

エンピリカルデータを対象とした マイクロプロセス分析

奈良先端科学技術大学院大学
情報科学研究科

森崎 修司 松村 知子 大蔵 君治
伏田 享平 川口 真司 飯田 元

あらまし

- 背景
- マイクロプロセス分析
 - エンピリカルデータ(作業記録)
 - プロセスモデル
 - プロセスモデルに基づくエンピリカルデータの解釈
- マイクロプロセス分析の試行
 - 試行結果
 - 考察
- まとめ

研究の背景: プロセス品質の計測

- ソフトウェアプロダクトから品質予測をするのと同様にプロセスからソフトウェアの品質を予測したい.
- 問題点
 - プロセスデータの多くは人手により収集されるため, 継続的, 統一的なデータ収集は難しい.
 - プロセスのみのデータを自動収集するのは難しい.
 - 例: HumphreyのPersonal Software Process
 - 作業記録は手動によるもの
 - 開発者に対するトレーニングが必要

研究の背景：エンピリカルデータ収集環境の普及

- エンピリカルデータ(開発作業の記録)を収集する研究が進み環境が整いつつある.

変更履歴の例)

日付	操作	ファイル名	Version	コメント
2003-08-07 10:5...	CHECKOUT	EmpiriPrj		
2003-08-07 11:5...	CHECKOUT	EmpiriPrj		
2003-08-07 12:0...	CHECKOUT	EmpiriPrj		
2003-08-07 12:0...	CHECKOUT	EmpiriPrj		
2003-08-07 12:5...	CHECKOUT	EmpiriPrj		
2003-08-07 13:1...	CHECKOUT	EmpiriPrj		
2003-08-07 13:1...	CHECKOUT	EmpiriPrj		
2003-08-07 13:1...	ADD	EmpiriPrj/XML/onTaira	1.1	実装作業の前に、Antを利...
2003-08-07 13:1...	ADD	EmpiriPrj/XML/build.xml	1.1	実装作業の前に、Antを利...
2003-08-07 13:1...	ADD	EmpiriPrj/XML/build.sh	1.1	実装作業の前に、Antを利...
2003-08-07 13:2...	CHECKOUT	EmpiriPrj		
2003-08-07 14:4...	ADD	EmpiriPrj/XML/docs/Empiri...	1.1	XMLを解析して、データ...
2003-08-07 15:3...	CHECKOUT	EmpiriPrj		
2003-08-07 15:4...	CHECKOUT	EmpiriPrj		
2003-08-07 15:4...	CHECKOUT	EmpiriPrj		
2003-08-07 15:4...	CHECKOUT	EmpiriPrj		
2003-08-07 15:4...	ADD	EmpiriPrj/translator cvs/C...	1.1	初期登録 : Translator (cv...
2003-08-07 15:5...	CHECKOUT	EmpiriPrj/translator cvs		
2003-08-07 15:5...	CHECKOUT	EmpiriPrj/translator cvs		
2003-08-07 20:3...	ADD	EmpiriPrj/XML/src/empiric...	1.1	XMLを解析し、値をデー...
2003-08-07 20:3...	ADD	EmpiriPrj/XML/src/empiric...	1.1	XMLを解析し、値をデー...

