リモートワークにおけるソフトウェア開発

2020年度　最先端ソフトウェア工学　個別ゼミ1報告会

黒川　一成
井浦　陽一郎
秦　理貴

背景

- **背景**
  - 新型肺炎ウイルス（Covid19）蔓延により、ソフトウエア開発にも大きく影響を及ぼしている。
  - コロナ前は関係者が密なコミュニケーションにより対応していた部分が、テレワークにより密なコミュニケーションが取りにくくなり、**新しい開発スタイルが必要となっている**
- **コロナによる変化点**
  - リモートワークの影響を大きく受ける以下二つの工程について、検証を行った。

	コロナ前	コロナ直後
要件定義	関係者が参加するミーティング/レビュー会を設けて直接擦り合わせ	ミーティングを設けた擦り合わせができなくなり、質問をメールでのやり取りやWebミーティングに変更
プログラミング	詰まっている部分があると、詳しい人に直接聞いたり、周りの人がサポート	詳しい人にメールで問い合わせたり、定期的なミーティング時に状況報告で周りが初めてサポート

リモートワークにおける要件定義スタイル

ユーザーのための要件定義ガイド：第２版（IPA）

- **要件定義でよくある課題**
  - ステークホルダとのコミュニケーションが不十分なため、要件定義に不備が発生する
- **勘どころ① コミュニケーション計画を立案する**
  - ステークホルダに対して、情報を伝える方法を意識して**効果的に組み立てる**ようコミュニケーション計画を立案する。コミュニケーション計画の際には、「伝達相手」「伝達内容」**「媒体」「頻度」**を適切に決めることがポイントとなる。

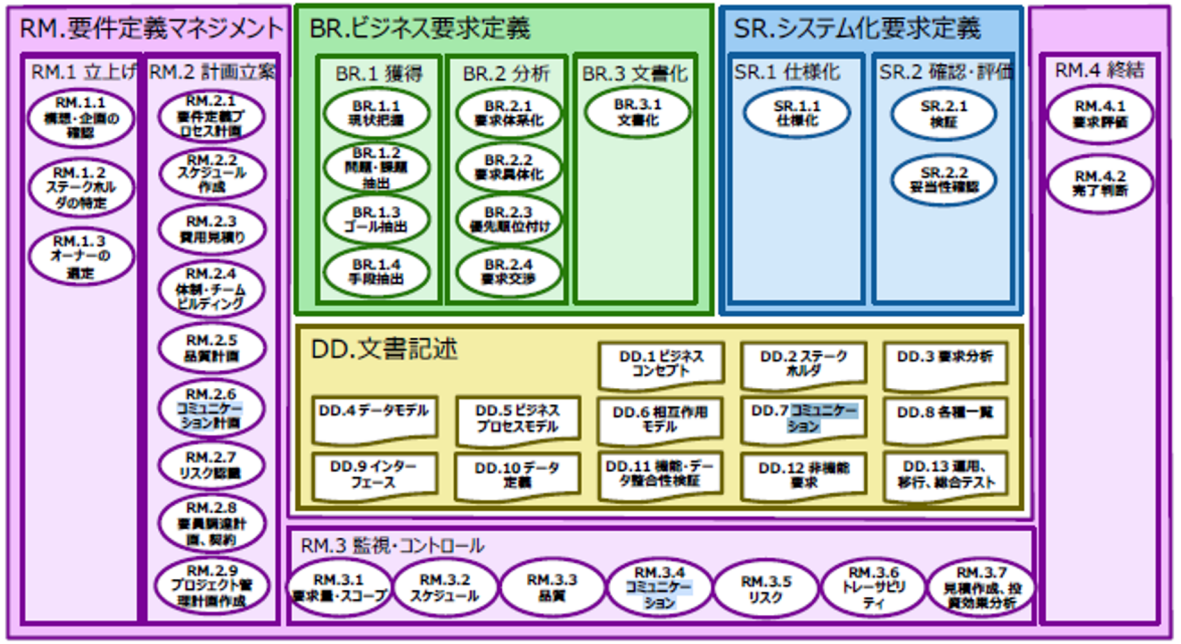
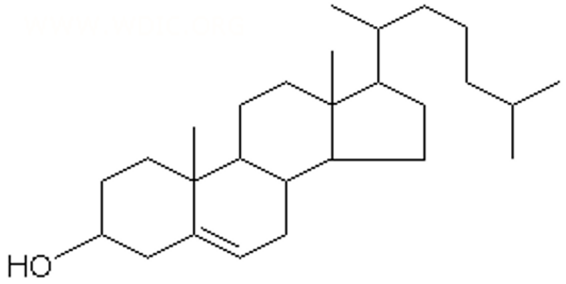


