オンラインイベント

失敗したWebプロダクト開発を復活に導いたClaris FileMakerの魅力

株式会社ライジングサン・システムコンサルティング
岩佐 和紀

セッション ID：U-03



RisingSun
System Consulting

株式会社ライジングサン・システムコンサルティング

代表取締役社長

岩佐 和紀

@kazukiiwasa





ITエンジニアとしての経歴

01
スタート

敷島製パン株式会社入社。情報システム部門に配属。

富士通メインフレームのCOBOLプログラマ・社内SEとしてキャリアをスタート

６年

02
SES

地方のソフトウェア開発会社に転職。

某国営企業の民営化PJに、多重下請けの最下位レイヤープログラマとして超巨大プロジェクトに派遣(ドナドナ)される

1年

03
外資系

SGS Far East
↓
BMWジャパン
ファイルナス
↓
DOLCE&GABBANA
ジャパン

…と３社の外資系企業を、社内SEとして転々とする

５年

04
ベンチャー

株式会社ベストブライダル（現ツカダ・グローバルホールディング）に社内SEとして入社。

基幹システムのPJリーダーとしてマザーズ上場・東証１部指定替えなどを経験

2年

05
独立

2006年、娘の誕生を期に株式会社ライジングサン・システムコンサルティングを設立

2010年から本格的にFileMakerプラットフォームを活用した事業をスタート

16年



CodeZineに連載した技術系記事

https://codezine.jp/author/2017

01

DBエンジン性能

実際に「どこまでのシステム」が構築可能？FileMaker Serverの強力なDBエンジン性能にフォーカス

02

外部システム連携

FileMakerプラットフォームと外部システムを連携する3つの手法

03

JavaScript連携

FileMakerとJavaScriptのマッシュアップでリッチなユーザインタフェースを実装しよう

04

ワンソース マルチデバイス

FileMakerがもつワンソース・マルチデバイスの強み

05

実行環境

FileMakerプラットフォームの豊富な実行環境

CodeZine人気記事ランキングで１位

ローコードはITプロフェッショナルを幸せにするのか？

FileMakerエンジニアに学ぶ、ローコード開発の本質

https://codezine.jp/article/detail/14843





事業内容

01

内製化支援

Claris FileMaker を用いた、業務システムの内製化を支援

2016年よりサービス開始

Zoomなどで、画面共有をしながらペアプログラミングで開発支援をするライトなサービスから、プロジェクト期間１年、基幹システムの構築と社内開発者の育成まで、様々な規模の事例がある。

02

スキル移行支援

Microsoft .net や Java、その他の一般汎用言語で開発してきたITエンジニアさんを、FileMakerプラットフォームの開発者としてスキルコンバートするプロセスを支援する

全国200店舗以上で運営するデイサービス事業の店舗運営システム等の構築を支援した実績がある

03

プロトタイプ開発

企画段階のWebサービスやパッケージシステムのプロトタイプを、FileMakerの高い生産性を活かして素早く開発することで、新規事業の立ち上げをサポートするサービス。

将来的にWeb系を始めとした、他の開発言語での焼き直しが前提だが、そのままFileMakerでサービスインするケースも有る。



事業内容

01

内製化支援

Claris FileMaker を用いた、業務システムの内製化を支援

2016年よりサービス開始

Zoomなどで、画面共有をしながらペアプログラミングで開発支援をするライトなサービスから、プロジェクト期間１年、基幹システムの構築と社内開発者の育成まで、様々な規模の事例がある。

02

スキル移行支援

Microsoft .net や Java、その他の一般汎用言語で開発してきたITエンジニアさんを、FileMakerプラットフォームの開発者としてスキルコンバートするプロセスを支援する

全国200店舗以上で運営するデイサービス事業の店舗運営システム等の構築を支援した実績がある

03

プロトタイプ開発

企画段階のWebサービスやパッケージシステムのプロトタイプを、FileMakerの高い生産性を活かして素早く開発することで、新規事業の立ち上げをサポートするサービス。

将来的にWeb系を始めとした、他の開発言語での焼き直しが前提だが、そのままFileMakerでサービスインするケースも有る。



事業内容

01

内製化支援

Claris FileMaker を用いた、業務システムの内製化を支援

2016年よりサービス開始

Zoomなどで、画面共有をしながらペアプログラミングで開発支援をするライトなサービスから、プロジェクト期間１年、基幹システムの構築と社内開発者の育成まで、様々な規模の事例がある。

02

スキル移行支援

Microsoft .net や Java、その他の一般汎用言語で開発してきたITエンジニアさんを、FileMakerプラットフォームの開発者としてスキルコンバートするプロセスを支援する

全国200店舗以上で運営するデイサービス事業の店舗運営システム等の構築を支援した実績がある

03

プロトタイプ開発

企画段階のWebサービスやパッケージシステムのプロトタイプを、FileMakerの高い生産性を活かして素早く開発することで、新規事業の立ち上げをサポートするサービス。

将来的にWeb系を始めとした、他の開発言語での焼き直しが前提だが、そのままFileMakerでサービスインするケースも有る。

事業内容



01

内製化支援

Claris FileMaker を用いた、業務システムの内製化を支援

2016年よりサービス開始

Zoomなどで、画面共有をしながらペアプログラミングで開発支援をするライトなサービスから、プロジェクト期間１年、基幹システムの構築と社内開発者の育成まで、様々な規模の事例がある。

02

スキル移行支援

Microsoft .net や Java、その他の一般汎用言語で開発してきたITエンジニアさんを、FileMakerプラットフォームの開発者としてスキルコンバートするプロセスを支援する

全国200店舗以上で運営するデイサービス事業の店舗運営システム等の構築を支援した実績がある

03

プロトタイプ開発

企画段階のWebサービスやパッケージシステムのプロトタイプを、FileMakerの高い生産性を活かして素早く開発することで、新規事業の立ち上げをサポートするサービス。