最新

刻

作業記録

目的とアプローチ

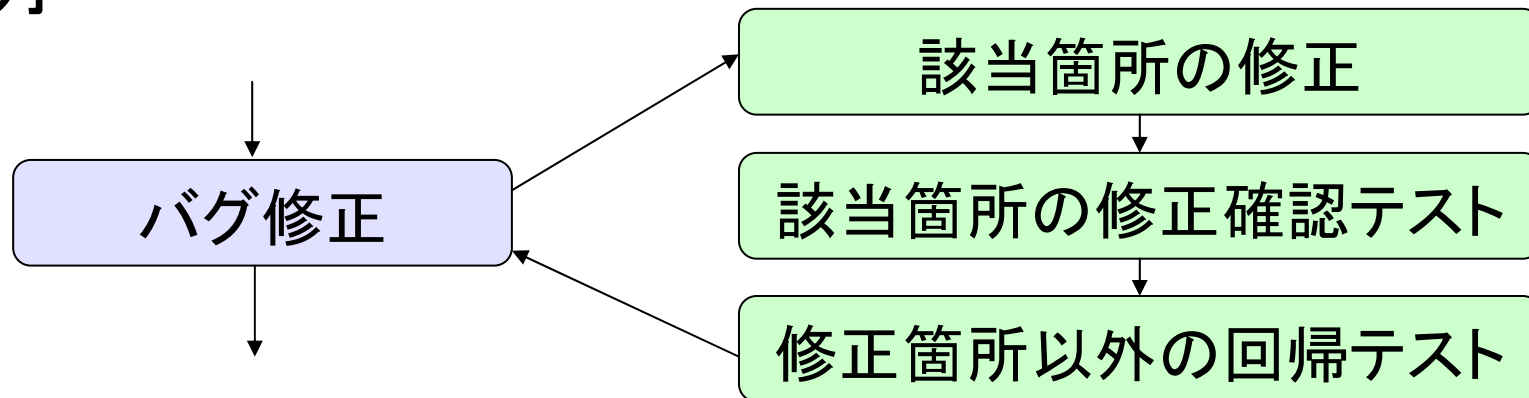
- 目的
定量的な指標を自動計測することによりプロセスの品質を測る(マイクロプロセス分析).
- アプローチ
 - 作業記録(エンピリカルデータ)から直接プロセスに関する情報(作業の順序, 時間, 作業量)を得る.
 - 基準となるプロセスモデルを与えることで作業記録からプロセスインスタンスを得る.
 - プロセスインスタンスからプロセスの品質を計測する.

作業記録

- $\langle t, s, a \rangle$ の3字組形式で記述する。
 t : 時刻, s : イベント種類, a : イベント属性の集合
- 成果物管理ツール, 編集ツールの実行履歴等から自動収集する.
- 例) $\langle 2006/12/2\ 13:07, \text{更新}, \{\text{ファイル名}=\text{”基本設計書.doc”}$ 更新者=木村, 更新内容=”レビュー指摘事項 No. 0004を反映した修正” } \rangle
→2006/12/2 13:07にファイル「基本設計書.doc」を更新者「木村」が変更し, 更新内容が「レビュー指摘事項No. 0004を反映した修正」

ソフトウェアプロセスモデル

- ここでの定義：ソフトウェア開発における作業間の順序関係や包含関係を記述したもの
例



- 利用可能な記法の例
 - WBS (Work Breakdown Structure)
 - アクティビティ図 (UML)

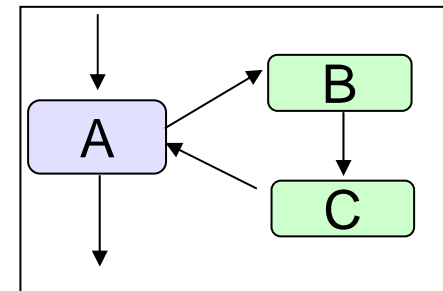
プロセスモデルに基づいた作業記録の解釈

- プロセスモデルを適用することにより，作業記録からプロセスインスタンスを取り出す。

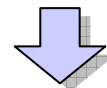
2006/11/24 11:08 イベントA_{開始}
2006/11/24 11:34 イベントB
2006/11/24 13:15 イベントC
2006/11/24 13:21 イベントA_{終了}
2006/11/24 18:10 イベントA_{開始}
2006/11/24 18:10 イベントB

作業記録

+



プロセスモデル



2006/11/24 11:08 イベントA₁開始
2006/11/24 11:34 イベントB₁
2006/11/24 13:15 イベントC₁
2006/11/24 13:21 イベントA₁終了
2006/11/24 18:10 イベントA₂開始
2006/11/24 18:10 イベントB₂

プロセス
インスタンス

様々なプロセスメトリクスの収集

- プロセスインスタンスからプロセスメトリクスを計測し品質予測に役立てる.
- 手順的メトリクス
手順漏れ, 遵守度合い, 登録情報漏れ, 並列度
- 回数的メトリクス
繰返し回数
- 時間メトリクス
所要時間, 全時間対やり直し時間比, 全時間対待ち時間比, 全時間対同期/合流待ち時間比,
遅延(ロスタイム)

