資料３

EPMデータに基づくプロジェクト仮想再現ツール『プロジェクトリプレイヤ』の開発

飯田元
奈良先端科学技術大学院大学/
EASE project

背景

- ソフトウェア開発プロジェクトの『失敗』の増加
  - →誤動作に基づく損失発生
  - →改修・再開発のためのコスト発生
  - →社会的信用の失墜
- 過去の失敗に学びにくい環境
  - プロダクトライフサイクルの短縮
  - 開発要員の流動化

▼

プロジェクトの事後分析による知見の蓄積と
再利用が必要

研究内容

- ソフトウェア開発プロジェクトの事後分析を支援し，開発者に知識を還元するサイクルフレームワーク「KFC」の提案
- プロジェクトを容易に振り返る為の**開発プロジェクト再現ツール**の開発
  - 様々な開発ツールのログデータをもとにした「振る舞い」をわかりやすく再現する
  - 数値にはあらわれない開発プロジェクトの**コンテキスト***をメールアーカイブから探る

*開発者同士の振る舞いや役割，開発現場の状況等

プロジェクト事後分析における課題(1)

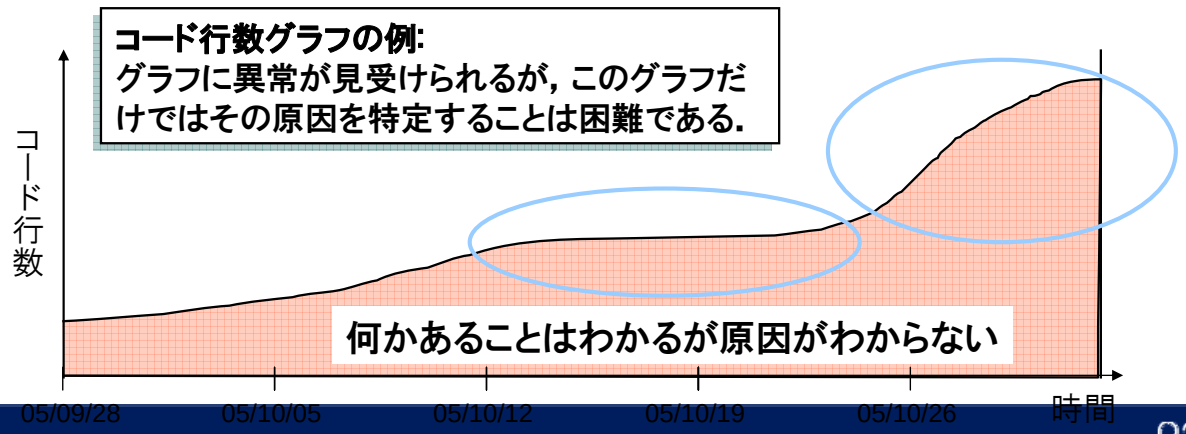
- プロジェクトを振り返るための時間やコストが少ない
- 異動や新プロジェクトへの参加などにより，開発者の協力が得られない
- 事後分析のためのドキュメントが十分に作成されていない

▼

EPMなどで自動的に収集されたデータを
振返りのために活用する

プロジェクト事後分析における課題(2)

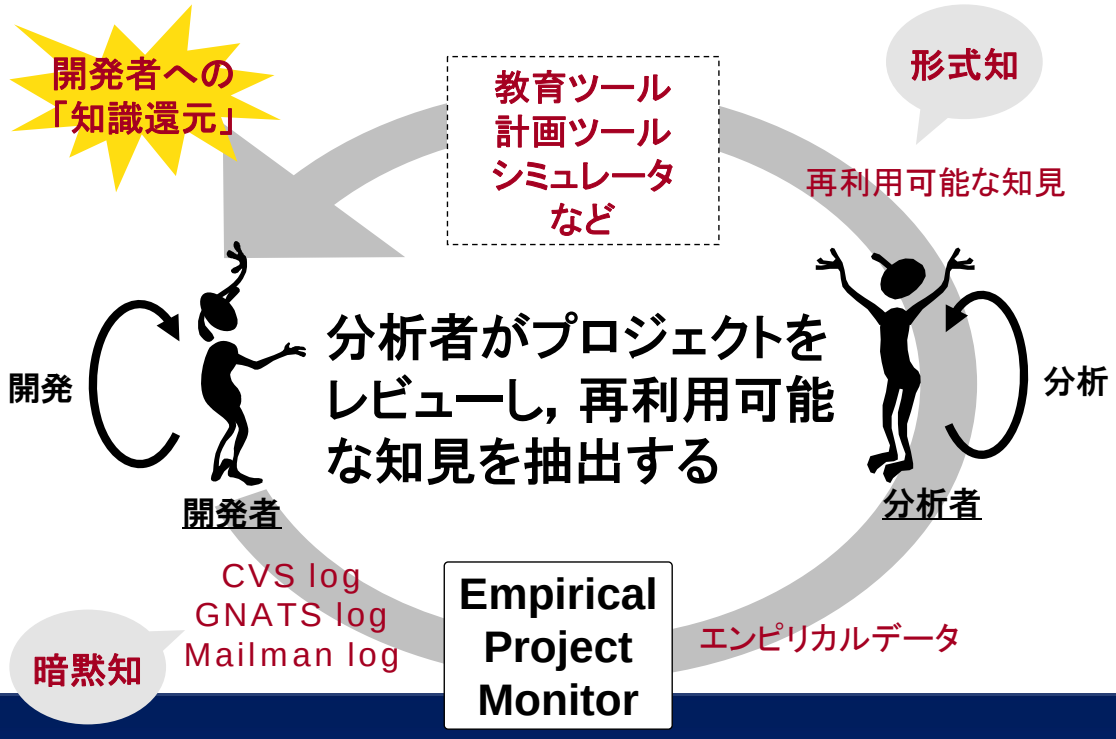
- 隠れたコンテキストの存在：開発ツールのログから得られる定量的なデータのみからわかることには限界がある



