

DynC要素技術: Inter-Community Metrics (ICM)

EPMを利用した
OSSプロジェクト間の
類似性・相違性分析

大平 雅雄 (Masao Ohira)
奈良先端科学技術大学院大学
エンピリカルソフトウェア工学ラボ
ohira@empirical.jp

発表の概要

- 前回までのおさらい
 - EASEプロジェクト
 - Empirical Project Monitor (EPM: データ自動収集・分析ツール)
 - EPMの分析結果例
- EPMを利用したOSSプロジェクト間の類似性・相違性の分析 (Inter-Community Metrics: ICM)
- 今後の予定

EASEプロジェクト

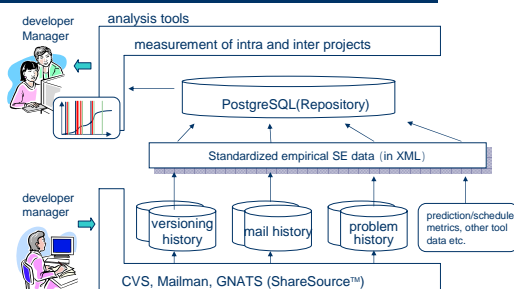


- 何万ものプロジェクトを扱うことのできるエンピリカルソフトウェア工学環境 (ESEE: Empirical software engineering environment) の構築
 - 大規模データ収集
 - 集約的データ分析
 - 改善のためのフィードバック
- 主なターゲットは多数のプロジェクトを抱える組織やコミュニティ

EPM: Empirical Project Monitor

- ESEEの一部として開発中
- プロジェクト制御に役立つデータの収集・分析・表示
- ソフトウェア開発において一般的に利用されているツールからのデータ収集
 - バージョン管理 (e.g. CVS)
 - メールリスト管理 (e.g. Mailman, Majordomo)
 - 障害管理 (e.g. Bugzilla, GNATS)

EPMのアーキテクチャ



EPMの特徴

- オープンソース開発において利用されているツール群から構成
容易に導入可能
- 低負荷のデータ収集
 - 既存の開発環境
 - データ収集のための追加的作業が生じない(CVSを中心に、メールや障害報告といった開発作業過程で蓄積されるデータを利用する)
- 他のツールからエンピリカルSEデータ形式への変換が容易 (他の開発環境との連携が可能)

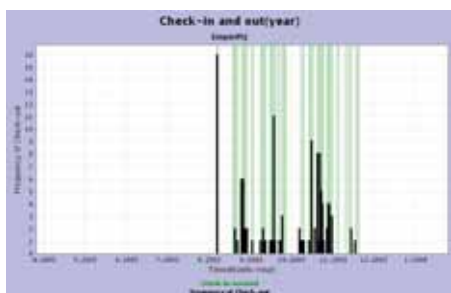
EPMの適用対象と利点

- 大規模プロジェクト
 - 情報の共有がリアルタイムで可能
 - 管理の負荷が低減
 - 人為的なデータ操作が入りにくい
- 小規模プロジェクト
 - 今まで手間等の問題で管理しにくかったプロジェクトにも適用しやすい
 - いろいろなプロセス(XPなど)や組織をまたがる分散開発にも適用可能