将来的にWeb系を始めとした、他の開発言語での焼き直しが前提だが、そのままFileMakerでサービスインするケースも有る。



メインテーマ

失敗したWebプロダクト開発を
復活に導いた
ローコード開発プラットフォーム の魅力

ITプロフェッショナルが、ローコード開発プラットフォームを
学習する意義とメリット



プロダクトのデモ

医療機関（病院）経営に特化した
管理会計システム

プロダクトのデモ



JavaScriptとの連携が大幅に強化



- Ver19より**[Web ビューアで JavaScript を実行]**スクリプトステップの追加、及びWebビューアにおける **FileMaker クラス(FileMaker.PerformScript)**の追加で、FileMakerとJavaScriptの親和性が強化された

- Ver19.3 より、Windows環境におけるWebビューアのレンダリングエンジンが、**InternetExplorer互換からEdge互換**に変更。

- Mac/Windows問わず、**Webフロントエンドで用いられるリッチな操作性**を、FileMakerのソリューションにも、容易に組み込み可能となった。

JavaScriptのコード記述が**「重荷」**になっては意味がない…

JavaScript UI library and framework for speeding up web development

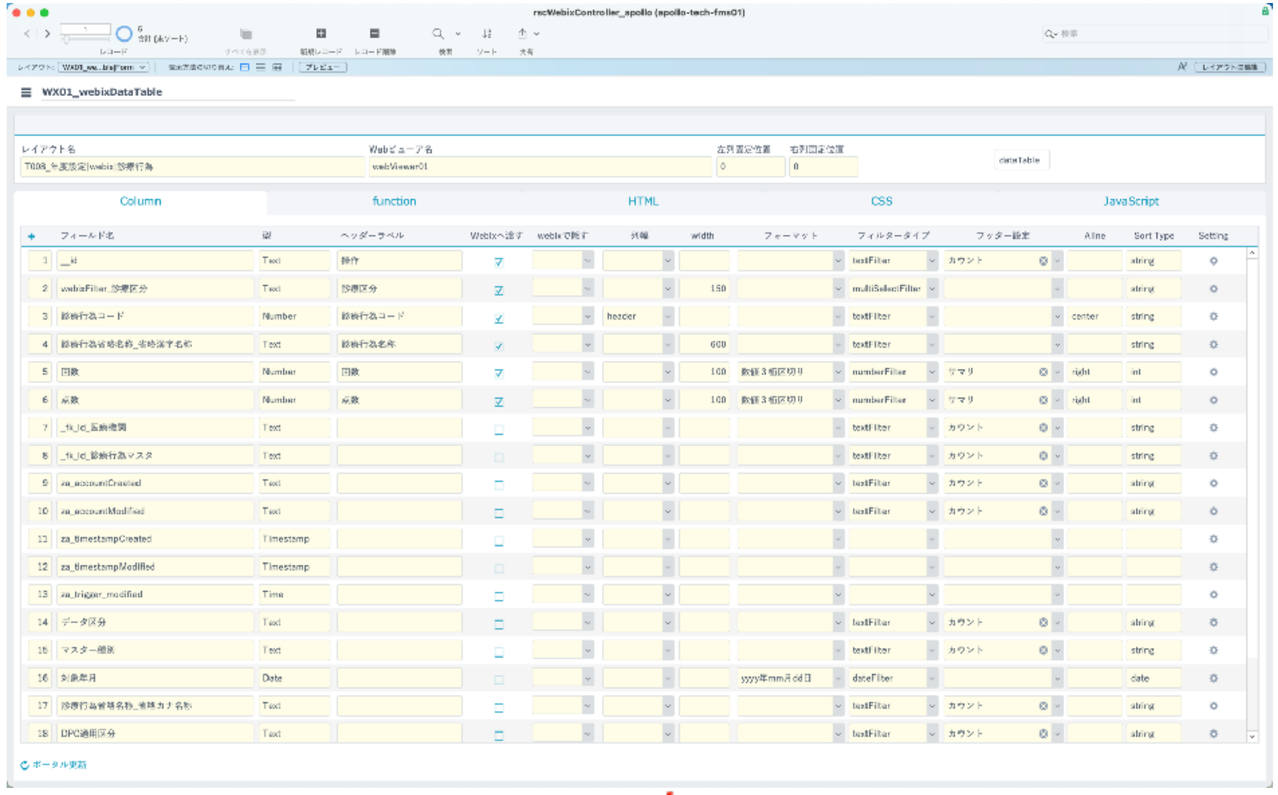
JS Framework for cross-platform web Apps development

102 UI widgets and feature-rich CSS / HTML5 JavaScript controls

Download Webix UI



JavaScriptのコードを自動生成するツール




```
function updateCellCss(obj, fieldName) {
  if (obj[fieldName] == 0) {
    // 数値がゼロの場合、CSSは適用しない
  } else if (obj[fieldName] < 0) {
    // 数値がマイナスの場合、CSSは適用しない
    grid.addCellCss(obj.id, fieldName, "chartjs-red-color", true);
  } else if (obj[fieldName] > 0) {
    // 数値がプラスの場合、CSSは適用しない
    grid.addCellCss(obj.id, fieldName, "chartjs-blue-color", true);
  }
}

function onClick_detail_button(e, id, node) {
  const item = grid.getItem(id.row);
  FileMaker.PerformScript(
    "T008_年度設定_診療行為_clickDetailButton",
    JSON.stringify(item)
  );
}
function render(param) {
  const targetFunction = function () {
    return function () {
      grid.clearAll()
      const res = {data: []}
      const obj = JSON.parse(param)
      const data = obj.response.data

      for (key in data) {
        const item = data[key]
        res.data[key] = item.fieldData
      }
      grid.parse(JSON.stringify(res), "json")
    }
  }
  executeFunction(targetFunction(param))
}
webix.i18n.setLocale("ja-JP");

const heightAadjust = 20;
webix.env.cdn =
  "https://app-webix-hub-cdn-extras.s3.ap-northeast-1.amazonaws.com";
webix.event(window, "resize", function () {
  grid.config.height = window.innerHeight - heightAadjust;
  grid.adjust();
```