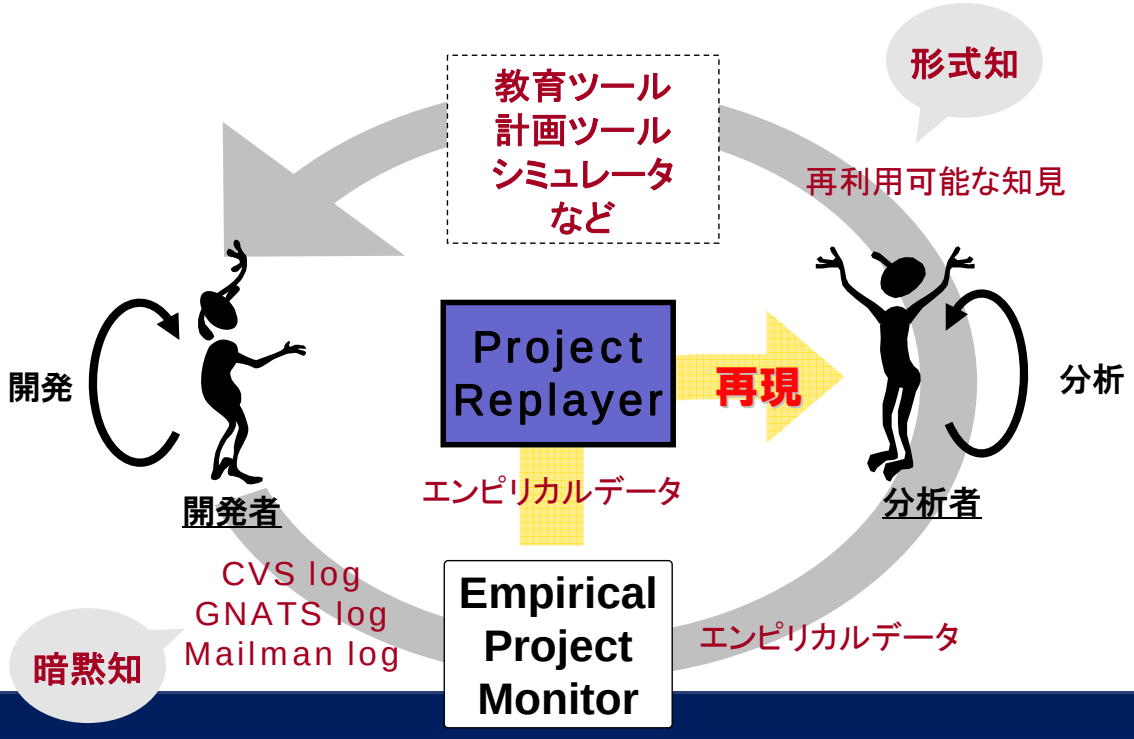
発表の流れ

- Knowledge Feedback Cycle (KFC)コンセプトの紹介
- プロジェクトリプレイヤの紹介
  - 基本機能（データ再生機能）
  - 拡張機能（メール探索機能）
- 予備実験とその結果
- まとめ

Knowledge Feedback Cycle：
開発者に知識を還元するための概念フレームワーク



プロジェクトリプレイヤ：
KFC内でのプロジェクト分析をサポートするツール



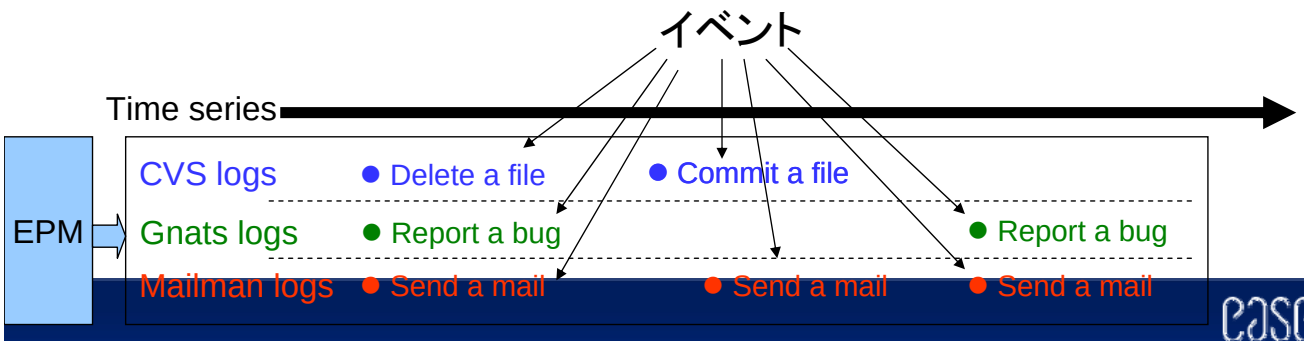
プロジェクトリプレイヤの基本機能

- 以下の情報を日付ごとに逐次表示する
  - CVSコミットログ・バグレポート・メール送受信記録の表示
  - 開発者の情報表示
  - ソースファイル完成度の表示
  - コード行数グラフの表示

