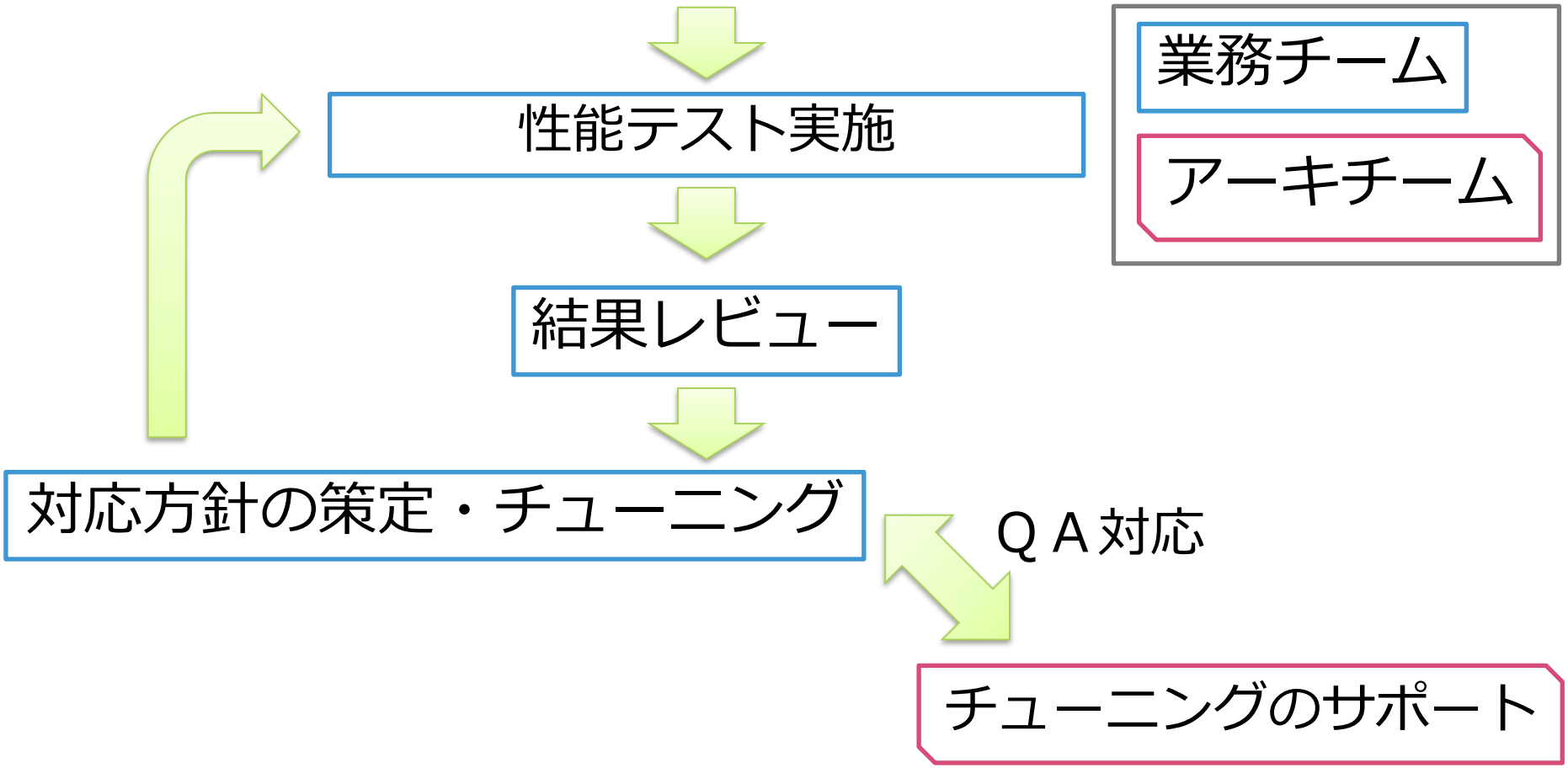
いやいや、PJの順調な進行に必要ならば、色々なチームと話をする。
以前、要員調整もやったじゃない。

結合テストの準備 (3/4)

結合テストの作業プロセスは、検討の結果以下のようになりました。



結合テストの準備 (4/4)



リリース手順の検討

リリース手順の検討(1/3)