HTML/CSS/JavaScritpのコード自動生成ツールを

FileMakerで開発



失敗したプロジェクトの概要

1.投資した金額

- イニシャルコスト：**約5,000万円**。主にハードウェアと開発ツールのライセンス等
- ランニングコスト：**年間約１億円**。主にエンジニアの人件費。最大17名のエンジニアが参加。全て内製。

2.開発プラットフォーム

- **ASP.NET + Microsoft SQL Server**
- ベタベタの業務系なので、言語としては.netかJavaの２択。技術者採用等の観点から.netを選択

3.開発手法

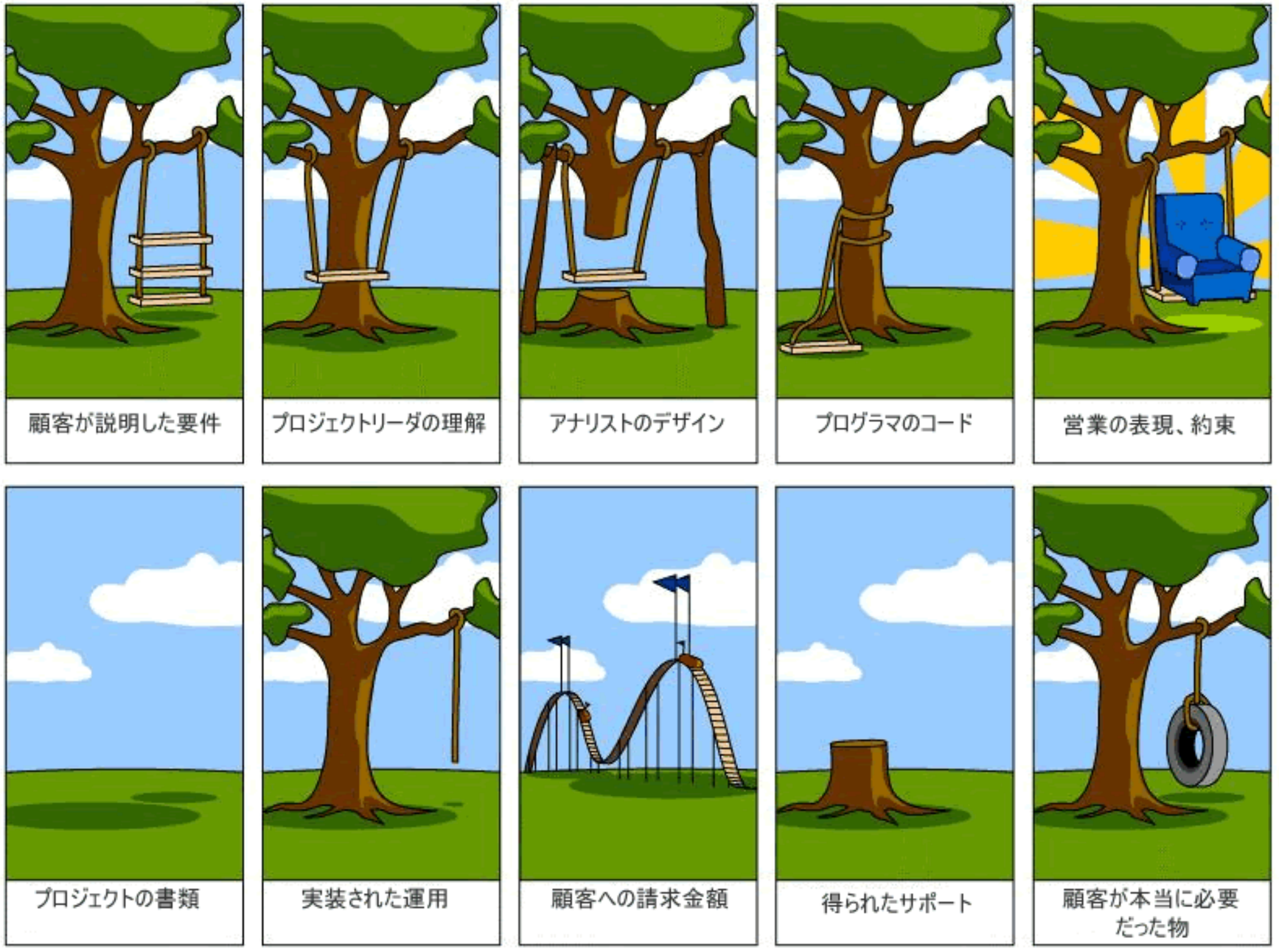
- 銀行やVC、そしてIT系監査法人との絡みもあって**ウォーターフォール**を採用することに…
- 事業者としてはアジャイルでやりたかったが、そもそも**アジャイル開発の経験者がいなかった**

ウォーターフォールモデルが失敗の元凶に…

ウォーターフォールモデルの不幸



- 新規ビジネス・新規プロダクトの開発なので、**既存業務は存在しない。**
- 現場にヒヤリングしてAs/Isモデルを可視化、そこから課題を抽出し、解決策を考えてTo/Beモデルを可視化…っという**いつもの定石が使えない。**
- それでもウォーターフォールモデルなので、仮にでも要件を定義しなければならない。結果的に**極めて曖昧な確度の要件定義書**を元に開発がスタート。
- 曖昧な要件定義書を元に、SEが設計し、プログラマがコーディングし、テスターがテストし…っと、**危険な状態での開発**が進んでいく…


顧客が説明した要件
プロジェクトリーダの理解
アナリストのデザイン
プログラマのコード
営業の表現、約束
プロジェクトの書類
実装された運用
顧客への請求金額
得られたサポート
顧客が本当に必要だった物