起動画面

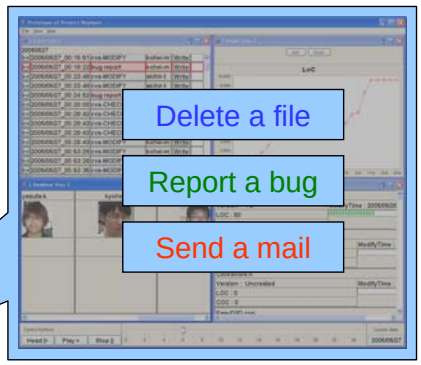
プロジェクトデータ再生のしくみ（１）

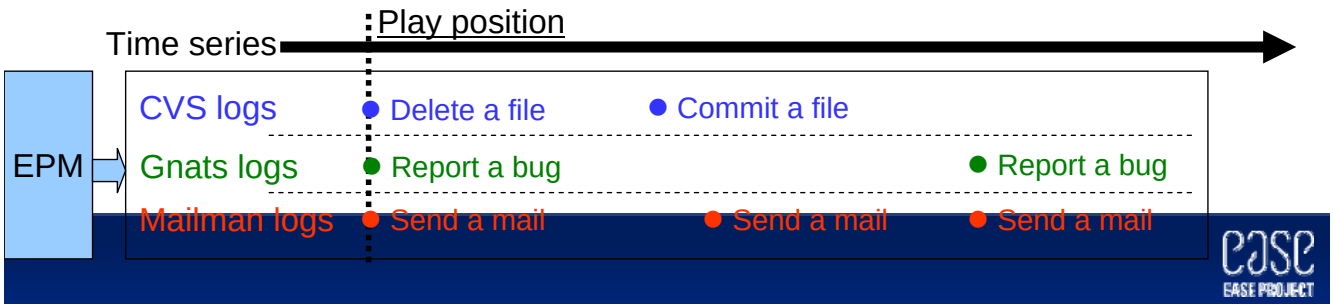
- リプレイヤはEPMにより収集されたデータを時系列順のイベントとして再構成する



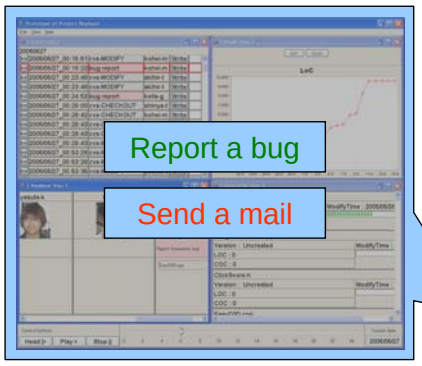
プロジェクトデータ再生のしくみ（２）

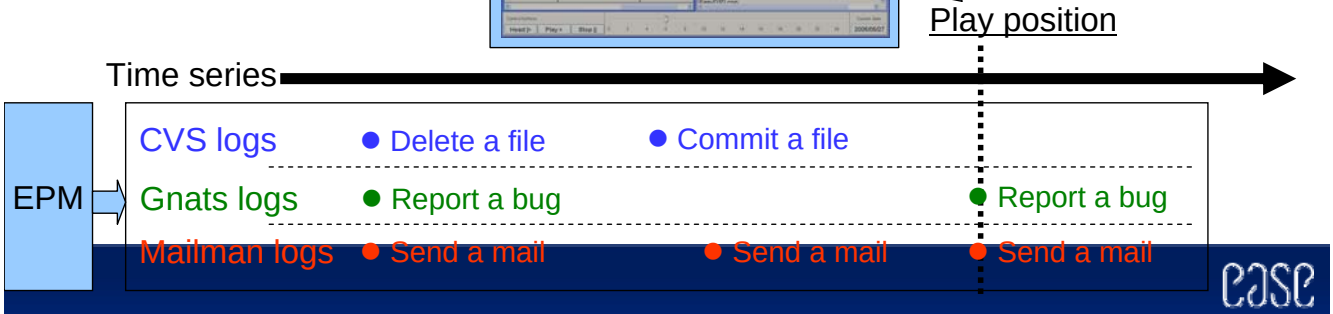
- 一定のタイムステップで再生時点を後方シフトさせ，その時点で発生しているイベント群をその種類応じて動的に表示する