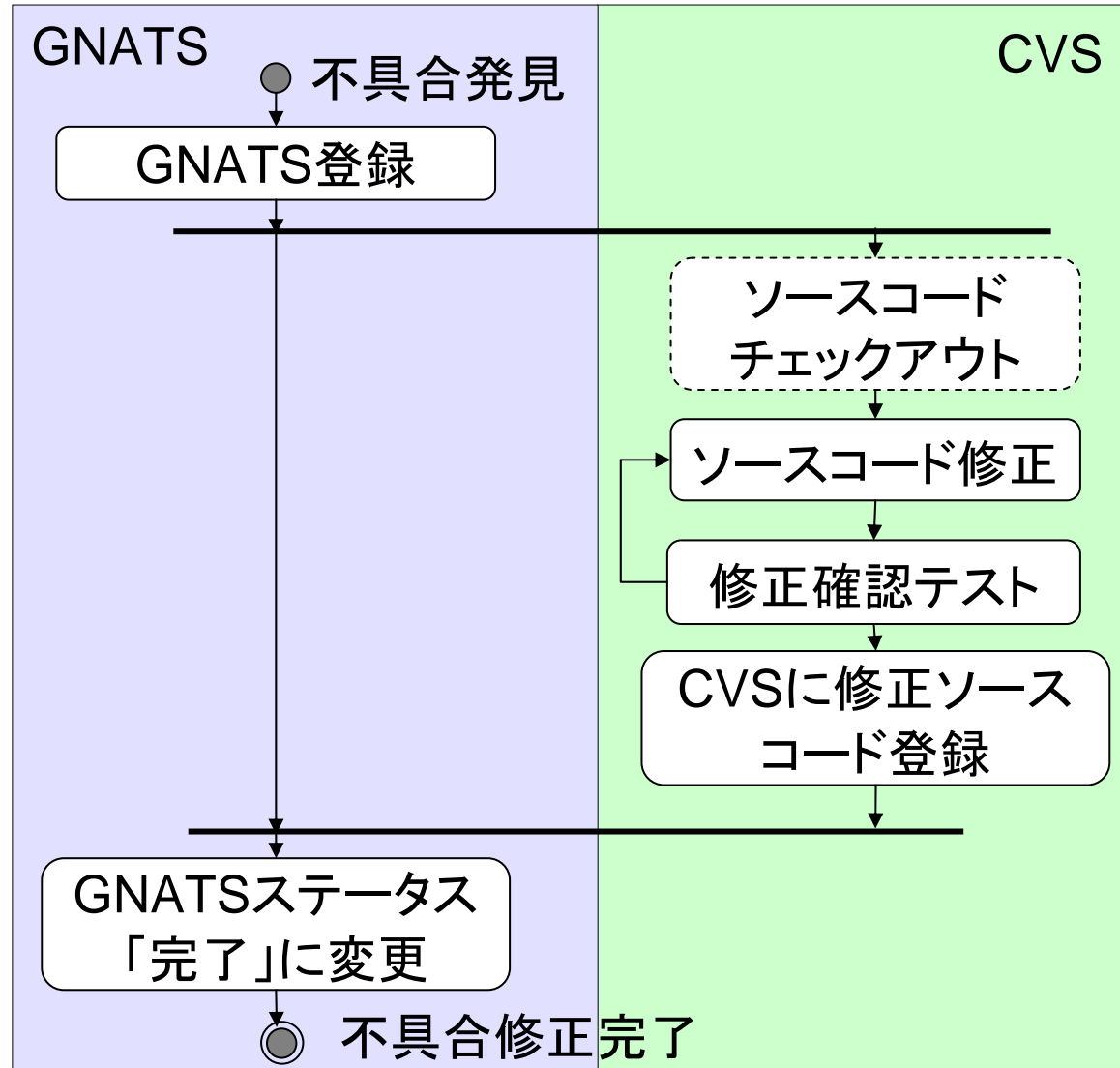
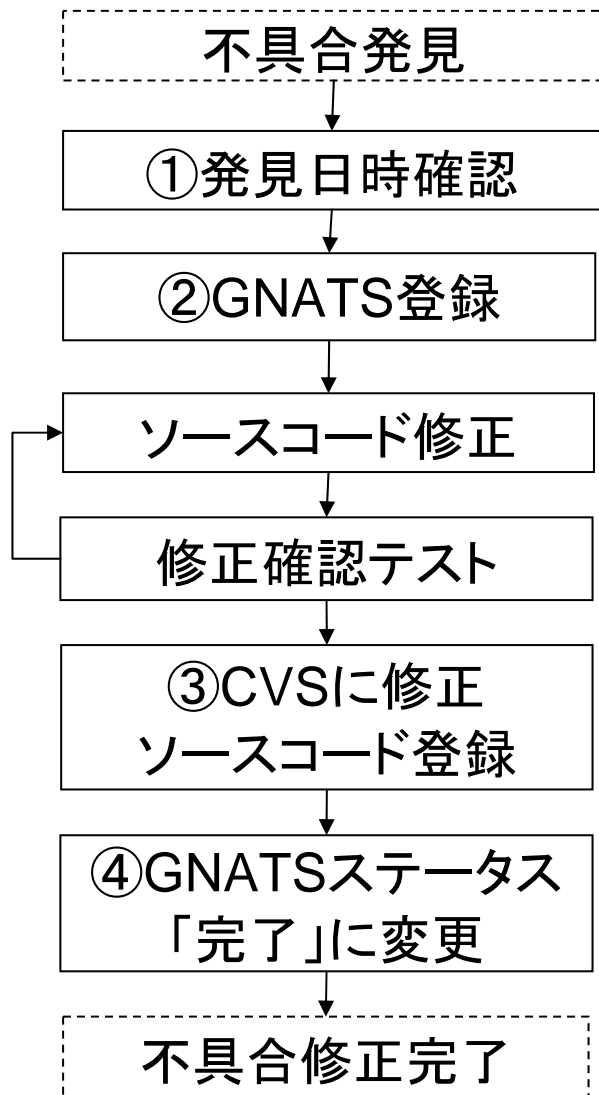
マイクロプロセス分析の試行