どうやって４ヶ月で完成させたのか？



- ベンチャーキャピタルや銀行が出資していた程の事業アイデアなので、**ビジネス的な潜在ニーズがある**ことは確かだった。
- 先のプロジェクトをクローズ後、発起人（後の弊社のお客様）さんは、別法人を登記して、**医療機関向けの経営コンサルティング事業**を開始
- コンサルティングレポートの作成は、PowerQuery等を用いた手作業で作成。１つの医療機関向け**レポートをつくるのに１週間の工数**が…
- レポート作成時に必要な各種データのクレンジング・前処理の自動化に**FileMaker**を使うことができないだろうか…

どうやって４ヶ月で完成させたのか？



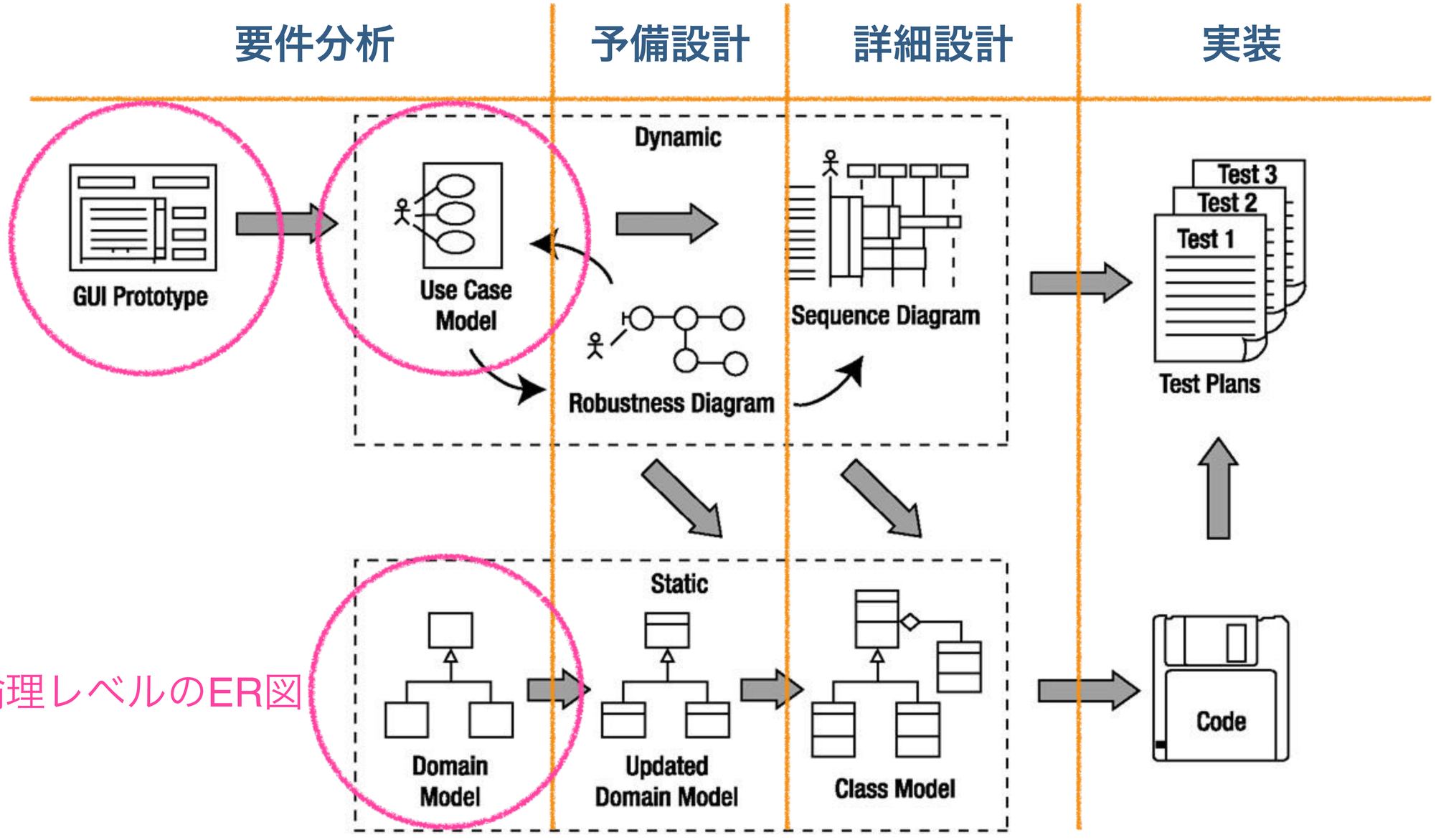
- FileMakerの学習を進めていく中で、**弊社のブログ記事**、そして**YouTube**動画に出会う。以降、弊社情報を教材に学習

- FileMaker がもつ潜在能力に可能性を感じ始める。もしかしたら**FileMaker でも以前作ろうとしたプロダクトが実装できる**のでは…？

- いくらFileMakerと言えど、非ITエンジニアが**プロダクトレベルのソフトウェアを実装することはとてもできない**…（特にJavaScript連携）

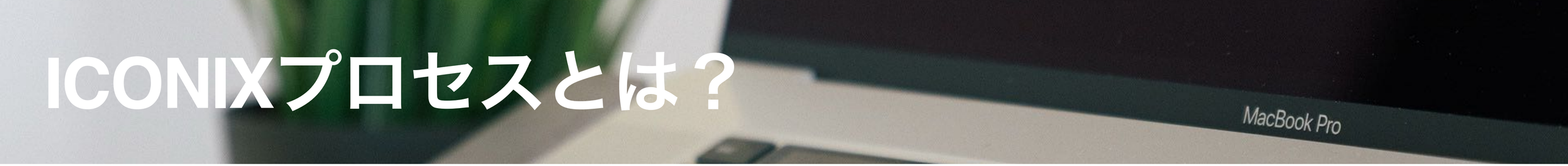
- **2021年５月**に弊社へお問い合わせ。**翌６月**にはプロトタイプシステム開発の契約を締結してプロジェクトをスタート