リーダー

結合テストからは、各業務チームが製造したプログラムをまとめ、1つのシステムとしてテスト環境にリリースする。その仕掛けを考えよう。

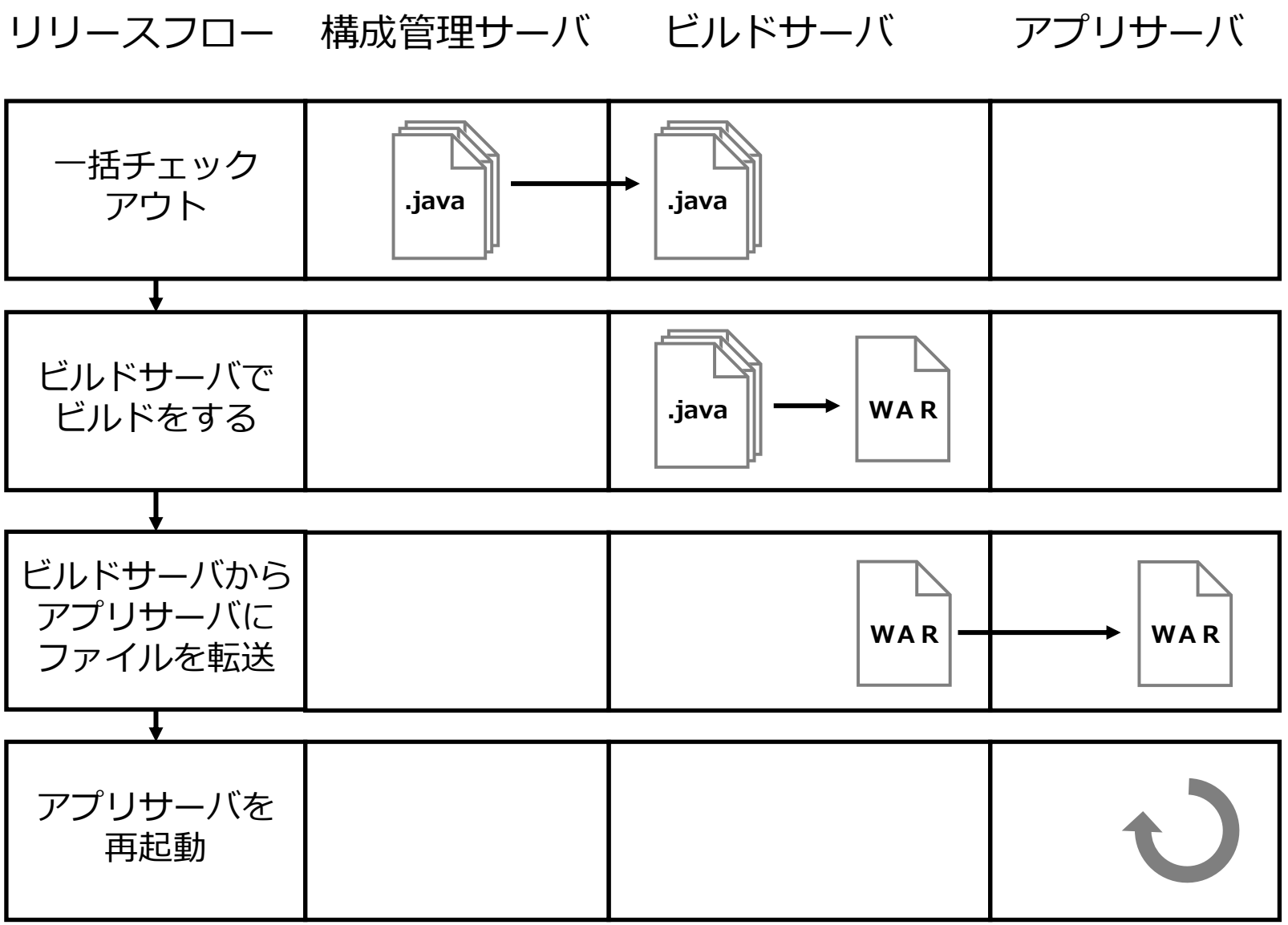
主人公

ソースコードは1カ所で管理しているから、そこから取り出してビルドすればよいですね。

テスト環境では、出来上がったアプリを、SFTPでアプリケーションサーバに送って、デプロイすればよさそうですね。

主人公の考えた仕掛けをもとに、テスト環境へのリリース手順は次のようになりました。

リリース手順の検討(2/3)



リリース手順の検討(3/3)

リーダー

あと、本番環境へのリリース方法も設計しなきゃね。

主人公

ええ。皆さんが作ったソースをコンパイルしてシステムに仕上げ、サーバに乗せて動かす手順を考えるというのは、とても重要な仕組みづくりを経験させてもらってる感じがします。

