リモートワークにおけるソフトウェア開発

「非同期コミュニケーションによる効率的な開発スタイルの提案」

2020年度　最先端ソフトウェア工学　個別ゼミ ２報告会

黒川　一成

井浦　陽一郎

秦　理貴

背景

個別ゼミ１での報告通り、新たなコミュニケーション手段（チャットツール／TV会議）の普及が進んだ

全体を通してのまとめ　　**個別ゼミ１報告会 資料**

■ 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

リモートワークなのでより
コミュニケーションが重要

＋

ツールによりコミュニーションが
より手軽に取れるようになった

同期的なコミュニケーションの増加による課題が生じている

- コミュニケーションパスの増加
- TV会議やチャットツールを妄信的に推進

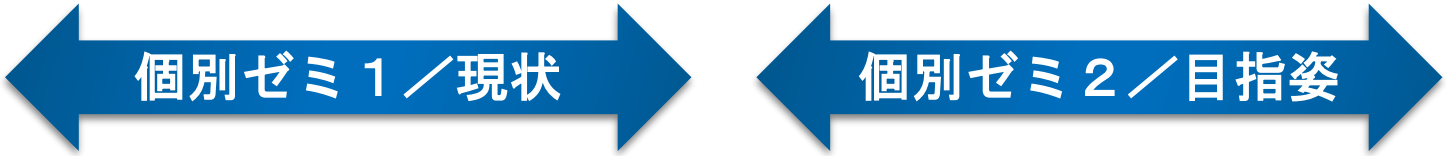
先端・グローバル企業のリモートワークのベストプラクティス

GitLab社の取り組み

- 65カ国以上に1200人以上の社員を持ち、全社員がリモートワークを行っている世界最大級のリモートワーク導入企業である
- リモートワークを下記の段階（非同期のフェーズ）があるとしている

オフィス環境を真似る　効果的・革新的に活用　誰もが非同期に仕事

コミットメント
・入力（時間）より出力（結果）
・グローバルなチーム



先端・グローバル企業のリモートワークのベストプラクティス

GitLab社の取り組み

- 65カ国以上に1200人以上の社員を持ち、全社員がリモートワーク導入企業である
- リモートワークを下記の段階（非同期のフェーズ）

自分たちのプロジェクトでは
非同期フェーズがちょうど良いと判断
今後，オフィス環境に戻っても
非同期な取り組みが活用できるような形を目指す

フェーズ１ スキュモーフィズム	フェーズ２ 機能	フェーズ３ 非同期	フェーズ４ 指向性
オフィス環境を真似る	効果的・革新的に活用	誰もが非同期に仕事	コミットメント ・入力（時間）より出力（結果） ・グローバルなチーム