- 目的: 作業記録からプロセスインスタンスが得られることを確認する.
- 対象プロジェクト
 - プローブ情報システム開発
 - マルチベンダによるウォーターフォール型中規模開発
- 分析の対象
 - 工程: コーディング工程後半～組織内結合試験工程
 - 作業記録: CVSとGNATS (バグ管理ツール)
 - CVS記録: 518レコード, バグ数: 31件

対象プロセスモデル

開発メンバーにお願いした手順

想定プロセスモデル(アクティビティ図)



各手順で記録される情報の詳細

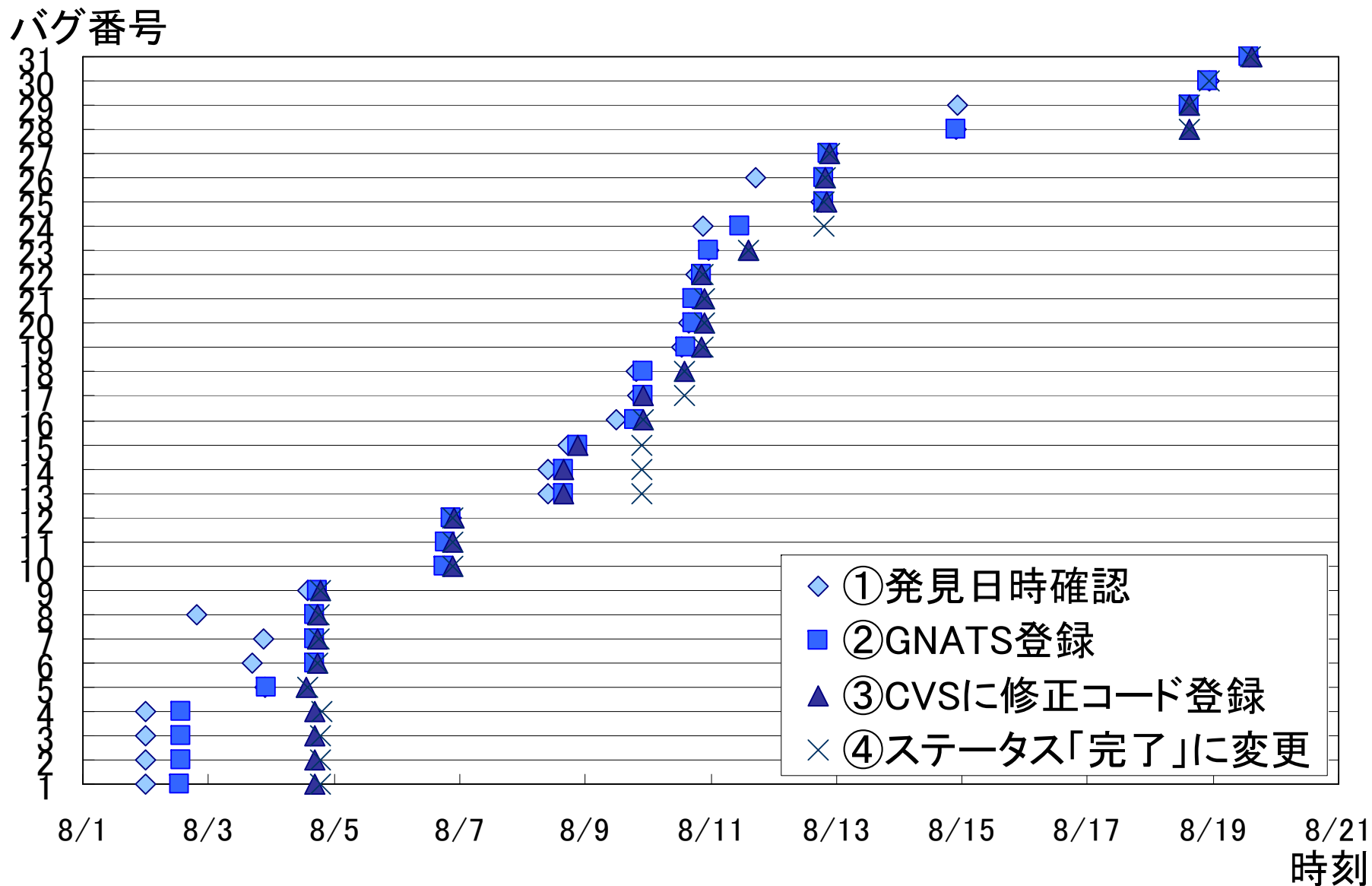
手順	登録内容
①発見日時確認	なし(日時の目視のみ)
②GNATS登録	①の発見日時, バグの説明, 再現方法, 修正担当者
③CVSに修正ソースコード登録	②の登録時に自動で振られるバグ番号, 修正担当者
④GNATSステータス「完了」に変更	なし