うん。システムを実際に稼働させるための仕組みだからね。
それがアーキの仕事の醍醐味でもあるんだ。

- はじめに
- ITアーキテクトの主な業務

  あるオンラインチケッティングシステムの開発
  - 提案工程
  - 要件定義工程
  - 設計工程
  - 製造工程
  - **結合テスト工程、システムテスト工程、移行展開**
- ITアーキテクトへのキャリアパス

アーキチームの仕事の流れ

工程ごとの仕事

結合テスト

- テストの準備
- 業務アプリチームのサポート

システムテスト

- 非機能テストの実施
- 業務アプリチームのサポート

移行展開

- 移行のサポート

作業内容(1/2)

リーダー

プロジェクトも終盤に差し掛かってきましたね。

主人公

そうだね。このあたりになると、各工程でやることは大体似てきて、課題対応や業務アプリチームのサポートの比重が大きくなってくる。今までと比べると、難易度やスピード感が上がってくるよ。

作業内容(2/2)

リーダー

テストの準備は前工程でやったことと基本的には変わらない。

主人公

業務アプリチームのサポートは、結合テストの不具合解決や、性能問題解決の支援がメインですね。

ＩＴアーキテクト経験者談

苦労と喜び

- 苦労

  課題が高度化してくるので、対応するにもそれなりの技術力が要求されるようになる。

- 喜び

  システムが形を見せはじめる(複数機能が連動して動き始めたりする）ので、自分が設計に関わった方式の通りにシステムが動くのを見るのは感慨深い。一つの機能やサブシステムを見ていた頃では味わえない達成感がある。

アーキチームの仕事の流れ

工程ごとの仕事

結合テスト

- テストの準備
- 業務アプリチームのサポート

システムテスト

- 非機能テストの実施
- 業務アプリチームのサポート

移行展開

- 移行のサポート

作業内容

リーダー

ここまでくると、業務チームのサポートがメインになってきますね。
解決すると、感謝されることが多くて、なんだかうれしくなっちゃいました。直接、感謝の言葉を聞くことはあまりないから、やる気が上がります。

主人公

そうだね。そういうやりがいもあるよね。
でも、残課題の解決も忘れてはいけないよ。
あと、非機能テストの実施もね。

アーキチームの仕事の流れ

工程ごとの仕事

結合テスト

- テストの準備
- 業務アプリチームのサポート

システムテスト

- 非機能テストの実施
- 業務アプリチームのサポート

移行展開

- 移行のサポート

移行のサポート

カットオーバー当日

主人公：とうとうこの日がやってきましたね。

リーダー：そうだね。全体進行はPMが担当している。我々は、動作検証などで何か不具合があった場合に対応できるよう、構えておこう。

何かトラブルが起きない限り、カットオーバー当日にアーキチームが大活躍するようなことはないと思うけど、最後までプロジェクトを支えていくよ。

[ご参考]ところで・・・

リーダー

FWのテストは？と思った方もいるでしょう。当然実施しています。

主人公

今回はOSSをベースにカスタマイズしました。この為、主にカスタマイズ部分に関して各工程で業務アプリチームと同様、テストを行っています。

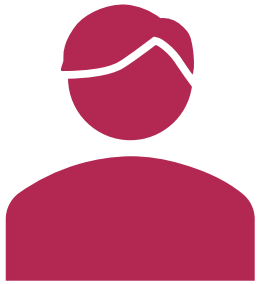
また、FWのテストは、「方式の担保」でもあります。設計した処理方式でシステムが成立することの検証でもあるんです。

～エピローグ～

本番カットオーバー数週間後…

本番移行数週間後、主人公はPMに会いました。

PM



特に大きな問題もなく、順調に稼働してるよ。
リリースの手順書など、作成者に君の名前の入った文書が今でも使われてるよ！

主人公

そうですか！なんだか嬉しいです！

主人公は今回の仕事ぶりが認められ、
次のPJでもアーキチームの一員として参画することに。

アーキチームを経験して気づいたこと

APSとして自分が作った機能が使われ続けていくことは当然嬉しい。でも、ITAとしてPJ全体のアーキテクチャを見て大きなものを動かすことにやりがいを感じた。
また、先輩たちの「先回り」の仕方も凄かったな。
あと、今までの経験が意外に役に立った。
今までは敷かれたレールを走っていたけれど、その経験があったからレールの敷き方も理解できたもの。敷かれたレールの意味が少しわかった気がする。