図 2.4 要件定義問題カテゴリマップ

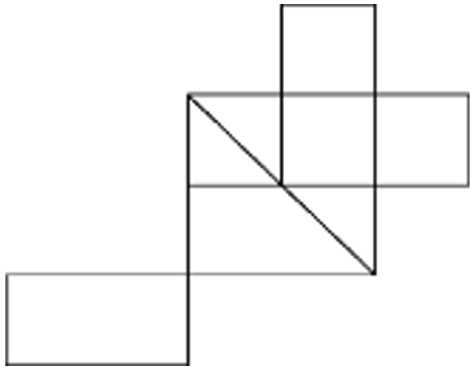
コロナ後において**特に媒体の変化**(TV会議・チャットツールの利用頻度の増大)が起き、リモート時代における効果的な組み合わせを提案する

提案にあたっての実験

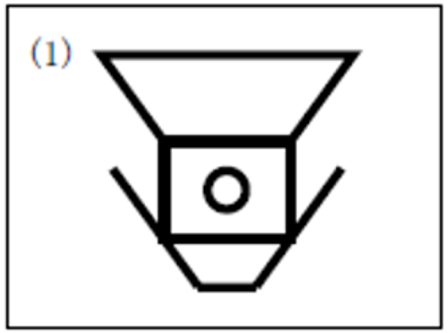
- **実験の目的**
  - 媒体のごとの要件の伝わり方を検証
- **実験内容**
  - 複数媒体（ＴＶ会議（要件元顔有り/顔無し）/チャット/メール）で、要件定義のロールプレイとして、難しい絵を伝える実験を実施。
  - 利用した絵は下記の３つ



A



B



C

- **評価方法**
  - 媒体毎の情報の正確性/やり取りの速度を比較

実験からわかったツール毎の特徴

ツール	情報の正確性	やりとりの速度	特徴
TV会議	◎	○	最も効率が良いが、打ち合わせを設ける必要がある。
チャット	○	△	手軽に行えるが、文字だけでは認識の違いが発生する可能性がある。
メール	○	×	最もやりとりに時間がかかる。別途資料ベースでのやりとりが必要。

- 顔ありでおこなっているゼミの方が、顔なしでおこなっている通常の仕事よりも、議論が活発にできている
  ⇒ TV会議において，顔有り／無しでどのような違いがあるのか？
- メールに比べると，チャットは気軽にやり取りができる
  ⇒リモートワークにおける要件定義に活用できるのでは？

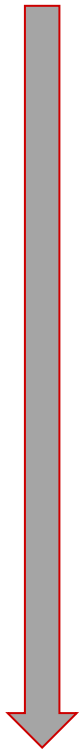
Ｗｅｂ会議における顔有り/無しに関する考察

	メリット	デメリット
顔有り	・相づちが取っていたりすると、相手の理解がわかる ・チームの一体感が出る気がする ・打合せのアイスブレイクは短くなる	・通信コストがかかる ・身だしなみを気にする必要がある
顔無し	・身だしなみを気にする必要がない ・重要でない打合せ等で流し聞きをしていても問題ない	・伝わっているかわかりにくい ・打合せのアイスブレイク時は難しい