! 今回のデータでは, CVSへの修正ソースコード登録時に GNATSに登録されたバグ番号を手動入力されているため, ツールログ間での手順の対応が容易に判定可能であった。

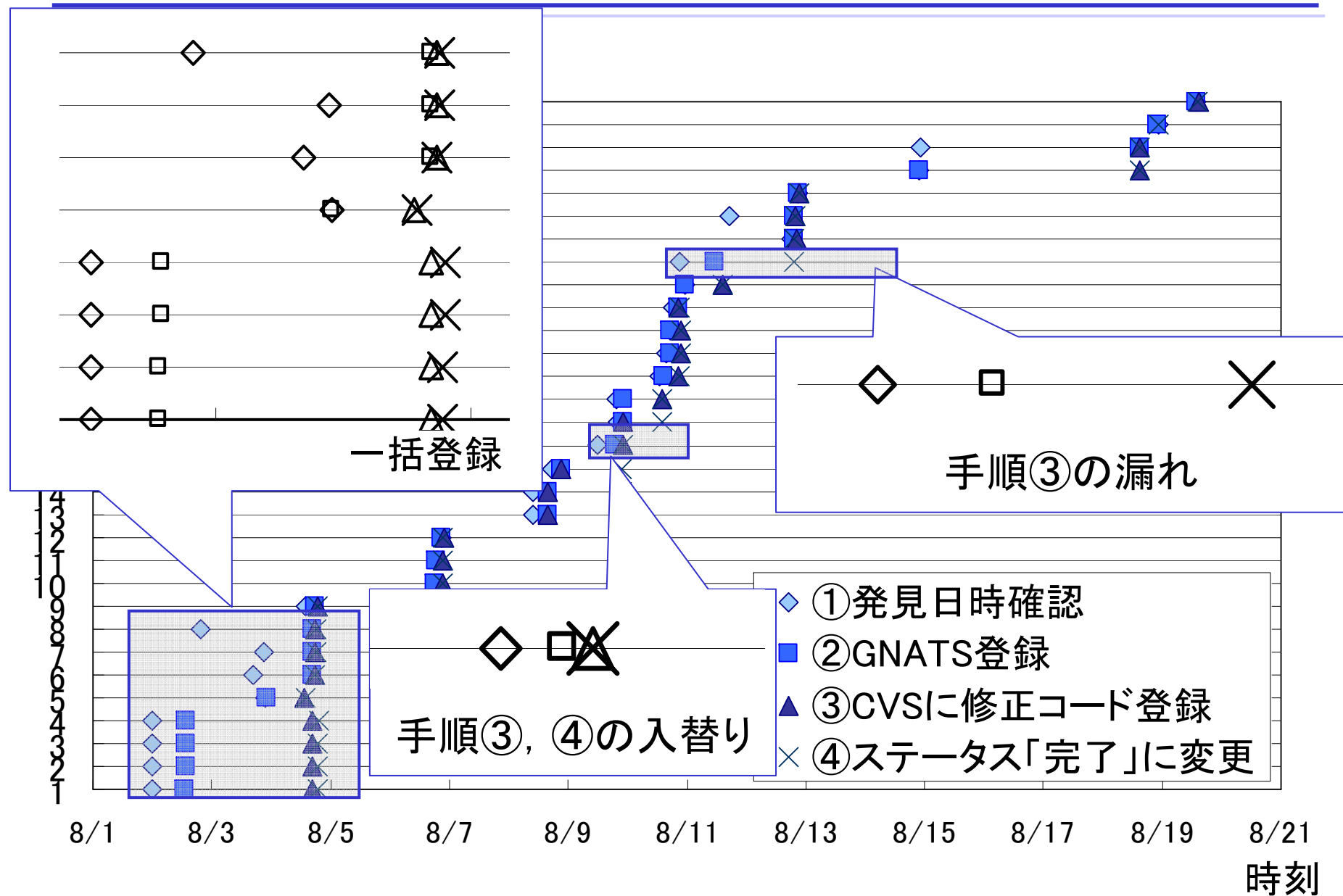
分析の実施

- プロセスインスタンスの同定
 - 人手により, 31個の作業系列をデバッグプロセスとして同定
 - 記述形式の整合性をとるための作業などが主
 - 系列のマッピング自体は機械的に実施可能
 - 所要時間: 3時間

結果: プロセスインスタンスの時系列プロット



結果: プロセスインスタンスの時系列プロット



結果：手順漏れ，不遵守，登録情報漏れ一覧

項目	該当件数	割合
手順漏れ(②)	0	0%
手順漏れ(③)	2	6.5%
手順漏れ(④)	3	9.7%
手順の不遵守(①→②)	7	22.6%
手順の不遵守(②→③)	0	0%
手順の不遵守(③→④)	7	22.6%
登録情報漏れ，誤り(②)	7	22.6%
登録情報漏れ，誤り(③)	0	0%

- ①発見日時確認，②GNATS登録，
- ③構成管理ツールに修正ソースコード登録
- ④GNATSステータス「完了」に変更

考察(1/3): プロセスインスタンスの同定

- 人手によっても比較的短時間(数時間)でエンピリカルデータからプロセスインスタンスを同定できた.
 - プロセスモデルに曖昧さがなく, エンピリカルデータ内に作業の特定に有益な情報が含まれるため
 - 曖昧さを含む(複数の解釈が可能な)モデルへの対応や, 作業特定のための情報が不足する場合への対応が今後の課題