- はじめに
- ITアーキテクトの主な業務

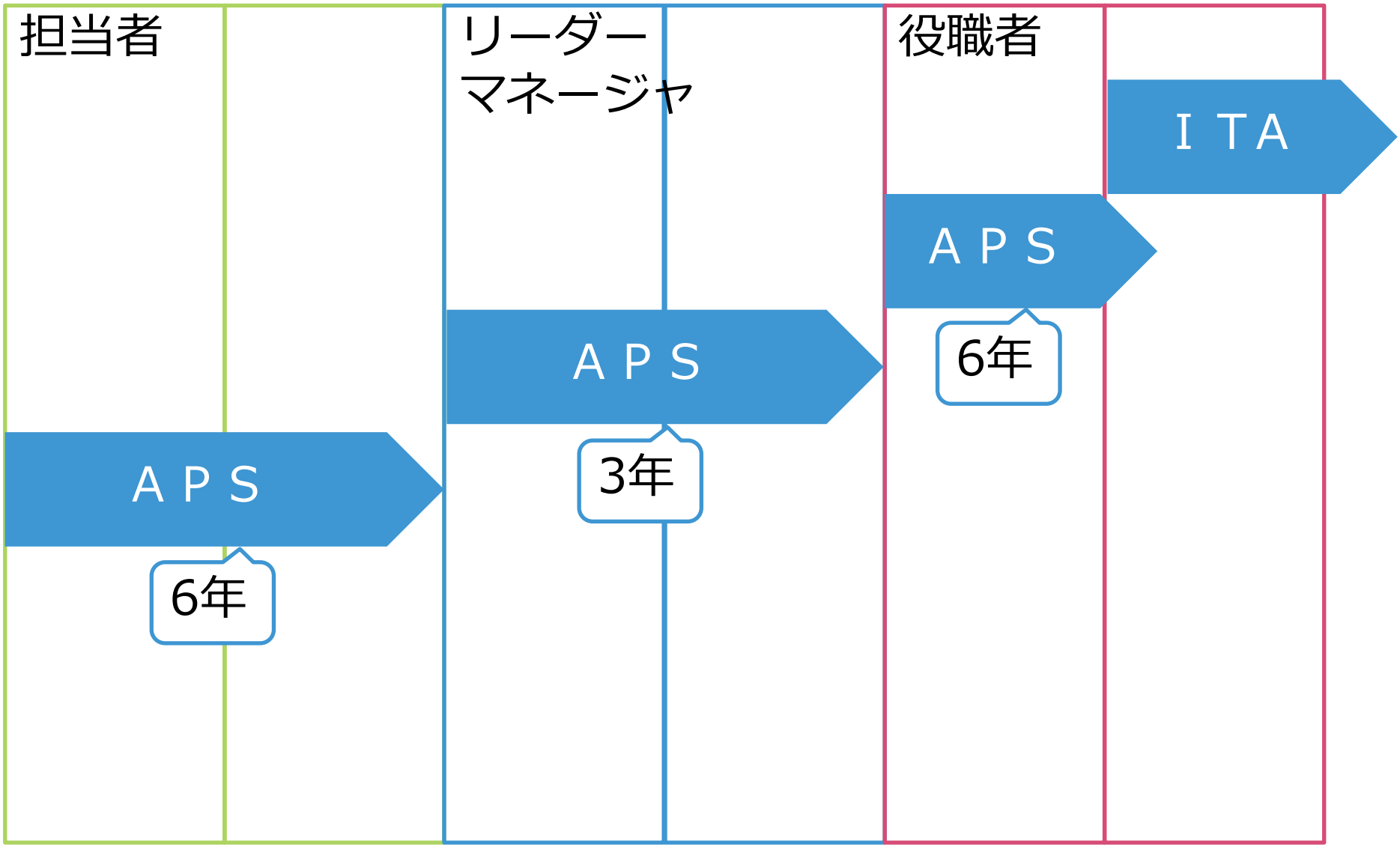
  あるオンラインチケッティングシステムの開発
  - 提案工程
  - 要件定義工程
  - 設計工程
  - 製造工程
  - 結合テスト工程、システムテスト工程、移行展開
- **ITアーキテクトへのキャリアパス**