チャットに関する考察（Slack）

	メリット	デメリット
要件定義	・最初の説明（TV会議や対面）を終えたのち、細かな議論をするのに向いている ・エクセルなどの詳細項目などをチャットで議論・確認するのに使える ・議論の途中経過が自動的に残る、議論に途中から入りやすい **・あいまいな議論の時ほど、文書で残るため、有効である**	・TV会議のようなリアルタイム性には対応できない ・メールや電話会議に比べ，文字でのやり取り（回数）が増加する 心理的な側面 • 人によっては，すぐに返信しないといけないと思う • 会話がすぐに流れていくともういいやとなることもある

要件定義の各工程における推奨ツール

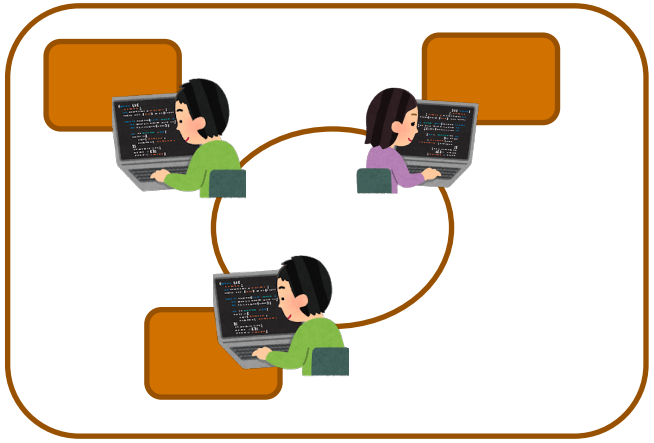


工程	求められること	推奨ツール
キックオフ	目的の合意形成	**TV会議（顔あり）**
要件定義作成中のやりとり	やりとりの迅速性 やりとりの蓄積	**チャット**
要件定義レビュー	意思疎通の確認	**TV会議**
要件定義レビュー 議事録の配布	決定事項の合意・蓄積	**メール**