開発プロジェクトの進め方



論理レベルのER図

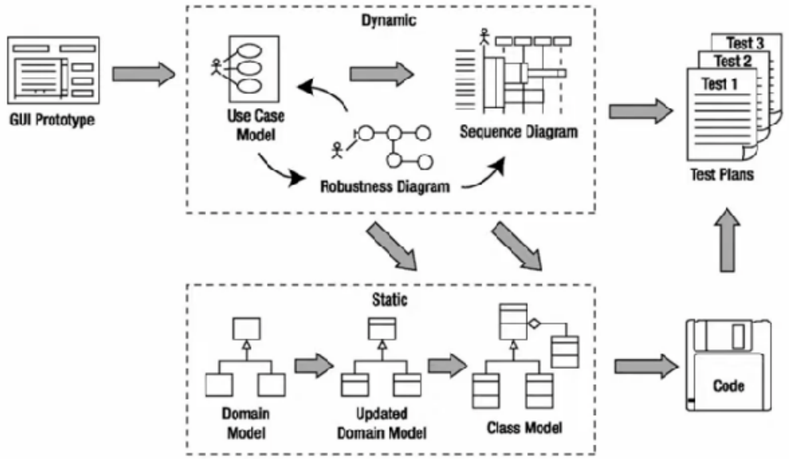
ICONIXプロセスとは？



Premium JP

ICONIXプロセス

ICONIX(アイコニクス)プロセスとは？



ICONIXは、ラショナル統一プロセス（RUP）やアジャイルソフトウェア開発よりも前から存在するソフトウェア開発方法論。RUPと同様にユースケース駆動であるが、RUPより軽量である。XPやアジャイルのアプローチとは異なり、ICONIXは十分な要求と設計のドキュメントを作成するのが特徴のひとつである。

ICONIXプロセス勉強会
株式会社ライジングサン・システムコンサルティング - 1/8

1. ICONIXプロセスVOL.1 :: ICONIXプロセスとは？ 6:47 — 株式会社ライジングサン・システムコ...
2. ICONIXプロセスVOL.2 :: なぜICONIXプロセスなのか？ 3:54 — 株式会社ライジングサン・システムコ...
3. ICONIXプロセスVOL.3 :: なぜBalsamiqMockupsが最適なツー... 9:45 — 株式会社ライジングサン・システムコ...
4. ICONIXプロセスVOL.04 :: ICONIXにおける要件分析とは？ 2:55 — 株式会社ライジングサン・システムコ...
5. ICONIXプロセス VOL.05 :: ドメインモデルを用いて用語を定義する 6:26 — 株式会社ライジングサン・システムコ...
6. ICONIXプロセス 6 :: 一味違うユースケースダイヤグラム 3:00 — 株式会社ライジングサン・システムコ...
7. ICONIXプロセス 7 :: GUIモックアップの重要性 4:13 — 株式会社ライジングサン・システムコ...
8. ICONIXプロセスVOL08 :: ロバストネス分析 8:04 — 株式会社ライジングサン・システムコ...