ITアーキテクトへのキャリアパス

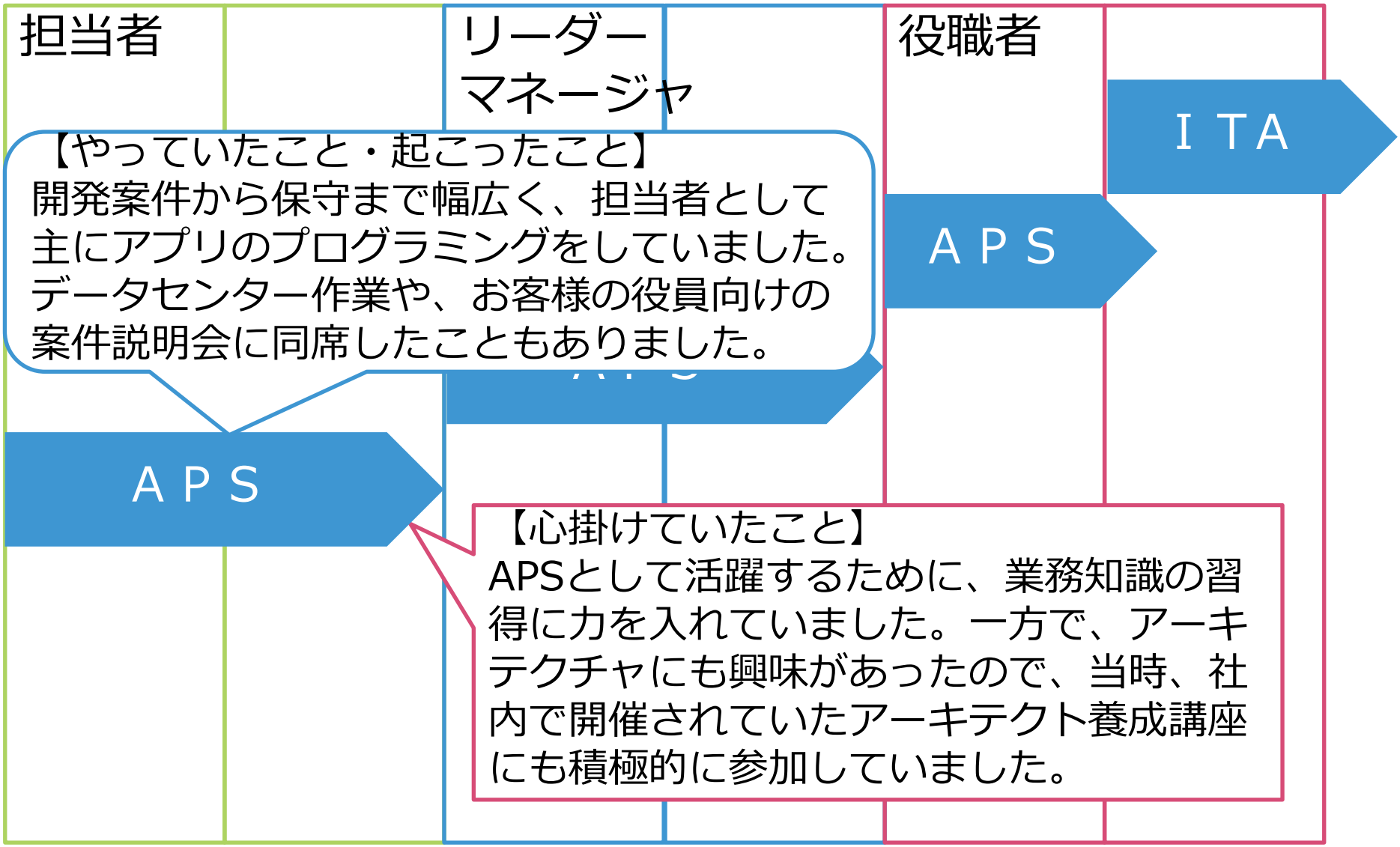
ITアーキテクトへのキャリアパスの事例として、業務領域のスペシャリスト（APS）から、ITアーキテクトへ進んだ例を紹介します。

