技術選定の審美眼（2023年版）

和田卓人 @t_wada

2023/09/23

自己紹介

- Name : 和田卓人 (Takuto Wada)
- Hatena : t-wada
- X(Twitter) : t_wada
- GitHub : twada, 代表作: power-assert
- BlueSky : twada
- 撮影OK (シャッター音を控えめにお願いします)
- Twitter での実況も大歓迎です
- 今日のハッシュタグは **#tbo2023**


97
Collective Wisdom from the Experts
プログラマが知るべき97のこと
97 Things Every Programmer Should Know
O'REILLY
オライリー・ジャパン
Kevlin Henney 編
和田 卓人 監修
夏目 大 訳


Avoiding the Pitfalls of Database Programming
SQL アンチパターン
Bill Karwin 著
和田 卓人 監訳
和田 省二 監訳
児島 修 訳
O'REILLY
オライリージャパン


Test-Driven Development By Example
テスト駆動開発
Kent Beck 著
和田卓人 訳
テスト駆動開発(TDD)を原点から学ぶ
本書は、Kent Beck, Test-Driven Development by Example (Addison-Wesley, 2003)の新訳版です。
Ohmsha


事業をエンジニアリングする技術者たち
株式会社 CARTA HOLDINGS 監修
和田卓人 編
フルサイクル
開発者がつくる
CARTAの現場
CARTA

ド
テストカイテナイトカ
オマエソレ
ド
t・-・w・a・d・a・ノ・マ・エ・デ・モ・オ・ナ・ジ・コ・ト・イ・エ・ン・ノ・？
ド
TDB
ド

よろしく
お願いします

どうしてこうなった

- JavaScript fatigue: 変化が速すぎる（ように見える）フロントエンド疲れ
- 限界集落: 停滞し、若者のいないプロダクト
- 極端なコントラストはどこから来るのか
- 変化の速さは陳腐化の速さでもあり、本質的でない変更を強いられる頻度にもなる
- **重要な変化と重要でない変化を見分けられない**
- 軽微な変化をスルーできずに消耗する
- 重要な変化をスルーして詰む

変わるもの × 変わらないもの

- 変化が予想できないことだけは予想できる
- だから技術の歴史から学ぶ
- 過去から学び、近未来の予測を試みる
- では、技術はどのような変化をしてきたのか


振り子


螺旋

振り子と螺旋

- 技術の変化の歴史は一見すると **振り子** に見える
- でも実は **螺旋** 構造。同じところには戻ってこない
- **差分** と、それを **可能にした技術** が重要
- それはそうと、変わらないように見える強固なものもある
- 「それは既に20年前に通った道だ」は老害発言かもしれない
- 技術の世界は基本的に若者有利。差分が見えるのがベテランの数少ないアドバンテージ

変わらないもの

長生きしている技術に学ぶ

- Unix
- REST / Web
- RDBMS / SQL

Unix

- Unix哲学
- **小さいのは良いことだ** (Small is Beautiful)
- **一つのことをうまくやる** (Make each program do one thing well)
- **すべて**はファイル
- プログラムをフィルタに
- 標準入力と標準出力
- 成功失敗は終了ステータスで示す
- KISS (Keep It Simple, Stupid)
- パイプを介して40年前のコードといま書いたコードがあっさりと連携できる凄さ

REST / Web

- 多様な URL
- **すべて**を URL で示されるリソースと考える (Adressable)
- **少なく、統一された**振る舞い (HTTP Method, Uniform Interface)
- 状態を持たない (Stateless)
- ハイパーリンクで状態遷移 (Connectedness)
- 成功失敗はHTTPステータスで示す
- **多様な名詞と少ない(制約のある)動詞**
- 良くできた分散システムとしての Web

RDBMS / SQL

- 強固で数学的な背景
- 閉じた系
- **すべて**が関係/集合（テーブルも、クエリの結果も関係/集合）
- **少なく、統一された**操作 (SELECT, INSERT, UPDATE, DELETE)
- 最近では SQL は RDBMS だけのものではなくなってきている
- NoSQL から NewSQL へ

