独立行政法人情報処理推進機構 委託

2013 年度ソフトウェア工学分野の先導的研究支援事業
「ソフトウェア品質の第三者評価のための基盤技術
－ソフトウェアプロジェクトトモグラフィ技術の高度化－」
成果報告書

平成 27 年 2 月

奈良先端科学技術大学院大学

本報告書は独立行政法人情報処理推進機構　技術本部　ソフトウェア高信頼化センターが実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の公募による採択を受けて奈良先端科学技術大学院大学情報科学研究科（研究責任者 松本健一）が実施した研究の成果をとりまとめたものであ

目次

研究成果概要

1. 研究の概要

本委託研究は，ソフトウェアプロジェクトトモグラフィ技術を高度化し，学術的にはもちろんのこと，産業界に対してもよりインパクトを与える「ソフトウェア品質の第三者評価の技術基盤」の確立を目指すものである．具体的には，ソフトウェアデータマイニング・アナリティクスの最新技術を取り入れると共に，「ソフトウェア品質の第三者評価」のニーズやプロセスにも焦点をあてた 6 つの研究目標，

(1) 品質評価の高度化
  ① 欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価
  ② ソフトウェア障害の波及度を定量的に評価する方式の設計と実証的評価
(2) 検証型解析・可視化技術の高度化
  ③ 非機能要件の自動評価方式の設計と実証的評価
  ④ 多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価
(3) 探索型解析・可視化技術の高度化
  ⑤ 低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価
  ⑥ ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価

を設定し，その妥当性・有用性をプロトタイプシステムの実装と実証実験を通じて示した．各研究目標に対する具体的な研究実施内容は 3.以降で述べる．

2. ソフトウェアプロジェクトトモグラフィ

「ソフトウェアプロジェクトトモグラフィ（Software Project Tomography)」とは，ソフトウェア品質の第三者評価で必要となる「ソフトウェアプロジェクトデータの提供」および「提供されたデータに基づくプロジェクト理解」を容易にするために，2012 年度の本先導的研究支援事業において本研究責任者らが提案した新しい概念，手法である．具体的には，医療におけるコンピュータ断層撮影，いわゆる，CT（Computed Tomography）を，ソフトウェア品質の第三者評価に適したデータ構造を考える上でのモデルとする．すなわち，ソフトウェア開発プロジェクトをからだに見立て，プロジェクト開始から終了まで（あるいは，現時点まで）のいくつかの時点において，プロジェクトの状況を定量的に表すスナップショットを断面画像のように作成する．CT において断面画像の系列から身体の 3 次元モデルが再構成できるように，スナップショットの数が十分であれば，その系列によってプロジェクトを様々な観点で可視化し，ソフトウェアやその品質が実現される過程（プロセス）を表すことも可能になる．「開発管理」のために収集・蓄積されているソフトウェア開発プロジェクトデータを，「解析と可視化」に都合の良い形式に変換する手法とも位置付けることができる．（図 ES-1 参照）．

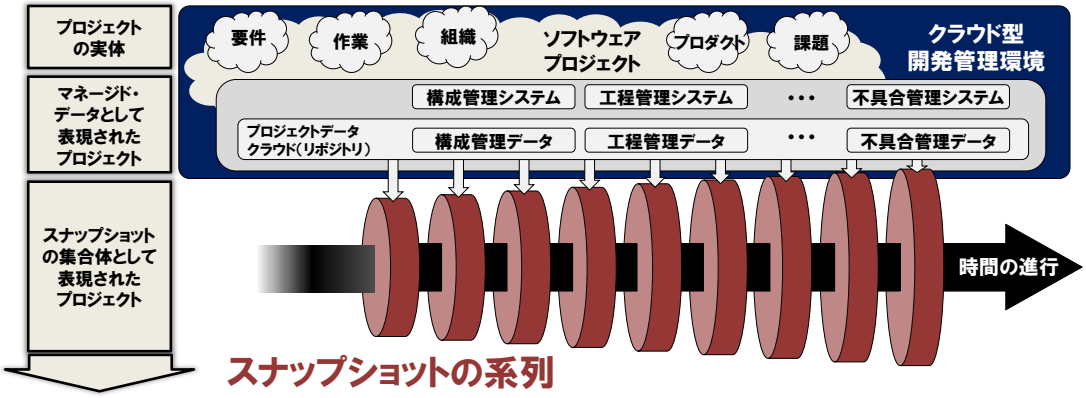


図 ES-1　ソフトウェアプロジェクトトモグラフィの基本概念

3. 品質評価の高度化

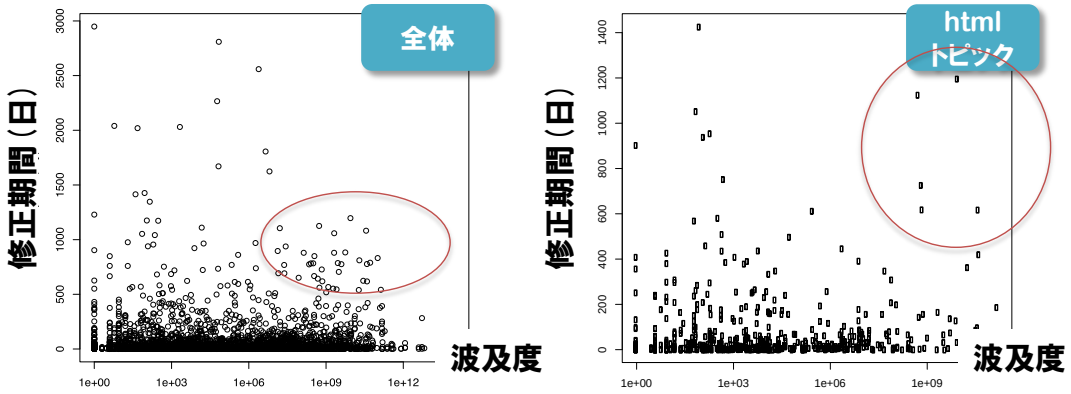
3.1 欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価

ソースコード中の欠陥（課題）が収束・発散するプロセスをそのトピック別に評価するモデルを構築した．トピック別評価は，技術的には，マルチラベル評価問題と位置づけることができる．具体的には，まず，ソフトウェア開発における欠陥レポート（障害レポート，バグ票,...）にトピックモデリングを適用し，欠陥トピックを自動特定する．次に，それら欠陥トピックやトピック間の関係を説明することのできるソフトウェアメトリクスを構造方程式モデリング（Structural Equation Modeling，SEM）により特定する．SEM で得られたメトリクス間（変数間）の関係情報に基づき，ベイジアンネットワークによって欠陥評価モデルを構築する．モデル構築と欠陥トピックの時系列分析のプロトタイプ実装には，統計解析向けプログラミング言語「R」を用いている．

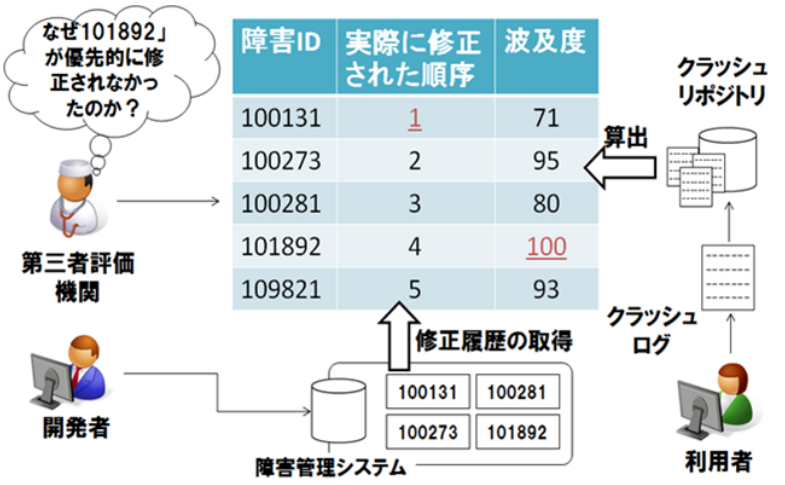
欠陥トピックを特定するプロセスの概要と欠陥の収束・発散プロセスのトピック別での可視化例を図 ES-2 に示す．トピックはいわば，評価対象とするプロジェクトや欠陥そのものの特性を反映した分類基準である．一般性の高い既定の分類基準では顕在化しないような欠陥をあぶり出し，その収束・発散プロセスを示すことができる．評価対象とするプロジェクトや欠陥に関する知識が必ずしも豊富とはいえない第三者が，プロジェクトの特性や実態を理解する上で大きな助けとなる．また，それら知識が豊富な者に対しても，プロジェクトの体制やプロセスの改善に新たな観点，論点を提供することになる．

3.2 ソフトウェア障害の波及度を定量的に評価する方式の設計と実証的評価

ソフトウェア開発で発生する障害の波及度を定量的に評価する方式を開発した，障害の波及度は，障害除去の優先度・優先順位を決定する上で重要な指標であるが，その定量的評価法はこれまで確立されておらず，暗黙知とされてきた．本委託研究では，ユーザから発信される「クラッシュレポート」から，障害の発生時期やユーザ環境（OS やバージョン）等の特徴量を抽出することで波及度を算出する．算出においては，ソフトウェア開発者へのヒアリングで得られた知見を定式化している．なお，プロトタイプ実装には，統計

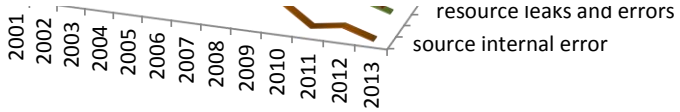


(a) 全障害　　(b) トピック「html」で表される障害



(c) 波及度に基づく第三者評価シナリオ例：障害修正プロセス評価

図 ES-3　障害の波及度と修正期間の評価・可視化例，および，第三者評価シナリオ



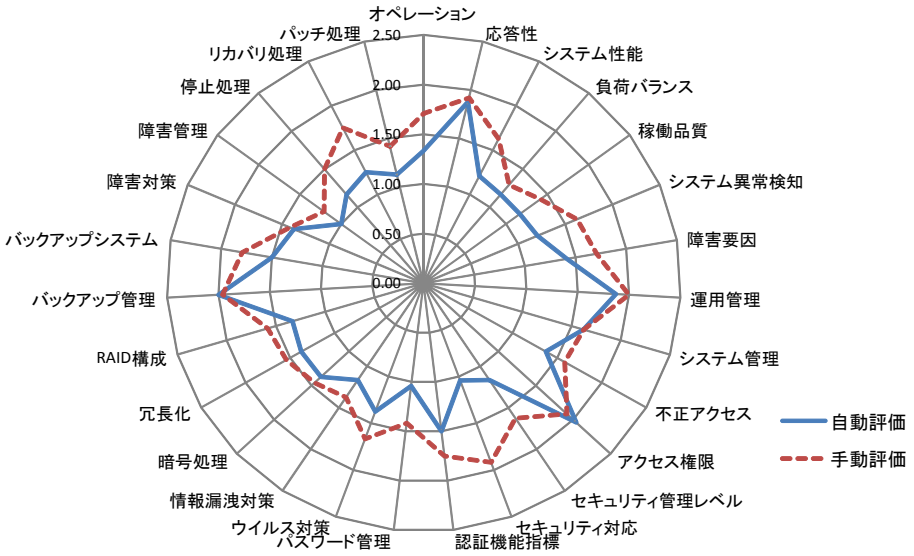
(b) 欠陥の収束・発散プロセスのトピック別での可視化例

図 ES-2　欠陥トピックの特定プロセスと可視化例

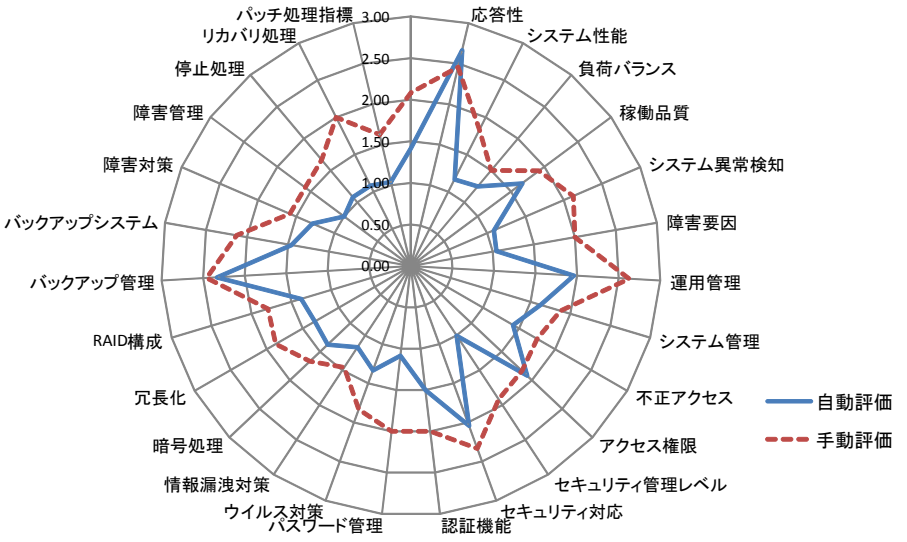
解析向けプログラミング言語「R」を用いている．

波及度の算出例，および，波及度に基づく第三者評価シナリオ例を図 ES-3 に示す．同図(a)(b)では，各障害における波及度と修正期間（除去までに要した期間．日数）の関係を散布図の形式で示している．同図(a)からは，対象システムで発生した全障害に関するもの

で，波及度が大きく，かつ，修正期間が長い障害はほとんど存在せず，波及度の大きな障害は優先的に除去されていることが分かる．一方，同図(b)は，トピックモデリング技術を使い，html という単語で表される障害のみを抽出し作成した散布図である．html で表される障害に限定すると，波及度が大きい障害のいくつかは修正期間が長く，波及度の大きな障害は必ずしも優先的に除去されていないことが分かる．こうした波及度と修正期間との比較は，障害除去の優先度・優先順位の決定をはじめとして，ソフトウェア障害への対応体制・プロセスの妥当性評価や改善点検討の基礎となる．特に，「クラッシュレポート」はソフトウェアユーザから発信される情報であり，ソフトウェア開発者（ベンダ）から発信される情報のみに基づく評価に比べ，より公正で透明性の高い観点を第三者評価にもたらすことになる．同図(c)に，障害修正プロセス（修正の優先順位）を波及度に基づき第三者が評価するとした場合のシナリオを示す．



(a) 3 段階評価



(b) 5 段階評価

図 ES-4　非機能要件 26 個の手動評価と自動評価の比較（RFP 70 件での平均値）

4. 検証型解析・可視化技術の高度化

4.1 非機能要件の自動評価方式の設計と実証的評価

ソフトウェア開発の上流工程で作成される RFP (Request For Proposal)において，非機能要件の記述の明確さを自動評価する方式を開発した．具体的には，非機能要件に関連する語句（キーワード）を説明変数とし，要件記述の明確さを目的変数とするモデルをランダムフォレスト法により構築する．RFP からの語句抽出には形態素解析を用い，モデル構築においては，TF-IDF 法（Term Frequency - Inverse Document Frequency Method）による

語句の重み付けも行っている．TF-IDF 法では，語句の重みが TF（Term Frequency，単語の出現頻度）と IDF（Inverse Document Frequency，逆文書頻度）の 2 つの指標に基づいて計算される．評価対象とする非機能要件は，開発者へのヒアリングや文献調査によって選定した 26 個である．なお，プロトタイプ実装には，統計解析向けプログラミング言語「R」を用いている．

Web 上から入手可能な 70 件の RFP(図書館情報システム 11 件,病院情報システム 10 件，大学情報システム 8 件，政府機関情報システム 14 件，自治体基幹情報システム 10 件，地方自治体業務システム 14 件，その他情報システム 3 件）を対象とした実証実験の結果を図 ES-4 に示す．同図より，エキスパートによる手動（主観）評価とほぼ同等の自動評価が可能となっていることが分かる．本委託研究では，RFP のみを評価対象としたが，ソフトウェア開発が進むにつれて作成される契約書や要求仕様書などのドキュメントでも同様のモデル構築，自動評価が可能である．それらドキュメント間で評価結果を比較することで，非機能要件を明確にするプロセスが妥当であったのかを第三者が評価し，その改善点を検討することも容易になる．

4.2 多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価

多数の開発者が携わるソフトウェア開発プロジェクトにおける開発タスクを「正当なデータ」に基づいて解析・可視化する方式を開発した．具体的には，ソフトウェア開発タスクの実施に伴うアプリケーション実行履歴を，識別のためのハッシュ値と共に記録すると共に，実行履歴の正当性を検証することのできるシステムを実現した．なお，多人数での開発作業を対象とするためサーバ／クライアント方式とした．アプリケーション実行履歴は，開発タスク計測システム TaskPit を通じて収集される．また，改ざんされたアプリ実行履歴の検出だけでなく，改ざんされたクライアントからのサーバ接続要求を拒否する機能も有する．収集されたアプリケーション実行履歴は，開発組織に閲覧可能な状態で蓄積されるため，プロジェクト管理者が組織内で行うプロセス改善に利用することも可能である．更に，アプリケーション実行履歴と開発作業・タスクの対応関係を機械学習によりモデル化し，それらの間の自動マッピングを 90%超の精度で可能にした．これにより，アプリケーション履歴を開発作業履歴とみなして評価や可視化に利用することが可能になった．なお，対応関係のモデル化には，統計解析向けプログラミング言語「R」を用いている．

実現したシステムにおけるハッシュ値の収集と正当性検証の画面表示例，および，同システムで実現されるソフトウェア開発組織と第三者（評価組織）との関係を図 ES-5 に示す．ハッシュ値に基づく検証が可能となることで，第三者（評価組織）は，データに改ざんのないことを確信し，ソフトウェア品質やその実現プロセスの評価を安心して行うことができる．ソフトウェア開発組織も，改ざんのないことが保証されている開発データを，第三者はもちろん，必要があれば，ソフトウェアの発注者やユーザにも提供できるようになる．また，アプリケーション実行履歴は，分単位の細粒度で収集，記録され，開発作業との自動マッピングも高い精度で実現されていることから，労務管理データを補完し，従来よりも詳細な工数管理が可能となる．

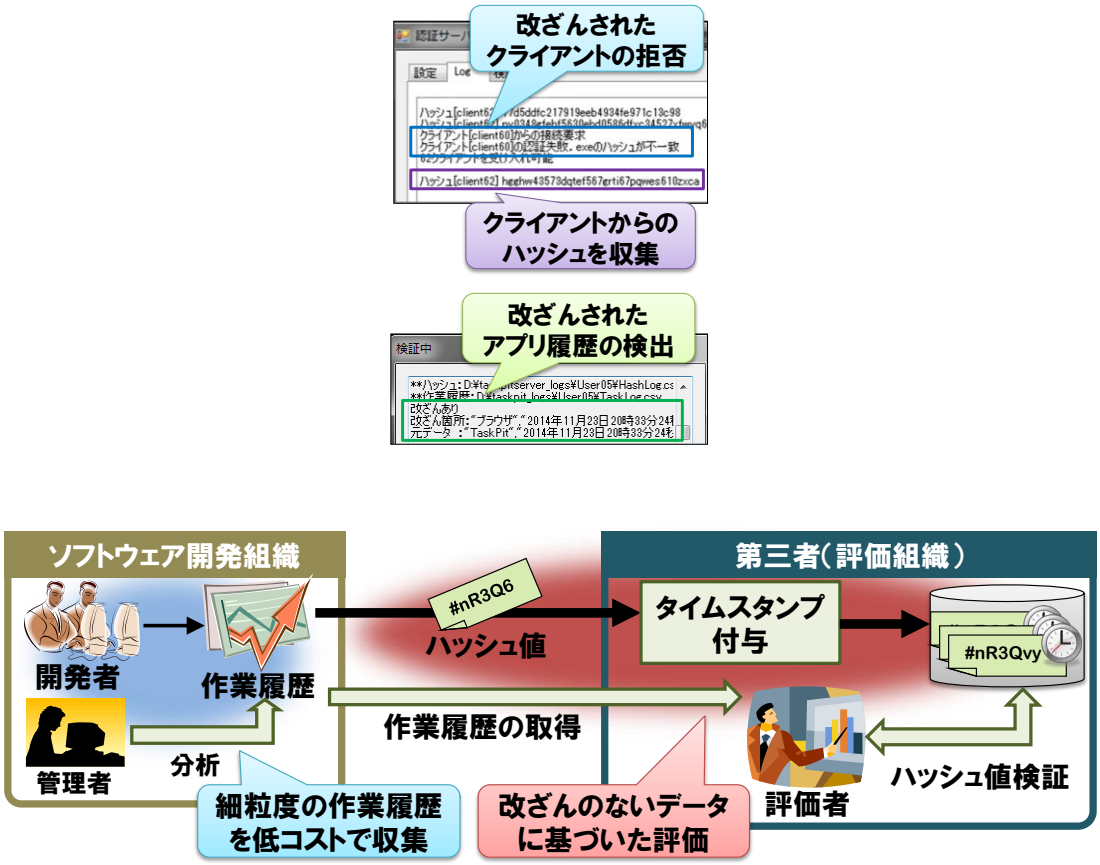


図 ES-5　システムの画面表示例，および，同システムが実現するソフトウェア開発組織と第三者（評価組織）との関係

5. 探索型解析・可視化技術の高度化

5.1 低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価

低品質モジュールが，プログラム内にどのように分布し，開発作業の進行に伴ってどのように変化していくのか，その構造的・時間的特性を解析・可視化する方式を開発した．具体的には，モジュール品質に関係すると考えられる属性 13 個について，その属性値（メトリクス値）の標準値域を，低品質なモジュール（欠陥ありモジュール）と高品質なモジュール（欠陥なしモジュール）それぞれについて明らかにし，人口ピラミッドグラフとして可視化した．更に，それら標準値域との比較に基づき，評価時点，および，将来のある時点において低品質なモジュールとなるかどうかを推定，予測するためのモデルを構築した．標準値域を明らかにした属性は，

複雑度，実行可能コード行数，クラス数，関数数，空白行数，コード行数，宣言コード数，実行可能ステートメント行数，宣言ステートメント数，コメント数，変更行数，追加行数，削除数