役割ごとに何をやっていて、どのようなことを心掛けていたかお話しします。

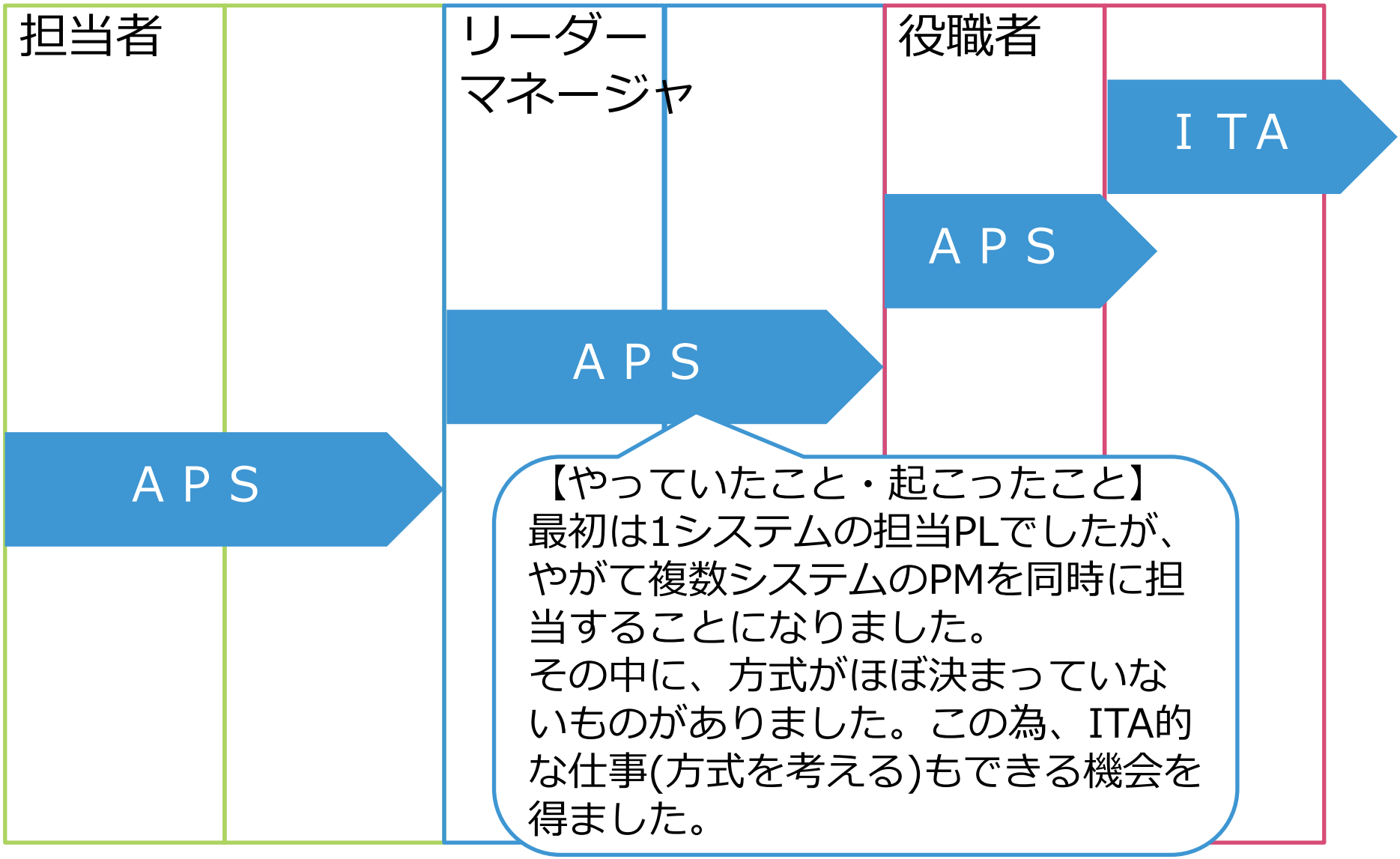
あるＩＴアーキテクトのキャリア(1/5)