個別ゼミ１／現状　　個別ゼミ２／目指姿

GitLabのフェーズ3での取り組み例
- ドキュメントファーストでのコミュニケーション
- ダイレクトメッセージ／メンションの禁止

なぜ、ドキュメントファーストに着目したか？

- リモートワーク前後でのドキュメントに対する変化

リモートワークへの移行前

- 仕様や設計判断理由や背景情報が残っていない
- 説明会などを通し、チームで共通認識を作成
- 同僚にオフィスで気軽に確認可能

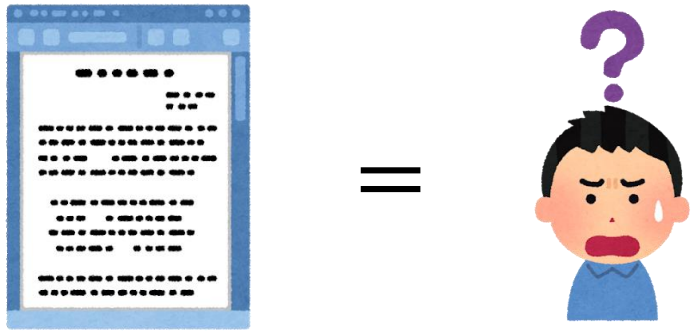
リモートワークへ移行後

- 仕様や設計判断理由や背景情報が残っていない
- 十分な共通認識を形成しずらい（TV会議）
- わからない場合、都度質問が必要

⇒ **同期的コミュニケーションが増加**

文書化されていないことが顕在化

設計書や仕様書　＋　皆からの説明　＝　理解できる！



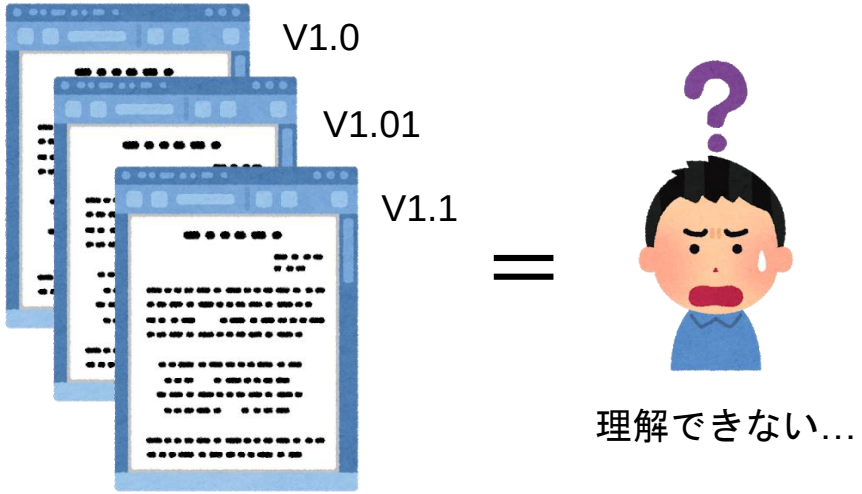
設計書や仕様書　＝　理解できない…

ドキュメントだけで理解できる仕組みが必要。

設計の経緯や背景の文書化手法

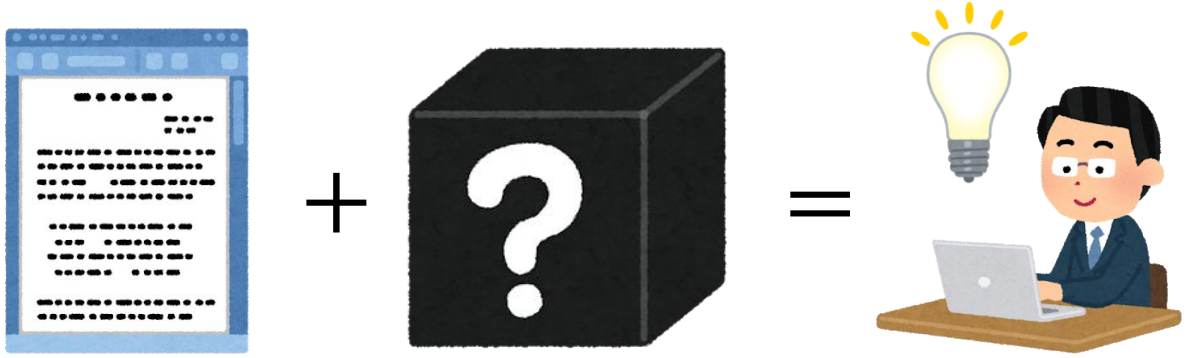
- **仕様・設計に関する情報の文書化手法の調査**
  - 設計における判断の記録が重要

ありがちな文書管理



- 複数の版数の仕様書を管理
- 差分は更新履歴からなんとかわかる
- Whyは残っていない・・・

理想のドキュメント管理



Architecture Decision Records(ADR)
様々なアーキテクチャの判断を記録する手法

引用：https://speakerdeck.com/vanto/a-brief-introduction-to-architectural-decision-records
https://adr.github.io/

Architecture Decision Records

- **ADRとは**

  軽量なテキストでArchitecture 設計の判断理由を記録を残すこと

- **ＡＤＲの特徴**
  - 軽量なテキスト・マークダウン形式
  - 可能な限りコードと同様に管理(Gitなど)
  - 決定の履歴・背景を記録
  - テンプレートの活用（Nygard 2011など）
- **効果的な利用ケース**
  - 公開インタフェースの変更
  - 優先度高の品質特性に直接影響を与える
  - 戦略的な技術的負債の受容
  - 開発プロセスを変更

 # 1. Use Elastic Search for exposing enterprise wide search API.

Date: 2018-05-20

Status

Accepted

Context

There is a need of having an API exposed which can be used to search enterprise wide common data model.

The data currently resides in a RDBMS database, it is difficult to expose micro-services directly querying out of RDBMS databases since the application runs out the same environment.