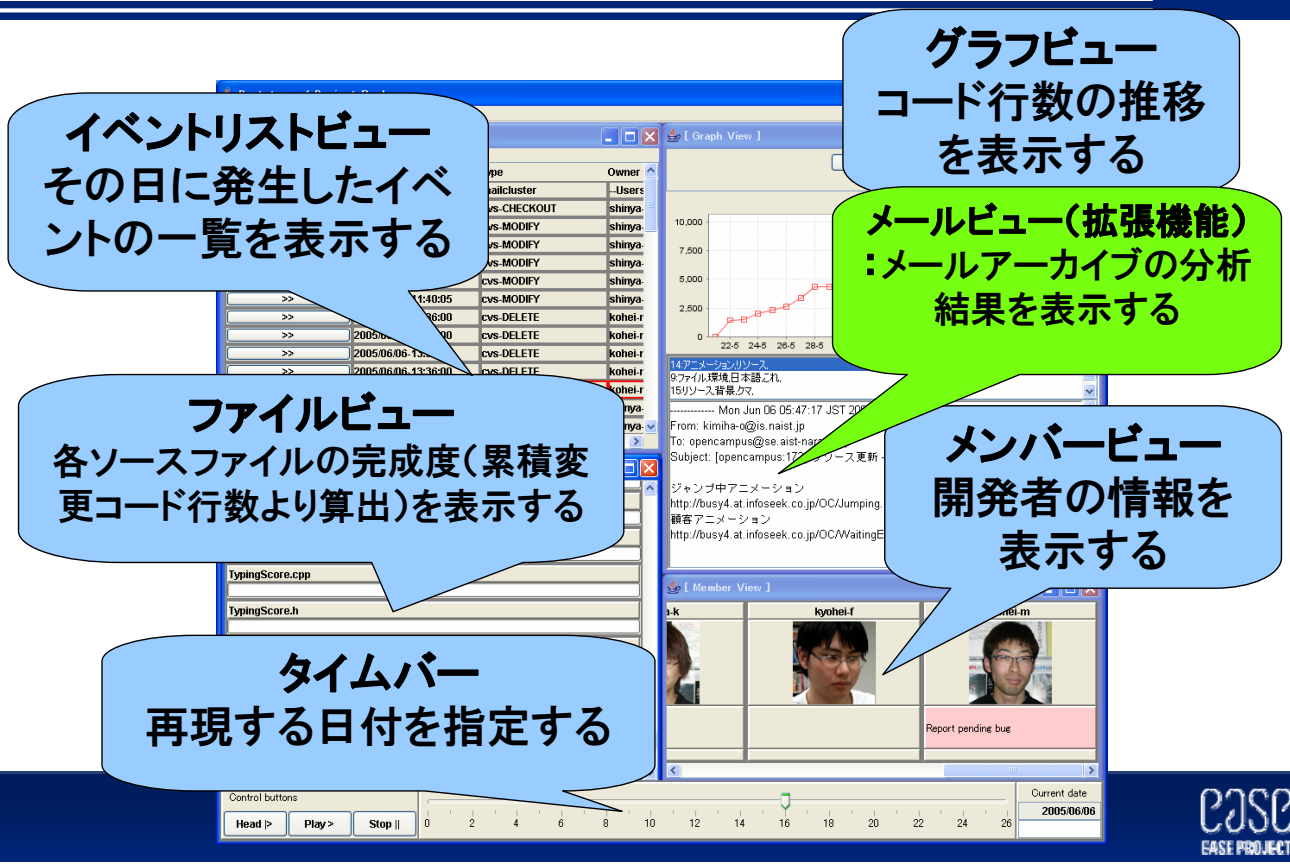


プロジェクトデータ再生のしくみ（２）





各ビューの説明



拡張機能:
メールビューによる再生時点での話題提示

- 開発用メーリングリストのアーカイブ分析によって開発プロジェクトのコンテキストを明らかにする. コンテキストには次のような事象が含まれる.
  - ミーティングの打ち合わせ
  - 細かなトラブルの情報
  - 開発に関するやりとり
  - etc...
- 自然言語処理, 特徴語抽出, クラスタリングを用いてメールアーカイブからコンテキストを抽出する.

メールアーカイブ分析手順

メールをクラスタリングすることで話題を自動分類し，
時系列上へマッピングする．

1. 類似度計算

   各メール同士で類似度を計算する

2. クラスタリング

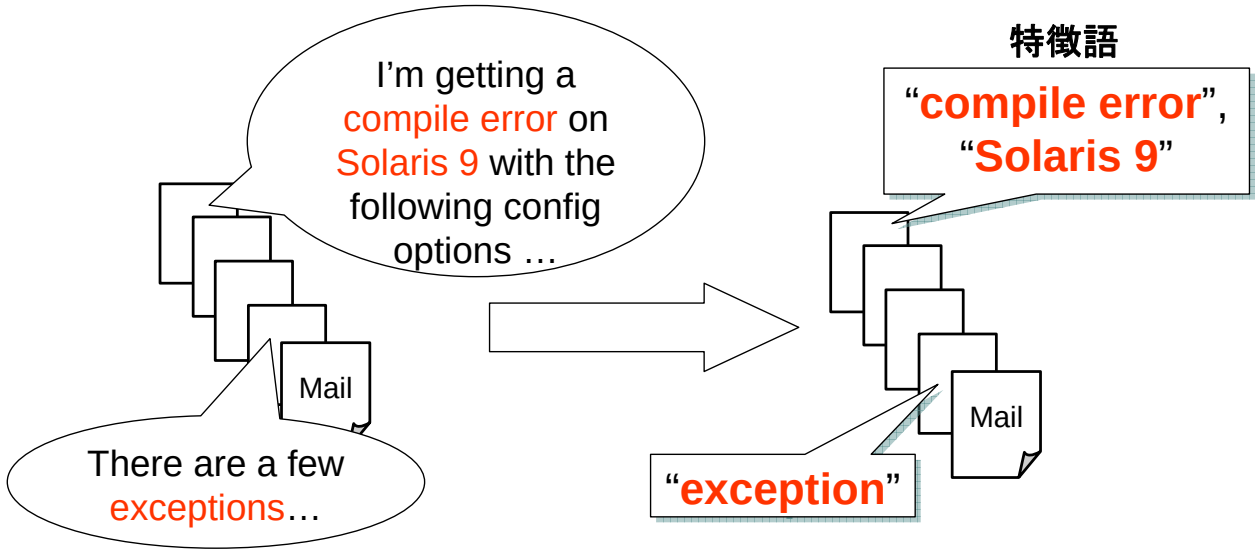
   類似度をもとにメールを話題ごとに分類する

3. 時系列表示

   時系列上にマッピングし，開発コンテキストを可視化する

類似度計算 - 特徴語抽出 -

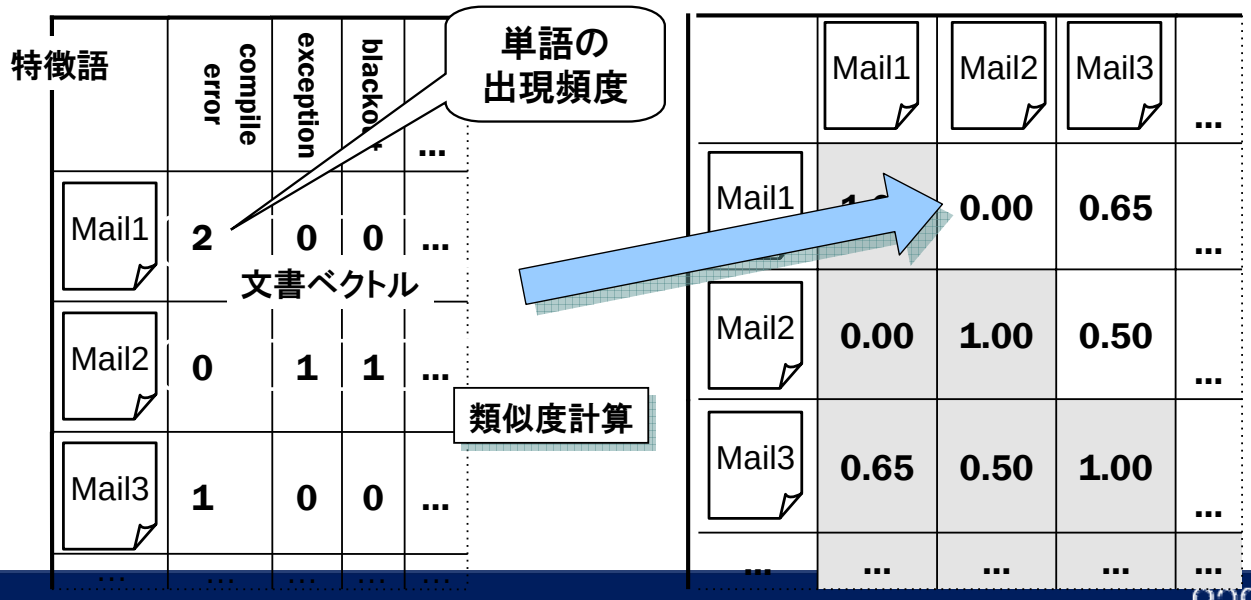
自然言語処理によってメール本文から単語を抽出，TF-IDF*を用いて単語の重み付けを行う．