の計 13 個である．

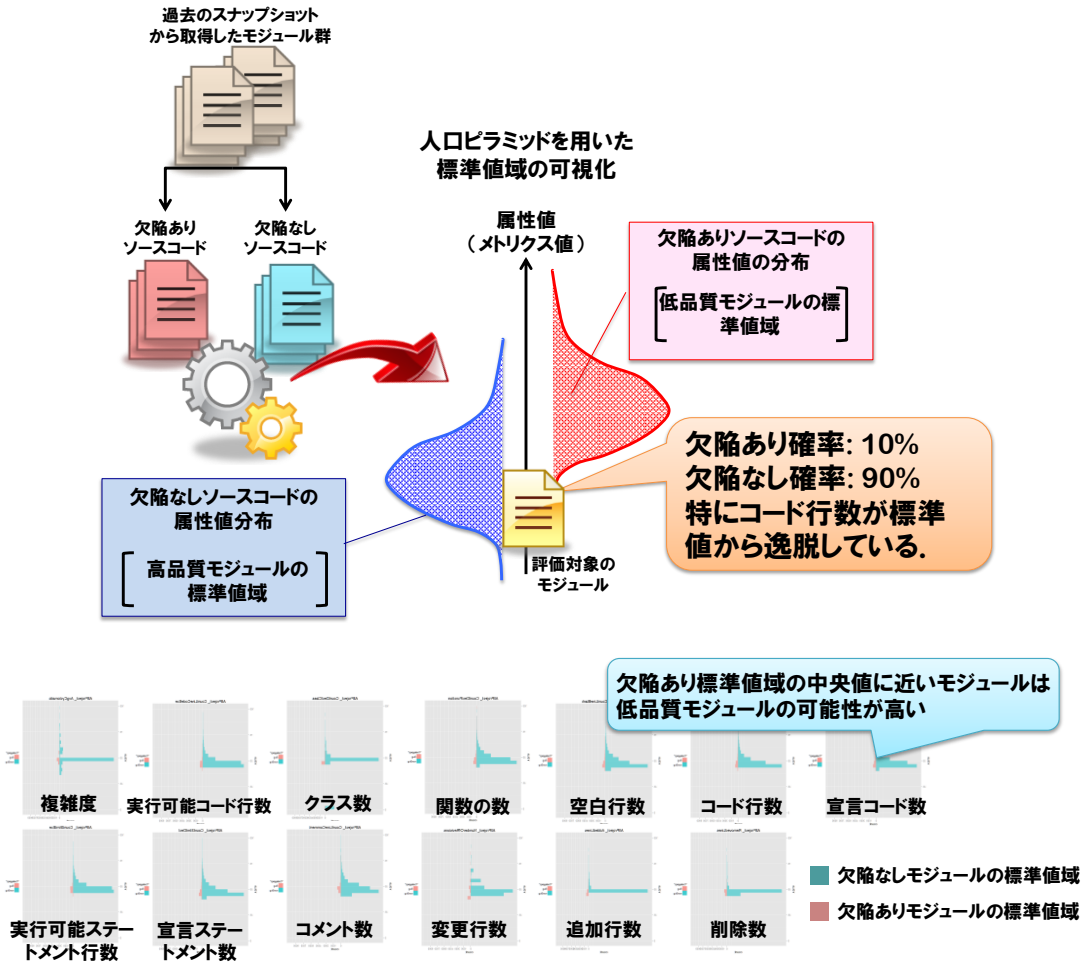
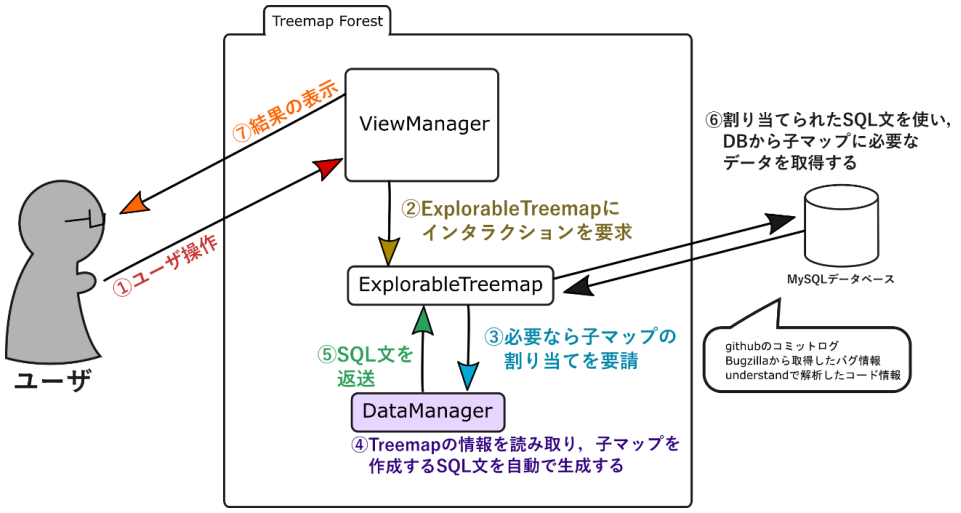


図 ES-6　オープンソースソフトウェアの開発データに基づいて算出したソースコード属性の標準値域

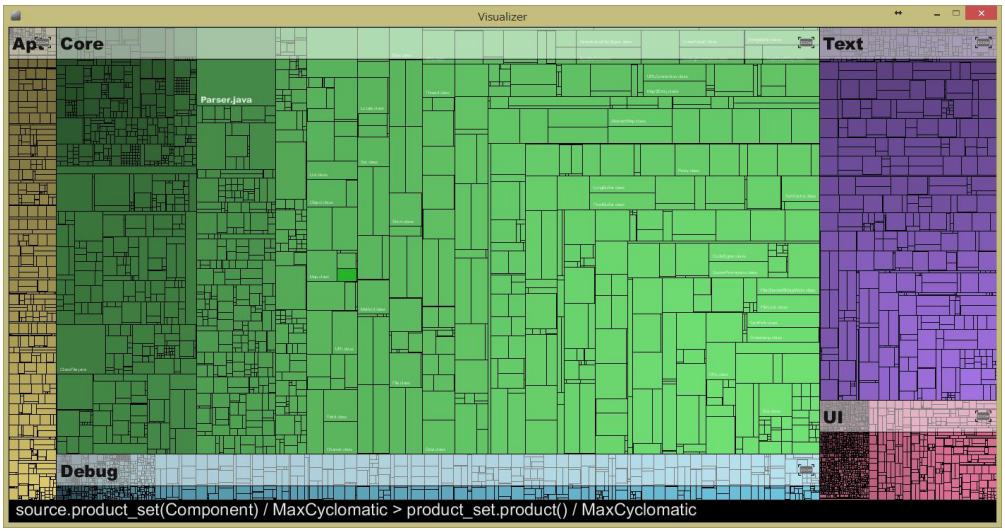
標準値域の利用例とオープンソースソフトウェアの開発データに基づいて算出した標準値域を図 ES-6 に示す．一般に，「欠陥ありモジュール」と「欠陥なしモジュール」とでは，属性値分布に違いがあり，特に，あるモジュールの属性値が欠陥ありモジュールの属性値の中央値に近い場合，そのモジュールは「欠陥あり」の可能性が高い．また，13 個の属性に基づく並行評価は，欠陥の特性や混入原因の解明につながるより詳細な情報を提供する．標準値域は，数多くのソフトウェア開発プロジェクトで収集されたデータ（ソフトウェアプロジェクトデータ）に基づいて算出されており，それとの比較は，第三者評価において重要な要素の一つである「外部妥当性の評価」と位置付けることができる．更に，標準値域の平均と分散に基づいて属性値を正規化（偏差値化）することで，プロジェクト間での開発データの比較，共有，共用が可能となり，第三者評価においてより多くのプロジェクトの開発データが利用できるだけでなく，開発データの蓄積が十分でない組織においても低品質モジュール予測等を行うことができる．

5.2 ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価

ソフトウェアプロジェクトトモグラフィで定義される 5 つの視点（要件，作業，組織，プロダクト，課題）で，ソフトウェアプロジェクトデータを俯瞰し，注目する子要素へのズームインや親要素へのズームアウトなどを繰り返すことで,データの量的属性を再帰的,かつ,探索的に解析・可視化する方式を開発した．俯瞰表示は Treemap 形式で行うが，ズームインにおける観点を自在に変更できるだけでなく，ランダムに変更する機能を持たせることで，偶発的発見を促すユーザインタフェースとなっている．なお，プロトタイプ実装には，Java の派生言語である Processing を用いており，Java からの直接利用，または,javascript 用ライブラリの利用により，ウェブアプリ化も可能となっている．



(a) Treemap Forest の動作モデル



(b) 表示例：観点「コンポーネント」→「ファイル」，サイクロマティック複雑度」

図 ES-7　Treemap Forest の動作モデルとソフトウェアプロジェクトデータ表示例

実装したプロトタイプTreemap Forestの動作モデルとソフトウェアプロジェクトデータの表示例を図 ES-7 に示す．主要な開発者やバグ報告数など，属性を変更し視点を変えながら俯瞰を繰り返すことで，ソフトウェア開発プロジェクトの全体像を，これまでよりもすばやく，多面的に把握することができる．また，Treemap では，対象データの全体を示す矩形領域が，その構成要素（子要素）によって分割表示され，各子要素の表示面積は，利用者が指定した属性値に比例して按分される．子要素の数や面積の片寄りにより「外れ値」となる子要素を容易に発見することができる．外れ値の発見は，第三者評価の初期段階における「(データや属性間の関係に関する）仮説の生成」に大いに役立つ．

6. まとめ

本委託研究では，ソフトウェアプロジェクトトモグラフィ技術を高度化し，学術的にはもちろんのこと，産業界に対してもよりインパクトを与える「ソフトウェア品質の第三者評価の技術基盤」の確立を目指した．学術的なチャンレンジとしては，トピックモデリングや人口ピラミッドグラフの応用，クラッシュレポートの活用，機械学習による自動評価・マッピングなどを挙げることができる．また，産業界へのインパクトとしては，企業実務者へのインタビュー等に基づく技術利用シナリオの作成，企業における実証実験，開発した技術のツール化やウェブサービス化などを挙げることができる．

これら成果は，ソフトウェア開発プロジェクトの透明性を高め，ソフトウェア品質に関する「シグナリング」の手段を提供し，ベンダとユーザ間の「情報の非対称性」を解消することになる．ソフトウェア品質の第三者評価制度といった「制度や組織」と相まって，市場をより健全化し，ベンダには競争力を，ユーザには安心感を与えることになる．

1 研究の背景および目的

1.1 背景

2010 年に米国で発生したトヨタ車の急加速問題では，当初，エンジンの電子制御ソフトウェアの欠陥が疑われた．しかし，NASA 工学・安全センター（NESC）が，「専門知識を有する中立的立場の第三者」として当該ソフトウェアの品質評価を行った結果，ソフトウェアの設計や実装に欠陥は発見されず，急加速の多くは，運転手の操作ミスに起因することが確認された．こうした「ソフトウェア品質の第三者評価」の必要性，重要性は，今後ますます高まると考えられており，国内においても，第三者評価の枠組みや制度に関する議論が活発に行われている[IPA1]．

本研究責任者らも，2012 年度の本先導的研究支援事業において，ソフトウェア品質の第三者評価の技術基盤の確立を目指して，「ソフトウェアプロジェクトトモグラフィ」と名付けた新しい概念を提案し，プロトタイプシステムの実装と実証実験を行った．「ソフトウェアプロジェクトトモグラフィ」とは，ソフトウェア開発プロジェクトをスナップショットの系列で表現する手法である．ここで，スナップショットとは，多様なソフトウェアプロジェクトデータとその解析結果の集合体であり．「要件」，「課題」，「プロダクト」，「組織」，「作業」の５つの観点を持つ．コンピュータ断層撮影（CT：Computed Tomography）において断面画像の系列から身体の 3 次元モデルが再構成できるように，スナップショットの数が十分であれば，その系列によってプロジェクトを様々な観点で可視化し，ソフトウェアやその品質が実現される過程（プロセス）を表すことも可能になると考える．プロトタイプ実装と実証実験を通じて，「品質評価に必要となるソフトウェアプロジェクトデータの提供」および「提供されたデータに基づくプロジェクト理解」が容易になるなど，ソフトウェア品質の第三者評価におけるソフトウェアプロジェクトトモグラフィ技術の妥当性・有用性が示された．

ただし，端緒についたばかりの研究ということもあり，概念的な議論やシーズ先行の技術開発と言える面もあった．産業界では，最近，ソフトウェア品質の第三者評価に関する実験や試行などが開始され，第三者評価に対する具体的なニーズや評価プロセスも明らかになりつつある[IPA2][IPA3]．それらニーズやプロセスに基づき，ソフトウェアプロジェクトトモグラフィ技術を高度化することができれば，より強固で実践的な「ソフトウェア品質の第三者評価の技術基盤」が確立されると考える．

1.2 研究課題

本委託研究では，ソフトウェアプロジェクトトモグラフィ技術を高度化し，学術的にはもちろんのこと，産業界に対してもよりインパクトを与える「ソフトウェア品質の第三者評価の技術基盤」の確立を目指す．特に，ソフトウェアデータマイニング・アナリティクスの最新技術を取り入れると共に，産業界において具体化しつつある「ソフトウェア品質の第三者評価」のニーズやプロセスにより焦点をあてたアプローチをとる．具体的には，2012 年度の本先導的研究支援事業で開発した技術のうち，産業界から高い関心が寄せられた技術を 3 つの観点「品質評価」，「検証型解析・可視化技術」，「探索型解析・可視化技術」で再構築・高機能化する．また，第三者評価のニーズやプロセスを「第三者評価シナリオ」として明確化・詳細化することで，得られる研究成果をよりニーズ指向なものとする．本

委託研究における具体的な研究課題①～⑥は次のとおり.

(1) 品質評価の高度化

① 欠陥の収束・発散プロセスのトピック別評価

② 障害の波及度評価

(2) 検証型解析・可視化技術の高度化

③ 非機能要件の自動評価

④ 作業解析・可視化システムの多人数対応

(3) 探索型解析・可視化技術の高度化

⑤ 低品質モジュールの構造的・時間的特性の解析・可視化

⑥ Treemap による広範で再帰的な解析・可視化

1.3 研究の意義

現在のソフトウェア開発では，ソフトウェア品質に大きな影響を与える開発過程の情報の多くは，開発者（ベンダ）のみが知り得るものである．すなわち，ベンダと他のステークホルダ，特に，利用者（ユーザ）との間には，ソフトウェア品質に関する情報量に大きな隔たりがある．こうした「情報の非対称性」は，

・逆選択：ユーザは低品質なソフトウェアしか選べなくなる．

・モラルハザード：ベンダが手抜きや虚偽報告をしがちになる．

といった問題を引き起こし，市場の健全な発展を阻害するとされている．対策としては，

・シグナリング： ベンダがユーザに積極的に情報発信する．

・スクリーニング：ユーザがいくつかの案をベンダに示し，選択させることを通じて情報を読み取る．

・制度と組織： 中立的立場の第三者が情報の非対称性の解消を手助けできるよう，認証や監視などのための制度や組織を整備する．

などが知られている．本委託研究で高度化を目指すソフトウェアプロジェクトトモグラフィ技術は，ソフトウェア開発プロジェクトの透明性を高め，ソフトウェア品質に関する「シグナリング」の手段を提供することになる．ソフトウェア品質の第三者評価制度といった「制度や組織」と相まって，市場をより健全化し，ベンダには競争力を，ユーザには安心感を与えることの意義は大きい．具体的な効果としては次の 2 つが期待される．

(1) 外注・再利用を伴うプロジェクトにおけるソフトウェア開発力の強化

ソフトウェア開発における外注比率は，人件費ベースで 35.6%に達している[IPA4]．外注先の 65.8%は国内中小企業であり，主な技術的課題として，

・品質管理が難しい．

・検収判定が難しい．

・納期・開発工程の管理が難しい．

が挙げられている．このうち，品質管理と検収判定に関しては，研究課題③「非機能要件の自動評価」の成果がその解決に役立つ．また，納期・開発工程の管理に関しては，研究課題④「作業解析・可視化システムの多人数対応」の成果がその解決に役立つ．

また，開発されるソフトウェアの約 60%は，既存ソフトウェアの再利用によって実現されている[IPA4]．再利用部分については，開発時の情報が得られない場合が多く，外注に

よる開発とは別の意味で品質管理が難しいとされている．研究課題①「欠陥の収束・発散プロセスのトピック別評価」,研究課題②「障害の波及度評価」,研究課題⑤「低品質モジュールの構造的・時間的特性の解析・可視化」の成果がその解決に役立つ.

(2) ソフトウェア品質説明力の強化

日本製ソフトウェアの品質は決して低くないが，その高い品質を具体的に利用者や市場に説明する力（品質説明力）が日本企業に十分備わっているとは必ずしも言えない，あるいは，その力は備わっているが説明の必要性を強く感じるに至っていない．しかし，国際市場では，欧米流の品質保証の考え方に基づき「証拠を示してユーザに信頼感を与える.」ことが重視される[Yasuda]．長年の実績を有し，きちんと開発管理を行っていれば，品質に関する説明は不要とする日本流の品質保証は通用しない.本委託研究で高度化される「ソフトウェアプロジェクトトモグラフィ技術」は，日本のソフトウェア企業の品質説明力を高めることになる．更に，ソフトウェア品質の第三者評価が普及，高度化されれば，国際市場における日本製ソフトウェアの品質に対する正当な評価システムが提供されることになる．これらは，我が国のソフトウェア産業の国際競争力の維持・強化に大きく貢献する.

2 実施内容

2.1 研究アプローチ

2.1.1 研究の全体像

研究課題間の関連を保ちながら，各研究課題を「準備→シナリオ策定→設計→実装→実証実験→評価」の順に進める．学会発表や企業との意見交換を重ね，中間報告で頂戴したコメントも勘案し，必要に応じて，設計や実装の見直し，実証実験のやりなおし，などを行う．

(1) 準備（既存の研究・システム・ツール等の調査）
(2) 「第三者評価シナリオ」の策定
（第 1 回中間報告）
(3) 同シナリオに基づく方式設計
（第 2 回中間報告）
(4) 同方式に基づくプロトタイプシステム実装
（第 3 回中間報告）
(5) プロトタイプシステムによる実証実験
(6) 成果のとりまとめ，研究の将来展望・フォローアップの検討.

2.1.2 関連するこれまでの研究について

本委託研究の土台をなす概念「ソフトウェアプロジェクトトモグラフィ」は，研究責任者らが提案する新しい概念であり，その独創性・新規性は極めて高い.

Bowring らは，ソフトウェアトモグラフィという概念を提案している[Bowring]．これは，多数のユーザのもとで稼働しているソフトウェアの振る舞いを，ユーザに負荷をかけずモニターする技術である．モニター用のプローブやタスクを細切れにし，多数のソフトウェアで実行し，その結果を再構成するところから，トモグラフィという用語が用いられているが，本委託研究とは，対象も目的も大きく異なる.

Sarma らは，ソフトウェア開発におけるプロダクト，組織，バグ，コミュニケーションなどに間に存在する Socio-technical な依存関係を見るためのブラウザを提案している[Sarma]．ブラウザに表示される情報は，本委託研究におけるスナップショットと共通する部分が多い．ただし，ソフトウェア要件に関する情報には言及しておらず，ソフトウェア品質の第三者評価に関する議論も行っていない.

研究責任者が当該委託研究以前に実施していた研究２件と本委託研究との関係は次の通りである.

(1) 文部科学省・委託研究「次世代ＩＴ基盤構築のための研究開発：ソフトウェア構築状況の可視化技術の開発普及（エンピリカルデータに基づくソフトウェアタグ技術の開発と普及)」（平成 19 年 8 月～平成 24 年 3 月）

ソフトウェアに対するトレーサビリティの概念を普及させ，世界最高水準の安心・安全なＩＴ社会を実現するため，ソフトウェア開発が適正な手順で行われたかどうかをソフトウェア発注者によって把握･検証可能とすることを目指し,エンタープライズ系ソフトウェア，組込み系ソフトウェアを問わず，オフショアを含むマルチベンダによるソフトウェア

開発に関する実証的データ（エンピリカルデータ）を収集し,「ソフトウェアタグ」としてソフトウェア製品に添付して提供する技術を開発することを目的とした委託研究プロジェクトである．当該研究プロジェクトは，ソフトウェア開発が適正な手順で行われたかどうかを表す実証データを,ユーザ･ベンダ間で共有するための技術（ソフトウェアタグ技術）を開発しようとしたものである．これに対して，本委託研究では，ソフトウェアタグ技術をベースとして，ユーザでもベンダでもない「第三者」によるソフトウェア品質評価の技術基盤を開発するものである．

(2) 情報処理推進機構・2012 年度ソフトウェア工学分野の先導的研究支援事業「ソフトウェア品質の第三者評価のための基盤技術－ソフトウェアプロジェクトトモグラフィの開発－」（平成 24 年 6 月～平成 25 年）

「ソフトウェアプロジェクトトモグラフィ」と名付けた新しい概念を提案すると共に，プロトタイプ実装と実証実験を通じて，ソフトウェア品質の第三者評価における妥当性・有用性を示した．ただし，概念的な議論やシーズ先行の技術開発が中心であった．これに対して，本委託研究は，ソフトウェアデータマイニング・アナリティクスの最新技術，および，産業界において具体化しつつある「ソフトウェア品質の第三者評価」のニーズやプロセスを取り入れることで，学術的にはもちろんのこと，産業界に対してもよりインパクトを与える強固で実践的な「ソフトウェア品質第三者評価の技術基盤」の確立を目指すものである．

2.1.3 研究目標

1.2 で示した 6 つの研究課題それぞれの実現に向け，次の挙げるより具体的な研究目標①～⑥を設定した．設定においては，研究成果の産業界への展開を見据え，明確な評価手順やモデルの構築，一部については自動評価，そして，手順やモデルの妥当性や有用性を確認するための実証的評価に重点を置いた．

(1) 品質評価の高度化
- ① 欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価
- ② ソフトウェア障害の波及度を定量的に評価する方式の設計と実証的評価

(2) 検証型解析・可視化技術の高度化
- ③ 非機能要件の自動評価方式の設計と実証的評価
- ④ 多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価

(3) 探索型解析・可視化技術の高度化
- ⑤ 低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価
- ⑥ ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価

2.2 研究の活動実績・経緯

・活動実績

本委託研究で設定した作業項目毎の実施スケジュールを図 2-1 に示す.