There are options like ElasticSearch or Solr where data can be replicated.

Decision

Use ElasticSearch for data indexing

Consequences

Data needs to be replicated across the ElasticSearch cluster. This separate cluster needs proper maintenance.

* Near-real time data replication is required.
* Additional cost of maintaining the ElasticSearch environments

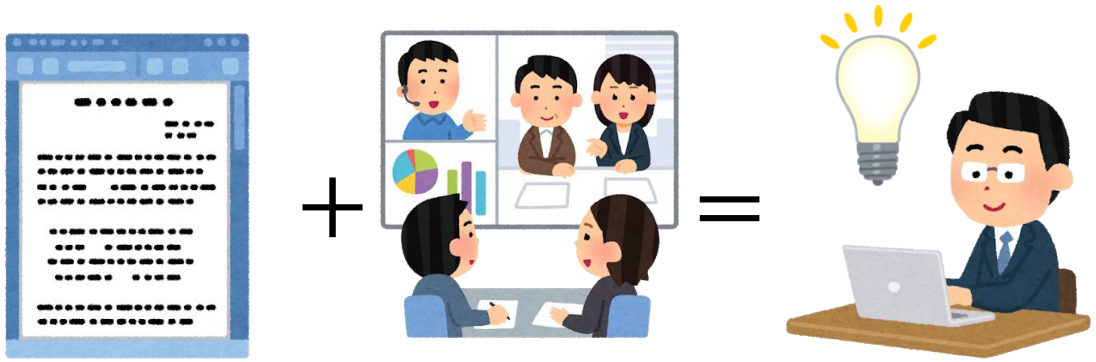
引用：https://speakerdeck.com/vanto/a-brief-introduction-to-architectural-decision-records

引用： https://github.com/deshpandetanmay/lightweight-architecture-decision-records/blob/master/doc/adr/0001-use-elasticsearch-for-search-api.md

リモートコミュニケーションを支援する手法の提案

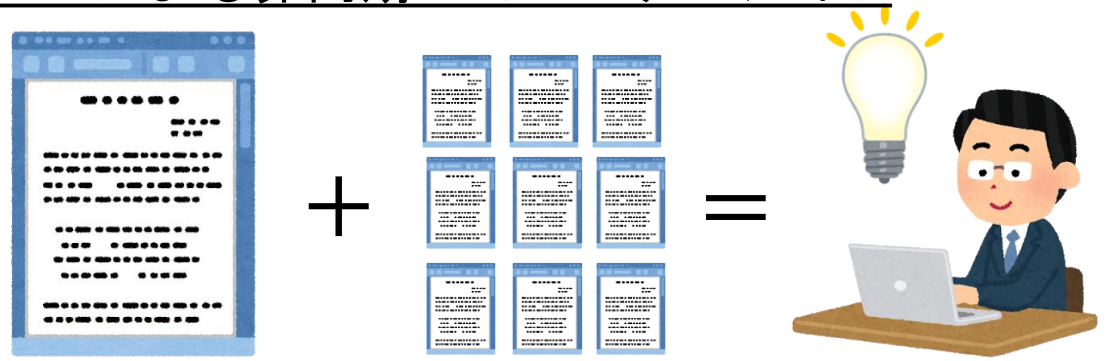
- **Remote Decision Records（RDR）を提案**
  ADRを適用したリモートワークでの
  非同期コミュニケーションを進めるための
  記録が蓄積される開発スタイルを提案する
- **RDRのプロセス概要** チームに適したツール
  - 開発用ツール(Git/Redmineなど)でRDRを管理
  - メンバーは既存のRDRを検索・参照
  - 決定・確認事項を事前にRDRで作成
  - チームに提案し，ツール上で非同期に議論
  - 決定時に，RDRの内容を更新
- **例**
  - リモートでの会議(TV会議）
  - リモートでの質問(Chat/電話）