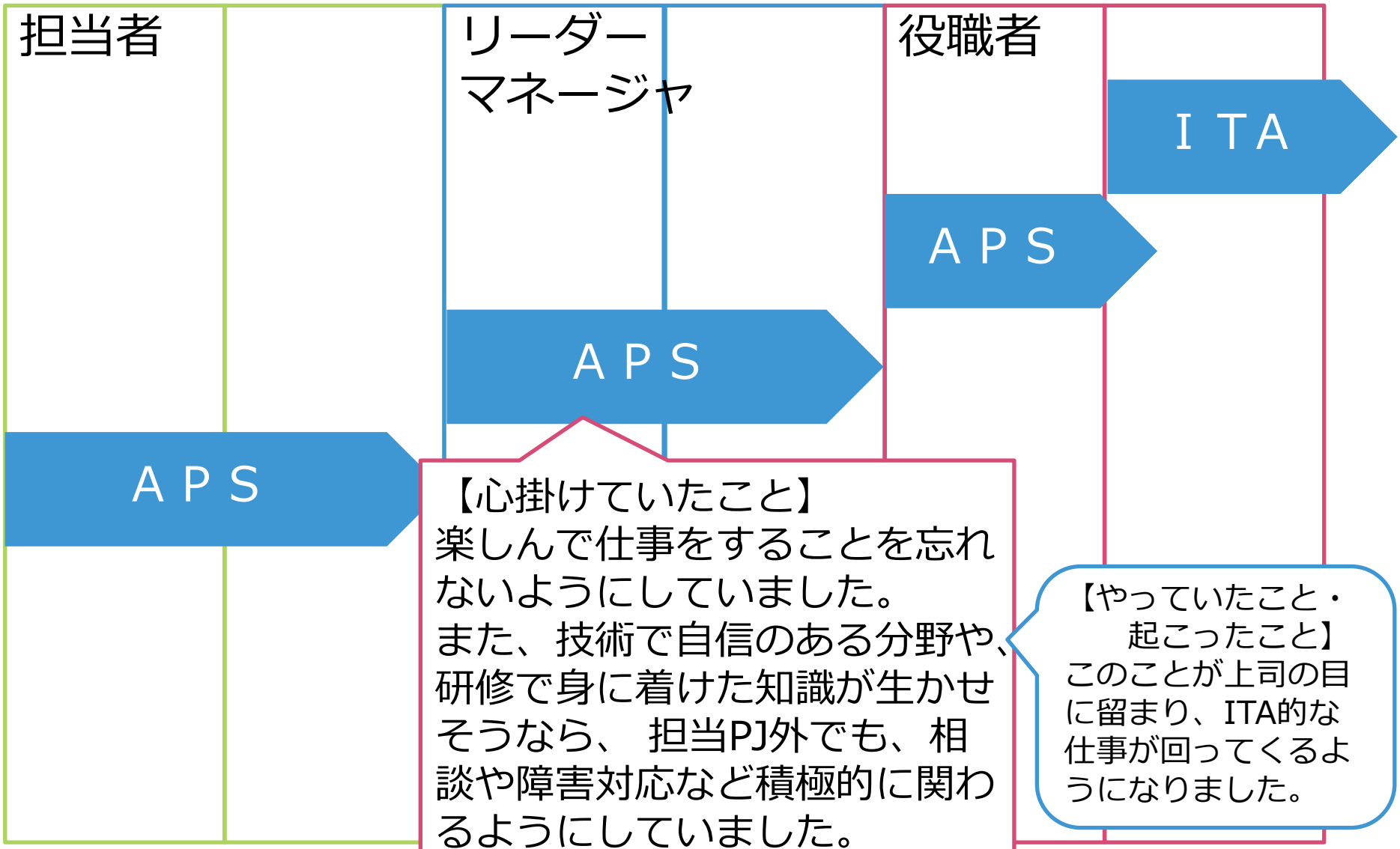
あるＩＴアーキテクトのキャリア(2/5)