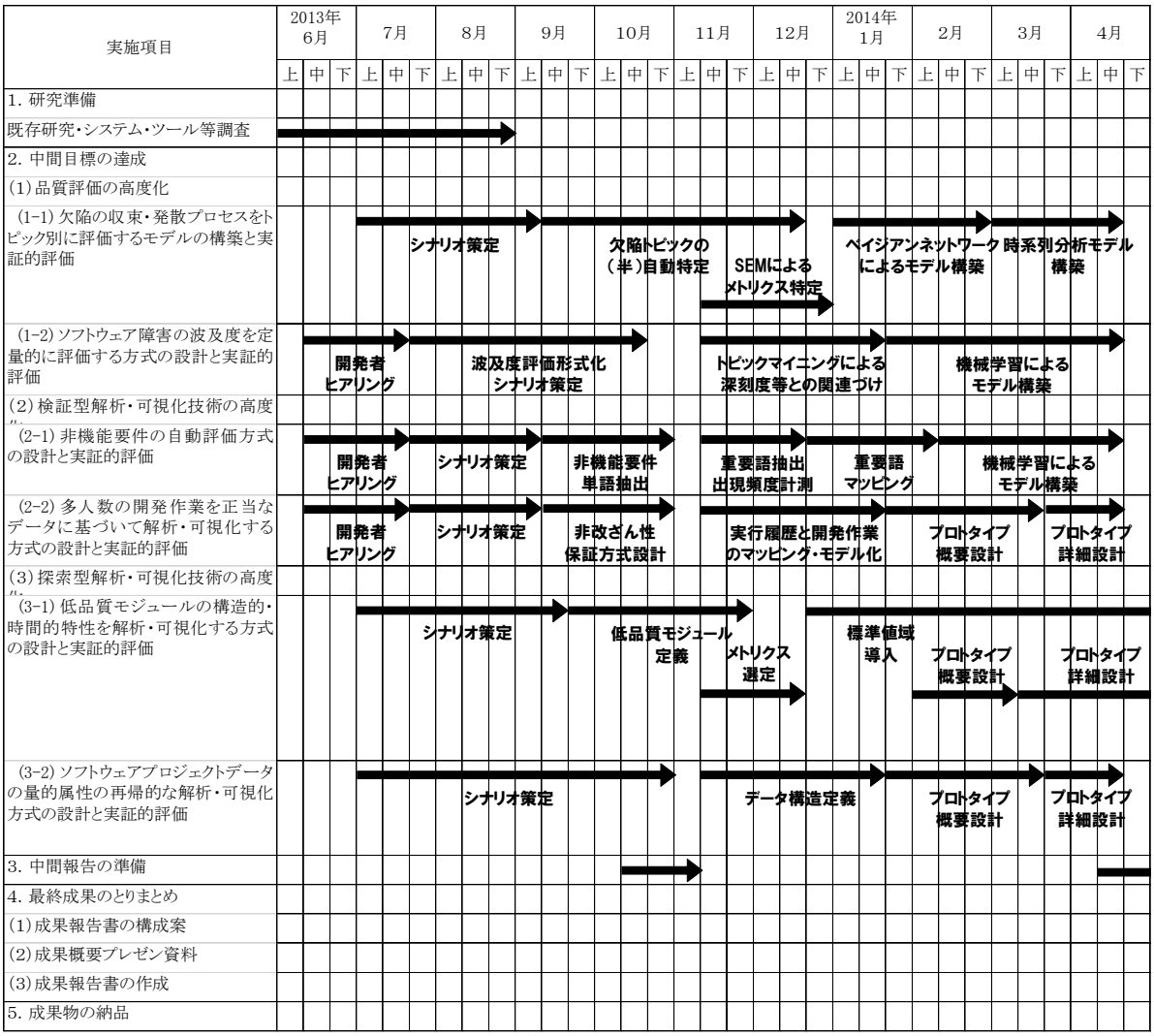


図 2-1 研究実施スケジュール

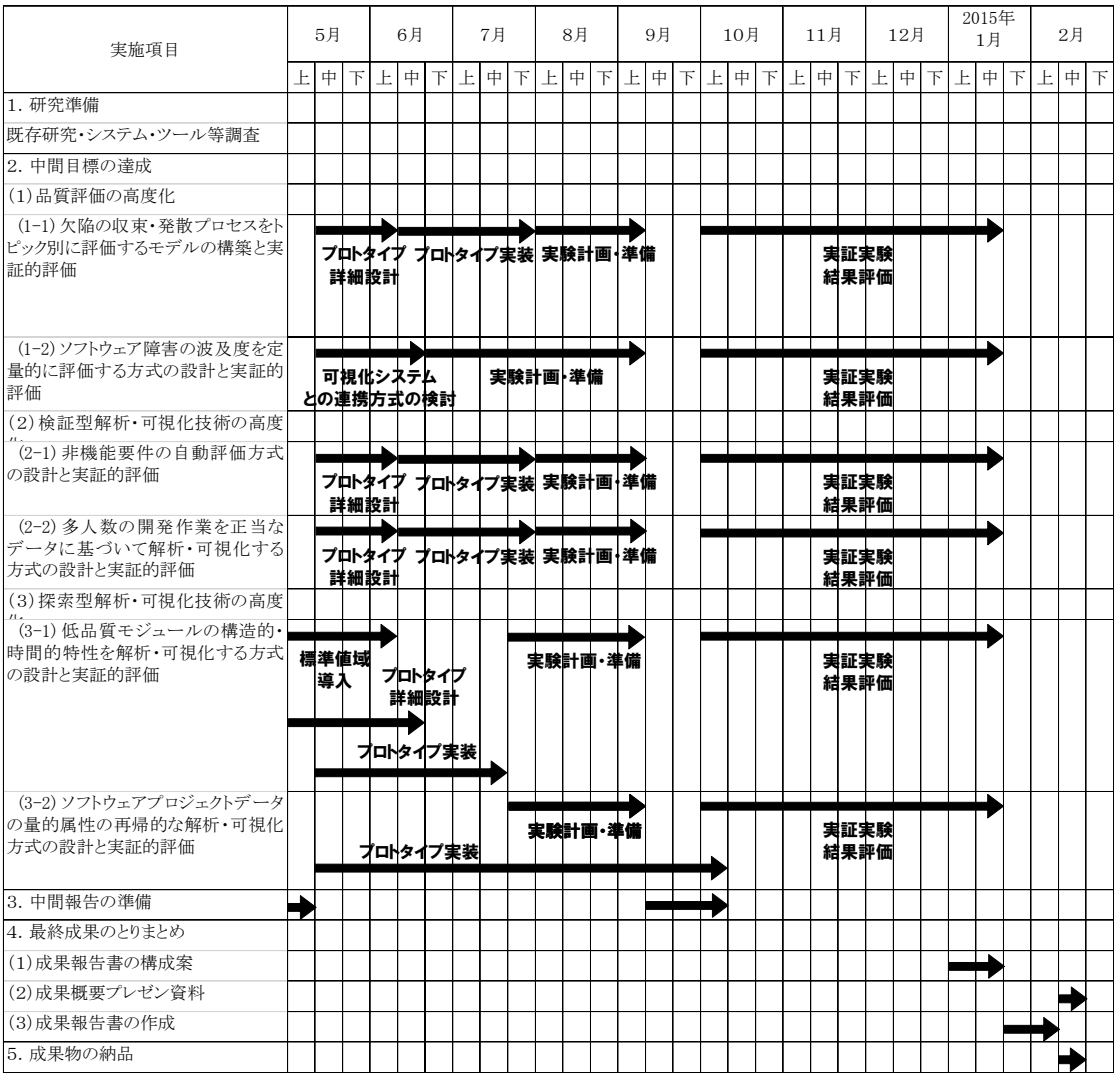


図 2-1 研究実施スケジュール（つづき）

・内部打合せの実施状況

研究責任者は，本委託研究メンバー（アドバイザ，研究技術員を除く）が参加する進捗報告会を隔週で開催し，各メンバーから，メンバー自身，および，研究技術員による研究開発の進捗状況の報告を受けると共に，IPA からの連絡事項の伝達，中間報告会や内部レビューの実施に向けた準備・調整などを行った．それらに基づき，各メンバーに必要な指示を行うと共に，各月の進捗報告書および進捗予定管理表を作成し，IPA へ提出した．なお，メンバーの所在地が，奈良県生駒市，奈良県大和郡山市，福岡県福岡市と分散しているため，進捗報告会の実施においては，必要に応じて，テレビ会議システムを利用した．また，議事，案件，成果物，報告資料等の共有と管理のため，オンラインストレージサービス Dropbox を用いた．

・外部打合せの実施状況

2013 年 7 月 1 日

相手組織名：アクセス，ファインバス，EASE 創研

打ち合わせ内容と得られた結論：

研究目標「多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価」について，アプリケーション実行履歴の計測頻度と計測した作業履歴に対する開発者，管理者，評価者に与えるべき閲覧・編集権限を検討し，プロトタイプの設計，および実装に反映した．

2013 年 7 月 1 日

出席者：亀井 靖高（九州大学），九大訪問研究員[通信開発系実務経験 10 年以上]

議題：障害除去の優先順位，およびおよび，波及度をどのように定式化するかについて

結論：深刻度（セキュリティ関係か否か等），波及度（どれだけの確率で障害が発生し，どれだけの利用者の環境で発生するか），および，修正コスト（修正に伴う影響範囲の大きさ，と修正範囲）が，障害の優先度（除去の優先順位）に大きく影響を与える 3 つの要因であることがわかった．また，障害の発生頻度，障害の発生しているユニークユーザ数，ユーザ間での障害の発生頻度のばらつき等で波及度を表現できることがわかった．

2013 年 7 月 5 日　15:00 - 16:00, NAIST 東京オフィス（田町）

出席者：松本健一（奈良先端科学技術大学院大学），山田淳（東芝）

遠隔出席：門田暁人，伊原彰紀（奈良先端科学技術大学院大学）

議題：非機能要件の評価シナリオ

結論：「記述の明確さ」を評価している点を明確にすること，また，RFP だけでなく，契約書や要求仕様書等も評価対象とする，あるいは，それらを含む一連のドキュメントを評価するシナリオも検討することとなった．

2014 年 6 月 2 日

出席者：井上克郎，Daniel M Germán，畑秀明

議題：研究課題①におけるプロトタイプ実装に向けての課題

結論：欠陥の識別に関して，Dormant bugs，Security bugs，Performance bugs などの既存研究についても考慮する．

2014 年 6 月 3 日

出席者：井上克郎，Daniel M Germán，畑秀明
議題：研究課題①におけるプロトタイプ実装に向けての課題
結論：より実践的な実証実験とするため，英語の欠陥レポートだけでなく，日本語の欠陥レポートも分析対象とすることになった.

・学会参加状況とその成果

① International Software Engineering Research Network Annual Meeting 2013 (ISERN 2013)

開催概要： 実証ソフトウェア工学の推進を目的として 1993 年創設に創設された研究者ネットワーク（ISERN）が年 1 回開催する総会である.

参加目的： 本受託研究における研究目標「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価」に関連する研究の最新動向，特に，「ソフトウェアプロジェクトデータの量的属性の再帰表現に適したデータ構造」に関する情報を入手すると共に，「量的属性の解析・可視化に基づく第三者評価シナリオ」について参加者との意見交換を行う.

主な成果： ISERN メンバーだけが参加できる会議であり，発表内容も，具体的な研究成果というよりも，実証的ソフトウェア工学において取り組むべき課題や問題意識に関するものが多く，量的属性のデータ構造や第三者評価シナリオと直接結びつけることのできる発表は多くなかった．しかし，当該分野で世界的に活躍する研究者が数多く参加する会議であり，セッション間の休憩時間などでの意見交換は大変有益であった．その結果，量的属性のデータ構造については，ソフトウェアの構成管理システムにおけるデータ構造をベースにすべきではないかという意見と，不具合管理システムを中心に検討すべきではないかという意見とに分かれた．いずれにしても，ソフトウェア開発者あるいはプロジェクト管理者にアピールすることが重要であり，その意味では，データ構造よりもシナリオが重要であるとの意見も多く聞くことができた.

② International Symposium on Empirical Software Engineering and Measurement 2013 (ESEM 2013)

開催概要： ソフトウェアの計測に基づいた実証的な研究に関する国際シンポジウムである.

参加目的： 本受託研究における研究目標「多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価」に関連する研究の最新動向，特に，「アプリケーション実行履歴と開発作業の自動マッピング」に関する情報を入手すると共に，「計測結果の非改ざん性保証」について参加者との意見交換を行う.

主な成果： Software Development and Experiences セッション，Study Methodologies and Tools セッションにおける発表を中心に聴講した結果，インタビューやアンケートなどによる情報収集を元にソフトウェア開発作業を分析している開発現場が多いことがわかった．本受託研究の研究目標「多人数の開発作業を正当

なデータに基づいて解析・可視化する方式の設計と実証的評価」は同様の分析を行う際の信頼性確保とコストの低下を目的としており,本受託研究の成果が幅広く適用可能である可能性を再確認できた.また,キーノート公演 "Towards Understanding Replication of Software Engineering Experiments" はエビデンスに基づいたソフトウェア開発に関する知識の構築について公演された.計測の条件がわずかに異なる複数の研究結果の違いから,それらの条件がソフトウェア開発に与える影響を明らかにする事と，そのために必要となる計測データの収集・共有を支援する仕組みの重要性についてである.本受託研究とは,証拠に基づいた検証とそのためのデータ収集・共有の仕組みと言う点で類似しており，データに基づいた分析結果の再現性(Replication)の概念はシナリオ構築，および体系化において参考になった.

③ International Conference on Automated Software Engineering (ASE 2013)

開催概要： ソフトウェア工学研究において提案された技術の自動化に関する世界トップランクの国際会議の一つ.

参加目的： 本受託研究における研究目標「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」に関連する研究の最新動向，特に,「低品質モジュールメトリクスの選定」に関する情報を入手すると共に,「低品質モジュールの定義」について参加者との意見交換を行う.

主な成果： Keynote や一般の研究発表において,human aspect に関する課題や研究成果が数多く紹介された.課題としては,分散開発における合意形成の困難さ,仕様書や欠陥報告書などに掲載される情報の欠落など,が指摘されていた.これら課題は,ソフトウェア品質と強く関連しており,低品質モジュールの解析や可視化の方式を検討する上で大いに参考になり,また,低品質モジュールの解析や可視化の実現は，それら課題の解決にもつながると感じられた.

④ International Conference on Software Maintenance 2013 (ICSM 2013)

開催概要： ソフトウェアの保守と進化に関する，最も歴史と権威のある国際会議の一つ.

参加目的： 本受託研究における研究目標「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価」に関連する研究の最新動向,特に,「量的属性の解析・可視化をベースとした第三者評価シナリオ」に関する情報を入手すると共に，参加者との意見交換を行う.

主な成果： 基調講演，Research Track における発表を中心に聴講した結果，ソフトウェア保守の研究分野においても,開発管理データ（保守に伴うデータを含む）の分析やマイニングが盛んに行われており，かつ，その多くで，オープンソースソフトウェア(OSS)開発を研究対象としていることが分かった.以上 2 点は,本受託研究の目的やアプローチと合致するものであり,本受託研究の成果をソフトウェア保守においても利活用できる可能性が高いこと改めて確認することができた．なお，Industry Track において企業研究者による発表が 6 件行われたが，Research Track 等における発表の多くは，大学の研究者によるも

のであり，企業研究者との共著論文はもちろん，企業との共同研究の成果を発表する論文もごくわずかであった.多くの研究でOSS開発が研究対象となっていることと呼応するが，産学連携については，その必要性や具体的なアプローチなどについて議論の余地があると感じられた.

⑤ Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering 2013 (ESEC/FSE 2013)

開催概要： 隔年開催される ACM/SIGSOFT 主催のソフトウェア工学における基盤に関する国際シンポジウムとヨーロッパソフトウェア工学会議のジョイントミーティング．ソフトウェア工学におけるトップ国際会議の一つ.

参加目的： 本受託研究における研究目標「欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」に関連する研究の最新動向，特に，「ソフトウェア開発における欠陥トピックのトピックモデリングによる(半)自動特定」に関する情報を入手すると共に，「ソースコード中の欠陥が収束・発散するプロセスに着目した第三者評価シナリオ」について参加者との意見交換を行う.

主な成果： チュートリアル「Statistics in Software Engineering」では，「ソフトウェア工学の多くのデータは正規分布しない」や「計測されるメトリクスには強い相関があることが多い」といった重要な注意事項がまとめられ興味深かった.欠陥を識別したモデル構築に関して講演者の一人と議論した際,現状の技術では難しいとの意見をもらった．本委託研究の「(IT1-1)欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」が達成すれば学術的にもインパクトを与えることができると感じた.

⑥ Asia-Pacific Software Engineering Conference 2013 (APSEC 2013)

開催概要： ソフトウェア工学の，アジア・環太平洋地域における中心的な国際会議.

参加目的： 本受託研究における研究目標「多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価に関連する研究の最新動向，特に，「多人数の開発作業を正当なデータに基づいて解析・可視化するツール・システムの設計・実装」に関する情報を入手すると共に，「アプリケーション実行履歴と開発作業の自動マッピング」について参加者との意見交換を行う.

主な成果： 基調講演，Research Track における発表を中心に聴講した．Research Track や Industry Track において，多くの日本人研究者の発表があった．特に，企業からの参加者や発表者が多かったことが印象的であった.開発履歴のマイニングや可視化は企業でも関心が高いことを確認した.企業でリポジトリマイニングの可視化が有用とされているという報告から,本受託研究の意義を再確認できた.

⑦ International Workshop on Empirical Software Engineering in Practice 2013 (IWESEP

2013)

開催概要： エンピリカルソフトウェア工学における，比較的新しい国際ワークショップ.

参加目的： 本受託研究における研究目標「欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」に関連する研究の最新動向，特に，「欠陥トピックやトピック間の関係を説明するソフトウェアメトリクスの構造方程式モデリング SEM による特定」に関する情報を入手すると共に，「ソフトウェア開発における欠陥トピックのトピックモデリングによる（半）自動特定」について参加者との意見交換を行う.

主な成果： 「ソフトウェア開発における欠陥トピックのトピックモデリングによる（半）自動特定」に関連する以下の投稿論文が採択され，発表を行った.

Natthakul Pingclasai, Hideaki Hata and Kenichi Matsumoto, “Classifying Bug Reports to Bugs and Other Requests Using Topic Modeling,” Proc. of IWESEP 2013, pp.13-18 (2013).

本論文は，トピックモデリングにより欠陥レポートを分類する．従来の単語を属性とした分類モデルと比べ分類精度が高いことを報告している.研究目標①「欠陥の収束･発散プロセスをトピック別に評価するモデルの構築と実証的評価」の初期成果であり，予定どおりの進捗である．本論文は，少数のデータに対する適用実験であり,結果の一般性について疑問が残るという指摘を受けた.また，今後の研究に活かすことができる，技術的コメントを得た.

⑧ International Conference on Software Engineering 2014 (ICSE 2014)

開催概要： ソフトウェア工学における世界最高峰の国際学術会議.

参加目的： 本受託研究における研究目標「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」に関連する研究の最新動向，特に，「同方式の実証実験」に関する情報を入手すると共に，「同方式に基づくプロトタイプの詳細設計」について参加者との意見交換を行う.

主な成果： 基調講演，research track における発表を中心に聴講した．継続的な変更はソフトウェアの成功に不可欠であり,変更の積み重ねはソフトウェア進化として研究対象となっている．論文「Software Evolution and Maintenance」ではソフトウェア進化研究の将来の課題として,進化的ソフトウェア開発の問題に則したプラクティスの提示，ソフトウェア変更のコンセプト捜索･インパクト分析の拡張，ソフトウェア進化向けの研究方法論の開発，ソフトウェア進化教育などをまとめている．モジュールの品質の構造的･時間的特性を解析する上で有用な観点となる．論文「Characterizing Defect Trends in Software Support」では，運用時の欠陥報告の特徴を商用とオープンソースのデータから分析している．分析の結果欠陥報告曲線の特徴を予測する特徴量として，プロダクトのタイプ，ライセンスモデル，リリース間のサイクル時間を用いるとよいことを報告している.本研究の低品質モジュールの時間的特性を解析する上で，重要な指標となる.

⑨ Working Conference on Mining Software Repositories 2014 (MSR 2014)

開催概要： 近年活発に研究されている研究領域マイニングソフトウェアリポジトリ(MSR)の中心となる国際会議.

参加目的： 本受託研究における研究目標「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」に関連する研究の最新動向，特に，「低品質モジュールの構造的・時間的特性を解析・可視化の実証実験」に関する情報を入手すると共に，「低品質モジュールメトリクスに対する標準値域の導入」について参加者との意見交換を行う.

主な成果： 低品質モジュールの分析に関する興味深い研究が発表された.論文「Works For Me! Characterizing Non-reproducible Bug Reports」では「再現性のない欠陥」が，論文「Characterizing and Predicting Blocking Bugs in Open Source Projects」では「ブロッキングバグ（プロジェクト全体の進行を阻害する，深刻な欠陥)」が，論文「An Empirical Study of Dormant Bugs」では「休眠バグ（混入されてから発見されるまで時間のかかる欠陥)」が，それぞれ研究対象となっていた．欠陥の種別・特徴別に分析を行う点は，研究目標「欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」のアイディアとも考え方が近い．特に論文「An Empirical Study of Dormant Bugs」は，欠陥を時系列で詳細に分析するものであり，研究目標「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」を進める上で参考になった.

⑩ International Conference on Software Maintenance and Evolution 2014 (ICSME 2014)

開催概要： ソフトウェアの保守と進化に関する，最も歴史と権威のある国際会議の一つ.

参加目的： 本受託研究における研究目標「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価」に関連する研究の最新動向，特に，「同方式の実証実験」に関する情報を入手すると共に，「同方式に基づくプロトタイプの実装」について参加者との意見交換を行う.

主な成果： 基調講演，Research Track における発表を中心に聴講した結果，ソフトウェア保守の研究分野においても，ソフトウェアプロジェクトデータ（保守に伴うデータを含む）を根拠データとして問題解決に取り組む実証的（Empirical）なアプローチが盛んに行われており，特に，ソースコードを対象とした実証実験を含む研究が増えつつある（復活しつつある）ことが分かった．こうした研究動向は，本受託研究の目的やアプローチと合致するものであり，実証実験の計画・実施において参考となる情報を得ることができた．また，Tools Track では，ソフトウェアプロジェクトデータの解析・可視化を目的としたツールにおけるユーザインタフェースの実例を数多く目にすることができ，プロトタイプ実装にも大いに参考となる情報を得ることができた.

・学会発表等の実施状況

・ Natthakul Pingclasai, Hideaki Hata and Kenichi Matsumoto, “Classifying Bug

Reports to Bugs and Other Requests Using Topic Modeling," Proc. of International Workshop on Empirical Software Engineering in Practice (IWESEP 2013), pp.13-18, 2013.
・ P. Phannachitta, J. Keung, A. Monden and K. Matsumoto, "Scaling Up Analogy-based Software Effort Estimation - A Comparison of Multiple Hadoop Implementation Schemes," Companion Proc. 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, Workshop on Innovative Software Development Methodologies and Practices, pp.65-72, 2014.
・ 池田祥平, 上野秀剛,"第三者によるソフトウェア開発作業評価のための作業記録の保護手法", 情報処理学会研究報告, Vol.2014-SE-185, No.2, pp.1-8, 2014.
・ Saya Onoue, Hideaki Hata, and Kenichi Matsumoto, "Software population pyramids: the current and the future of OSS development communities," Proc. of 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2014), Article 34 , 4 pages, 2014.
・ 藤野啓介, 坂口英司, Papon Yongpisanpop, 伊原彰紀, 松本健一, "偏差値メトリクスを用いた欠陥モジュール予測モデルの提案", ウィンターワークショップ 2015・イン・宜野湾, 2015.

2.3 研究実施体制

本委託研究における研究実施体制を図 2-2 に示す.

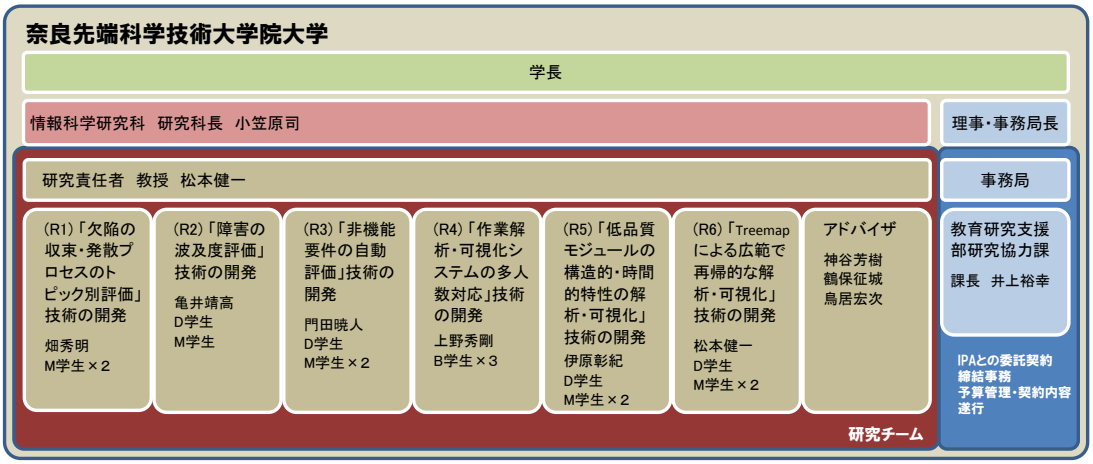


図 2-2 研究実施体制

研究責任者のプロフィールとメンバーの役割分担は次の通りである.