現状の同期コミュニケーション



設計書や仕様書　＋　決定・背景・判断理由　＝　理解できる！

RDRによる非同期コミュニケーション



設計書や仕様書　＋　RDRで小さな単位で管理
どんどん追加する　＝　理解できる！

RDRの適用例： リモートでの会議(TV会議）①

2つの適用ステップ

１．議論のまとめを形式化：RDRテンプレートの活用（議事録として）

議題をRDRで記載

判断内容・決定事項などをRDRに追記

議論となったドキュメントの近くで保存

⇒ 非同期的に議論の共有可能

テンプレートの例：

```
# Title(議題)
<議論すべき事項>
## Status(状態)
<RDRの状態(オープン/一時中断/クローズ)>
## Context(経緯)
<議題の背景や調査・判断状況など経緯を簡潔に記載>
## Decision(決定事項)
<決定・合意事項や質問の回答を記載>
## Links(リンク)
<対象の文書・関連するRDR・Wikiなどを記載>
```

２．同期的な会議の削減：RDRベースで議論

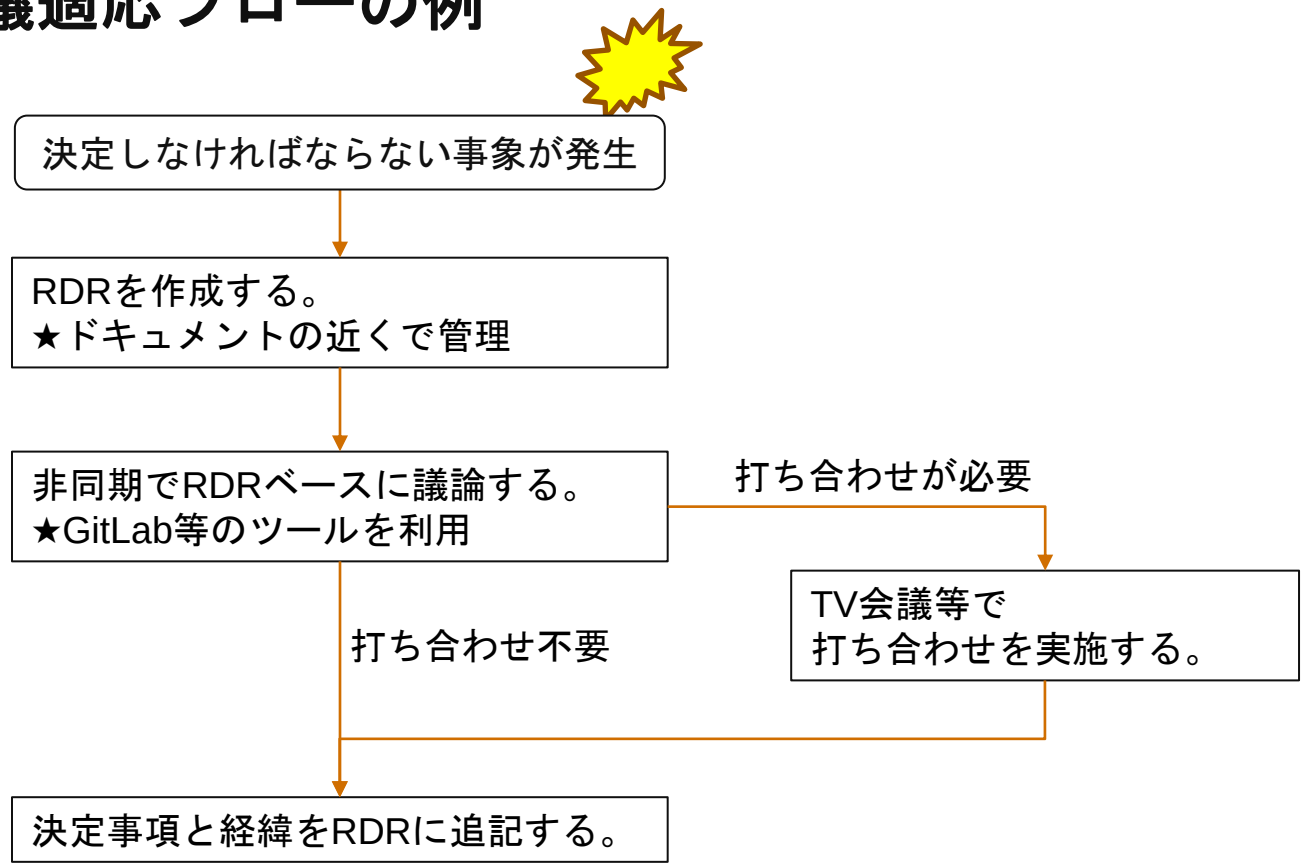
非同期で済む議論は打ち合わせしない

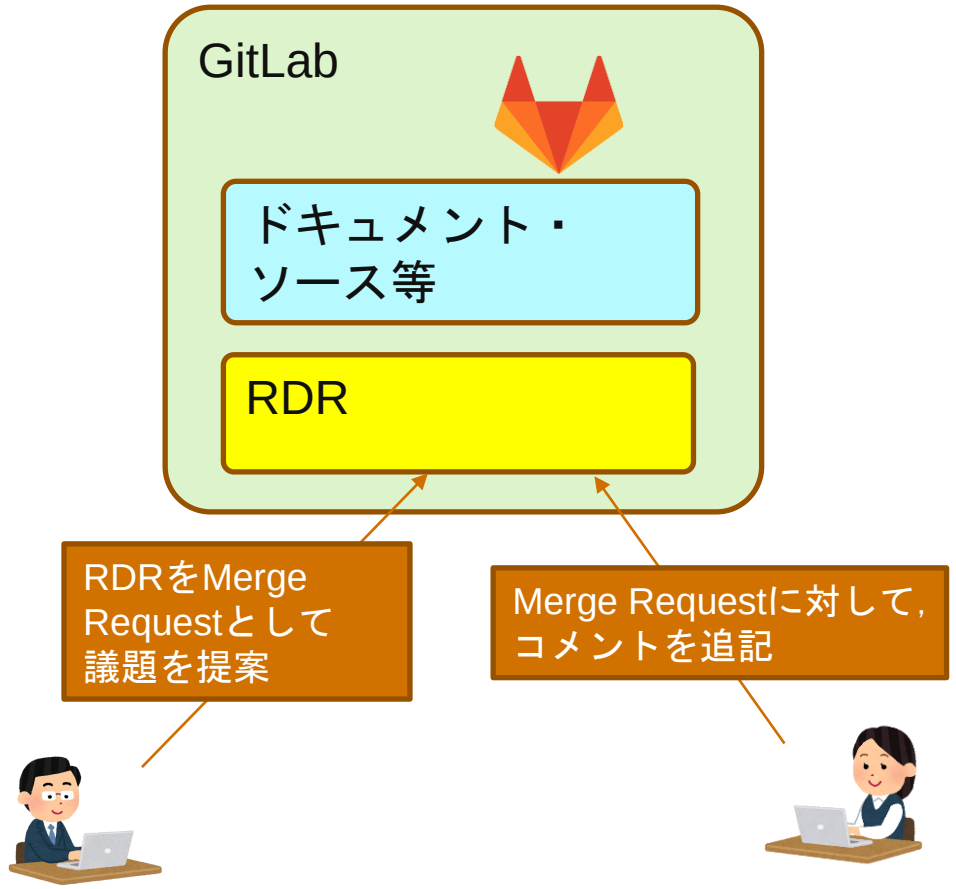
議論をGitLabなどのツールで実施

ソースコードのPullRequestなどと同様に議論する

RDRの適用例：リモートでの会議(TV会議）②

会議適応フローの例





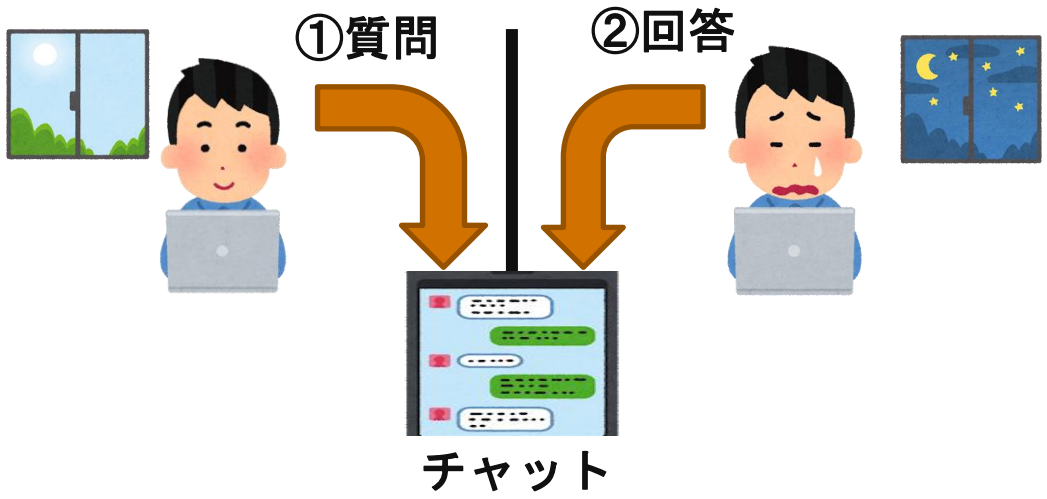
RDRの適用例：リモートでの質問(Chat/電話）

ちょっとした質問に対して　→　チャットの質問をRDR化

■Before

・同期通信を強いられる。

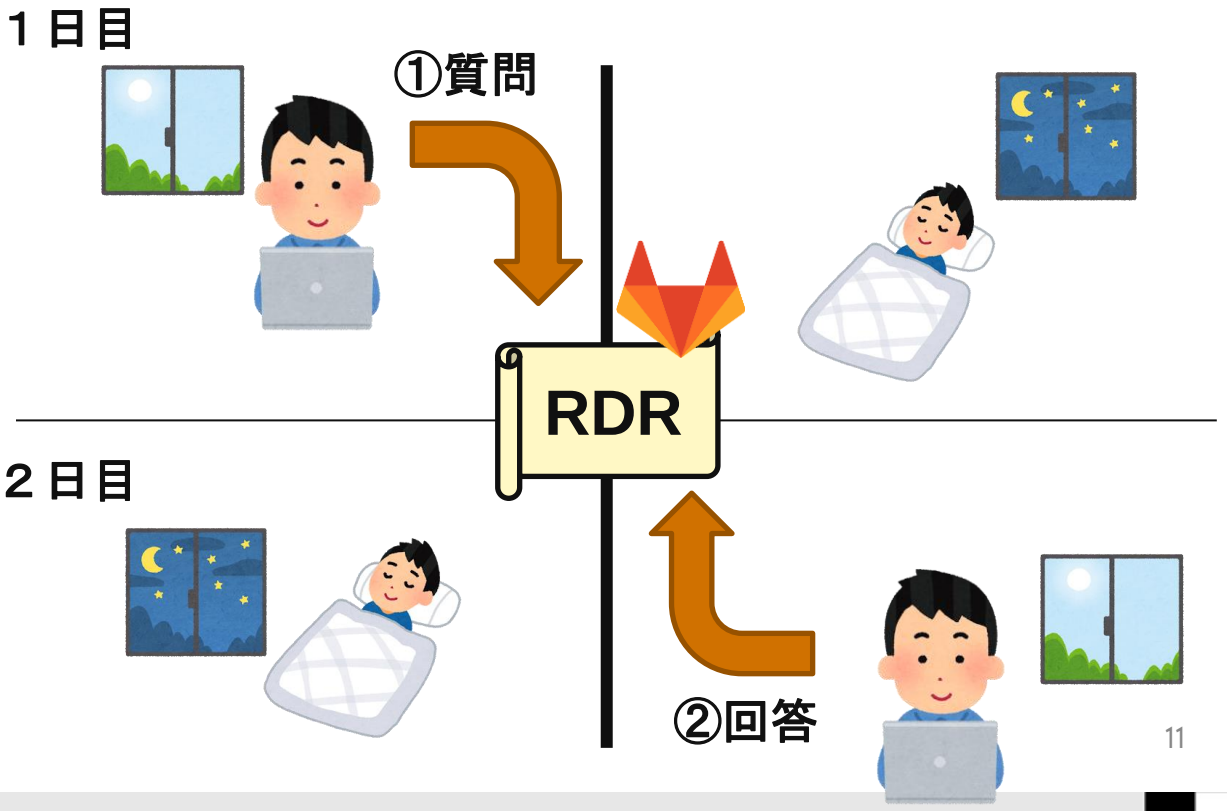
・ログが体系的に残らない。（チャットログのみ）



■After

・非同期通信で対応可能

・ログが体系的に残る。



まとめ

- **リモートワークにおける同期的なコミュニケーションによるコストが増大**
- **先端企業Gitlabを参考に，リモートワークでの非同期コミュニケーションの重要性を認識**
- **ソフトウェア開発において，設計書などの共有で課題があり，ドキュメントファーストに着目**
- **Architecture Decision Recordsを参考に，Remote Decision Recordsを提案**
  - 決定・背景・判断理由を残すことにより、非同期コミュニケーションのアシストになる
  - 開発用ツール(Git/Redmineなど)でRDRを管理
  - メンバーは既存のRDRを検索・参照
  - 決定・確認したい事項がある場合にを事前にRDRで作成
  - RDRをもとに、チームに提案し，ツール上で非同期に議論
  - 決定時に、RDRの内容を更新
- **非同期コミュニケーションを実現して，効率よいソフトウェア開発を行う必要がある.**
- **残課題**
  - 実運用ではプロセスやツール・テンプレートはチームに適したものを準備する必要がある
  - チーム内で実際に普及(動機付け)していくこと