すべて　関連動画　提供: 株式会社ライジング

ICONIXプロセスVOL.1 :: ICONIXプロセスとは？

500 回視聴 • 2015/03/13

6　低評価　共有　オフライン　クリップ　保存



ローコード開発プラットフォーム Claris FileMaker でつくる基...
株式会社ライジングサン・システム...
487 回視聴 • 1 年前

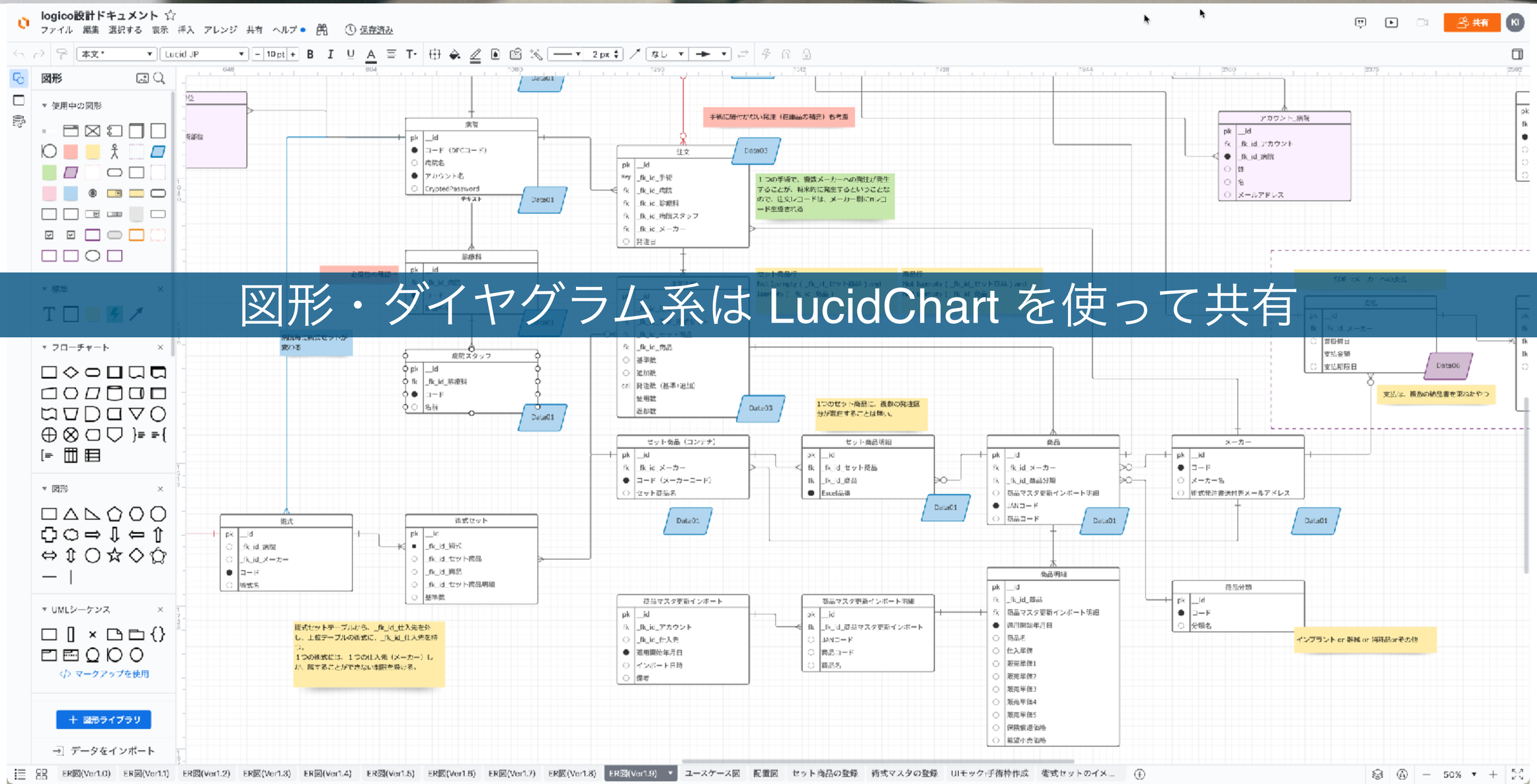
ICONIXプロセスとは？



https://www.youtube.com/watch?v=34ajTQs9z1U&list=PLVPVgIlnLiUKwjuyPCMiit4MwEoVP3i7p&index=1

開発プロジェクトの進め方

図形・ダイヤグラム系は LucidChart を使って共有

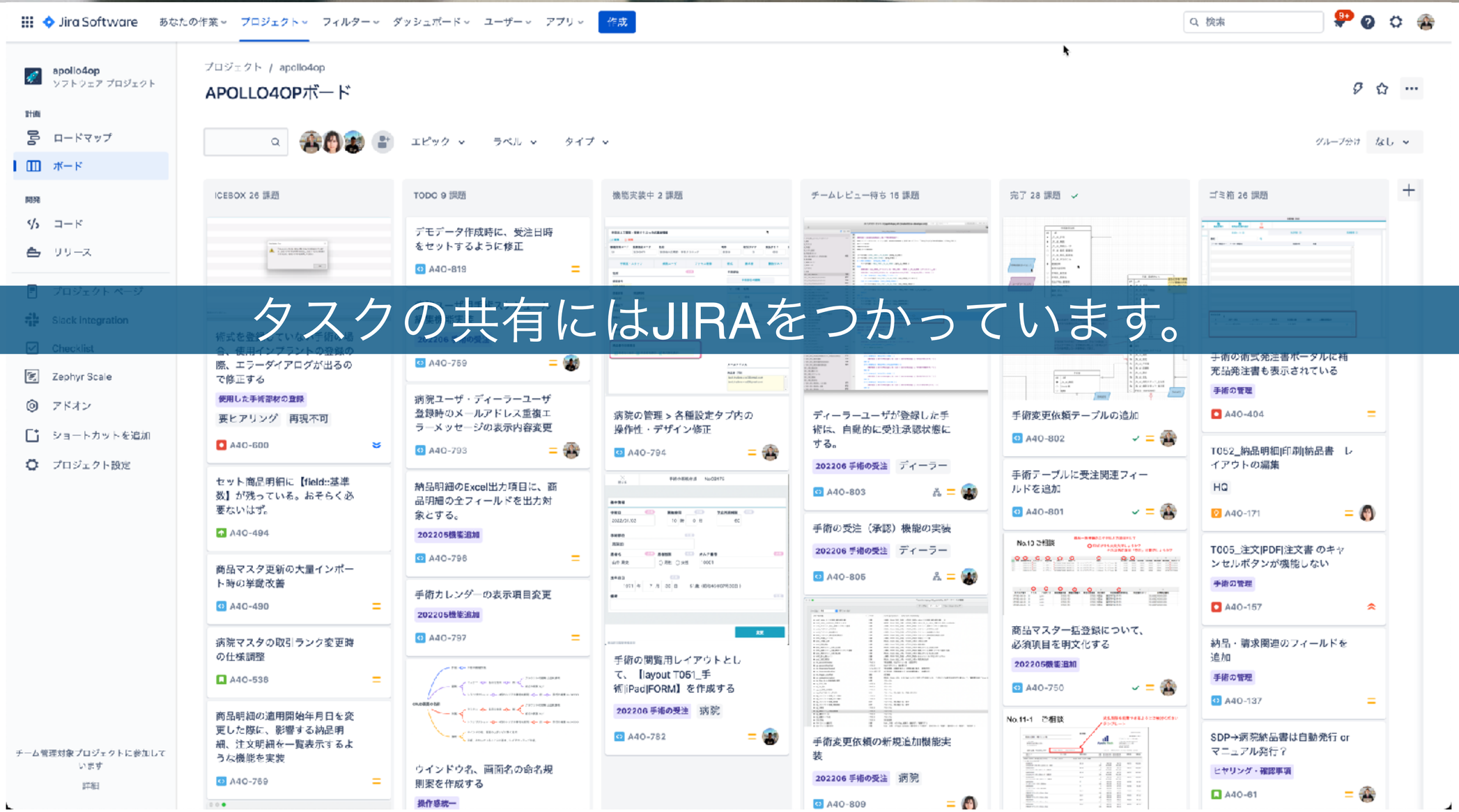




https://www.lucidchart.com/pages/ja

開発プロジェクトの進め方

タスクの共有にはJIRAをつかっています。





フィーチャー	バグ	その他のタスク	リリース	UIデザイン	リファクタリング	仕様検討
実装すべき機能	機能上の不具合バグ	直接的な実装作業ではが、プロジェクト進行上では必ず必要な作業	リリース作業に伴って発生する一時的な作業	LucidChartを使ったUIモックアップの作成	保守性や応答性と行った観点から課題のあるコードの改善	仕様設計が不十分な要求機能に対する検討や確認