・研究責任者のプロフィール

(ふりがな) 氏　名	まつもと　けんいち 松本　　健一
生年月日	昭和 37 年 8 月 21 日
所属機関・ 団体名	国立大学法人　奈良先端科学技術大学院大学
所属 (部署名)	情報科学研究科
役　職	教授
住　所	〒630-0192 奈良県生駒市高山町８９１６番地の５
TEL	0743-72-5310
E-mail	matumoto@is.naist.jp

【学歴（大学卒業以降）】		【職歴】	
昭和 60 年 3 月	大阪大学基礎工学部卒業	平成元年 4 月	大阪大学基礎工学部・助手
昭和 62 年 3 月	大阪大学基礎工学研究科博士前期課程修了	平成 5 年 4 月	奈良先端科学技術大学院大学情報科学研究科・助教授
平成元年 4 月	大阪大学基礎工学研究科博士後期課程中途退学	平成 13 年 4 月	同・教授

・役割分担

<table>
<tr><td>実施機関名</td><td colspan="3">国立大学法人　奈良先端科学技術大学院大学情報科学研究科</td></tr>
<tr><td>研究責任者</td><td colspan="3">教授　松本　健一</td></tr>
<tr><td>連絡担当者</td><td colspan="3">教授　松本　健一</td></tr>
<tr><td rowspan="15">主な実施場所
および研究者</td><td colspan="3">国立大学法人　奈良先端科学技術大学院大学
〒630-0192
奈良県生駒市高山町８９１６番地の５
TEL　0743-72-5310　　FAX 0743-72-5319
e-mail　matumoto@is.naist.jp</td></tr>
<tr><td>氏名</td><td>職種</td><td>担当内容</td></tr>
<tr><td>松本健一</td><td>教授</td><td>全体のとりまとめ,
「Treemap による広範で再帰的な解析・可視化」技術の開発</td></tr>
<tr><td>門田暁人</td><td>准教授</td><td>「非機能要件の自動評価」技術の開発</td></tr>
<tr><td>伊原彰紀</td><td>助教</td><td>「低品質モジュールの構造的・時間的特性の解析・可視化」技術の開発</td></tr>
<tr><td>畑秀明</td><td>助教</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
<tr><td>上野秀剛</td><td>博士研究員</td><td>「作業解析・可視化システムの多人数対応」技術の開発</td></tr>
<tr><td>亀井靖高</td><td>博士研究員</td><td>「障害の波及度評価」技術の開発</td></tr>
<tr><td>神谷芳樹</td><td>非常勤講師</td><td>研究の内容や進捗への助言</td></tr>
<tr><td>鶴保征城</td><td>非常勤講師</td><td>研究の内容や進捗への助言，TERAS との連携</td></tr>
<tr><td>鳥居宏次</td><td>非常勤講師</td><td>研究の内容や進捗への助言</td></tr>
<tr><td>藤原賢二</td><td>研究技術員</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
<tr><td>藤原雄介</td><td>研究技術員</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
<tr><td>中山直輝</td><td>研究技術員</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
<tr><td>湯月亮平</td><td>研究技術員</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
<tr><td>尾上紗野</td><td>研究技術員</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
<tr><td>槇原絵里奈</td><td>研究技術員</td><td>「欠陥の収束・発散プロセスのトピック別評価」技術の開発</td></tr>
</table>

	藤原新	研究技術員	「欠陥の収束･発散プロセスのトピック別評価」技術の開発
	上村恭平	研究技術員	「欠陥の収束･発散プロセスのトピック別評価」技術の開発
	川島尚己	研究技術員	「欠陥の収束･発散プロセスのトピック別評価」技術の開発
	齋藤雄輔	研究技術員	「欠陥の収束･発散プロセスのトピック別評価」技術の開発
	福島崇文	研究技術員	「障害の波及度評価」技術の開発
	大坂陽	研究技術員	「障害の波及度評価」技術の開発
	小須田光	研究技術員	「障害の波及度評価」技術の開発
	Passakorn Phannachitta	研究技術員	「非機能要件の自動評価」技術の開発
	河居寛樹	研究技術員	「作業解析･可視化システムの多人数対応」技術の開発
	大橋亮太	研究技術員	「作業解析･可視化システムの多人数対応」技術の開発
	池田祥平	研究技術員	「作業解析･可視化システムの多人数対応」技術の開発
	一ノ瀬智浩	研究技術員	「作業解析･可視化システムの多人数対応」技術の開発
	幾谷吉晴	研究技術員	「作業解析･可視化システムの多人数対応」技術の開発
	應治沙織	研究技術員	「作業解析･可視化システムの多人数対応」技術の開発
	Papon Yongpisanpop	研究技術員	「低品質モジュールの構造的･時間的特性の解析・可視化」技術の開発
	藤野啓輔	研究技術員	「低品質モジュールの構造的･時間的特性の解析・可視化」技術の開発
	Chakkrit Tantithamthavorn	研究技術員	「低品質モジュールの構造的･時間的特性の解析・可視化」技術の開発
	内垣聖史	研究技術員	「低品質モジュールの構造的･時間的特性の解析・可視化」技術の開発
	坂口英司	研究技術員	「低品質モジュールの構造的･時間的特性の解析・可視化」技術の開発
	中川尊雄	研究技術員	「Treemap による広範で再帰的な解析・可視化」技術の開発

3 研究成果

3.1 研究目標１「欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」

3.1.1 当初の想定

(1) 研究内容

ソースコード中の欠陥（課題）が収束・発散するプロセスをそのトピック別に評価するモデルを構築する．トピック別の評価は，技術的には，マルチラベル評価問題と位置づけることができる．トピックモデリング，構造方程式モデリング（Structural Equation Modeling，SEM），および，ベイジアンネットワークの技術により評価モデルを構築する．まず,ソフトウェア開発における障害レポートにトピックモデリングを適用し,欠陥トピックを（半）自動で特定する．次に，それら欠陥トピックやトピック間の関係を説明することのできるソフトウェアメトリクスを SEM により特定する．SEM で得られたメトリクス間（変数間）の関係情報に基づき，ベイジアンネットワークでモデルを構築する．また，時系列分析を行い，欠陥の収束・発散プロセスを評価する．

トピックモデリング，構造方程式モデリング，ベイジアンネットワークによる欠陥評価モデル構築と，欠陥トピックの時系列分析のプロトタイプ実装には，統計解析向けプログラミング言語「R」を用いる．欠陥の収束・発散プロセスについての分析結果は，把握しやすくするため，可視化することを検討する．実証実験では，オープンソースソフトウェアのプロジェクトデータを利用する．具体的には，ソースコード版管理システムと障害管理システムのデータを取得し，提案するモデルと分析手法を適用する．提案手法の結果と実際の開発履歴とを比較することで，提案手法の妥当性，有用性，精度などを評価する．

(2) 当初の到達目標と期待される効果

到達目標は,「欠陥の収束・発散プロセスをトピック別に評価する技術の実現」である．具体的には，ソースコード中の欠陥（課題）が収束・発散するプロセスをそのトピック別に評価する技術を開発する．これにより，例えば,「評価対象プロジェクトでは，欠陥をおおむね効率よく除去している（収束させている）が，GUI に関する欠陥だけは除去し切れていない（収束に成功していない）.」,「モジュール A は主にファイル処理を行うモジュールであるが，ファイル処理に関する欠陥は意外に少なく（収束に成功しており），むしろ，例外処理に関する欠陥の増加（発散）を低品質の原因と考えるべきである.」といった，より詳細な品質評価が可能となる．「ソフトウェア品質の第三者評価」では,「プロセス実施の妥当性」が主要なスコープの一つとされている．本研究目標は，欠陥除去のプロセスがどのように実施されたのか，その妥当性評価を可能にしようとするものであり，第三者評価を支える主要な技術の一つと位置付けることができる．なお，トピック毎の評価は，技術的には，マルチラベル評価問題と位置づけることができる．これに対して，ソフトウェアの欠陥を対象とした従来研究の多くは，シングルラベル評価（欠陥の有無のみの評価・判別）を行うものである．2012 年度の本先導的研究支援事業でも,「低品質モジュール予測」として，欠陥が今後収束するのか発散するのかを評価（予測）してきているが，欠陥の種別には言及していない．

3.1.2 研究プロセスと成果

(1) 研究プロセス

トピックモデリング，構造方程式モデリング（Structural Equation Modeling，SEM），および，ベイジアンネットワークの技術により評価モデルを構築する．まず，ソフトウェア開発における障害レポートにトピックモデリングを適用し，欠陥トピックを（半）自動で特定する．次に，それら欠陥トピックやトピック間の関係を説明することのできるソフトウェアメトリクスを SEM により特定する．ただし，SEM は，メトリクス間（変数間）の関係分析に有用ではあるがプロセス評価には適していないとされている．そこで，SEM で得られた関係情報に基づき，ベイジアンネットワークにより欠陥トピックの評価モデルを構築する.最後に,構築した評価モデルをソースコードの構成管理情報に適用することで，欠陥の収束・発散プロセスのトピック別の評価を可能とする．なお，評価モデルの実装には，統計解析向けプログラミング言語「R」を用いる．具体的な作業項目は次の通り．

シナリオの策定
欠陥トピックの（半）自動特定
SEM によるメトリクスの特定
ベイジアンネットワークによるモデルの構築
プロトタイプの詳細設計
プロトタイプの実装
実証実験の計画・準備
実証実験の結果評価

(2) 具体的な研究成果（結果）

① シナリオ策定

学会での情報収集や意見交換を通じて，継続的で多面的なモニタリングや，欠陥の種類を識別した分析が求められていることを再認識した．第三者評価に向けて，欠陥トピックを継続的かつ多面的にモニタリングし，開発プロセスが適切に実施されたかを評価可能にすることを目指す.

② 欠陥トピックの（半）自動特定

欠陥レポートの欠陥記述内容から，トピックモデリングとクラスタリングによって，ほぼ自動的に欠陥トピックを特定する技術を開発した.

本技術は，以下のプロセスで構成されている（図 3-1-1 参照）.

1. 【テキスト取得】欠陥レポートから欠陥を記述するテキストを抽出する.
2. 【前処理】和文テキストは語への分かち書き．また，英文・和文とも不用語とストップワードを除去する.
3. 【トピックモデリング】前処理されたデータへトピックモデリングを適用する．ここで階層ディリクレ過程（Hierarchical Dirichlet Process，HDP）を採用したため，トピック数を明示的に設定する必要がない．出力は，各欠陥レポートに対して，各トピックの占める割合を値とするトピックベクトルとなる.

4. 【クラスタリング】先のトピックベクトルに基づき，似たトピックをまとめるようクラスタリングを行う．ここで EM アルゴリズムを採用したため，クラスタ数を明示的に設定する必要がない．
5. 【クラスタラベリング】NLP chunking によってクラスタに関連したフレーズを抽出する．これをクラスタのラベルとする．

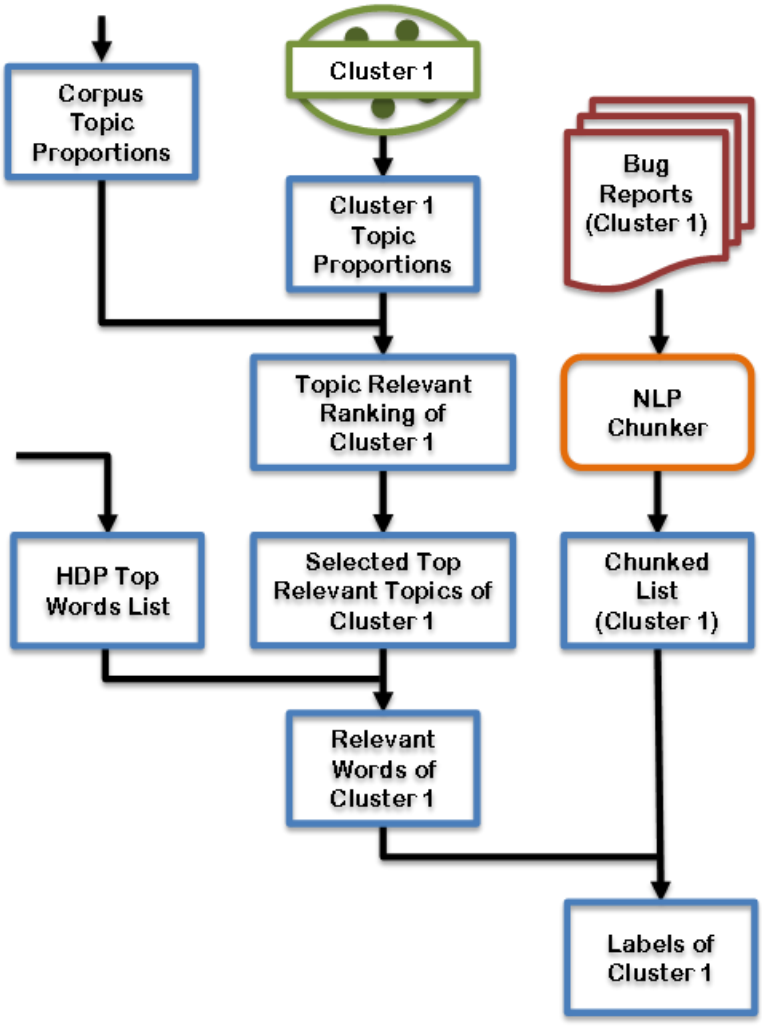


図 3-1-1 欠陥トピック特定プロセス

③ SEM によるメトリクスの特定

予測モデル構築のために以下のメトリクスを計測する．これらは多くのバグ予測研究で有用性が報告されている．

- PTi：Previous bug Topic i．以前に修正された欠陥トピック i の欠陥発生回数．1 つの欠陥レポートの修正を 1 回と数える．すなわち，同じ欠陥レポートの修正が対象のソースコードで何度も行われた場合も 1 回としている．また，1 つのコミットで複数の欠陥レポートが修正された場合（1 つのコミットメッセージに複数の欠陥レポート ID が含まれている場合），それぞれカウントする．以前に欠陥があったソースコー

ドには，同様のバグが再発しやすいのではないかと考えられる．これまでのシングルラベルのバグ予測においては，以前に修正されたバグ数は効果的なメトリクスと考えられている[Graves]．

- LOC：対象コードの行数（Lines Of Code）．サイズが大きいほどリスクが高いと考えられる．多くの複雑度メトリクスがLOCとの相関が高いと報告されている[Graves]．
- AddDel：最初の版と計測対象の版を比較した場合の追加行数と削除行数の和．大きく変更されたソースコードはリスクが高いと考えられる[Nagappan]．
- AGE：ソースコードの存在日時．新規に作成されたソースコードは欠陥を含む可能性が高いと考えられる[Kim]．
- HCM：変更プロセスの複雑度（History Complexity Metic）[Hassan]．シングルラベルのバグ予測でよいメトリクスだと知られている．

予測対象とするメトリクスは以下である．

- ATi：After bug Topic i．将来発生する欠陥トピック i のバグ発生回数． PTi と同様に，1つの欠陥レポートIDの発生を1回と数える．

予測対象メトリクスATiを，潜在変数を介して，収集したメトリクスから説明するモデルを構造方程式モデリングによって構築する．構造方程式モデリング（SEM）では，分析者が仮定した観測変数と潜在変数の因果関係を分析する．構造方程式モデリングは，変数間の関係を表すモデルの記述，パラメータの推定，モデルの改善の手順を繰り返して最終的なモデルを得る．モデルは，観測変数や潜在変数から影響があると思われる独立変数にパスを与えて記述する．最初は，計測メトリクスから潜在変数，また潜在変数から応答変数への全てのパスを与える．パラメータ推定は自動的に行われる．記述が不適切な場合は，エラーが出るので記述を改める．回帰係数の値が小さい（本稿では0.1に満たないものとした）パスを除去して，モデルを改善する．余分なパスやメトリクスを除去し，シンプルなモデルが得られたら終了する．プロジェクトによって違いはあるが，LOC，AGE，HCM が重要なメトリクスとして特定された．

④ プロトタイプ設計・実装

欠陥レポートを入力として欠陥トピックを特定するとともに，版管理システムから欠陥トピックの時系列変遷を分析するシステムを実装した．

⑤ 実証実験

活動期間も長く，健全に運営されている2つのOSSプロジェクト（Eclipse JDT Core，Eclipse PDE UI）を対象に実験を行った．これらのOSSプロジェクトでは英文で欠陥レポートが報告，管理されている．また，日本語の欠陥レポートにおいても欠陥トピック特定の効果を検証するため，企業プロジェクトの和文欠陥レポートを対象に実験を行った．

表3-1-1と表3-1-2にOSSプロジェクト（英文欠陥レポート）を対象として自動的に得られた欠陥トピックを示す．また，表3-1-3に企業プロジェクト（和文欠陥レポート）を対象として自動的に得られた欠陥トピックを示す．ただし，企業プロジェクトデータは版

管理システムのデータが得られなかったので，以降の分析は行っていない.

表 3-1-1 OSS プロジェクト Eclipse JDT Core の欠陥トピック

ID	欠陥ラベル
0	source internal error
1	pde sources test projects
2	type error class bart
3	resource leaks and errors
4	assist constructor completion
5	javadoc target attribute
6	created attachment proposed patch
7	ast node class initializations
8	don't delete link folder
9	string boolean int line
10	public class types search
11	same key type bindings
12	external class file folders
13	name references jdt core

表 3-1-2 OSS プロジェクト Eclipse PDE UI の欠陥トピック

ID	欠陥ラベル
0	created attachment full patch code
1	type view
2	created attachment fix problem
3	imported binary manifest
4	exporting plugins features
5	dialog font plugin search page
6	api type containers
7	package explorer right click point
8	features plugins
9	my ide eclipse launcher dialog
10	context help editor command field
11	maintenance http
12	executable extension point
13	product configuration wizard npe
14	plugin import wizard
15	example https
16	some unresolved method error
17	many unresolved method errors
18	target platform preference page press
19	feature based launch configuration

表 3-1-3 企業プロジェクトの欠陥トピック

ID	欠陥ラベル
0	条件 sfd 記載 内容 合う
1	画面 previous screen ボタン
2	黄色 車両 表示
3	ソフト ｔｃｄ ブザー
4	lcu operation mode ato モード
5	no 表示 信号
6	セグ 表示 見る
7	送信 一緒 停止 伝送 異常
8	送信 来る データ ブロック 格納
9	試験 standby mode key

図 3-1-2 と図 3-1-3 に OSS プロジェクトを対象として得られた構造方程式モデリング(SEM)のパス図を示す．モデルは，各予測対象メトリクス ATi を，対応する過去の欠陥トピック修正メトリクス PTi と潜在変数で説明するよう構築した．また，潜在変数はその他のメトリクスで説明させる．これによって，潜在変数で一般的な欠陥リスクを表し，特定の過去欠陥トピックでその欠陥についてのリスクを表すことを意図している．対象のうち活動期間がより長い Eclipse JDT Core では，潜在変数 Factor1 へは AGE と HCM のみが影響があるとして残った（図 3-1-2 参照）．すなわち，本プロジェクトでは，存在日時が短く変更プロセスが複雑なものが一般的な欠陥リスクと考えられる．

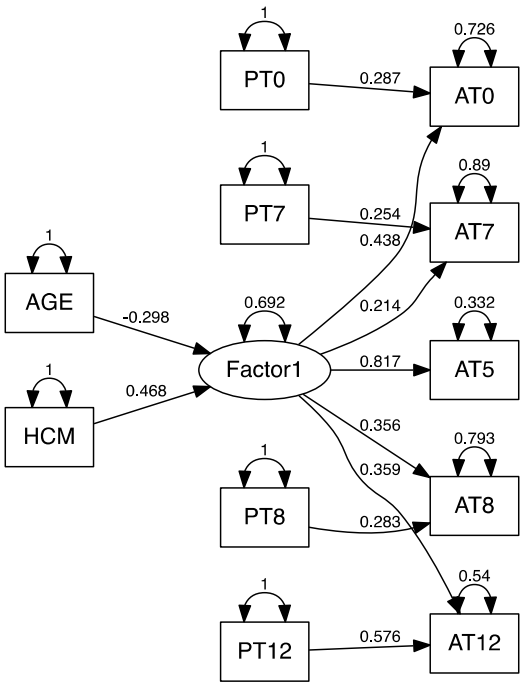


図 3-1-2 OSS プロジェクト Eclipse JDT Core の構造方程式モデリングのパス図

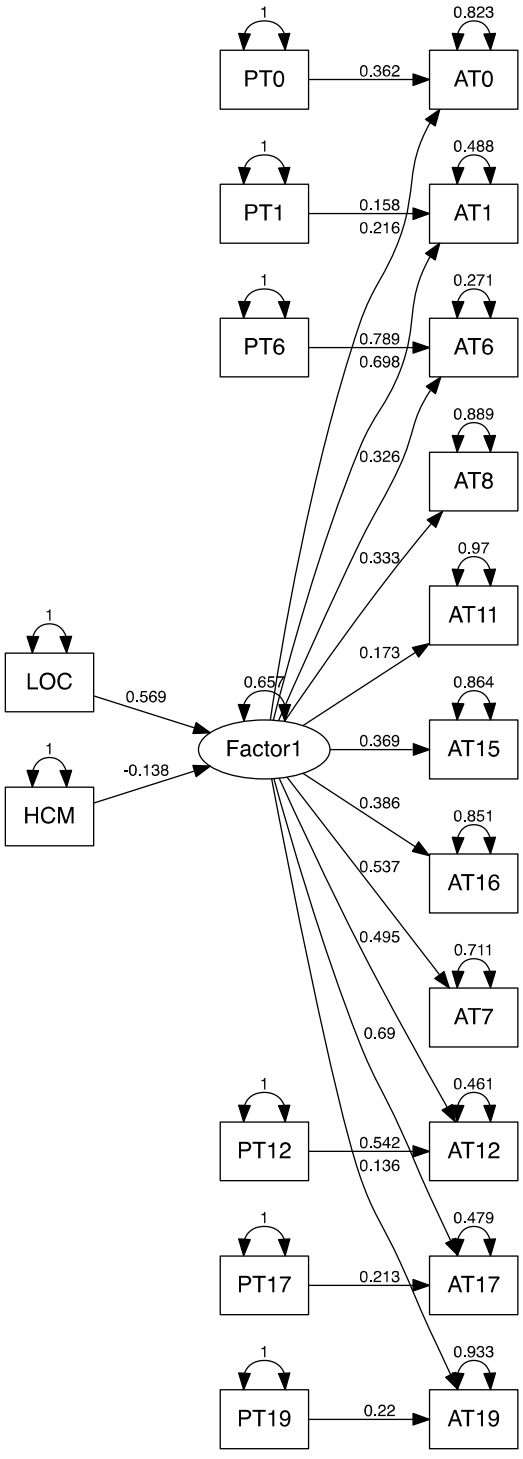


図 3-1-3 OSS プロジェクト Eclipse PDE UI の構造方程式モデリングのパス図

一方図 3-1-3 のプロジェクト Eclipse PDE UI では，潜在変数 Factor1 へは LOC と HCM のみが残った．すなわち，本プロジェクトでは，行数と変更プロセスの複雑度が一般的なリスクに関わると考えられる．予測対象メトリクス ATi によっては潜在変数が影響を与え

ないものもあった．これらは図からは省略している．

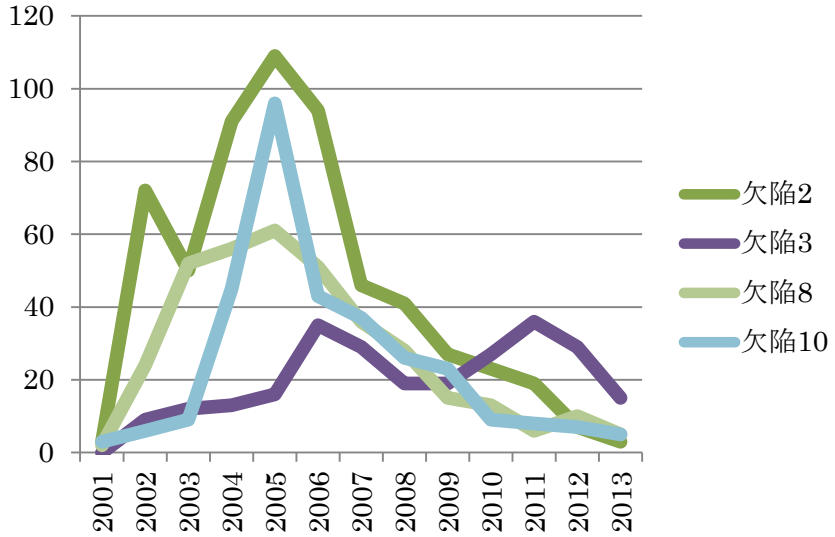


図 3-1-4 OSS プロジェクト Eclipse JDT Core の欠陥トピック発生数推移

図 3-1-4 に，欠陥トピックを版管理システムとリンクすることで得られた，プロジェクト Eclipse JDT Core の欠陥トピック（一部）発生数推移の様子を示す．これから，欠陥 2，欠陥 8，欠陥 10 は 2000 年代前半に多く発生したが，期間の後半には収束していることが分かる．一方，欠陥 3 は 2006 年と 2011 年に発生数が増加していることから，期間の後半も注意が必要なことが分かる．

構造方程式モデリングで得られたパス図から，ベイジアンネットワークによって予測モデルを構築した[Hata]．階層ベイズモデルは，統計モデルのパラメータに階層構造を持たせてベイズ推定するモデルであり，ベイジアンネットワークの一種である．将来の欠陥トピック発生数を一般化線形混合モデルで表現し，ファイルの個体差も含めた階層ベイズモデルを構築した．まず，それぞれの将来欠陥トピック発生数 ATi をポアソン分布で表現した．これは，欠陥トピック数が離散値で，ゼロ以上の範囲となるためである．ポアソン分布は，平均λをパラメータとして持つ．このパラメータλを，構造方程式モデリングで得られた説明変数となるメトリクスで表現した．潜在変数 Factor は，平均 μ と分散 σ のパラメータを持つ正規分布で表現した．パラメータ μ は計測したメトリクスでモデル化し，非負のパラメータ σ は無情報事前分布で与えた．無情報事前分布とは，$[-\infty, \infty]$ の範囲で任意の値をとってよい分布で，ここでは[0, 100]の範囲の一様分布とした．潜在変数を含む全てのパラメータはマルコフ連鎖モンテカルロ（MCMC）サンプリングで推定した．将来欠陥トピック発生数 ATi について得られたサンプル列の中央値を予測値とした．

本実験では，直前の版を学習データとして学習させ予測する．具体的には，3.4 版を学習し 3.5 版の将来欠陥トピックを予測した．結果を表 3-1-4 に示す．予測対象データには 1400 程度のモジュールがあるが，各欠陥トピックの発生するモジュールの割合は大変小さい．このような場合ほとんど予測できないこともあるが，提案手法で 0.11 以上の F 値を得

られた.