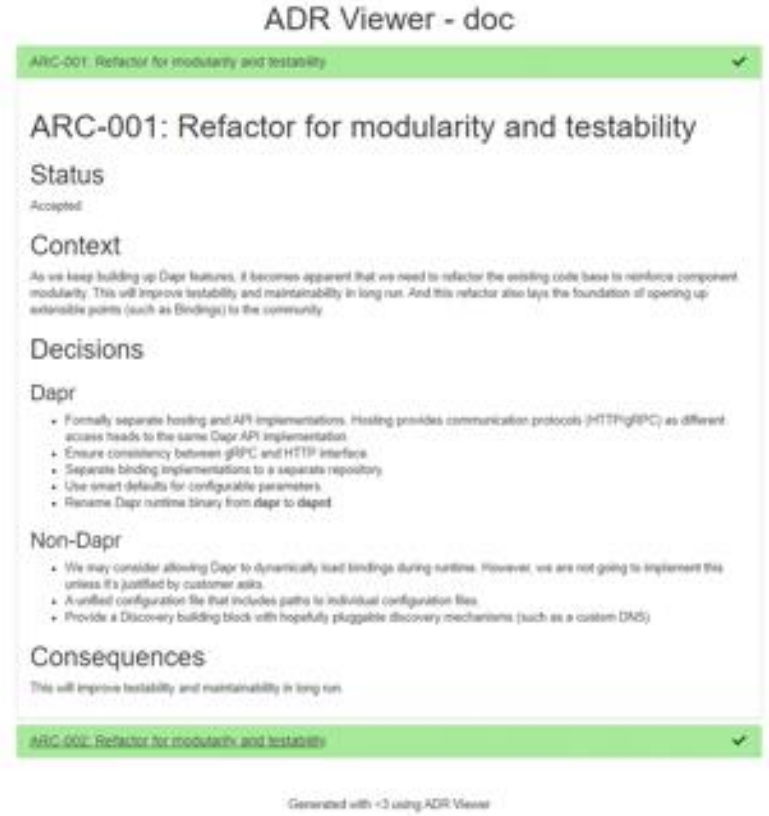
付録：RDRを職場での普及させ方

エヴェリット・ロジャースの普及学にあるイノベーション要件を元に普及させる要件をまとめた

要求	説明	何ができるか？
比較優位	従来のアイデアや技術と比較した優位性。まったく新しい技術の場合でも、同じ役目を担っていた代替案との比較になる。	・これまで、コードと別々に管理していた運用との比較。 →コードと共に管理、編集、検索を行える。
適合性	その個人の生活に対しての近さ。新規性が高くても、大きな生活の変化を強要するものだと、採用されにくい。	・コードの運用と同じである為、大きな変化が無いことを提示。 →チームで利用中の開発用ツールで実施できる。
わかりやすさ	使い手にとってわかりやすく、易しいものが採用されやすい。	・入力しやすいテンプレートを作成する。 →チームで共通の形式で理解を促進できる。
試用可能性	実験的な使用が可能だと、採用されやすい。	・小さな決定事項から試用してみる。 →部分的に重要なところから適用できる
可視性	採用したことが他者に見える度合い。新しいアイデアや技術が採用されていることが、周囲の人から観察されやすい場合に、そのイノベーションに関するコミュニケーションを促し、普及を促進する。	・RDRの一覧表示（専用ビューワーの利用） →誰でもRead/Writeできる

付録：RDRを職場での普及させる具体的な施策

可視性「見やすくする」：ADR-Viewrを利用する



わかりやすさ「書く手間の排除」：テンプレートを準備

- title
- status
- context
- decision
- consequence

ADR-TEMPLATE-01.md

ポイント：将来のアイデアも記載しておく