Withコロナ時代の新しい実装・プログラミングスタイル

- コロナ前後で実装環境が大きく変化し、新しいスタイルが必要となった。

コロナ前

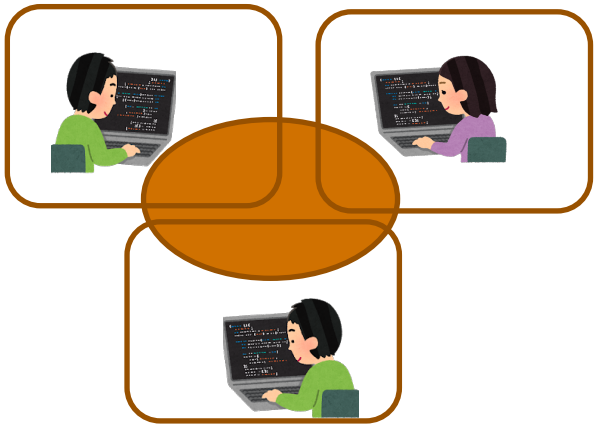


詰まっても
気軽に聞ける環境

コロナ後

気軽にきけなくなった
気づけなくなった

目指す姿



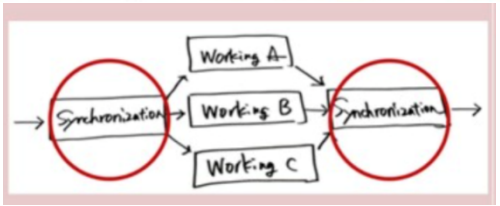
皆で同じ環境を共有

リモートモブプログラミングの提案

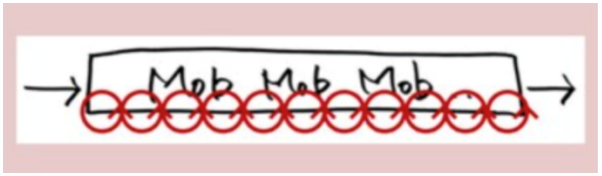
モブプログラミングとは？

・一つの開発環境を、複数人で共有する。
・作業者（ドライバ）を一人決め、その他の人は意見を述べ、ドライバをナビゲートする。
・ドライバは定期的に交代する。

分担作業

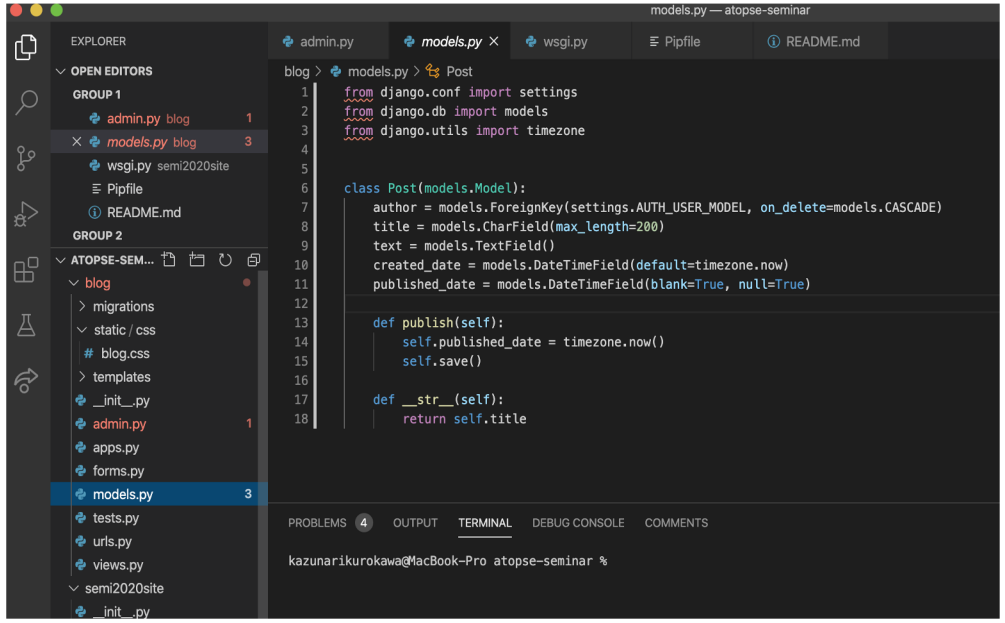


モブプログラミング

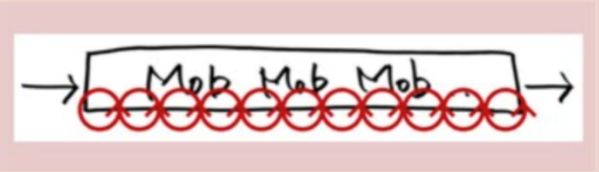


提案するにあたっておこなった実験

- **実験の目的**
  - リモートモブプログラミングを体験し，特徴を明確にする
- **おこなった実験内容**
  - Ｗｅｂアプリケーション開発の為の環境構築をリモートモブプロで実施
  - ドライバ：黒川、　ナビゲータ：井浦、秦
  - ツール：VS Code LiveShareを利用
- **評価方法**
  - リモートモブプロをおこなっての気づきを挙げる