表 3-1-4 OSS プロジェクト Eclipse JDT Core の欠陥トピック予測

欠陥トピック	割合	F 値
AT0	1.8%	0.27
AT5	0.48%	0.12
AT7	0.77%	0.11
AT8	4.5%	0.14
AT12	2.0%	0.23

3.1.3 課題とその対応

欠陥トピックの特定における，トピックモデリングとクラスタリングにはパラメータの設定が必要であり，実務者には難しいと思われる．そこで本研究では，パラメータをも推定する技術を実装することで，ほぼ自動的に実行することが可能となった.

3.2 研究目標 2「ソフトウェア障害の波及度を定量的に評価する方式の設計と実証的評価」

3.2.1 当初の想定

(1) 研究内容

ソフトウェア開発で発生する障害の波及度を定量的に評価する方式を設計する．ソフトウェア開発者に対してヒアリングを行い，現在は暗黙知である「波及度の算出方法」を形式化し，障害の「優先度」(除去の優先順位) として決定する手順を，第三者評価シナリオとしてまとめる．ヒアリングに際しては，事前に仮想シナリオを準備する．仮想シナリオにおける，ヒアリング対象者の各障害に対する優先順位付けの結果を分析し，一般化することで第三者評価シナリオとする.

波及度の算出には，クラッシュレポートを利用する．クラッシュレポートからは，障害の発生時期やユーザ環境 (OS やバージョン) 等の特徴量を抽出する．次に，算出した波及度と，障害管理システムが蓄積・管理するデータとの関連づけを，トピックマイニング技術を用いて行う．トピックマイニング技術には，教師データを必要とせず，かつ，近年高精度での推定が可能であると示唆されている LDA(Latent Dirichlet Allocation)を採用する．また，障害除去の実績データ (実際の除去順序) と波及度との関係を機械学習の手法の１つであるランダムフォレストを用いて，改めてモデル化する.

得られた成果に基づき，統計解析向けプログラミング言語「R」を用いてプロトタイプとして実装する．モデルの構築のパラメータ推定には，OSS のクラッシュリポジトリから入手予定のデータを用いる．また，実証実験では，プロトタイプとして構築したモデルの性能評価を行う．モデル評価には交差検証法 (データセットを分割し，一部をモデル構築用に用いて，残りを性能の評価に用いる方法) を用いる.

(2) 当初の到達目標と期待される効果

到達目標は,「ソフトウェア障害の波及度を定量的に評価する技術の実現」である．具体

的には，ソフトウェア開発で発生する障害（課題）の波及度を定量的に評価する技術を開発する．障害の「波及度」とは，当該障害が発生するユーザ環境の多寡を表す指標である．ソフトウェアがインフラ環境として利用される昨今，一部のユーザのみで発生する障害か，それとも，社会全体に影響を及ぼしかねない障害かを正しく，かつ，迅速に把握することの意義は大きい．なお，波及度とは別の指標として，当該障害によりユーザが被る不利益の程度を表す「深刻度」が広く用いられている．2012 年度の本先導的研究支援事業でも対象とした「障害管理システム」の多くは，深刻度を管理対象の１つとしているが，波及度を自動計測し，評価・管理する枠組みは用意されていない．このため，ソフトウェア開発者は，どの障害から優先的に取り除くべきかを経験的に判断しているのが現状である．本研究目標の達成により，障害除去における意思決定（除去の優先順位の決定）における不透明性が解消され，より詳細な「ソフトウェア品質の第三者評価」の実現につながる．

3.2.2 研究プロセスと成果

(1) 研究プロセス

まず，ソフトウェア開発者に対してヒアリングを行い，現在は暗黙知である「波及度の算出方法」を形式化し，障害の「優先度」（除去の優先順位）として決定する手順を，第三者評価シナリオとしてまとめる．波及度の算出には，クラッシュレポートシステムが生成する情報（クラッシュレポート）を利用する．クラッシュレポートシステムは，ユーザ環境下で障害が発生すると，スタックトレース等のデータを自動的に収集し，クラッシュレポートとして開発プロジェクトに送信する．クラッシュレポートを解析することで，障害の発生時期やユーザ環境（OS やバージョン）等の特徴量を抽出することができる．

次に，算出した波及度と，（ソフトウェア開発で従来から広く利用されている）障害管理システムが蓄積・管理するデータ（深刻度を含む）との関連づけを，トピックマイニング技術を用いて行う．これにより，障害の波及度を高くする要因を細粒度で分析することが可能となる．また，障害除去の実績データ（実際の除去順序）と波及度との関係を機械学習の手法で，改めてモデル化する．具体的な作業項目は次の通り．

開発者へのヒアリング
波及度評価の形式化．シナリオの策定
トピックマイニングによる深刻度等との関連づけ
機械学習によるモデルの構築
可視化システムとの連携方式の検討
実証実験の計画・準備
実証実験の結果評価

(2) 具体的な研究成果（結果）

① 開発者へのヒアリング，および，既存研究調査に基づく波及度評価の定式化と第三者評価シナリオの策定

1) 開発者へのヒアリング

障害除去の優先順位，および，波及度をどのように定式化するかについて，開発者へのヒアリングを行った．ヒアリングは，通信系ソフトウェア開発の開発経験が 10 年以上の実

務者２名を対象に行った.

結果として，深刻度（セキュリティ関係か否か等），波及度（どれだけの確率で障害が発生し，どれだけの利用者の環境で発生するか），および，修正コスト（修正に伴う影響範囲の大きさ，と修正範囲）が，障害の優先度（除去の優先順位）に大きく影響を与える３つの要因であることがわかった．また，障害の発生頻度，障害の発生しているユニークユーザ数，ユーザ間での障害の発生頻度のばらつき等で波及度を表現できることがわかった.

2) 既存研究調査

クラッシュレポートを活用した研究として，オープンソースプロジェクトを対象とした研究が行われていることがわかった[Dang][Dhaliwal][Kim].例えば,Kim らは,クラッシュレポート全体の大半を占めている,頻出する種類のクラッシュレポート(トップクラッシュ)に着目している．トップクラッシュとなりうるクラッシュの種類を特定するため，製品のリリース後から間もないクラッシュレポートの特徴から，トップクラッシュか否かを判定する予測モデルの構築を行った．また，Khomh らは，クラッシュレポートに含まれる情報（インストール日時や OS のバージョン）を用いて，クラッシュレポート群から同一の人物から送信されてきたものを推定する方法を紹介している.

また既存の障害管理システムを対象とした研究について行われている[Herraiz][Mockus]．例えば，Herraiz らや Mockus らの報告では，オープンソース開発プロジェクトにおける深刻度の数値は，初期値のまま報告されることや，経験の少ないユーザから報告されることが多く，バグ修正を決めるための優先度しては有用ではない可能性を示唆している．本調査によって，クラッシュレポートによって算出可能である，波及度を用いた各障害の優先度付けが有用であるというより強い根拠を得ることができた.

3) 第三者評価シナリオの策定

暗黙知である「波及度の算出方法」を形式化し，障害の「優先度」（除去の優先順位）として決定する手順を，第三者評価シナリオとしてまとめた.

波及度に関しては，1)開発者へのヒアリング，および，2)既存研究調査を通して，クラッシュレポートの「発生件数 × 発生人数（ユニークユーザ数）」とした．そして，波及度を用いた各障害に対する優先順位付けの結果を図 3-2-1 としてまとめた.

図 3-2-1 におけるステークホルダは，対象システムの「利用者」,「開発者」，および，障害除去の優先順位が妥当であるかを分析する「第三者評価機関」である.「利用者」が対象システムの使用中に予期せぬ不具合が発生すると，クラッシュレポートが自動的に生成されて，開発プロジェクトのクラッシュリポジトリに蓄積される.「開発者」は，障害管理システム中の既知の障害を随時修正する．今回の例では，同時期に５つの障害が既知であるとする.

「第三者評価機関」は，障害管理システムに蓄積された修正履歴を取得し，実際に修正された順序と，クラッシュリポジトリから算出される波及度の大きさを比較する.「第三者評価機関」は，実際に修正された順序と波及度に乖離がないかを確認することで，障害修正の優先度を評価できる．例えば，今回の例では，障害 ID「100131」が最も早期に修正されたものの，同時期に発生した「101892」の波及度が高いため，なぜユーザにとって波及

度の高い障害「101892」が優先的に修正されなかったのかを「第三者評価機関」は「開発者」に確認できる.

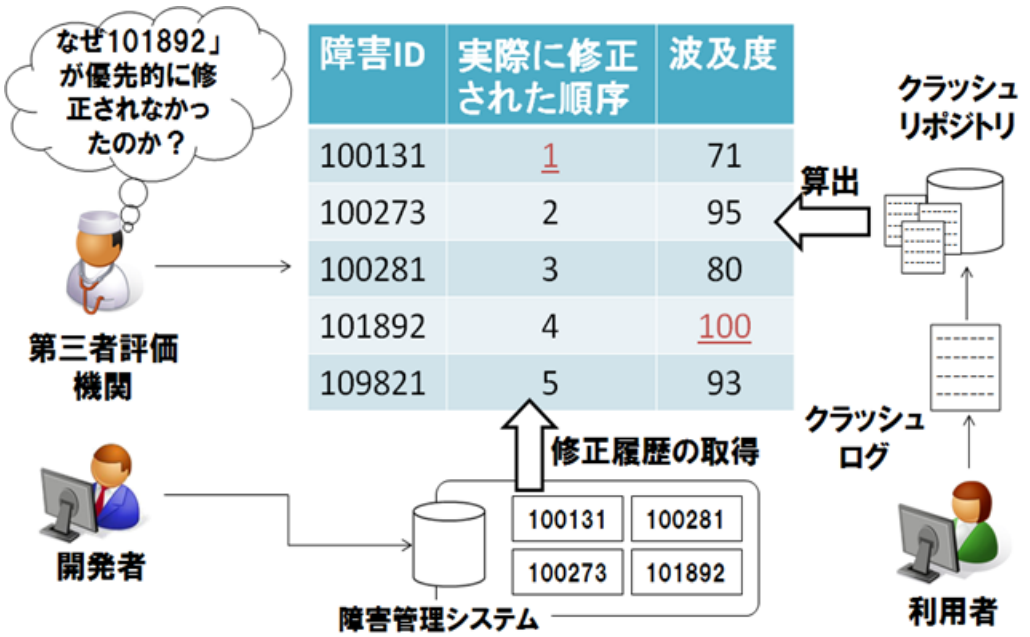


図 3-2-1　波及度に基づく第三者評価シナリオ例：障害修正プロセス評価

② クラッシュレポート解析ツールの作成・開発

クラッシュレポートシステムが生成する情報（クラッシュレポート）を利用して，波及度を算出する. クラッシュレポートシステムは, ユーザ環境下で障害が発生すると, スタックトレース等のデータを自動収集し，クラッシュレポートとして開発プロジェクトに送信する. クラッシュレポートを解析することで, 障害の発生時期やユーザ環境 (OS やバージョン）等の特徴量を抽出することができる.

クラッシュレポート群が与えられると，ア）ユニークユーザ数の計測，イ）モデル化のための波及度，および，ウ）特徴量の計測を実行し，出力するツールを作成した．プログラミング言語の Ruby によって実装され，規模はおおよそ 300 行である.

1）ユニークユーザ数の計測

波及度の計測には，各障害に対するクラッシュレポートの送信件数に加えて，当該障害に対するユニークユーザ数が必要である．しかしながら，クラッシュレポートシステムでは，個人情報保護の観点から，そのままではユーザを特定できる情報を得ることはできない.

既存研究[Khomh]で用いられている手法を本研究でも活用することで，ユニークユーザ数を近似的に求めることができた．Khomh らと同様に，クラッシュレポートにおける項目の内容が完全に一致するクラッシュレポートは同一のユーザであるとした．その上で，全クラッシュレポート中のユニークユーザ数を求めた．今後の実験における再現性向上のために，以下に本研究で用いたクラッシュレポートの項目一覧を示す.

- client_crash_date(ユーザ環境基準でのクラッシュ発生時間)
- install_age(該当ソフトのインストールからの経過時間)
- product(ソフトウェア)
- version(ソフトウェアのバージョン)
- build(プロダクトのビルド識別子)
- branch(ブランチ)
- os_name(OS 名)
- os_version(OS のバージョン)
- cpu_info(CPU 情報)
- addons_checked(アドオンの種類)
- flash_version(flash Player のバージョン)

2) ユニークユーザ数の計測

後述する③の（ア）の評価実験用データセット 2,882,365 件のクラッシュレポートを用いてユニークユーザ数を求めた場合（実験環境は CPU：Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz, CPU MHz ： 1200.000, cache size ： 12288 KB), 300 時間かかっても計算を終えることができなかった．各クラッシュレポートそれぞれの組み合わせに対して，クラッシュレポートの項目の比較を行う必要があったためである．そのため，実装上の工夫を行い，最終的に 3 時間以内にユニークユーザ数を求めることができた（詳細は 3.2.3 の「対応と課題」).

3) モデル化のための波及度，および，特徴量の計測

各クラッシュの発生件数，および，（ア）のユニークユーザ数から波及度を計測する．今回は，開発者が各障害に対してクラッシュレポートが送られてきていると認識するまで（障害管理システムに登録されている障害とクラッシュレポートが関連付けられるまで）に届いたクラッシュレポート数から波及度を算出した（図 3-2-2 参照）．

本ツールでは今後のさらなる分析のため，クラッシュレポート群から下記の特徴量の計測も可能にした．これらは，ユニークユーザ数の算出にも利用する．

- product(ソフトウェア)
- version(ソフトウェアのバージョン)
- os_name(OS 名)
- os_version(OS のバージョン)
- cpu_info(CPU 情報)
- addons_checked(アドオンの種類)
- flash_version(flash Player のバージョン)

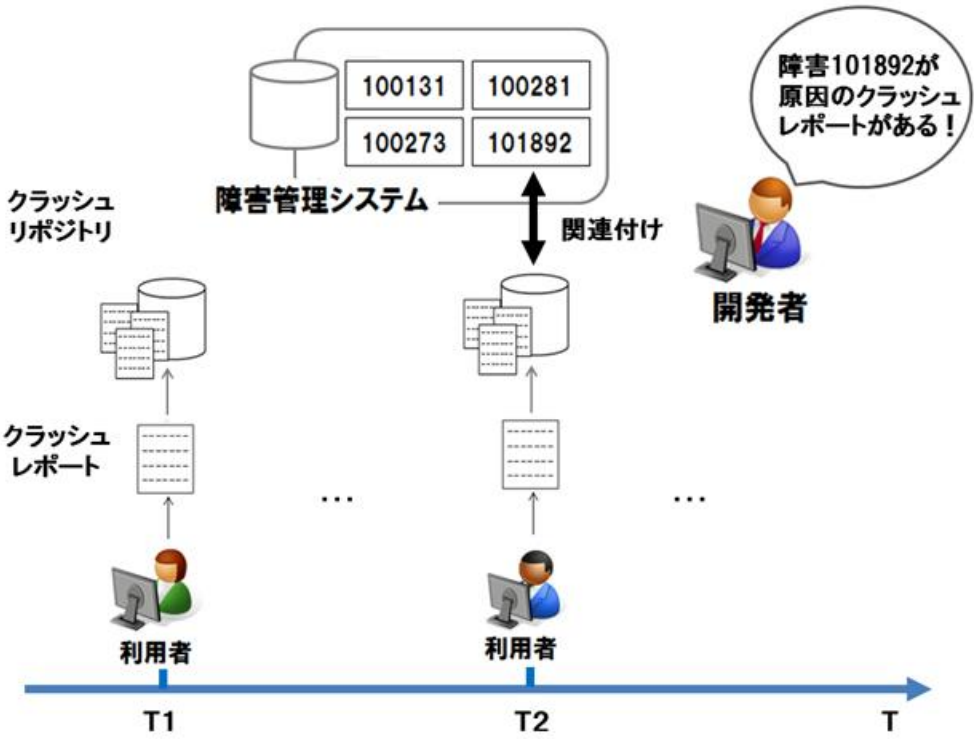


図 3-2-2　クラッシュレポートの発生から開発者が認識するまでの流れ

③ 波及度と障害修正順序の関係の可視化とモデル化

1) 評価実験用データセット

波及度によって障害除去の順序をどの程度予測できるかの評価実験を行うための評価用データセットとして，Firefox プロジェクトからクラッシュレポート，および，障害報告の収集を行った(表 3-2-1 参照).Firefox プロジェクトを選んだ評価対象とした理由は，当プロジェクトが 5 年以上のクラッシュレポートの導入実績があり，かつ，そのデータをオンライン上に公開しているためである.

これらのデータセットから，各障害の波及度について調査した．調査により得られた波及度のヒストグラムを図 3-2-3 に示す．横軸が波及度の値，縦軸が波及度の各値の頻度を表している.

［モデルの目的変数］除去順序を優先度として扱う予定であったが，障害が発生した日時等によるノイズが多く含まれてしまうため，除去にかかった期間を除去の優先度（短い期間ほど優先度が大きい）とした．除去の期間は，バグレポートに記載されている，障害の除去の開始，完了の日付から求めた．そして，全障害の中でも上位 10%以内の修正期間のものを，除去優先度の高い障害であると定義した.

表 3-2-1　評価用データセットの概要

	クラッシュリポジトリ	障害管理システム
対象期間	2011/01/01~2012/12/31	2011/01/01~2012/12/31
レポート数・障害数	2,882,365	422

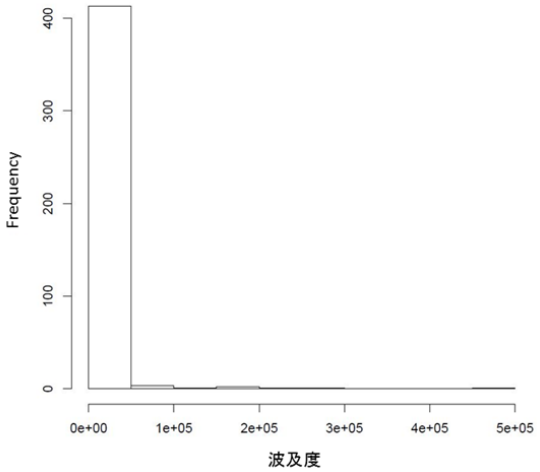


図 3-2-3　障害ごとの波及度の分布

2）波及度と障害修正順序の可視化

波及度と障害順序（修正期間）を図 3-2-4 に示す． 図 3-2-4 に示すように，波及度が相対的に大きいにも関わらず，修正期間が大きい障害が見てとれる（丸で囲まれた部分）．波及度と障害除去順序の関係を可視化し，障害除去順序の妥当性の評価，議論を容易にした．

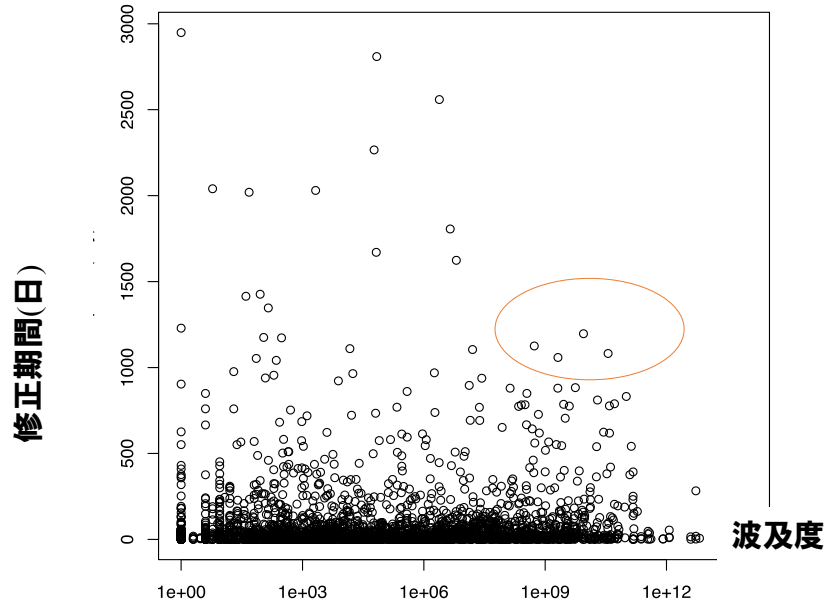


図 3-2-4　波及度と障害順序（修正期間）

3）モデルの性能評価

評価基準として，適合率（Precision），再現率（Recall）と F1 値（F1-measure）を用いた．適合率は，優先度が高いと予測したうち，実際に優先度が高い障害である割合を示す．再現率は，すべての優先度が高い障害のうち，優先度が高いと予測した割合を示す．F 値は，適合率と再現率の調和平均で与えられる．

10-交差検証法による性能評価の結果を，図 3-2-5 に示す．再現率については，中央値が 95%以上の結果を得ることができた．再現率と比べると適合率は小さい結果であった（おおよそ 20%）．しかしながら，今回のデータセットでは，優先度の高い障害は 10%しか含まれておらず，ベースライン（ランダムに優先度が高いか低いか）の適合率は 10%である．そのため，ベースラインと比較すると，波及度を用いることでベースラインと比べると，適合率の観点において約２倍の性能を達成することができた．

また，比較対象として，障害管理システムから計測可能な情報を用いて予測を行った場合，適合率，再現率，F1 値すべてにおいて，波及度を用いた場合と比べて小さい値であった．各中央値は，0.21，1.00，0.34 であった．

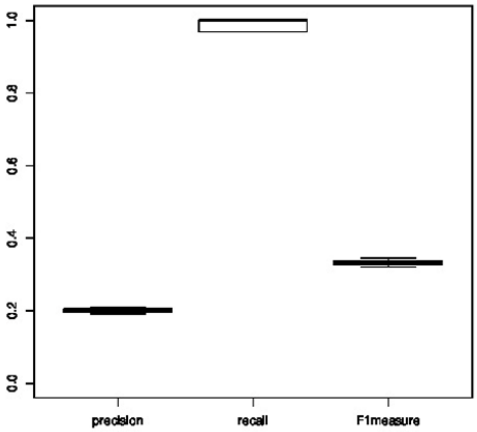


図 3-2-5　波及度による予測モデルの予測精度

④ 障害管理システムにおけるトピック事の波及度表示システムの作成

算出した波及度と，（ソフトウェア開発で従来から広く利用されている）障害管理システムが蓄積・管理するデータ（深刻度を含む）との関連づけを，トピックマイニング技術を用いて行った．これにより，障害の波及度を高くする要因を細粒度で分析することが可能となった．

1）障害管理システムデータに対するトピックモデリング

障害管理システムの Bugzilla に保存されている障害報告（バグレポート）を取得して，バグレポートに対してトピックモデリングを行った（表 3-2-2 参照）．今回はバグレポートの short description（1 行にまとめられた短いコメント）をもとにトピックモデリングツール MALLET を用いてトピックモデリングを行った.今回は MALLET でトピック数を 10，出力結果として，(1)各トピックにおける頻出頻度上位 20 単語を記述したファイルと(2)バグレポートの short description に記述されている単語がどのトピックに属しているか記述したファイルを出力した．(1)により各トピックがどのような単語の集まりなのかを頻出頻度上位 20 単語より手動で推定し定義することで，表 3-2-2 に示すトピックが含まれることを明らかにした．

2）Web サービス環境準備（プロトタイプシステム）

Web サービス上において，ソフトウェア障害の波及度を可視化するシステムを実現した．本サービスは，CentOS 上の Ruby on Rails で動作する．バグレポートのトピックモデリングの結果とクラッシュレポートから算出した波及度の値などをデータベースに格納しており，利用者の動作に応じてトピックに応じた波及度を表示する．