共通するもの

制約がもたらす共有、相互接続性の高さ

- インタフェースが狭く限定的
- 振る舞いの種類が少ない
- 実装に依存しない
- 接続と再利用が時間をまたいでいる

すべては○○○であるという抽象化

○○○を介して梃子の効果が生まれる

- Unix: すべてはファイルとフィルタである
- REST/Web: すべてはリソースである
- RDBMS/SQL: すべては集合である

閉包性: **結果も○○○に閉じる**ような系の強さ

- SQLの結果とテーブルの同一視
- ファイルの中身とコマンド結果の同一視


変わるもの

螺旋1: 集中と分散

- Good Old Web (PHP, Java Servlet)
- EJB と SOAP と XML と SOA
- Ruby on Rails とモノリシックな Web システムの事業立ち上げの早さ
- REST と PoX over HTTP
- JSON サーバとコンテナ技術と k8s
- microservices と逆コンウェイ戦略
- OpenAPI (Swagger) と gRPC と GraphQL
- クラウドネイティブとサーバレスコンピューティング
- フロントエンドは CDN でさらに分散処理を行う時代に
- 一方バックエンドはマイクロサービス疲れとモジュラモノリスへの回帰

螺旋1の差分

- クラウドが柔軟で動的な構成変更を可能にした
- 分散システムとしての Web の理解が進んだ
- シェアドナッシング+スケールアウトという選択肢
- ムーアの法則の限界とその打破は分散コンピューティングだった
- 要素技術として Docker や k8s などのコンテナ技術が登場
- 新時代のアーキテクチャパターン The Twelve-Factor App
- 開発組織を跨ぐスキーマやクエリ(例: OpenAPI, gRPC, GraphQL)
- XML から JSON から型付けされたやりとりへ
- GraphQL と Fragment Colocation による SQL 時代の問題の解決
- CDN とエッジコンピューティングの拡大
- 位置透過性との戦いは続く

螺旋2: アプリとブラウザ、フロントエンドとバックエンド

- VB (オープン系): 2階層システム
- Good Old Web Site (PHP, Java Servlet): 3階層システム
- Flash: リッチクライアントの誕生
- Ajax: Re-discover JavaScript
- Web フロントエンドのフレームワーク乱立時代とキャッチアップ疲れ
- HTML5: 大きな標準の誕生
- Native iOS/Android App: ブラウザでは満たせない UX の実現
- React 登場後のフロントエンド
- React Native, Flutter, PWA, 夢と現実
- Core Web Vitals という人参
- RSC, Remix によるバックエンドへの回帰

螺旋2の差分

- アプリであれ、Web 技術であれ、主戦場はモバイルに収束している
- モバイル端末は、可処分時間の奪い合い
- Next Billion Users や最近の学内システム事情
- リリースプロセスの違いが、品質のベースの違いをもたらす（MTBF と MTTR）
- Web の要素技術が、モバイルにおけるネイティブアプリとの差を縮められるかどうか
- エンゲージメントの強さと、モバイル端末を占有させる UX の両立
- CWV はパフォーマンスを品質特性の上位に押し上げ、フロントエンドのアーキテクチャへ強い影響を与えた

螺旋の中心の移動: Web 技術への大きな流れ

- 螺旋の軸そのものが移動している（不可逆なシフトだろうか）
- 昔はエディタでメールを読み書きしていたのが（※このあたり老人感あり）、ブラウザ（Gmail）に移り、とうとうエディタ（Atom / VSCode）も Web 技術ベースになった
- プログラマの生活の場が Web 技術の上に移り、イノベーションの中心も Web 技術に移動してきている
- 例: Visual Studio Live Share, GitHub Copilot

螺旋3: OSSと組織

- 前史: クローズドソースの時代
- Linux の成功と『伽藍とバザール』による言語化
- sourceforge, ASF: まず開発組織があり、コミット権が付与される
- GitHub 革命: まず個人があり、後に必要に応じてチームができる
- OSS 燃え尽き問題: 個人開発者が毀誉褒貶にさらされ開発をやめてしまう
- OSS 成功モデル: 『伽藍とバザール』では説明できない現代のOSS像
- OSS の理想の崩壊: 強者によるフリーライド問題

螺旋3の差分

"What success really looks like in open source, And how we can support them."