分析結果例 (1): ソースコード規模の推移とチェックインとの関連



分析結果例 (2): チェックイン時期とチェックアウト数との関連



分析表示例 (3):メール投稿数の推移と 障害発生/障害解決報告との関連



分析表示例 (4): 障害報告の累計推移とチェックインとの関連



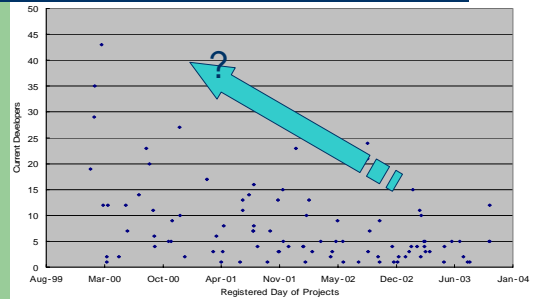
分析表示例 (5): 障害発生/解決報告と 平均障害滞留時間との関連



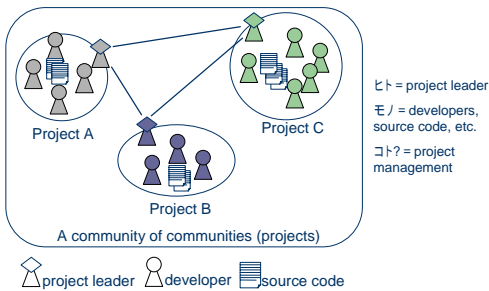
オープンソースシステム開発プロジェクトからのデータ収集と分析

- SourceForge.net (世界最大のOSS開発コミュニティ)
 - 登録プロジェクト: 75,871 (Feb. 12)
 - 登録者: 789,568 (Feb. 12)
 - 様々な種類の協調作業支援ツールを提供
 - SourceForge Collaborative Development System (CDS) web tools
 - Project Web Server
 - Tracker: Tools for Managing Support
 - Mailing lists and discussion forums
 - MySQL Database Services
 - Project CVS Services
 - etc.
- } EPMのデータ収集源として利用可能

Summary of 100 OSS projects @SF.net: Evolution?



Inter-Community Metrics: EPMを利用したOSSプロジェクト間の類似性・相違性の分析



今回の分析に利用したメトリクス概要

Data sources for EPM (97 Active projects @ SF.net)

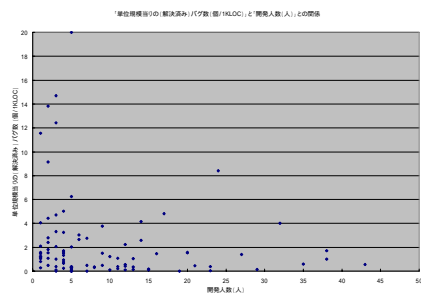
- CVS data
 - Lines of code
 - Number of check-out
 - Number of check-in
 - Number of developers
 - Term of development
- Mailing lists data
 - N/A
- Issue (Bug) reports data (not collected using EPM)
 - Number of resolved Issues
 - Number of unresolved Issues

解決済み/未解決バグ数に着目した分析

- 使える単位
 - 「解決済みバグ数(個)」
 - 「未解決バグ数(個)」
 - 「単位規模当りの(解決済み)バグ数(個/KLOC)」
 - 「単位規模当りの(未解決)バグ数(個/KLOC)」
 - 「バグ解決率(%)」
 - 「開発人数(人)」
 - 「開発期間(月)」
 - 「開発規模(人月)」
- 組み合わせ
 - 「(解決済みバグ数(個))と(開発人数(人))」「(開発期間(月))」「(開発規模(人月))」
 - 「(未解決バグ数(個))と(開発人数(人))」「(開発期間(月))」「(開発規模(人月))」
 - 「単位規模当りの(解決済み)バグ数(個/KLOC)と(開発人数(人))」「(開発期間(月))」「(開発規模(人月))」
 - 「単位規模当りの(未解決)バグ数(個/KLOC)と(開発人数(人))」「(開発期間(月))」「(開発規模(人月))」
 - 「バグ解決率(%)と(開発人数(人))」「(開発人数(人))」「(開発期間(月))」「(開発規模(人月))」