チーム開発

- 新しいメンバーの参画、及び新しいテクノロジーの採用にあたり、アジャイルソフトウェア開発のプラクティスのひとつである**ペアプログラミング**を１ヶ月ほど行った。

- ペアプログラミングは、**開発ルール・開発標準の共有**を目的として、Zoomで画面共有をしながら行った。

- Zoomでの画面共有は**レコーディング可能**なので、理解が難しい部分も反復学習が可能。

- 必要に応じて JIRA と連携可能な **Confluence** というソフトウェア開発に最適化された Wiki サービスを使ってドキュメント化した。



開発ルール標準化の重要性

01

属人性の排除

実装された機能が、実装を担当した特定のエンジニアにしかわからない場合、ソフトウェアの運用・保守性が極めて低くなる。

誰が作っても、他の開発メンバーがいつでも保守目的などで、該当機能を修正、改良できるようなルールを作り、実際にそれを適用することが求められる。

02

品質管理

既に稼働実績のあるロジックを、カスタム関数やコンテクストフリーなスクリプト、もしくはスニペットとしてチームメンバーと共有することで、品質が担保された機能を「部品（モジュール）」として再利用できる。

稼働実績のあるロジックなので、バグの発生確率は、ゼロからつくるよりも遥かに低下する。

03

開発の効率化

該当ソフトウェアの UIデザインや、基本的な「振る舞い」をルールとして定めておくことで、ソフトウェアとしてのルック＆フィールに一貫性をもたせることができる。

また、上記の点を予めルールとして定めておけば、個々のエンジニアがゼロからデザイン・設計する必要が無いので開発スピードが向上する。



開発ルール標準化の重要性

01

属人性の排除

実装された機能が、実装を担当した特定のエンジニアにしかわからない場合、ソフトウェアの運用・保守性が極めて低くなる。

誰が作っても、他の開発メンバーがいつでも保守目的などで、該当機能を修正、改良できるようなルールを作り、実際にそれを適用することが求められる。

02

品質管理

既に稼働実績のあるロジックを、カスタム関数やコンテクストフリーなスクリプト、もしくはスニペットとしてチームメンバーと共有することで、品質が担保された機能を「部品（モジュール）」として再利用できる。

稼働実績のあるロジックなので、バグの発生確率は、ゼロからつくるよりも遥かに低下する。

03

開発の効率化

該当ソフトウェアの UIデザインや、基本的な「振る舞い」をルールとして定めておくことで、ソフトウェアとしてのルック＆フィールに一貫性をもたせることができる。

また、上記の点を予めルールとして定めておけば、個々のエンジニアがゼロからデザイン・設計する必要が無いので開発スピードが向上する。



開発ルール標準化の重要性

01

属人性の排除

実装された機能が、実装を担当した特定のエンジニアにしかわからない場合、ソフトウェアの運用・保守性が極めて低くなる。

誰が作っても、他の開発メンバーがいつでも保守目的などで、該当機能を修正、改良できるようなルールを作り、実際にそれを適用することが求められる。

02

品質管理

既に稼働実績のあるロジックを、カスタム関数やコンテクストフリーなスクリプト、もしくはスニペットとしてチームメンバーと共有することで、品質が担保された機能を「部品（モジュール）」として再利用できる。

稼働実績のあるロジックなので、バグの発生確率は、ゼロからつくるよりも遥かに低下する。

03

開発の効率化

該当ソフトウェアの UIデザインや、基本的な「振る舞い」をルールとして定めておくことで、ソフトウェアとしてのルック＆フィールに一貫性をもたせることができる。

また、上記の点を予めルールとして定めておけば、個々のエンジニアがゼロからデザイン・設計する必要が無いので開発スピードが向上する。



開発ルール標準化の重要性

01

属人性の排除

実装された機能が、実装を担当した特定のエンジニアにしかわからない場合、ソフトウェアの運用・保守性が極めて低くなる。

誰が作っても、他の開発メンバーがいつでも保守目的などで、該当機能を修正、改良できるようなルールを作り、実際にそれを適用することが求められる。

02

品質管理

既に稼働実績のあるロジックを、カスタム関数やコンテクストフリーなスクリプト、もしくはスニペットとしてチームメンバーと共有することで、品質が担保された機能を「部品（モジュール）」として再利用できる。

稼働実績のあるロジックなので、バグの発生確率は、ゼロからつくるよりも遥かに低下する。

03

開発の効率化

該当ソフトウェアの UIデザインや、基本的な「振る舞い」をルールとして定めておくことで、ソフトウェアとしてのルック＆フィールに一貫性をもたせることができる。

また、上記の点を予めルールとして定めておけば、個々のエンジニアがゼロからデザイン・設計する必要が無いので開発スピードが向上する。



開発ルール標準化の重要性

01

属人性の排除

実装された機能が、実装を担当した特定のエンジニアにしかわからない場合、ソフトウェアの運用・保守性が極めて低くなる。

誰が作っても、他の開発メンバーがいつでも保守目的などで、該当機能を修正、改良できるようなルールを作り、実際にそれを適用することが求められる。

02

品質管理

既に稼働実績のあるロジックを、カスタム関数やコンテクストフリーなスクリプト、もしくはスニペットとしてチームメンバーと共有することで、品質が担保された機能を「部品（モジュール）」として再利用できる。