表 3-2-2　バグレポートのトピックモデリング

トピック名	主な単語	総数
HTML	css, html, page, gt, lt, table	156,156
ERROR MESSAGE	error, dialog, message, javascript	111,327
DOWNLOAD	update, download, version, release	143,149
BUILD	build, remove, add, support	192,026
TEST	test, intermittent, assertion, bug, fail	77,405
MAIL	mail, message, address, account	166,553
BROWSER	menu, bar, button, bookmarks	211,601
CRASH	crash, hangs, mozilla, error	180,299
ACCESS	access, server, password, http, request	126,215
MISC	doesn, work, shouldn, isn, open, java	127,583

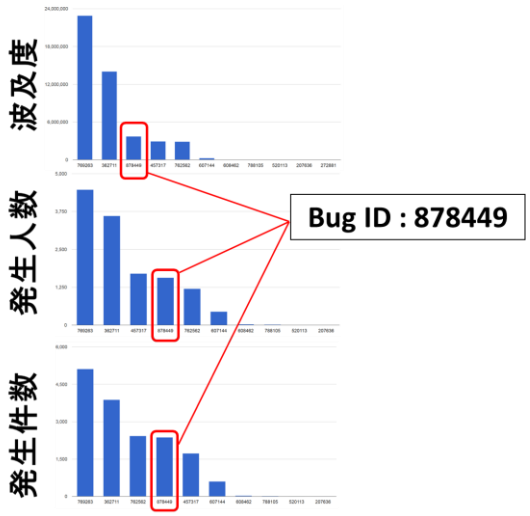


図 3-2-6　各障害報告の波及度可視化

（ア）各障害報告の波及度可視化

本機能では障害報告それぞれの波及度を視覚化することができる（図 3-2-6 参照）．表示したい障害報告の数は，5・10・20・50 の 4 種類を選択できる．なお，波及度の数値は高速化のために事前に計算しており，データベースに保存している．サービスはユーザの選択した表示数を上限として波及度の高い順番に棒グラフとして表示する．表示対象が選択した表示数に満たない場合は選択した表示数よりも少ない数の棒グラフが表示される．

波及度の他にも，波及度を計算するために用いられた発生件数と発生人数（ユニークユーザ数）を同様の形式で表示している．これにより，「発生件数は少ないが波及度が高い」というような Bug ID の存在を知ることができる可能性がある．

図 3-2-7　トピックの選択

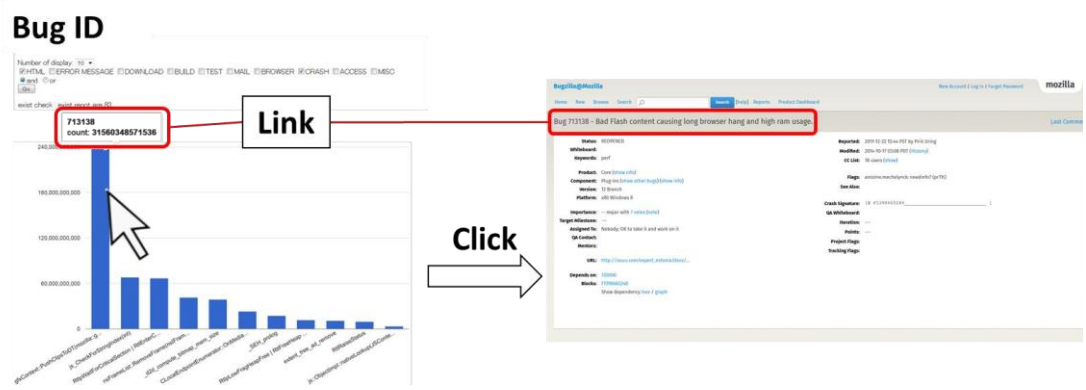


図 3-2-8　各障害報告の詳細への遷移

（イ）トピックの選択

波及度を棒グラフとして表示する際に，トピックに絞った分析のためにトピックの選択を実現した（図 3-2-7 参照）．複数のトピックを選択する場合は *and*，もしくは *or* を選択することで論理式と同様の絞り込みが可能となる．表示される棒グラフの数は選択した表示数を上限とするが，いくつの障害報告が条件に当てはまったかが数値で示される．

（ウ）各障害報告の詳細への遷移

表示された波及度の棒グラフをクリックすると，該当する Bugzilla 上の障害報告へ遷移することができる（図 3-2-8 参照）．これにより，構築したサービスには表示しきれていない障害の詳細を簡単にアクセスすることが可能となる．

3.2.3 課題とその対応

ユニークユーザ数の算出を現実的な時間で求めることができなかった． そこ で，ユーザ情報を全て連結したものをハッシュ化により圧縮し，必要なメモリ容 量を減少させることで，処理の高速化を実現した．また，障害と関連付けられて いるクラッシュレポートの種類を事前に調査することで，今回の対象でないク ラッシュレポートの判別に対する高速化を実現した．

3.3 研究目標 3「非機能要件の自動評価方式の設計と実証的評価」

3.3.1 当初の想定

(1) 研究内容

非機能要件の自動評価を行うための方式を設計する．まず，開発者へのヒアリングを通して，自動評価が必要なコンテキストやシナリオを明確にする．次に，形態素解析を用いて，RFP から非機能要件に関連する語句（キーワード）を抽出する方式について検討する．非機能要件は多数の種類があるため，要件ごとに，関連するキーワード群を選定する必要がある．次に，各キーワードの出現頻度を説明変数とし，要件の明確さを目的変数とするモデリングを行う方式について検討する．また，多数の RFP に基づいて，モデルのパラメータ推定を行う方式について検討する．以上の各手順において，どのようなキーワードを抽出すべきか，どのようなモデリング手法を用いるべきか，TF-IDF 法などによるキーワードの重み付けを行うべきか等を研究目標へ向けて検討していく．

プロトタイプ実装では，統計ソフトウェア R を用いて，キーワードと要件の明確さの関係を表すモデルを実際に構築する．モデル構築にあたっては，Web 上から入手可能な RFP（政府機関情報システム，地方自治体情報システム，病院情報システムなど）を用いて，エキスパートの判断により教師信号を与え，モデルのパラメータ推定を行う．

実証実験では，構築したモデルの性能評価を行う．評価においては，モデル構築用の RFP とは別に，モデル評価用の RFP を用意し，要件の明確さの予測を行う．そして，エキスパートの判断とモデルによる予測の誤差を評価する．また，精度よく予測できる要件とそうでない要件について，傾向や原因の調査を行う．

(2) 当初の到達目標と期待される効果

到達目標は，「非機能要件の自動評価技術の実現」である．具体的には，対象ソフトウェアに関する知識が十分でない第三者（ソフトウェアの要件定義書や RFP（Request For Proposal）の作成経験・知識が十分でない非エキスパート）でも，ソフトウェアの非機能要件の評価を可能とする技術を開発する．2012 年度の本先導的研究支援事業では，要件定義書や RFP を対象として，非機能要件が漏れなく明確に記載されているか否かを第三者が評価するための枠組み（非機能要件評価表，および，要件の明確さの評価指標）を開発し，システム発注・開発に 10 年以上携わってきたエキスパート 1 名によるケーススタディを実施し，テキストマイニングと機械学習の手法を応用することで，非機能要件の自動評価を実現する．ソフトウェア委託開発を成功させるためには，開発初期にユーザ要件を明確にすることが求められる．特に，非機能要件は，ソフトウェアアーキテクチャの決定や，保守コストに大きく影響することが知られており，その自動評価の実現は，「ソフトウェア品質の第三者評価」の普及と発展にも大きく資する．

3.3.2 研究プロセスと成果

(1) 研究プロセス

まず，評価対象となる要件定義書や RFP に対して形態素解析を行い，非機能要件に関する単語を抽出する．次に，TF-IDF 法（Term Frequency - Inverse Document Frequency method）などのテキストマイニング手法を用いて重要語を抽出すると共に，各重要語の出現頻度を計測する．次に，得られた抽出語と非機能要件とのマッピングを行う．最後に，機械学習

により，重要語と非機能要件の明確さの関係をモデル化する．得られたモデルを多数の RFP に適用し，その結果に基づいて手法の改良を繰り返す．具体的な作業項目は次の通り．

開発者へのヒアリング
シナリオの策定
非機能要件に現れる単語抽出
重要語抽出，出現頻度計測
重要語のマッピング
機械学習によるモデルの構築
プロトタイプの詳細設計
プロトタイプの実装
実証実験の計画・準備
実証実験の結果評価

(2) 具体的な研究成果（結果）

① 開発者へのヒアリングに基づくシナリオの策定

ソフトウェア開発企業の管理者，及び，開発実務経験者へのヒアリングに基づき，計画フェーズから要求仕様の確定に至る前の間に，３つのフェーズ（計画フェーズ，プロポーザルフェーズ，設計・製作フェーズ）において非機能要件が適切に決められていたのかを検証するというシナリオを策定した．

図 3-3-1 に示すように，本シナリオでは，計画フェーズにおける RFP の妥当性検証，プロポーザルフェーズにおける契約書の妥当性検証，設計・製作フェーズにおける要求仕様書の妥当性検証をそれぞれ行う．これにより，各フェーズにおいて非機能要件が網羅的に記述されているかを評価することが可能となるとともに，RFP において記述漏れがあった場合に後のフェーズにおいて改善されているかどうかや，RFP における記述が後のフェーズでも抜け落ちていないことを確認できる．

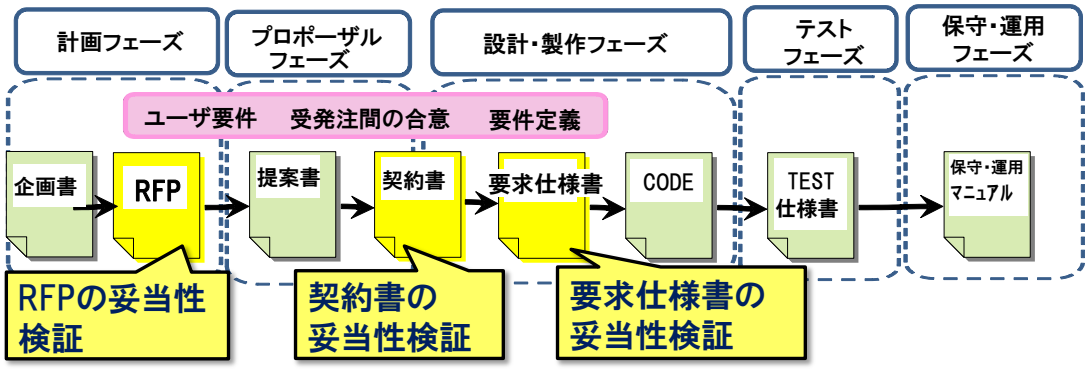


図 3-3-1 非機能要件の評価シナリオ

② 非機能要件に現れる単語抽出

非機能要件に現れる単語抽出を行うためには，まず，評価対象となる非機能要件を確定

する必要がある．本事業では，広くソフトウェアシステムにおいて重要となる次の 27 個の非機能要件を対象とした自動評価を実現した．

オペレーション，応答性，システム性能，負荷バランス，稼働品質，システム異常検知，障害要因，運用管理，システム管理，不正アクセス，アクセス権限，セキュリティ管理レベル，セキュリティ対応，認証機能，パスワード管理，ウイルス対策，情報漏洩対策，暗号処理，冗長化，RAID 構成，バックアップ管理，バックアップシステム，障害対策，障害管理，停止処理，リカバリ処理，バッチ処理

各非機能要件に対応する単語，重要語（非機能要件キーワード）を選定するための題材として，インターネット上で公開されている 70 件の RFP を用いた．その内訳は，図書情報システム（11 件），病院情報システム（10 件），大学情報システム（8 件），政府機関情報システム（14 件），自治体基幹情報システム（10 件），地方自治体業務システム（14 件）およびその他情報システム（3 件）である．

単語抽出においては，70 件の RFP に対して形態素解析を行い，名詞およびその複合語（名詞の組み合わせ・例えば操作と手順を組み合わせた「操作手順」）を抽出した．

③ 重要語抽出，出現頻度計測，重要語のマッピング

②で抽出した単語の出現頻度を計測し，その結果に基づいて重要語（非機能要件キーワード）の抽出を行った．まず，単語のうち出現頻度が 1 回のみのものを除外した．次に，TF-IDF 法に基づいて，出現頻度が高いが汎用的である語句を除外した．さらに，各非機能要件を表すのに適した単語のみを選定し，非機能要件とのマッピングを行った．その結果を表 3-3-1 から表 3-3-11 に示す．

④ 機械学習によるモデルの構築

非機能要件の記述の明確さを自動評価するのに適したモデリング手法を選定し，モデル構築を試行した．モデリング方法として，外れ値やノイズに対する頑健性と予測性能を兼ね備えた方法であるランダムフォレストを採用した．非機能要件キーワードの出現頻度を説明変数とし，非機能要件の記述の明確さを目的変数として，統計ツール R のランダムフォレストパッケージを用いて自動評価モデルを構築できることを確認した．

⑤ プロトタイプの詳細設計と実装

次の機能を持った自動評価システムについて，データ変換方式と入出力ファイル形式の詳細設計を行うとともにプロトタイプ実装を行った．

(1) 与えられた RFP をテキスト形式に変換し，HTML タグでマーキングする．
(2) (1)で得られたテキスト形式の RFP について，表 3-3-1 から 3-3-11 に基づいて，27 件の非機能要件の全てのキーワードの出現頻度を計測する．
(3) (1)(2)を 70 件の RFP に対して行い，非機能要件ごとに csv 形式にまとめる．
(4) (3)のうちモデル構築に用いるデータのみを用いてランダムフォレストモデルを構築する．
(5) (3)のうち評価に用いるデータに対し，構築したモデルによる自動評価を行う．

表 3-3-1 操作容易性に関する非機能要件とそのキーワード

オペレーション	操作性	操作説明	操作ガイド
	操作マニュアル	操作手順	操作ガイド機能
	システム操作方法	システム運用手順書	オンラインマニュアル
	操作方法	運用手順書	画面遷移
	操作手順書	運用手順	
	システム操作	操作環境	
	操作説明書	入力ミス	

表 3-3-2 稼働品質特性に関する非機能要件とそのキーワード

応答性	平均読み出し遅延	最大スループット	応答速度確保
	転送応答性	安定的レスポンス	メモリ使用率
	応答性	最小レスポンス	システム応答速度
	ハードディスク応答性能	最小応答	データ量
	画面レスポンス	ネットワーク応答	主記憶容量
	オンライン応答時間	VPNスループット	ハードディスク容量
	平均処理応答	平均応答	ネットワーク転送容量
	スループット	応答時間	ターンアラウンド
	レスポンスタイム	秒以内	秒以下
	秒程度	ms以下	MB/s以上
システム性能	性能監視機能	処理性能	アクセス性能向上
	処理速度	アクセス速度	ディスクIO負荷率
	システム性能	CPU性能	性能耐久性
	MPU使用率	CPU負荷率	サーバ性能
	CPU使用率	性能評価	同時接続数
	性能目標値	ネットワーク使用率	性能管理機能
	オンライン処理性能	%以下	演算性能
	性能	CPU	中央演算処理装置
	CPU	SPEC	
負荷バランス	負荷率	負荷分散方式	負荷分散機能
	負荷分散性能	負荷分散構成	負荷計測
	負荷分散	負荷低減	動的負荷分散機能
	負荷許容範囲	負荷運用	運用負荷
	最小サーバ負荷	サーバ負荷	ロードバランス
	ネットワーク負荷	負荷分散装置	ピーク時
	最大負荷時	負荷監視	CPU負荷
	負荷予測	ロードバランシング	
稼働品質	稼動実績	トランザクション数	稼働率
	安定稼働実績	トラフィック量	システム稼働率
	システム正常性	業務稼働率	年間稼働率
	トランザクションデータ	ダウンタイム	連続稼働
	アクセス頻度	トランザクション管理	安定運用
	通信速度	アクセス量	連続運転
	稼動安定性	平均稼働率	システム停止許容
	パフォーマンス管理	サービス稼働率	正常稼働
	365日安定稼働	稼働状況	システム稼働状況
	稼動状況	安定システム	システム稼動情報
	運用スケジュール	稼働実績	ネットワーク管理状況
	稼働期間	%以上	%以下
	24時間365日	サービス稼働率	

表 3-3-3 障害検知特性に関する非機能要件とそのキーワード

応答性	平均読み出し遅延	最大スループット	応答速度確保
	転送応答性	安定的レスポンス	メモリ使用率
	応答性	最小レスポンス	システム応答速度
	ハードディスク応答性能	最小応答	データ量
	画面レスポンス	ネットワーク応答	主記憶容量
	オンライン応答時間	VPNスループット	ハードディスク容量
	平均処理応答	平均応答	ネットワーク転送容量
	スループット	応答時間	ターンアラウンド
	レスポンスタイム	秒以内	秒以下
	秒程度	ms以下	MB/s以上
システム性能	性能監視機能	処理性能	アクセス性能向上
	処理速度	アクセス速度	ディスクIO負荷率
	システム性能	CPU性能	性能耐久性
	MPU使用率	CPU負荷率	サーバ性能
	CPU使用率	性能評価	同時接続数
	性能目標値	ネットワーク使用率	性能管理機能
	オンライン処理性能	%以下	演算性能
	性能	CPU	中央演算処理装置
	CPU	SPEC	
負荷バランス	負荷率	負荷分散方式	負荷分散機能
	負荷分散性能	負荷分散構成	負荷計測
	負荷分散	負荷低減	動的負荷分散機能
	負荷許容範囲	負荷運用	運用負荷
	最小サーバ負荷	サーバ負荷	ロードバランス
	ネットワーク負荷	負荷分散装置	ピーク時
	最大負荷時	負荷監視	CPU負荷
	負荷予測	ロードバランシング	
稼働品質	稼動実績	トランザクション数	稼働率
	安定稼働実績	トラフィック量	システム稼働率
	システム正常性	業務稼働率	年間稼働率
	トランザクションデータ	ダウンタイム	連続稼働
	アクセス頻度	トランザクション管理	安定運用
	通信速度	アクセス量	連続運転
	稼動安定性	平均稼働率	システム停止許容
	パフォーマンス管理	サービス稼働率	正常稼働
	365日安定稼働	稼働状況	システム稼働状況
	稼動状況	安定システム	システム稼動情報
	運用スケジュール	稼働実績	ネットワーク管理状況
	稼働期間	%以上	%以下
	24時間365日	サービス稼働率	

表 3-3-4 システム監視特性に関する非機能要件とそのキーワード

運用管理	稼動状態	運用管理機能	稼働状況
	運用維持管理	運用監視ソフト	運用状況
	運用支援	運用監視	運用管理要件
	運用管理システム	運用マニュアル	正常運用
	運用管理ソフトウェア	運用状態	運用時間
	運用管理ソフト	運用権限	運用管理マニュアル
	運用権限	運用管理ツール	運用保守
	運用管理	運転状況	運用・管理
	運用ルール	運用スケジュール	運用時間帯
	運用計画		
システム管理	ネットワーク監視装置	システム監視用	ファイル監視
	ネットワーク監視機能	システム監視機能	リモート監視
	ネットワーク監視機能	システム稼働管理	サーバ運用監視
	ネットワーク監視	システム監視	サーバ運用管理
	ネットワークシステム管理	システム管理	サーバ管理機能
	ネットワーク管理	システム管理機能	サーバ管理
	システム稼働状況	システム運用管理機能	サーバ監視機能
	システム運用支援機能	システム運用管理	内部機器監視サーバ
	システム監視手順	システム運用管理担当者	システム管理者
	運用監視システム	運用管理サーバ	運用管理障害監視
	システム運用機能	稼動監視	監視ソフトウェア
	運用監視機能		遠隔監視

表 3-3-5 セキュリティ対策特性に関する非機能要件とそのキーワード（1/3）

不正アクセス	漏えい防止	不正アクセス防止	アクセス制御
	不正操作対策機能	不正アクセス対策	ファイアウォール装置
	不正アクセス対策機器	不正侵入防止	ファイアウォール機能
	不正アクセス	侵入監視	ファイアウォールシステム
	侵入対策	アクセス制御機構	ファイアウォール
	ファイルアクセス制御	脆弱性	利用者パスワード
	脆弱性対策	不正アクセス防御対策	不正アクセス対策
	脆弱性情報	不正アクセス検知	
	脆弱性	不正侵入	
アクセス権限	アクセス権制御	アクセス権限	ユーザ数
	アクセス権管理システム	操作権限	ユーザ権限
	アクセス権制御	権限レベル	アカウント管理
	アクセス権管理機能	ユーザ権限	利用権限
	アクセス権	アクセスポリシー	利用者権限登録機能
	権限管理	アクセスコントロール	アクセス管理基盤
	管理権限	権限設定	利用者権限管理
	アクセス制限	管理者権限	利用者数コントロール
	アクセスコントロール機能	制限ユーザ	権限要件
	権限区分	業務権限	権限レベル
セキュリティ管理レベル	情報セキュリティ対策統一	セキュリティ運用	セキュリティ管理計画書
	情報セキュリティ対策基準	セキュリティポリシー管理機	セキュリティ管理機能
	情報セキュリティレベル	セキュリティポリシー	セキュリティ監査
	情報セキュリティ実施手順	情報セキュリティポリシー	セキュリティ対策レベル
	情報セキュリティ実施手順	ローカルセキュリティポリシー	セキュリティレベル
	情報セキュリティ管理シス	セキュリティポリシー	セキュリティリスク分析機能
	情報セキュリティ管理	セキュリティの確保	インシデント管理
	監査業務	セキュリティー確保	セキュリティリスク
	監査情報	セキュリティ管理要領	セキュリティ教育
	監査証跡	セキュリティの維持	セキュリティ診断
	セキュリティに関する研修		
	セキュリティ確保	セキュリティ管理システム	セキュリティ管理

表 3-3-6 セキュリティ対策特性に関する非機能要件とそのキーワード（2/3）

セキュリティ対応	セキュリティワイヤー	セキュリティチェック	仮想化セキュリティサーバ
	セキュリティホール	セキュリティモニタリング	セキュリティ設定
	セキュリティパッチ	セキュリティホール対策	データセキュリティ
	セキュリティシステム	セキュリティパッチ管理	ネットワークセキュリティ
	スパムブロック機能	セキュリティゾーン	ネットワークセキュリティシステム
	サイバー攻撃	セキュリティーシステム	セキュリティ監視
	シンクライアント方式	シャドウパスワード機能	セキュリティ情報
	セキュリティロック	VPN接続機能	セキュリティ対策状況
	シンクライアント方式	検疫ネットワーク	セキュリティ対応
	シンクライアント端末	セキュリティ実施手順	セキュリティ対策
	VPN接続装置	セキュリティソフトウェア	セキュリティ侵害監視
	セキュリティパッチ適用	パッチ管理システム	セキュリティ侵害
	セキュリティ侵害監視	セキュリティ対策ソフト	セキュリティレベル
	セキュリティ対策レベル	セキュリティーレベル	情報セキュリティ対策
	セキュリティテスト	ネットワークセキュリティ	情報セキュリティ
	セキュリティ機能	セキュリティソフト	セキュリティ脅威
認証機能	利用者認証機能		ユーザ認証基盤
	本人認証	ユーザ認証システム	ユーザ認証サービス
	ユーザID	ユーザ認証	システムログイン
	利用者認証	職員認証	不正ユーザ
	操作者認証	利用者認証機構	認証情報
	クライアント認証	ユーザ認証システム	主体認証情報
	生体認証	ログオン認証	静脈認証
	不正監視	認証機能	端末認証
	認証方式	認証管理	ネットワーク認証
	認証基盤	統合認証システム	MAC認証
	認証システム	ネットワーク認証サーバ	サーバ認証
	認証VLAN	ネットワーク認証	認証サーバ
	ログイン認証	認証基盤システム	認証データ
	主体認証方式	認証ソフトウェア	端末認証
	認証方式	認証クライアントソフトウェア	統合認証データサーバ
	SSLクライアント認証機能	認証対象者数	