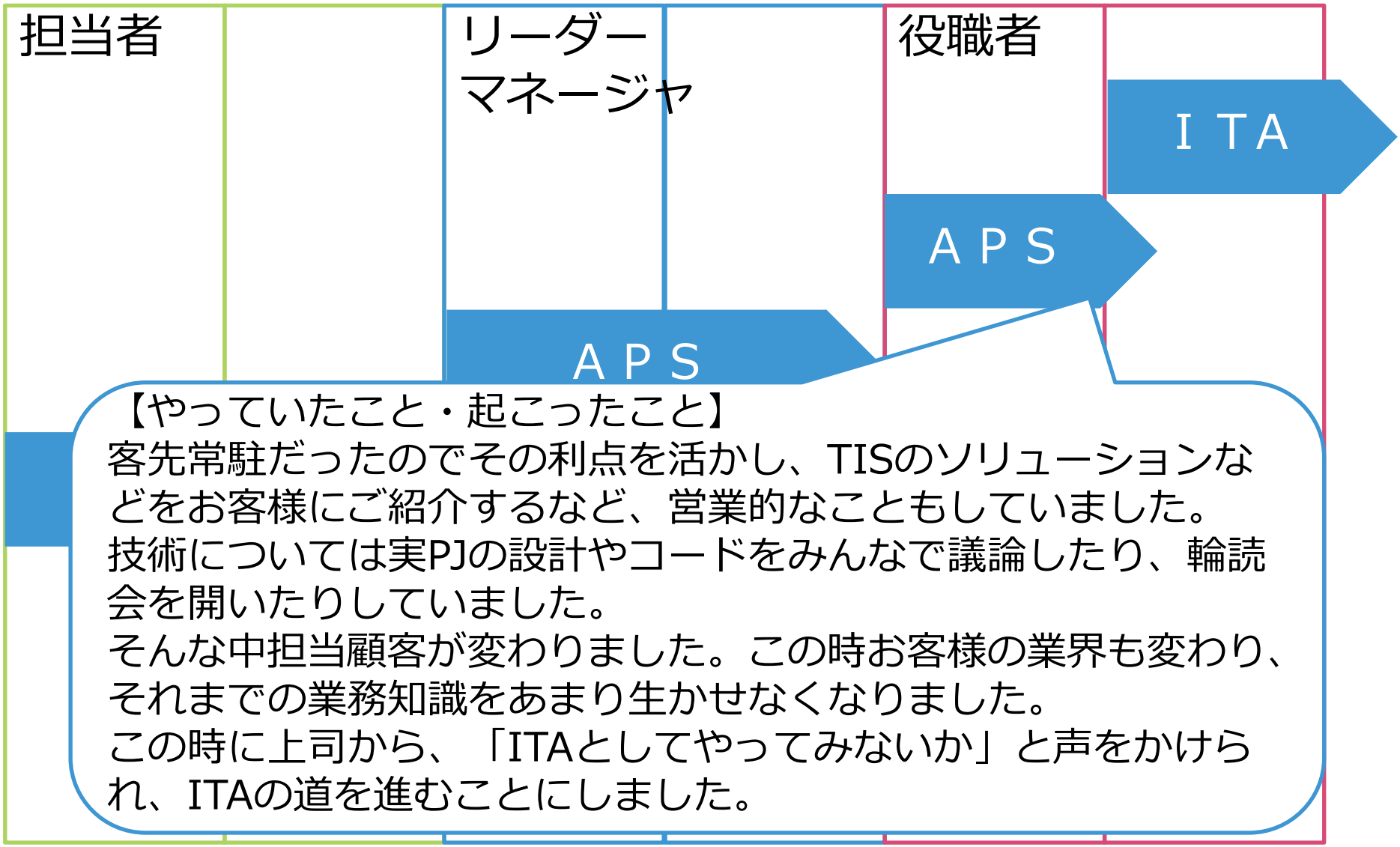
あるＩＴアーキテクトのキャリア(3/5)



あるＩＴアーキテクトのキャリア(4/5)



あるＩＴアーキテクトのキャリア(5/5)



【やっていたこと・起こったこと】
客先常駐だったのでその利点を活かし、TISのソリューションなどをお客様にご紹介するなど、営業的なこともしていました。
技術については実PJの設計やコードをみんなで議論したり、輪読会を開いたりしていました。
そんな中担当顧客が変わりました。この時お客様の業界も変わり、それまでの業務知識をあまり生かせなくなりました。
この時に上司から、「ITAとしてやってみないか」と声をかけられ、ITAの道を進むことにしました。

あるＩＴアーキテクトのキャリア(まとめ)

- 技術をアピールする。
  - 技術を習得することは大切。しかし、自分の知識が生かせそうなら、 PJ外の話題でも積極的に関わったり、草の根活動をしたりする。そうやって、技術に興味がある、知識があることをアピールすることも大切。
    - 「技術を持っている事をアピール→上司の目に留まりITA向きの仕事がくる→成果を出す→ITA向きの仕事があつまる」の好循環ができる。
- 技術で得意な分野を持つ。
  - まずは職種にこだわらず技術で得意な分野をもつ。そして、それを土台に好きな方向に進んでいく。

ITで、社会の願い叶えよう。

TIS INTEC
Group

付録

本文で参照した資料のリンク

- ウォーターフォール開発における役割分担シート
  - https://fintan.jp/?p=1326
- アプリケーション方式設計書サンプル
  - https://fintan.jp/?p=1323
- 非機能要求グレード
  - https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html
- 設計書フォーマット、コーディング規約（Nablarch開発標準）
  - https://fintan.jp/?p=1145
- PG・UT作業の完了条件チェックリスト
  - https://fintan.jp/?p=1367
- プログラマー向け成果物セルフチェックリスト
  - https://fintan.jp/?p=1369