*TF-IDF (Term Frequency - Inverse Document Frequency)
文書中のある単語が他の単語と比べて相対的にどれだけ重要か(その文書を特徴づけているか)を表す指標

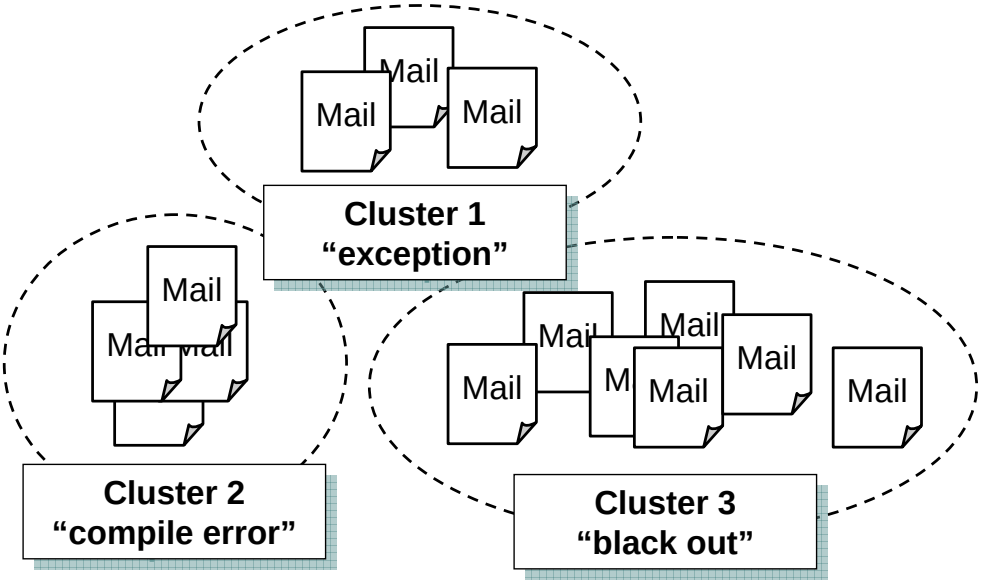
類似度計算 - ベクトル空間法 -

ベクトル空間法を用いて全メール間の類似度を計算する.



メールアーカイブのクラスタリング

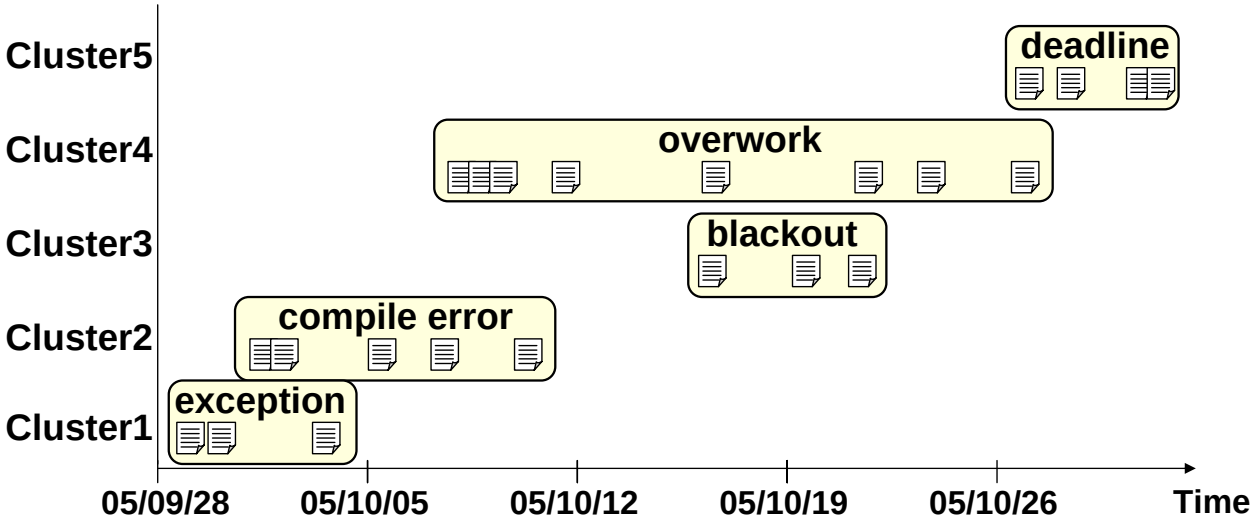
クラスタリングアルゴリズム*に基づいて，メールを話題別に分類する.