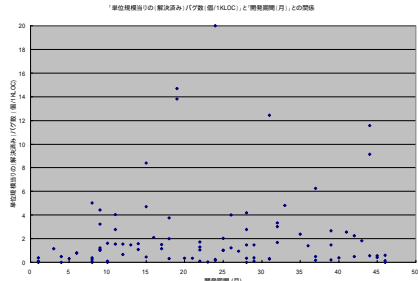
単位規模当たりの解決済みバグ数と開発者数との関係

* 桁外れの2プロジェクト除く



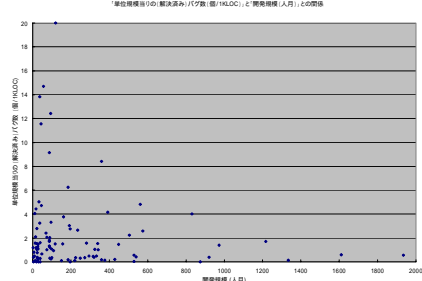
単位規模当たりの解決済みバグ数と開発期間との関係

* 桁外れの2プロジェクト除く



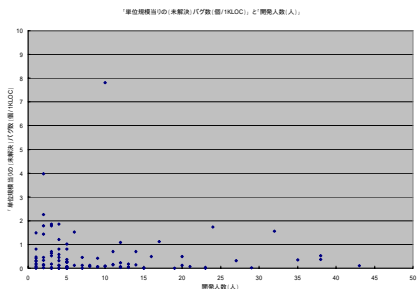
単位規模当たりの解決済みバグ数と人月との関係

* 桁外れの2プロジェクト除く



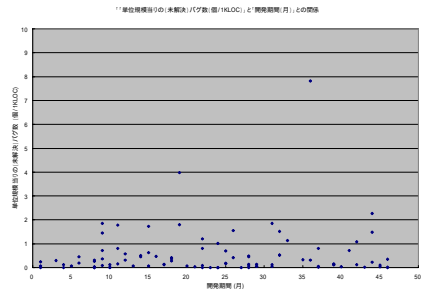
単位規模当たりの未解決バグ数と開発者数との関係

* 桁外れの2プロジェクト除く



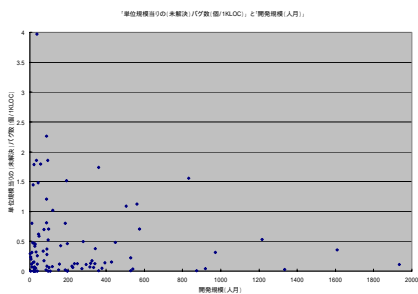
単位規模当たりの未解決バグ数と開発期間との関係

* 桁外れの2プロジェクト除く

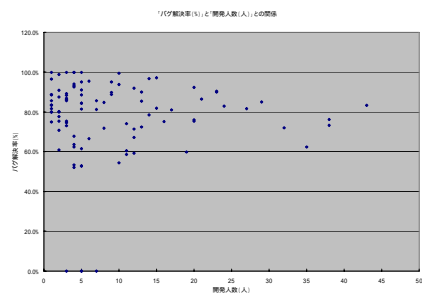


単位規模当たりの未解決バグ数と人月との関係

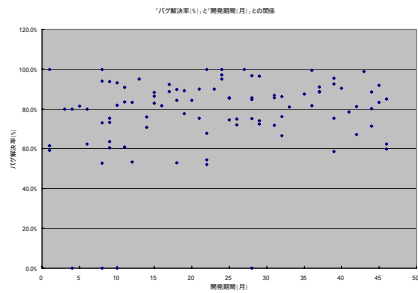
* 桁外れの2プロジェクト除く



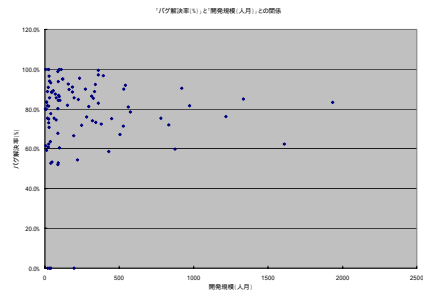
バグ解決率と開発者数との関係



バグ解決率と開発期間との関係



バグ解決率と人月との関係



モノの特性としてのバグ数

- 解決済みバグ数/未解決バグ数/バグ解決率
 - 人月との相関が高い? (議論の余地あり)
 - 規則性のあるグラフ
 - 他のプロジェクトと比較して優秀なのか、普通なのか、劣っているのかの確認
 - 異常値の早期検出
- 今回とは異なる標本を測定し比較する必要あり
 - 今回収集したプロジェクトデータは、現在も活発に活動しているプロジェクトのデータ
 - 失敗プロジェクトの方がたくさんあるはず
DynCで支援すべき対象

今後の予定

- チェックイン・チェックアウト、メール(投稿数、サイズなど)からの分析
- active project と non-active project との比較
今回収集した97プロジェクトは、現在まで活動が活発に続いている、いわば「成功プロジェクト」、おそらく多くのプロジェクトが“一人プロジェクト”や“死んだ”状態で途中で終わる「失敗プロジェクト」。
- “旬な”プロジェクトの要因・基準の同定
プロジェクトが活発になる適度な(最小限の?)人数、ソースコード規模、開発期間などがあるのではないかな?