稼働実績のあるロジックなので、バグの発生確率は、ゼロからつくるよりも遥かに低下する。

03

開発の効率化

該当ソフトウェアの UIデザインや、基本的な「振る舞い」をルールとして定めておくことで、ソフトウェアとしてのルック＆フィールに一貫性をもたせることができる。

また、上記の点を予めルールとして定めておけば、個々のエンジニアがゼロからデザイン・設計する必要が無いので開発スピードが向上する。

開発プロジェクトの進め方



- 月曜日の午前中に、お客様とZoomミーティング。画面を共有しながら、当週の開発予定機能の確認や、その他の重要事項を共有。

- 月曜日の午後から木曜日にかけて、FileMakerでプロトタイプを実装。その間も、必要があればZoomやChatでコミュニケーションを取る

- 金曜日の午前中に、お客様へ木曜日までに実装したプロトタイプをZoomで画面共有しながらデモ。

- 金曜日の午後に、お客様の検証環境サーバに最新バージョンをデプロイ。実際に触ってもらってフィードバックをいただく。



プロジェクトの成功要因

01

プロトタイピングと準委任契約

一般的には、受託側が有利、そして委託側には不利とされる**準委任契約を受け入れてくださったことは、最も大きな成功要因。**

準委任契約は、スクラップ＆ビルドを繰り返しながら、プロトタイプを少しづつ育てていくプロトタイピングモデルを採用する上で必須の契約条件。

02

FileMaker × Webix ローコードの採用

FileMakerは当然のこと、今回のプロダクトが要求する必須機能を実装するにはJavaScriptとの連携が必須だった。

FileMakerはVer19になって**JavaScriptの連携が強化**されたこと。

更にJavaScriptベースの **Webix** を採用できたことがプロジェクトの成功に大きく貢献した。

03

FileMaker × JavaScript 両刀使いの開発チーム

アジャイルソフトウェア開発において、開発チームのメンバーに求められるのは**「多能工」**。

今回の開発チームメンバーは、ともにFileMakerとJavaScriptの両方に対応できるエンジニアだったので、クライアントが求める**「柔軟性」と「俊敏性」に応える**ことができた。



プロジェクトの成功要因

01

プロトタイピングと準委任契約

一般的には、受託側が有利、そして委託側には不利とされる**準委任契約を受け入れてくださったことは、最も大きな成功要因。**

準委任契約は、スクラップ＆ビルドを繰り返しながら、プロトタイプを少しづつ育てていくプロトタイピングモデルを採用する上で必須の契約条件。

02

FileMaker × Webix ローコードの採用

FileMakerは当然のこと、今回のプロダクトが要求する必須機能を実装するにはJavaScriptとの連携が必須だった。

FileMakerはVer19になって**JavaScriptの連携が強化**されたこと。

更にJavaScriptベースの **Webix** を採用できたことがプロジェクトの成功に大きく貢献した。

03

FileMaker × JavaScript 両刀使いの開発チーム

アジャイルソフトウェア開発において、開発チームのメンバーに求められるのは**「多能工」**。

今回の開発チームメンバーは、ともにFileMakerとJavaScriptの両方に対応できるエンジニアだったので、クライアントが求める**「柔軟性」と「俊敏性」に応える**ことができた。



プロジェクトの成功要因

01

プロトタイピングと準委任契約

一般的には、受託側が有利、そして委託側には不利とされる**準委任契約を受け入れてくださったことは、最も大きな成功要因。**

準委任契約は、スクラップ＆ビルドを繰り返しながら、プロトタイプを少しづつ育てていくプロトタイピングモデルを採用する上で必須の契約条件。

02

FileMaker × Webix ローコードの採用

FileMakerは当然のこと、今回のプロダクトが要求する必須機能を実装するにはJavaScriptとの連携が必須だった。

FileMakerはVer19になって**JavaScriptの連携が強化**されたこと。

更にJavaScriptベースの **Webix** を採用できたことがプロジェクトの成功に大きく貢献した。

03

FileMaker × JavaScript 両刀使いの開発チーム

アジャイルソフトウェア開発において、開発チームのメンバーに求められるのは**「多能工」**。

今回の開発チームメンバーは、ともにFileMakerとJavaScriptの両方に対応できるエンジニアだったので、クライアントが求める**「柔軟性」と「俊敏性」に応える**ことができた。



プロジェクトの成功要因

01

プロトタイピングと準委任契約

一般的には、受託側が有利、そして委託側には不利とされる**準委任契約を受け入れてくださったことは、最も大きな成功要因。**

準委任契約は、スクラップ＆ビルドを繰り返しながら、プロトタイプを少しづつ育てていくプロトタイピングモデルを採用する上で必須の契約条件。

02

FileMaker × Webix ローコードの採用

FileMakerは当然のこと、今回のプロダクトが要求する必須機能を実装するにはJavaScriptとの連携が必須だった。

FileMakerはVer19になって**JavaScriptの連携が強化**されたこと。

更にJavaScriptベースの **Webix** を採用できたことがプロジェクトの成功に大きく貢献した。

03

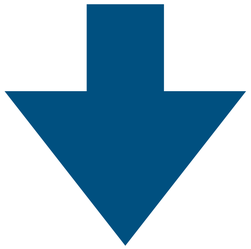
FileMaker × JavaScript 両刀使いの開発チーム

アジャイルソフトウェア開発において、開発チームのメンバーに求められるのは**「多能工」**。

今回の開発チームメンバーは、ともにFileMakerとJavaScriptの両方に対応できるエンジニアだったので、クライアントが求める**「柔軟性」と「俊敏性」に応える**ことができた。

ITエンジニアがFileMakerを学ぶ意義

- ソフトウェア開発に対しては、常に俊敏性と柔軟性、つまり**アジャイルなソフトウェア開発**を要求されている。
- アジャイルなソフトウェア開発対応するには、**多能工（フルスタック）**なエンジニアが必須。しかし**その育成には多大な時間と費用が必要**。



ローコード、そしてFileMakerであれば、育成に必要な**時間と費用を圧縮**することができる！



限界のないローコード。
スキルを磨けば、ビジネスが輝く。

Claris Engage Japan 2022

#ClarisEngageJP