考察(2/3): メトリクス値の計測

- 手順的メトリクス(手順漏れ, 手順の不遵守, 登録情報漏れ), 時間的メトリクス(所要時間)の値が得られた.
 - 今後起こりうる作業漏れや手戻りを未然に防ぐための判断材料になる.
例)一括登録は作業漏れにつながるので, その都度の更新を徹底する.
 - プロジェクト進行中に手順逸脱を指摘できる.
例)「③修正ソースコード登録」から24時間以内に「完了」にならなければ指摘する.

考察(3/3): プロセスモデル自体の評価

- プロセスモデル自身の評価や改善が期待できる。
 - 手順の入替えや入替りの許容
例) 発見→起票の手順の不遵守が多く発見された。
2つの手順を1つに集約, 短時間であれば手順が入れ替わっていても許容する。
 - 手順の並列化/一括登録の許容
例) 複数バグに対して複数のソースコード修正の登録を一括登録していた。
時間間隔が短ければ問題にならない上に, 作業時間が効率化できる。

まとめ

- プロセスの品質を計測するためにマイクロプロセス分析を提案した。
 - プロセスモデルを用いて作業履歴からプロセスインスタンスを得る.
 - プロセスインスタンスからプロセスメトリクス(手順の抜けや情報漏れ)を計測する.
 - バグ修正のプロセスが定義された商用開発プロジェクトのデータに適用した.

今後の課題

- プロセスインスタンスの(半)自動解釈
 - あいまいさを含む文法の解析手法
 - 統計的な推定

- プロセスメトリクスの洗練
 - リスク要因との関連づけ
 - メトリクスの分類
 - プロダクト品質との相関