＊ 現在はk-means法を用いている

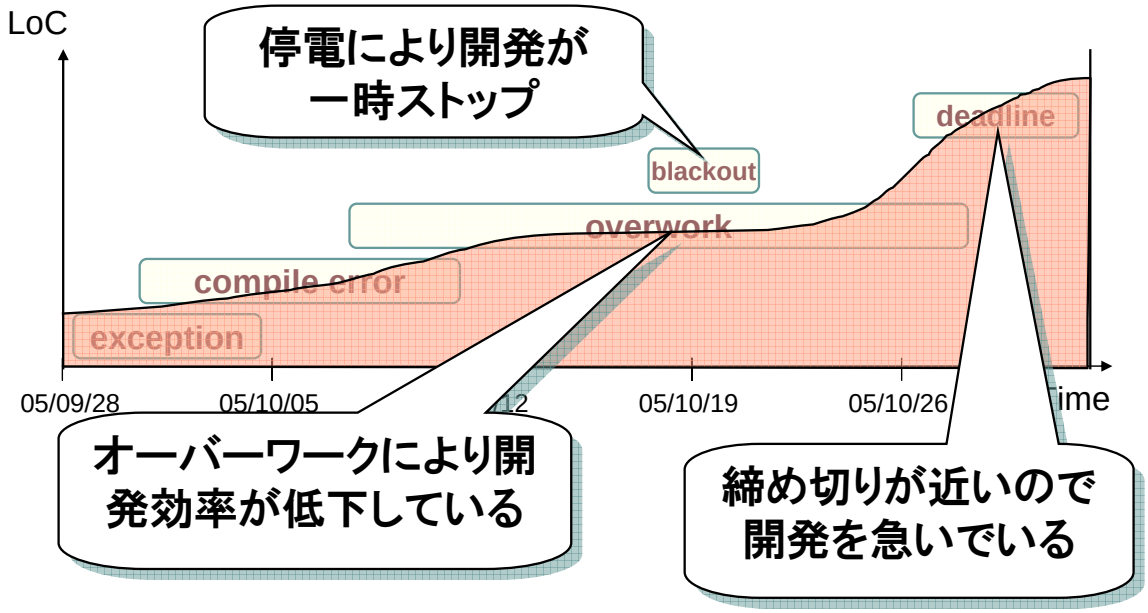
時系列表示

メール送信日時を参照し，各クラスタを時系列上にマッピングする．



時系列表示により，他の分析ツールと連携が行いやすくなる．

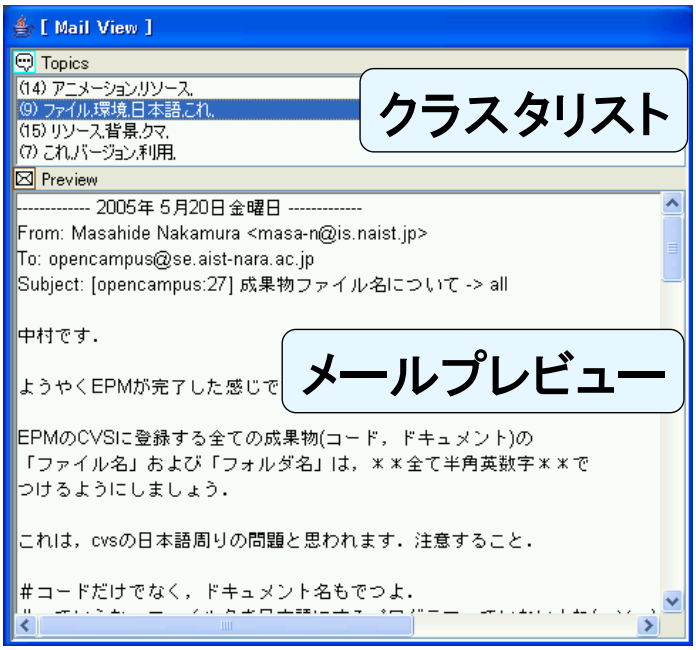
他グラフとの連携イメージ（開発中）



数値には表れないプロジェクトのコンテキストが明らかになる．

現在の実装

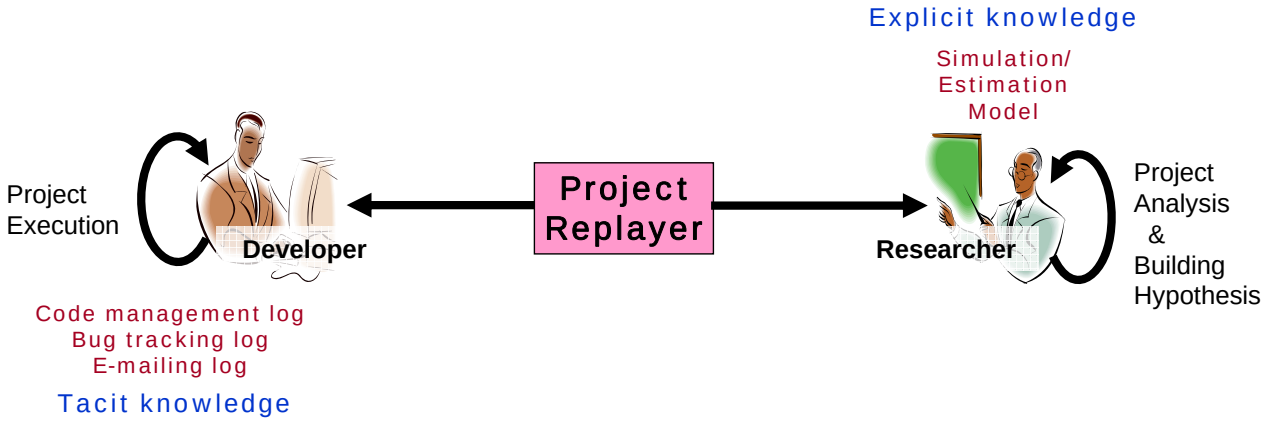
- プロジェクトリプレイヤの拡張モジュールとして実装
- クラスタ(話題)リストとそのクラスタに含まれるメールのプレビューウィンドウから構成される



予備実験とその結果

実験の目的

- リプレイヤの有効性，可用性の確認
  - 開発者，分析者が手軽にレビューできるか
  - 実際に何らかの知見を得ることができるか
  - メールビューによりコンテキストを発見できるか



実験１：リプレイヤを用いた過去のプロジェクトレビューから何らかの仮説を立て，その検証を試みる

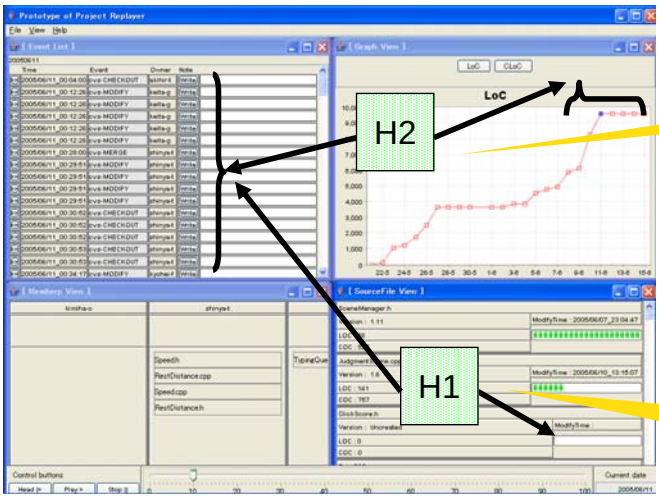
- レビュー対象プロジェクト
  - 開発対象：タイピングゲーム（NAISTオープンキャンパスのデモ用）
  - 開発要員：大学院生6名
  - 開発期間：24日（全期間をEPMにより計測済み）
  - 最終プロダクトサイズ: 9,578 lines in C++
- 実験参加者
  - 1名の分析者（プロジェクト外）
  - 3名の開発要員（元プロジェクトメンバ）