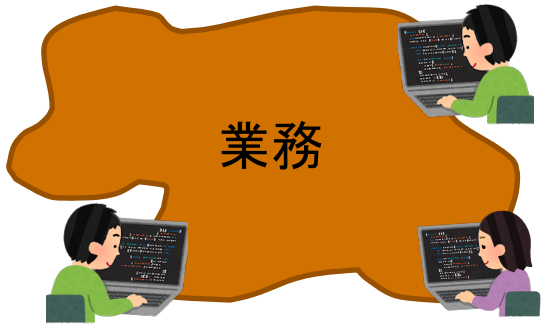
リモートモブプログラミングの特徴に関する考察

	リモートモブプログラミング
イメージ	 
長所	・LiveShare等のツールが充実しており、**手軽に実施できる。** ・ナビゲータの協力により、**詰まるリスクが低下** ・コードの品質が高まり、**レビュー工数が少**なくて済む。
短所	・**詰まらなければ**、分担作業の方が効率的。 ・ドライバが特定のナビゲータに依存する場合、他のナビゲータは**疎外感が発生。** ・休憩のタイミングがとりにくい。
向いている ケース	・**教育**など、参加者にスキル差がある ・**難しい課題**があり、全員で取り組む必要がある ・**コードの品質**が特に求められる ・作業が**明確に分担できない**（開発初期等）

リモートにおけるプログラミング手法の利用ケース

■本テーマで推奨する利用ケース

■開発初期



■ 分担が明確化できない。
■ トライアンドエラーの頻度が高い。
■ 分担したとしても同期の頻度が高い。
■ 解決方法が明確化されていないと、詰まるリスクが高い。
■ 新人などのスキルが十分でないケースがあり得る。