- A **popular** project is a project that is being used by a lot of people, and ideally growing.
- A **healthy** project is a project where maintainers have an active, reciprocal relationship with their community.
- A **supported** project is one where maintainers have the resources to manage the popularity of a project and grow a healthy community.

https://medium.com/@nayafia/what-success-really-looks-like-in-open-source-2dd1facaf91c

Game Changer たち

決定的な違いをもたらした技術たち

- Ruby on Rails
- Cloud Computing
- Docker
- React
- **LLM**

量的な変化が質的な変化をもたらした

- Ruby on Rails / Cloud Computing / Docker / LLM
- Rails の生産性はスタートアップ企業に必要な初期速度と仮説検証を可能にした
- クラウドはシステム開発のコスト構造やボトルネックを大きく変え、開発と運用の関係の変革を促した(DevOps)
- 従来の仮想化技術に比べた Docker の立ち上がりの圧倒的な速さと、再現性の高さ
- LLM の出現自体が量的->質的変化であり、結果出現した世界も量的->質的変化
- 圧倒的な安さ速さの獲得で不可能が可能になり、新たなものが見えてくる
- 本当にやりたいことの必須要件として安さ速さの獲得がある
- コードが事業そのものになる時代へ (Why Software Is Eating The World)

考えることを大きく減らした変化

- React / LLM
- 仮想 DOM により富豪的にデータを更新してもレンダリングコストの心配をしなくて良くなった
- サーバサイドでテンプレートを毎回レンダリングするのと同じ感覚でクライアントのコードを書ける
- データ更新の方向が一方通行なので状態が一意に定まり、混乱しない (Single source of truth)
- React により手数は増えたが、心配は減った
- 壁打ち相手、たたき台作りとしての ChatGPT の優秀さ
- ChatGPT は「一手目」を考えるコストを劇的に下げた

LLMはスルーできない

- 今まさに世界を不可逆に変えている Game Changer
- 去年の新人研修と今年の新人研修で景色が違いすぎる
- GitHub、プロンプトでコード生成やデバッグを指示できる「GitHub Copilot Chat」を個人ユーザーにも提供開始
- 生成AIを統合した新Office、会話だけでタスクが消化されていくのがやばい
- プログラミングの姿は変わるが、プログラミングがなくなるわけではない
- サンドイッチワークフロー
- 道具としてのLLM、壁打ち相手としてのLLM
- 「愚者は知識を問い、賢者は議論をする」 by ところてん
- 労力は外注できるが、能力は外注できない
- 人間と AI が互いの強みを活かし、協力して新しい価値を創造する

Takuto Wada
@t_wada

GitHub Copilotはどういうコードを書くべきか分かっている人が最速でコードを書くのを支援する（タブを多数開いておくと「おまえは俺か」というレベルの提案精度が出る）

Copilot Chatはどういうコードを書くべきか分かっていない場合でもコードを書けるように支援する（疑問を環境内で解決できる）

Translate post

12:26 PM · Jul 26, 2023 · **101.5K** Views

View post engagements

131 806 176

スルーするか、しないか

- Game Changer そのものを採用する必要は必ずしも無いが(例: Rails, React)、それらが変えた後の世界は知っておく必要がある
- 開発に関わる労力(コスト、心的コスト)を矢印で表すとする
- 矢印を圧倒的に短くして、コスト構造を大きく変えるもの
- あるいは矢印を折るような、ジャンルそのものを不要にするもの
- それらインパクトのある Game Changer とその後の世界は、スルーしがたい
- 逆に言えば、そこまでではない変化は「お好みで」となる
- その技術が何のために出てきて、何をどのくらい変化させるのか見切れればスルーできる

まとめ

- 変わるもの。技術の変化は螺旋状。螺旋の差分と、それを可能にした技術を理解する
- 変わらないもの。生き残る技術には特徴がある。その背後にある抽象の力を理解する
- Game Changer を見抜く目を養う

ご清聴
ありがとうございました