実験1の手順

- フェーズ1：仮説を提示（分析者）
  - 分析者がリプレイヤを用いてプロジェクトを分析（特に手順は指定せず）し，何らかの仮説をたてる
  - 仮説の正当性確認するために開発者向けの質問を作成する．
- フェーズ2：質問への回答作成（開発者）
  - 分析者からの質問への回答を作成する
  - 必要ならリプレイヤを用いる
- フェーズ3：仮説の確認・修正（分析者）
  - 回答に基づいて，仮説を確認・修正する
  - 必要ならリプレイヤを用いる

結果（1）：フェーズ1で得られた仮説

- (H1)「プロジェクト末期に新規作成されたモジュールの品質が低い」
- (H2)「CVS操作の振る舞いとバグレポートやメールの振る舞いが揃っていない期間はプロジェクトが混乱しており，その間に開発されたモジュールの品質が低い」



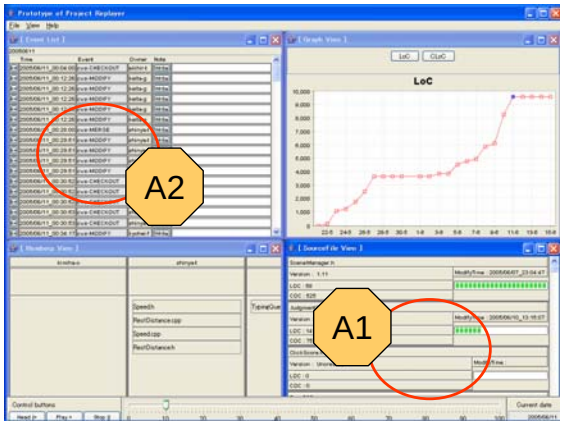
最後の三日間は，バグ報告やメール送信は行われているのに総行数が変化せず，必要な修正が行われていないのではないか．

いくつかのモジュールはプロジェクト終了のわずか数日前に作成されており，品質が低そうである

結果(２):フェーズ1で作成された質問とフェーズ2での答え

Q1:「プロジェクト末期に新規作成したモジュールの品質に問題はなかったか？」
　A1:「ほとんどのモジュールの品質に問題はなかった」
Q2:「なぜ，最後の三日間はCVSの更新がなかったの？」
　A2:「実際の納期は三日前でそのあとは保守情報の整理をしていた」

解答の作成にはリプレイヤが補助的に活用された

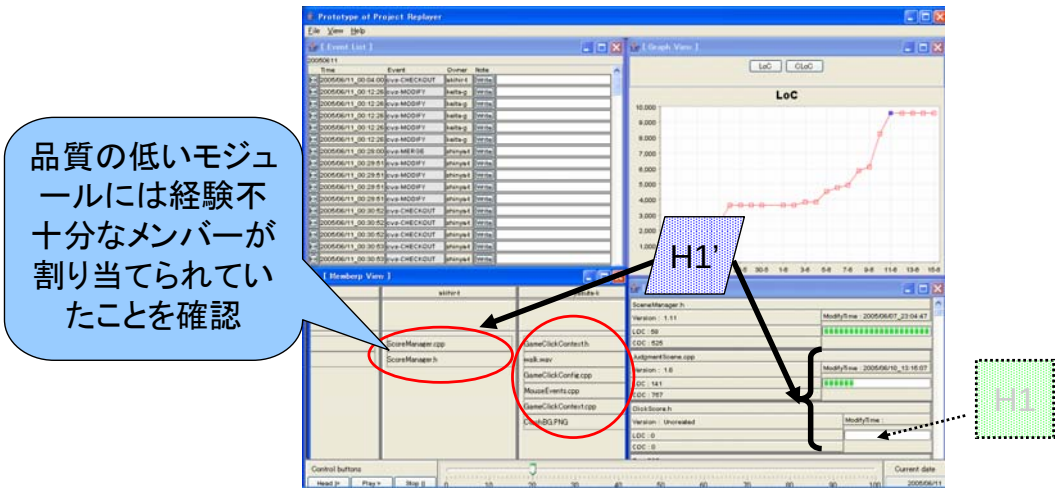


A2:最後の三日間の作業系列の確認

A1:プロジェクト末期に開発されたモジュール名の確認

結果(３):回答に基づいた仮説の再評価

- H2は単に棄却
- H1はリプレイヤを用いてさらに検討し仮説を修正
  (H1')「プロジェクト末期に類似機能の開発経験の無いメンバーにより新規作成されたモジュールの品質が低い」



実験2：メールビューを追加したリプレイヤの試用1

- メールビューを追加した新プロジェクトリプレイヤを用いることによって，対象プロジェクトの流れ・コンテキストを把握できるかどうか確認する
- レビュー対象：実験1と同じ（学生による開発）
- 実験参加者：分析者1名（開発に参加していない外部研究者）