表 3-3-7 セキュリティ対策特性に関する非機能要件とそのキーワード（3/3）

パスワード管理	ワンタイムパスワード パスワード認証 パスワード管理 パスワード情報 パスワードポリシー パスワードロック 利用者パスワード 旧パスワード パスワードの変更機能 パスワードの有効期限	パスワード入力 パスワード一元管理 初期設定パスワード ログイン制御 ロックアウト処理 ログオン制御 利用者のパスワード パスワード有効日数 強制パスワード変更 パスワード登録	ログインパスワード ユーザ認証パスワード パスワード設定 パスワード変更 認証管理 パスワード管理機能 パスワード入力 初期パスワード パスワードの暗号化
ウイルス対策	不正プログラム ウィルス検出 ウィルス感染 パターンファイル ウィルスパターンファイル ウィルスチェックソフト ウィルスチェック ウィルスパターンファイル 検疫ネットワーク機能 検疫サーバ 検疫対象 検疫ネットワーク 検疫エージェント 検疫機能 検疫セグメント ウィルス除去	ウィルスソフトウェア ウィルスソフト ウィルススキャンエンジン ウィルス検知 ウィルスチェック ウィルス定義ファイル アンチウィルスソフトウェア アンチウィルス機能 ウイルス対策ソフトウェア ウィルス対策システム ウィルス対策ソフト ウィルス対策 ウィルス対策ソフトWebサー ウィルス対策サーバ ウィルス管理システム ウィルス管理ソフト	ウィルス監視 攻撃パターン ウィルス検知報告 ウィルスソフト ウィルスチェックサーバ ウィルス ウィルス定義 ウィルスパターン更新 ウィルスチェックソフトウェア ウィルススキャン機能 コンピュータウイルス対策 ウィルス対策機能 ウィルスチェック対策 ウィルス監視 ウィルス防御
情報漏洩対策	情報漏洩対策機器 情報漏洩対策エージェント 情報漏えい防止 改竄防止 データの漏洩	情報漏洩対策ソフトウェア 情報漏洩対策 情報漏えい 改ざん防止	情報漏洩対策サーバ 情報漏洩 データ漏洩
暗号処理	HDD復号化 復号化 暗号化 ハシュ化 ssh暗号化通信 SSL暗号化通信 暗号化通信 暗号方式	通信暗号化 公開鍵基盤 鍵管理定義書 鍵管理 秘密鍵 公開キー交換 暗号化キー 暗号化通信路	暗号化対策 ファイル暗号ソフトウェア 暗号化通信機能 暗号化強度 鍵管理定義書 暗号通信モード 暗号化ファイルサーバ機能

表 3-3-8 冗長化特性に関する非機能要件とそのキーワード

冗長化	二重化構成 電源二重化 多重化対策 多重化 冗長化機構 冗長構成 冗長化構成 冗長化 仮想化技術	冗長性 冗長化電源 冗長化度合い デュアルディスプレイ構成 冗長電源 バックアップ電源 リダンダント機能 ホットスペア	ディスクアレイシステム ディスクアレイ コールドスタンバイ ホットスペアディスク コールドスタンバイ方式 分散システム 仮想化管理サーバ 仮想化サーバ
RAID構成	ハードウェアRAID-1構成 メモリミラーリング機能 ミラー構成 ミラーリング機能 ミラー化 ミラーリング ハードウェアRAID クラスタリング	RAIDコントローラ RAID0+1構成 RAIDレベル RAIDグループ RAID RAID機能 RAID構成 クラスタシステム	RAIDレベル RAID01構成 RAID5ストレージ RAID5構成 RAID5 クラスタ構成 RAID

表 3-3-9 データバックアップ特性に関する非機能要件とそのキーワード

バックアップ管理	日中バックアップ	フルバックアップ	データバックアップ機能
	定期バックアップ	一括バックアップ	データバックアップ方式
	日次バックアップ	フルバックアップイメージ	バックアップ対象データ
	定期的バックアップ成功率	フルバックアップ運用	バックアップイメージ
	自動バックアップ	フルバックアップ	バックアップデータ
	システムバックアップ	暗号化バックアップ機能	バックアップ方式
	バックアップデータ	各種ログバックアップ	データバックアップ
	データバックアップ	データバックアップスケジュー	データ退避保存
	差分ブロック	バックアップスケジュール	データ保護
	差分データ	バックアップデータ項目	バックアップ管理
	差分バックアップ	バックアップ期間	バックアップ仕様
	世代保管	バックアップ速度	バックアップ運用
	差起動	バックアップ手順	データのバックアップ
	差分データバックアップ	データバックアップ方法	バックアップ対象
	バックアップを自動	定期的にバックアップ	世代管理
バックアップシステム	バックアップ転送	バックアップソフトウェア	専用バックアップサーバ
	バックアップ実行	バックアップソフト	バックアップ専用サーバ
	バックアップ機能	バックアップシステム	バックアップ装置
	バックアップツール	バックアップソフト	バックアップ媒体
	バックアップ統合基盤	バックアップ用	バックアップ電源
	バックアップ基盤	バックアップ機構	バックアップ用サーバ
	バックアップ処理	データバックアップ用機器	バックアップサーバ

表 3-3-10 障害予防特性に関する非機能要件とそのキーワード

障害対策	耐障害性	システム障害対策	障害対応
	障害予防	故障対策	障害時
	障害対応マニュアル	障害未然防止	
	障害対策マニュアル	雷対策	障害調査
	障害切り分けマニュアル		停電対策
	障害対応方法	障害検知ソフトウェア	ホットスワップ
	障害対策	緊急対応機能	ホットプラグ機能
	停電装置		フェイルオーバー機能
	システム障害対応		
障害管理	障害履歴	計画停電対応	障害再発防止
	障害票	障害発生頻度	障害管理方法
	障害手順書	障害事後対策	障害管理
	障害対応結果報告書	障害管理機能	障害対応計画
	障害再発限度	障害防止管理	電源管理ソフトウェア
	障害管理計画	障害発生限度	障害管理サーバ
	障害管理台帳		
停止処理	冗長稼働	計画停止	自動シャットダウン用サーバ機能
	停止期間	停止システム	自動シャットダウン
	停電処理	自動システム停止機能	リモートシャットダウン
	停止稼動	計画停電対応	シャットダウン
	業務停止	縮退稼働	停電状態
	停止処理	保守停止	縮退
	非常時緊急運用	通常稼働	正常稼動
	障害連絡	緊急停止	マルウエア対策
	シャットダウン方式	シャットダウン	システムクローズ処理
	システムクローズ	停止順	正常に停止
	強制停止	自動停止	正常なシステム停止

表 3-3-11 障害復旧特性に関する非機能要件とそのキーワード

リカバリ処理	復旧方法	システム復旧	回復レベル
	復旧対応	システム回復	回復内容
	復旧対策	リカバリプロセス	回復作業
	復旧状況	リカバリ処理	障害回復用
	復旧作業	セッションリカバリ機能	障害回復
	復旧作業対応	データリカバリ処理	回復手順
	障害復旧	データリカバリ	システム障害回復
	復旧対応作業	リカバリーディスク	システム回復作業
	復旧処置	自動リカバリ	データ復旧操作
	復旧機能	回復処理	データ復旧
	フェイルオーバー	リカバリー	迅速な復旧
	復旧措置	復旧時間	復旧期限
	回復時間		
パッチ処理	プログラム修正パッチ	パッチ適用	動作確認手順書
	パッチ適用管理機能	パッチファイル適用状況	動作確認
	パッチプログラム	プログラム修正パッチ適用状	システム起動
	パッチ管理	パッチ対応	パッチ
	修正パッチ	パッチファイル	

⑥ 実証実験の計画・準備

モデル構築の教師信号として，システム発注・開発に 10 年以上携わってきたエキスパート 1 名によって非機能要件の明確さの評価を行った．具体的には，70 件の RFP，27 個の非機能要件の全てについて，非機能要件の明確さを 5 段階で評価した．

⑦ 実証実験の結果評価

モデルの構築と評価のためのデータとして，70 件の RFP のうち無作為に選択した 2/3 を学習データとして用い，残りの 1/3 を評価に用いた．モデルの評価においては，目的変数の出力を 3 段階にした場合と 5 段階にした場合のそれぞれについて，自動評価と人手による評価(手動評価)の結果を比較した．3 段階評価の結果を図 3-3-2 に，5 段階評価の結果を図 3-3-3 に示す．いずれの図においても，レーダーチャートは評価値の平均値を示している．

3 段階評価においては，自動評価と手動評価の一致率は 69.8%であった．±1 差の誤差を許容した±1 差一致率は 97.2%であった．このことから，自動評価は手動評価と比べて大きく外れることは少ないといえる．また，5 段階評価においては，自動評価と手動評価の一致率は 62.0%，±1 差一致率は 85.5%であった．評価 3 段階評価より精度は劣るものの，2 段階増えたにも関わらず一致率の低下は 7.9%と小さかった．これらの結果より，3 段階評価，5 段階評価のいずれにおいても，手動評価が困難な状況における自動評価の有効性が示されたと考える．

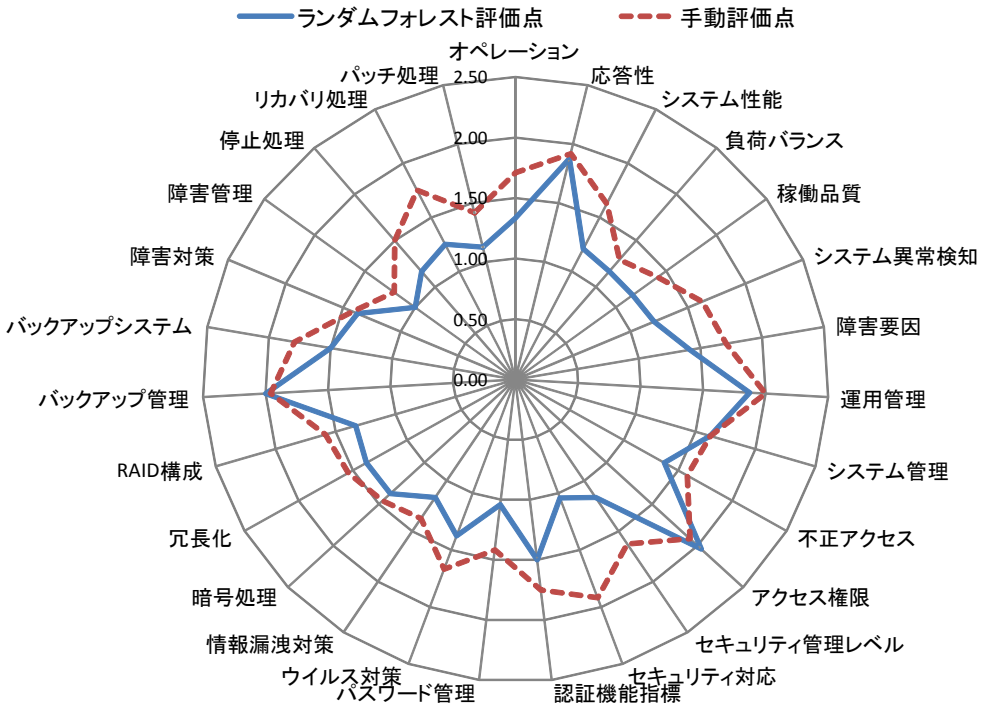


図 3-3-2 非機能要件の 3 段階評価の結果

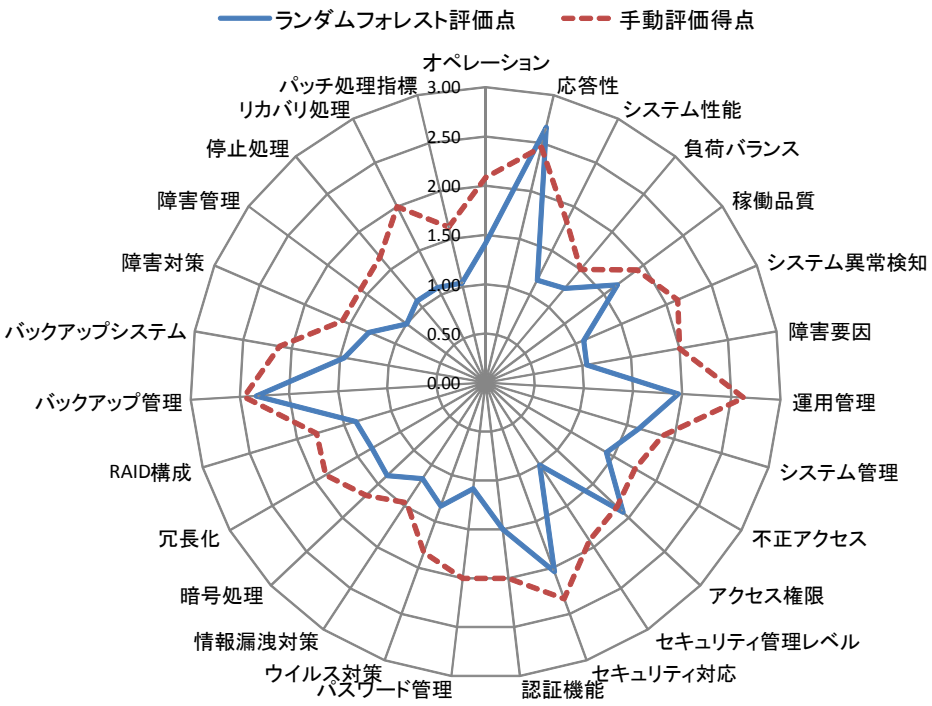


図 3-3-3 非機能要件の 5 段階評価の結果

3.3.3 課題とその対応

非機能要件の自動評価において，2012 年度の本先導的研究支援事業における 55 個の非機能要件は重要語が互いに似通っており，区別して自動評価を行うことが困難であることが分かった．そこで，55 個の非機能要件を整理し直し，大項目（7 項目），中項目（18 項目），小項目（59 項目）に再構成し，小項目である 59 個の非機能要件を評価対象とすることを考えた．

ただし，本研究で題材として用いた 70 件の RFP においては，これら 59 個の中には評価点がほとんどゼロ（つまり，全く記載されていない）ものが含まれていた．例えば，テストに関する項目は，ほとんどの RFP において記載されていなかった．その原因は，70 件の RFP は発注側に情報部門を持たない公共機関や病院のシステムのものであり，テストに関する知識がなかったものと推察される．このような非機能要件については，自動評価モデルを構築できないため，自動評価の有効性を確認することが困難である．そこで，本研究では，59 個のうち，公共機関や病院のシステムにおいて特に重要となる 27 個を対象としてモデル構築と評価を行うこととした．

3.4 研究目標 4「多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価」

3.4.1 当初の想定

(1) 研究内容

多数の開発者が携わるソフトウェア開発プロジェクトにおける開発タスクを「正当なデータ」に基づいて解析・可視化する方式を設計する．タスク計測結果が開発者等により改変されていないことを保証するために開発者，プロジェクト管理者，評価者など役割ごとに計測データに対する権限（閲覧，変更，削除など）を設定する．また，ソフトウェア開発者に対するヒアリングを元に，プロジェクト管理者による改善活動を妨げることなくデータの正当性を確保できる計測・記録手順を第三者評価シナリオとして役割ごとにまとめ，必要な権限を定義する．

開発作業の計測データであるアプリケーションの実行履歴を時系列情報と見なし，前後に実行されたアプリケーションの種類や操作の特徴から開発作業にマッピングする．開発組織で規定されている利用アプリケーションの一覧やヒューリスティクス，部分的なヒアリングなどから部分的なマッピングを行い教師データとする．得られたマッピングから機械学習によりアプリケーション実行履歴と開発作業の対応をモデル化する．機械学習にはランダムフォレスト法など複数の手法を用いて比較する．

設定した権限に基づいて，作業履歴データの取得，保存，サーバ送信を行うプロトタイプを実装する．実装は開発タスク計測システム TaskPit をベースとして開発する．作業履歴と開発作業のマッピングによるモデルの作成には統計解析向けプログラミング言語「R」を用いる．

実証実験では，モデル構築に用いるマッピング情報の量を変化させたときの推定精度に基づいて，モデルの性能を評価する．また，実装したプロトタイプを用いた作業計測におけるデータの正当性を評価する．具体的には，開発者や管理者，評価者ごとにシナリオ通りの利用が可能か，不適切な閲覧，変更がどの程度発生しうるかで評価を行う．

(2) 当初の到達目標と期待される効果

到達目標は,「多人数の開発作業を正当なデータに基づいて解析･可視化する技術の実現」である．具体的には，2012 年度の本先導的研究支援で用いた「開発タスク計測システム TaskPit」に次に示す 2 つの機能拡張･強化を施すことで,多数の開発者が携わるソフトウェア開発プロジェクトにおける開発作業（タスク）を「正当なデータ」に基づいて解析・可視化する技術を開発する．

① 計測結果の非改ざん性保証：クライアント・サーバ型での計測において，タスク計測結果が開発者等により改変されていないことを保証し，第三者評価に用いるデータとしてその正当性を確保する．

② アプリケーション実行履歴と開発作業の自動マッピング：タスク計測における「アプリケーション実行履歴と開発作業のマッピング」を自動化する．多くのアプリケーション（例えば Microsoft Word）は複数の開発作業（仕様策定，詳細設計等）で用いられる．これまで，両者間の対応関係を正確に把握するため，開発者へのヒアリングなどが行われてきた．マッピングを自動化することで，多数の開発者が携わるプロジェクトでもタ

スク計測が可能，あるいは，容易になる．

データの正当性保証と解析コストの大幅削減は,「ソフトウェア品質の第三者評価」の普及と発展に大きく貢献する．

3.4.2 研究プロセスと成果

(1) 研究プロセス

まず，開発者，プロジェクト管理者，評価者など役割ごとに権限（閲覧，変更，削除など）を設定することで，計測結果の非改ざん性を保証する．具体的には，役割ごとにシステムの利用シナリオを策定し，必要な権限を明らかにする．次に，第三者評価の正当性を確保しながらプロジェクト管理者による改善活動の妨げにならない柔軟な方式を設計する．最後に，設計に基づいてプロトタイプを実装し，シナリオ通りの利用が可能か，また，不適切な閲覧，変更が発生する可能性がないか評価し，評価結果に基づいたシステムの改善を行う．

次に,アプリケーションの実行履歴を時系列情報と見なし,前後に実行されたアプリケーションの種類や操作の特徴から開発作業への自動マッピングを実現する．具体的には，対応する開発作業を一意に特定できるアプリケーション実行履歴を明らかにする．開発組織で規定されている利用アプリケーションの一覧やヒューリスティクス，部分的なヒアリングなどからマッピングする.次に,得られたマッピングから機械学習によりアプリケーション実行履歴と開発作業の対応をモデル化する方法を開発する．最後にモデルに基づいてすべてのアプリケーション実行履歴に対応する開発作業をマッピングする．モデル構築前に行うヒアリングの量を少なくするほど，第三者評価のコストが小さくなる．ヒアリングの量を変化させた時の精度を比較し，本機能によってどれだけモデル構築前のヒアリングの量を削減できるか評価する．具体的な作業項目は次の通り．

開発者へのヒアリング
シナリオの策定
非改ざん性保証方式の設計
実行履歴と開発作業のマッピング・モデル化
プロトタイプの概要設計
プロトタイプの詳細設計
プロトタイプの実装
実証実験の計画・準備
実証実験の結果評価

(2) 具体的な研究成果（結果）

① 開発者ヒアリングと既存研究調査に基づいたシナリオ策定

開発作業に基づいた第三者評価において，ソフトウェア開発組織に属するプロジェクト管理者と開発者，および第三者評価組織に属する評価者それぞれが作業計測システムに求める要件を明らかにするために，役割ごとにシステムの利用シナリオを策定した．シナリオを策定するために開発者へのヒアリングおよび既存研究について調査した．

1)開発者ヒアリング

開発組織における利用シナリオを策定するために，パッケージソフトウェアの開発 を行っている中規模の開発企業に所属するベテランの開発者１名，および，同社のプロジェクト管理者１名にヒアリングを行った．ヒアリングの結果，開発者は開発作業履歴に対して閲覧・編集の権限は不要で，管理者に閲覧権限が必要であることがわかった．また，アプリケーション実行履歴の収集頻度について，システムによって発生する開発者の端末への負荷と，実行履歴と開発作業のマッピングに必要な作業量のバランスから，5 分から 10 分程度の収集頻度が適していることが分かった．

2)既存研究調査

第三者評価に関する既存技術および，作業推定の手法について調査した．調査の結果，第三者評価のためにデータの原本性（作成された時点から変化がないこと）を保証するための仕組みとしてハッシュとタイムスタンプを用いた方法がよく用いられていることがわかった．代表的な利用例としては，電子カルテや電子契約，電子政府などが挙げられる[IPA5]．

作業推定の手法として，アクティブなウインドウタイトルとアプリケーション名，ウインドウ内に表示されたコンテンツから推定する手法[Granitzer]や,操作履歴に基づいてウインドウ間の関連性と各ウインドウの重要度を算出する手法[Bernstein]が提案されていることがわかった．

3)シナリオ策定

開発者ヒアリングと既存研究調査の結果を元に，評価者と開発組織それぞれの利用シナリオを策定した．

・評価者の利用シナリオ

　評価対象となる開発組織に，システムを用いて原本性が保証されたアプリケーション実行履歴を計測させることで改ざんのない正確なデータに基づいた評価をしていることを示したい．

・開発組織の利用シナリオ

　第三者評価によって自組織の開発プロセスや製品の品質を示したい一方で，評価の元になるアプリケーション実行履歴の収集や分析にできるだけ負担をかけたくない．特に，複数開発者の計測においても管理者の負荷が大きくならないようにしたい．

② 非改ざん性保証方式の設計

評価者の利用シナリオに基づいて，計測したアプリケーション実行履歴に対する改ざんを検出する方法を開発した．電子カルテにおける非改ざん性保証システムを参考に，ハッシュ値とタイムスタンプを用いて改ざんを検出することで，アプリケーション実行履歴の信頼性を保証する．

図 3-4-1 に改ざん検出方法の概要を示す．開発した改ざん検出方法はサーバ・クライアントモデルのシステムから構成される．ソフトウェア開発組織内では，開発者が利用した

アプリケーションの履歴をクライアントシステムが計測する際に，履歴情報からハッシュ値を計算し，定期的に第三者評価組織に送信する．評価組織はサーバで受信したハッシュ値にタイムスタンプを付与して保存する．評価を実施する際には，開発組織が計測した履歴を受け取り，ハッシュ値を再計算した上で，これまでに受信したハッシュ値と比較することで改ざんの有無を検出する．開発した改ざん検出方法は，作業履歴から計算されるハッシュ値のみを送信するため，作業履歴自体を送る場合と比べて以下のメリットがある．

・データ量が小さく，既存のネットワークを圧迫することなく導入が可能．また，多数の開発組織を対象に評価業務を行う場合であっても，評価組織のデータ管理が容易．
・開発者の作業内容やプライバシー情報，業務機密を含まないため，セキュリティ上のリスクが小さい．

また，実際とは異なるアプリケーション実行履歴を送信するようにクライアントの実行ファイルを改ざんされる可能性も考慮した．クライアントは自身の実行ファイルからハッシュ値を計算し，サーバに送信することで改ざんのないクライアントであることを確認する．この時，クライアントのハッシュを通信経路中で盗聴されないよう，ワンタイムパスワードを用いて暗号化することでクライアントの改ざんや通信内容の書き換えを防止する設計を実現した．

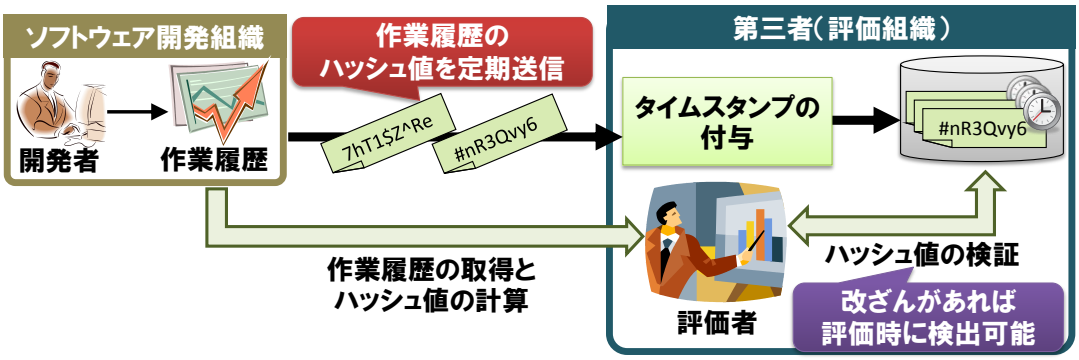


図 3-4-1 ハッシュとタイムスタンプによる改ざん検出

③ 実行履歴と開発作業のマッピング・モデル化

クライアントが計測するアプリケーション実行履歴と開発者の開発作業を機械学習によって自動的にマッピングする方法を開発した．開発した方法は機械学習アルゴリズムの 1 つであるランダムフォレストを用いており，ノイズや外れ値を含む作業履歴を対象とした場合でもマッピングの精度が低下しにくい．計測する履歴の例を表 3-4-1 に示す．アプリケーションの種類とウィンドウタイトル，アプリケーションに対する操作量（右クリック数，左クリック数，打鍵数）を計測する．また，自動マッピングの精度を高めるために，マッピングの対象となるアプリケーション履歴の前後にある履歴の特徴を利用する手法を考案した．指定した個数の前後アプリケーション履歴の内，1 番目に打鍵数が多いアプリ，