リモートだと特に顕著に現れ、分担作業は困難

⇒リモートモブプログラミングを推奨

■開発中期～

■ 業務が明確に分担できる。
■ トライアンドエラーの頻度が低い。
■ 同期の頻度は低い。
■ コミュニケーションはチャット等で十分

リモートでも分担作業は可能

⇒分担作業＋チャットを推奨

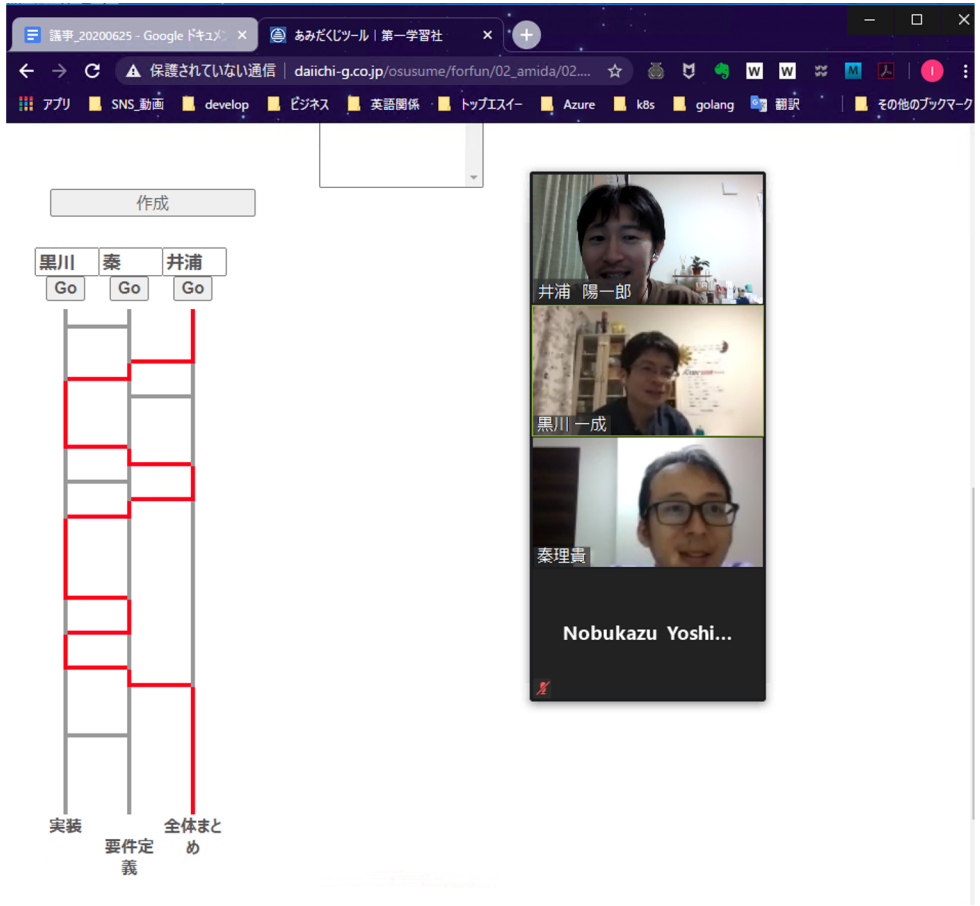
全体を通してのまとめ

- リモートワークにおけるソフトウェア開発の各工程で，
  どのようなコミュニケーション手段や媒体が適しているか検証
- 要件定義（設計）
  - リアルタイムでの意思疎通や合意が必要
    TV会議などリアルタイム性のあるツール
  - ちょっとしたやり取りで詳細を詰める
    チャットツール
  - 決定事項などを多数に通知する場合
    メール
- プログラミング・実装
  - 開発初期など分担作業が行いにくい
    モブプログラミング
    リモートの場合，強力なツールの活用し，
    なるべく職場に近い雰囲気が望ましい
  - 開発が順調に進み始めると
    分担作業＋チャットの方が効率が良い。
- 導入における課題
  - 社外クラウドサービスなど場合，社内のセキュリティ規約などを変更しないといけない
  - 企業間の利用ツール違いによって，他社との打ち合わせなどに利用できない

どのようなツールを使うにしても，リモートワークは孤立しがち，
適度なアイスブレイクを忘れないようにすることも重要

おまけ

- **タスク・作業分担がリモートだと決まらない・・・**
  - リモートじゃんけん
    タイミングが難しく、後出しではないか・・・
  - リモートあみだくじ
    ちょっとしたドキドキ感で，意外と盛り上がることも
    実際に資料作成の分担で活用（右のキャプチャ）



ご清聴ありがとうございました

ソフト開発工程毎の目的と、ツール種別の分類

工程	同期的なコミュニケーション（電子メール、電話、テキストチャット、テレビ会議システム）	非同期のコミュニケーション（ドキュメント、ソースコードのバージョン、打ち合わせの決定事項等）
プロジェクト管理	ミーティング （グループ内，関連部署，顧客）	スケジュール管理 決定項目の管理
要件定義	要件定義書レビュー	要件定義書の管理
設計	設計書レビュー	設計書の管理
実装	コードレビュー	コードのバージョン管理
単体テスト	単体テスト仕様書レビュー 成果物レビュー	単体モジュールのバージョン管理
結合テスト	結合テスト仕様書レビュー 成果物レビュー	結合モジュールのバージョン管理
出荷テスト	出荷テスト仕様書レビュー 成果物レビュー（外部）	最終成果物のバージョン管理

モブプログラミングにおけるVS Code Live Share

VS Code Live Shareのメリットと制約

メリット①：リアルタイム性

- 同時に複数のナビゲータとソースコードをシェアできる
- リアルタイムでコードの変更が反映され、即時レビューが可能

メリット②：独立性

- 画面共有と異なり、ナビゲータはそれぞれ異なるファイルを参照・編集できる。
- 指摘や参照箇所を、リアルタイムに通知できる。（自分でソースコードを探さなくてよい。）

メリット③：統合性

- 同時に編集するため、複数人の作業のマージが不要

制約

- エディタの共有だけでなく、他のツールでの画面共有も必要になる
- GUIなどの動作確認などでは、Live Shareだけでは共有できない
- VS Codeの他のエクステンションなどの機能は共有できない
  Linterなどの表示は共有者しか見えないことがある
- 外部サービスを利用するので、企業によっては利用できない。

⇒以上を踏まえて、今後の業務に活かしていく