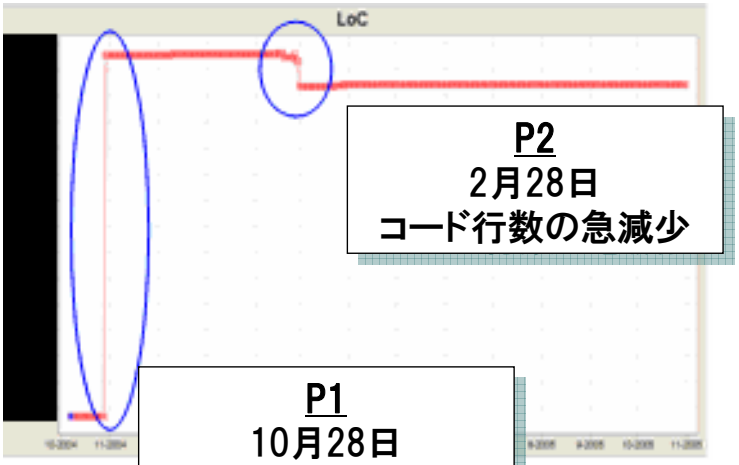
実験2の結果 -得られたコンテキスト-

- ドキュメントに残されていない小規模なミーティングの存在を，メールの分析により知ることができた
- 5月27日から6月2日までの間，開発が停止していた．
  - CVSコミット・バグレポートも停止
  - メールビューを用いても停止の理由は明らかにできなかった
  - このプロジェクトではメールを主に次のような用途のみに使用していた：
    - ミーティングスケジュールの調整
    - コード修正のアナウンス
    - リソースのアップロード報告

実験3：特定現象に対する原因の追跡

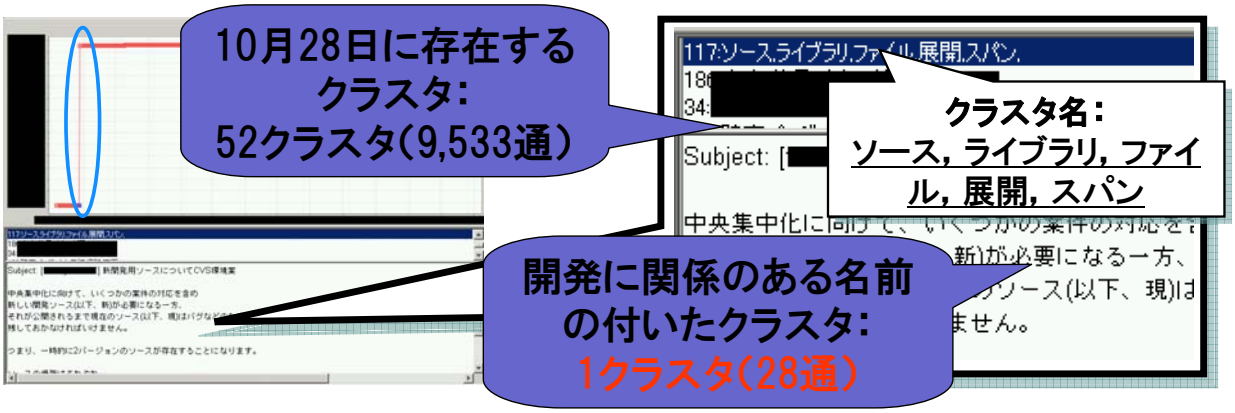
Project Replayerのグラフビューによって，コード行数の異常な変化が2カ所発見された（P1，P2）．これらの原因を，メールアーカイブ分析を用いて明らかに出来るか検証する．

- 対象：ソフトウェア開発企業のデータ（保守フェーズ）
- 規模：大規模（データ収集期間約500日・開発者数10人以上）
- 被験者：2人（開発に参加していない外部研究者）



実験3 -結果- (1/2)

- P1：10月28日の急増原因 -

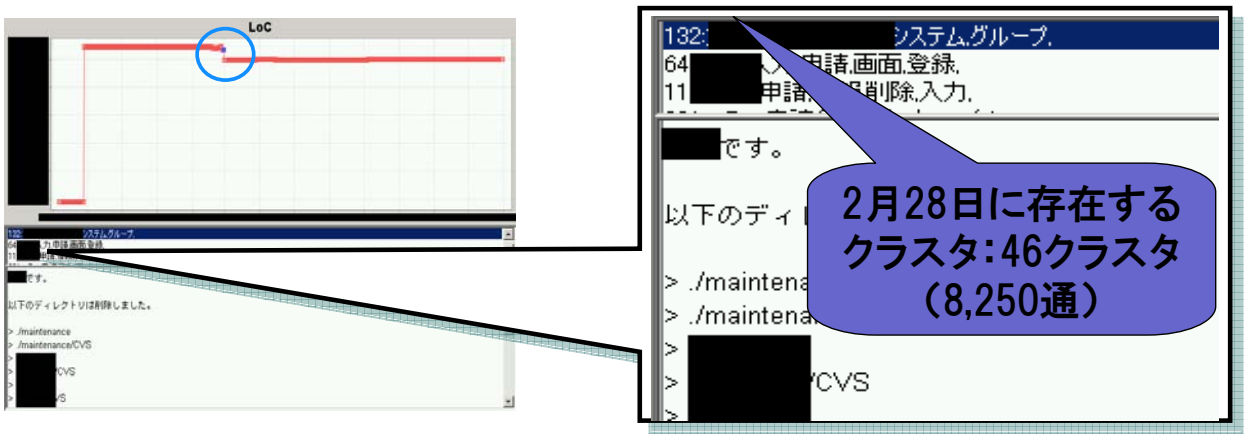


上記28通のメールから，コード行数急増の原因が「開発データの中央集約化の為に行ったCVS環境の新構築」であることがわかった．

更に，中央集約化を行うことになった理由が「旧CVSサーバのハードディスク障害」であることがわかった．

実験3 -結果- (2/2)

- P2：2月28日の減少原因 -



クラスタリストを巡覧することにより，原因を明らかにすることができた．原因は「開発言語の変更に伴う不要ファイルの一斉削除」であった．

まとめ，総論

- リプレイヤの評価
  - 試用者からはおおむね好評
  - 低コストで事後レビューを実現可能
- 課題
  - ツールの性質上，有用性に対する客観的評価が困難
    - 実プロジェクトに対する適用と知見の蓄積が必要
  - 効果的な利用方法のガイドライン
    - レビュー手順
    - 知見のまとめ方，再利用手段