2 番目に打鍵数が多いアプリ，3 番目に打鍵数が多いアプリの名前を特徴として追加する．同様に左クリックと右クリックについても特徴として追加する．この手法は，利用されるアプリケーションの内，どれが主たる作業か表すデータを学習に加えた方法といえる．

表 3-4-1 アプリケーション実行履歴

アプリ	ウィンドウ タイトル	開始時間	終了時間	左 クリック	右 クリック	打鍵数
プログラミング	Visual Studio	13:25:35	13:28:48	11	2	98
文書作成	Microsoft Word	13:28:48	13:29:54	7	0	0
プログラミング	Visual Studio	13:29:54	13:33:48	23	3	120
デスクトップ	explorer.exe	13:33:48	13:33:56	1	0	0
プログラミング	Visual Studio	13:33:56	13:34:48	4	0	75
データ分析	Microsoft Excel	13:34:48	13:36:41	24	0	64
デスクトップ	explorer.exe	13:36:41	13:36:59	2	0	0

④ プロトタイプの設計・実装

開発した非改ざん性保証方式を実現するサーバ・クライアントシステムを設計，実装した．システムは産業界への導入の容易性を考慮して，サーバ・クライアント共に Windows 上で動作する.Net フレームワークアプリケーションとして作成した.クライアントは 2012 年度の本先導的研究支援で用いた「開発タスク計測システム TaskPit」をベースとし，アプリケーション実行履歴のハッシュ値計算，クライアントのハッシュ計算，およびサーバとの通信機能を拡張した．また，インストーラとバッチによる複数台への自動インストールが可能である．サーバはクライアントの認証とハッシュ値の受信とタイムスタンプの付与，および改ざんの検出を行う．複数のクライアントを同時に処理できるように，クライアントごとにスレッドを生成して並列に処理する設計とした．

図 3-4-2 にクライアントの起動からサーバによる接続認証までの処理の流れを示す．クライアントは起動直後にサーバからワンタイムパスワードを受け取り，クライアントの実行ファイルから計算したハッシュ値を暗号化した上でサーバに送信する．サーバは受信したハッシュ値を元にクライアントの実行フィアルが改ざんされていないか確認し，改ざんされている場合，クライアントの起動を許可しない．

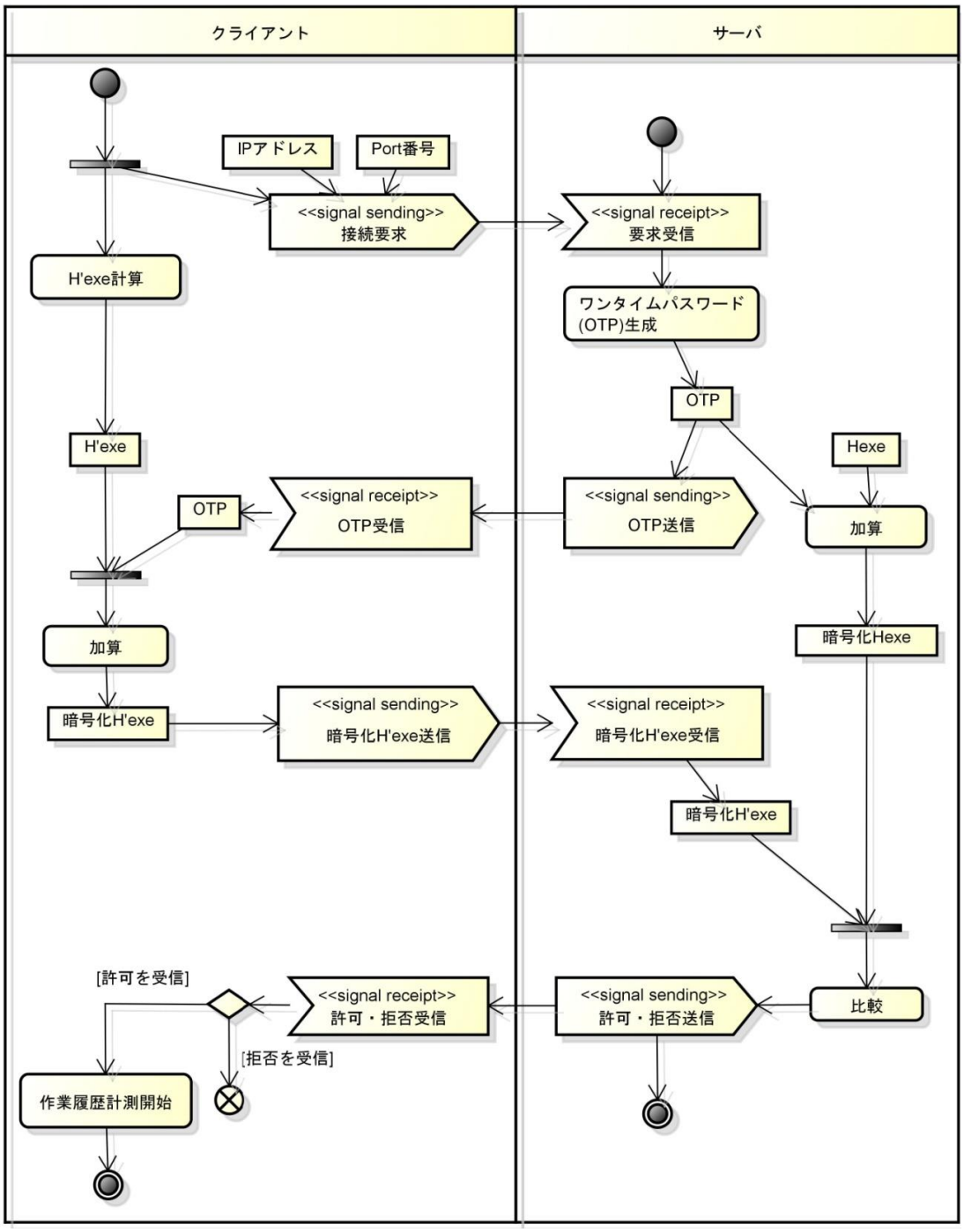


図 3-4-2 クライアントの認証

図 3-4-3 に改ざんのないクライアントから接続要求が来た時のサーバのスクリーンショットを，図 3-4-4 に改ざんされたクライアントから接続要求が来た時のスクリーンショットを示す．改ざんのないクライアントでは，表示メッセージの 3 行目でクライアントの認証が成功しているのに対して，改ざんのあるクライアントでは認証に失敗し，接続が切断されることが確認できた．

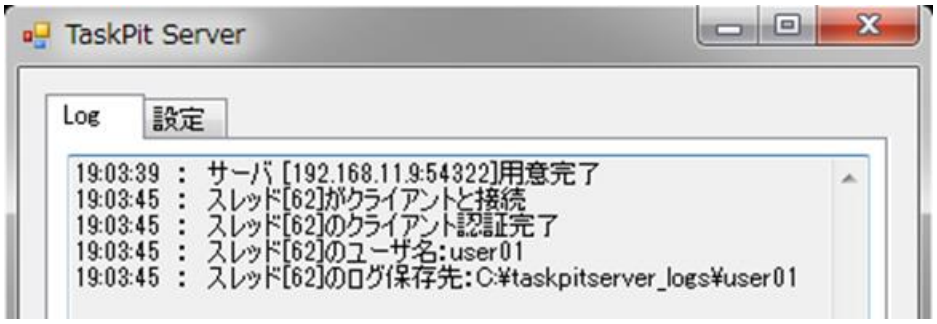


図 3-4-3 クライアントの認証（改ざんなし）

図 3-4-4 クライアントの認証（改ざんあり）

図 3-4-5 に実行履歴のハッシュ送信から実行履歴に対する改ざん検出までの処理の流れを示す．起動を許可されたクライアントは，端末上で開発者が利用するアプリケーション名と各アプリに対する操作量を記録する．アクティブな（最前面に表示された）アプリケーションが変化するたびに履歴を生成し，履歴の文字列からハッシュ値を計算する．負荷を軽減するために一定期間の履歴とハッシュをまとめてサーバに送信するような設計とした．サーバは受信したハッシュ値にタイムスタンプを付与した上で保存する．

第三者機関による評価を実施する際に，開発組織からアプリケーションの実行履歴を受け取り，ハッシュ値を計算する．計算したハッシュと受信したハッシュが一致しなければ，サーバはアプリケーション実行履歴が改ざんされたと判断する．ハッシュをアプリケーションが切り替わるごとに計算，送信しているため，改ざんがあった行を特定することが可能である．改ざんが検出されなければ改ざんがされていないと判断し，アプリケーション実行履歴に基づいた第三者評価を行う．

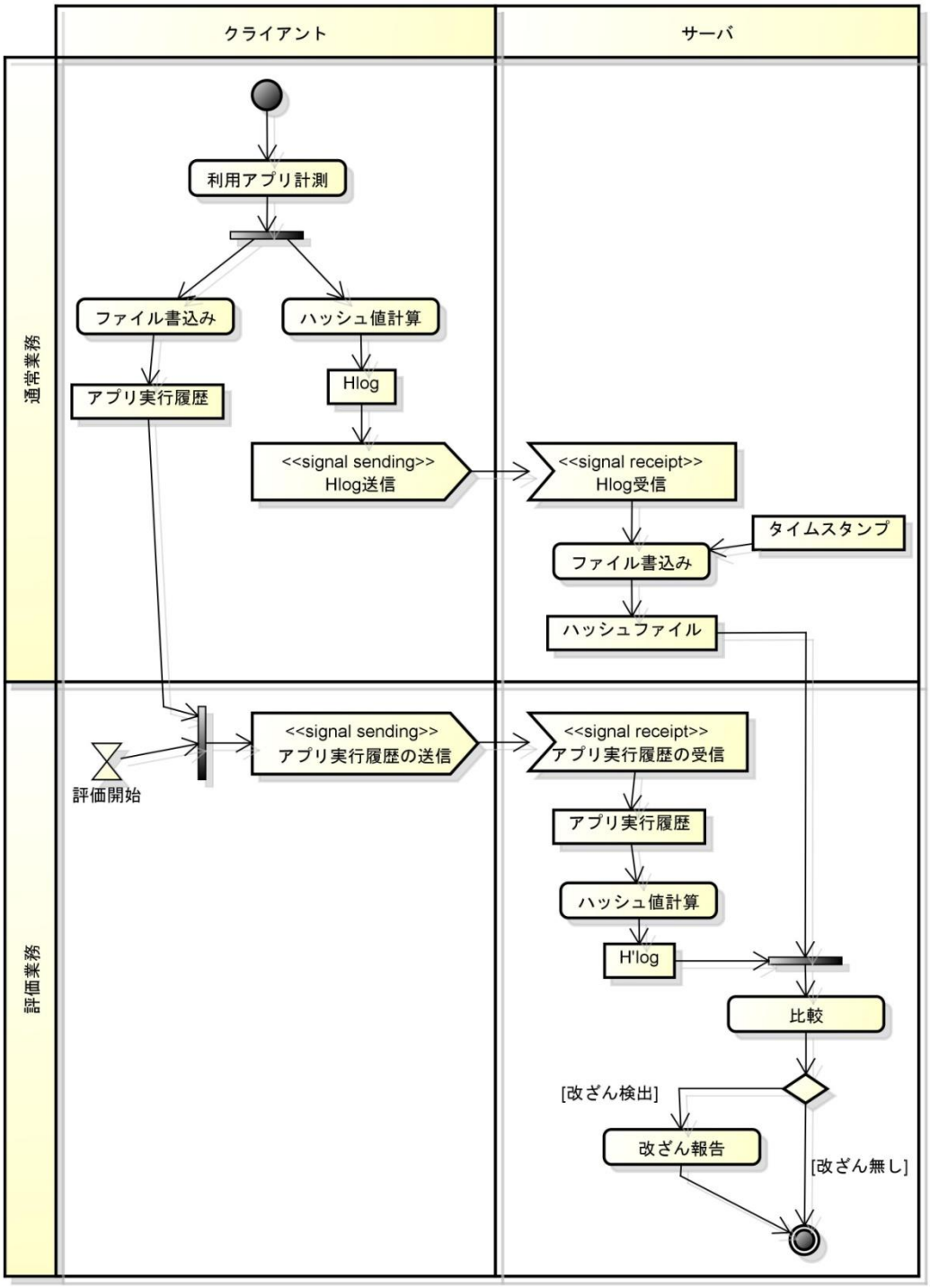


図 3-4-5 ハッシュの送信と改ざん検出

図 3-4-6 にクライアントから受信したアプリケーション実行履歴のハッシュを示す．クライアントの負荷を抑えるために，一定期間中に生成された複数のアプリケーション実行履歴が同時刻にまとめて送信されている．図 3-4-7 にクライアントで計測した履歴を改ざ

んし，サーバで改ざん検出した結果を示す．履歴の 14 行目に記録されていた Microsoft Word による作業を eclipse による作業に書き換えた結果，改ざんを検出し，改ざんの箇所も正しく特定された．

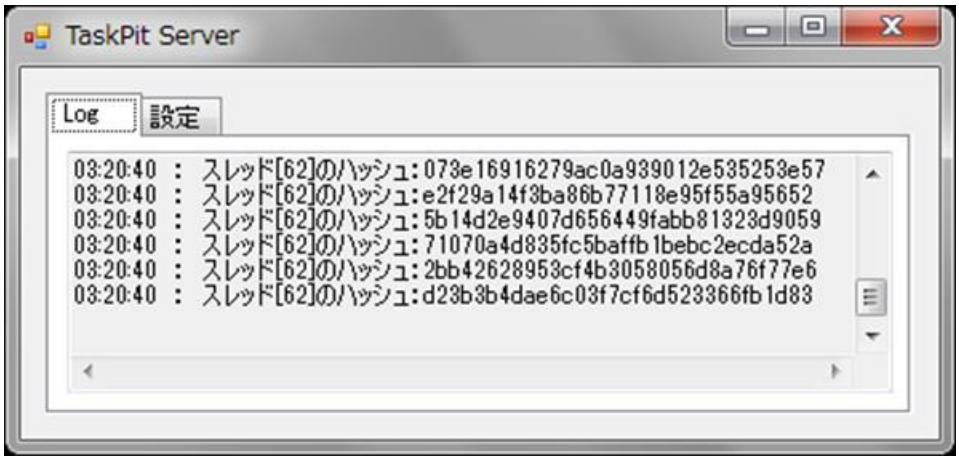


図 3-4-6 ハッシュの受信

図 3-4-7 アプリケーション実行履歴の改ざん検出

⑤ 実証実験

1)複数台同時計測

学内のLANに接続された端末にサーバとクライアント6台を導入し,同時接続試験を行った．クライアントは，作成したインストーラを用いて正常に導入が完了し，サーバへの同時接続も成功した.5 日間の接続試験の結果,クライアントの記録しているアプリケーション作業履歴に対応するハッシュをサーバが受信していることを確認した．また，サーバを LAN 外で起動した場合でも正常に接続し，ハッシュの送受信ができることを確認した．

2)実行履歴と開発作業のマッピング精度

実務者のアプリケーション実行履歴から開発作業へのマッピングを行い，その精度を評価した．開発組織で開発プロジェクトに従事する開発者 3 名の端末にクライアントを導入

し，34 営業日のアプリケーション実行履歴（6185 件）を計測した．この開発者は計測期間中に以下の 4 種類の作業を行っていた．

・実装
・テスト
・休憩
・その他

開発したシステムによるマッピングの精度を確認するために，計測されたアプリケーション履歴のウィンドウタイトルの精査および，開発組織の管理者へのインタビューから開発作業を特定した．また，予測に用いる前後の履歴の数を 0（前後の履歴を用いない）から前後 10 個まで変化させ，精度の変化を評価した．

図 3-4-8 に評価結果を示す．前後のアプリ履歴を用いた時の精度は，用いなかった時に比べ大幅に向上した．履歴数を 0 から 1 にすると，開発者 A では，69.2%から 86.1%と 16.9%精度が向上している．また開発者 B では，72.2%から 85.8%と 13.6%，開発者 C では 87.8%から 95.1%と 7.3%精度が向上した．さらに履歴数を 10 まで増やすと精度がさらに向上し，平均で 97%と高い精度でマッピングできた．以上の結果から，アプリ履歴について前後の履歴に関する情報を加えることでマッピング精度が向上するといえる．

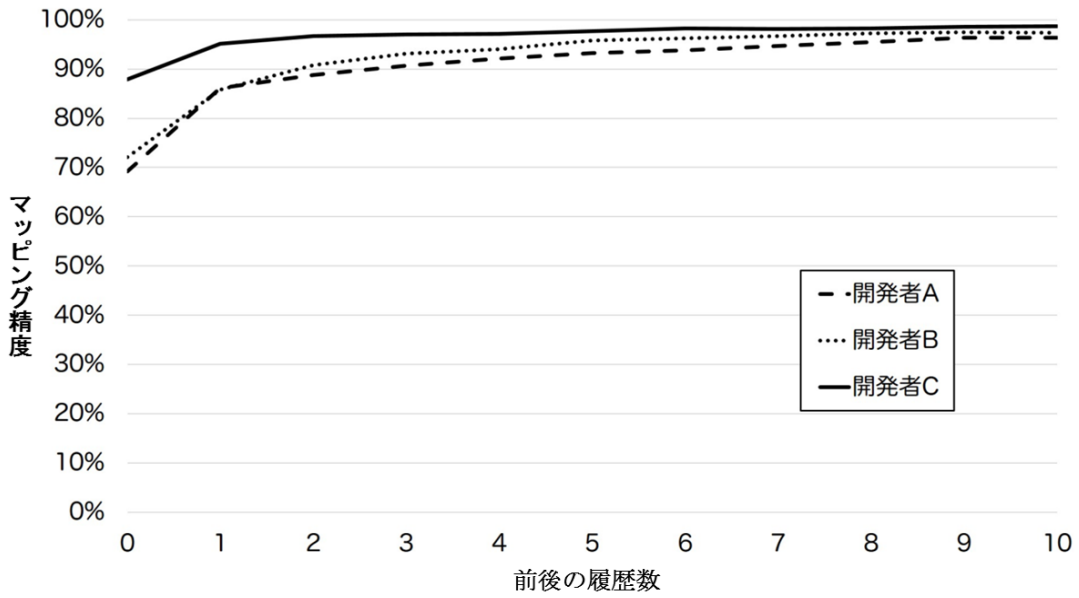


図 3-4-8 アプリケーション実行履歴の数とマッピング精度

3.4.3 課題とその対応

外部委託やアウトソーシングなどで開発組織が海外にある場合，インターネットを介した通信が安定せず，速度低下やパケットロスを起こす可能性がある．作成したシステムを通信品質が高くないインターネットを介して利用する場合を考慮して，バッファリング処

理やデータの再送処理など，設計を一部変更することで対応した．

開発したシステムを開発現場へ適用する場合，計測対象となる多数の開発者の端末に導入し，開発組織の方針や規則に則した環境設定を支援する必要がある．そこで，システムを自動でインストールするためのインストーラと，インストールおよび環境設定を複数の端末に一括して自動で行うためのバッチプログラムを作成した．また，導入手順を記載した手順書を作成した．

3.5 研究目標 5「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」

3.5.1 当初の想定

(1) 研究内容

低品質モジュールが，プログラム内にどのように分布し，開発作業の進行に伴ってどのように変化していくのか，その構造的・時間的特性を解析・可視化する方式を設計する．具体的には，2012 年度の本先導的研究支援事業で開発した「低品質モジュール予測モデル」を改良し，改良モデルに基づく解析・可視化システムを実現する．まず，低品質モジュールを定義する．低品質モジュールには，単に「欠陥が混入しているモジュール」だけでなく，既存研究に基づき，保守性や可読性が低いモジュールも含める．次に低品質モジュールメトリクスを選定する．更に，「低品質となるモジュールに共通する主な属性（メトリクス値）」の標準値域を，多数の大規模 OSS プロジェクトのデータから算出し，低品質モジュールの予測（判別）に活用する．同様に，モジュール間の構造的関係（データや制御の流れから見たモジュールの接続関係等）を示す情報も予測（判別）に活用する．低品質モジュールに共通する属性（メトリクス値）は，たとえ同一プロジェクト内であっても，終始不変であるとは限らない．予測（判別）対象である低品質モジュールそのものの変性（コンセプトドリフト）の検知，予測モデルの再構築，モデル適用範囲・条件の明確化などにより，予測モデルの内的妥当性の強化も行う．

プロトタイプ実装では，構築した予測モデルによって低品質と判断されたモジュールの可視化を実現する．実証実験では，大規模 OSS プロジェクト 20〜30 件のデータ（ソースコード版管理システム，課題管理システムに蓄積されているデータ）を用いて，提案方式の妥当性や精度などを評価する．特に，モジュールが低品質と判断された根拠を，プロトタイプにより可視化された情報に基づき理解することができるかどうかを評価する．

(2) 当初の到達目標と期待される効果

到達目標は，「低品質モジュールの構造的・時間的特性を解析・可視化する技術の実現」である．具体的には，2012 年度の本先導的研究支援事業で開発した「低品質モジュール予測モデル」を次に示す 3 つの観点で改良し，改良モデルに基づく解析・可視化システムを実現することで，低品質モジュールが，プログラム内にどのように分布し，開発作業の進行に伴ってどのように変化していくのか，その構造的・時間的特性を解析・可視化する技術を開発する．なお，低品質モジュールには，単に「欠陥が混入しているモジュール」だけでなく，保守性や可読性が低いモジュールも含まれるものとする．

① 外的妥当性（External Validity）の向上：「低品質となるモジュールに共通する主な属

性（メトリクス値)」の標準値域を，多数の大規模オープンソースソフトウェア（OSS）プロジェクトのデータから算出し，低品質モジュールの予測（判別）に活用する．(2012年度の本先導的研究支援事業では，同一ソフトウェアの異なるバージョン間であれば，開発初期であっても，開発終了時と同程度の精度で予測が可能なこと，すなわち，モデルの内的妥当性（Internal Validity）のみを実験により示した.）

② モジュール構造情報の活用：モジュール間の構造的関係（データや制御の流れから見た接続関係等）を示す情報を予測（判別）に活用する．

③ コンセプトドリフトへの対応：低品質モジュールに共通する属性（メトリクス値）は，たとえ同一プロジェクト内であっても，終始不変であるとは限らない．予測（判別）対象である低品質モジュールそのものの変性（コンセプトドリフト）の検知，予測モデルの再構築，モデル適用範囲・条件の明確化などにより，予測モデルの内的妥当性を強化する．(2012年度の本先導的研究支援事業では,構築する予測モデルは１つのみであり，予測時点も3つのみであった.）

ソフトウェア開発プロジェクトにおいて低品質モジュールがどのように生成され，対処されてきたのかを詳細に見ることは，品質そのものの評価はもちろんのこと，開発プロセスの妥当性の評価にもつながるものであり,「ソフトウェア品質の第三者評価」の普及と発展に大きく資する．

3.5.2 研究プロセスと成果

(1) 研究プロセス

2012年度の本先導的研究支援事業で開発した「低品質モジュール予測モデル」を次の手順で改良し，同事業で開発した「低品質モジュール可視化システムのプロトタイプ」を改良モデルに基づき機能拡張することで，目標とする解析・可視化システムを実現する．

まず，既存研究[Buse][Marco]等に基づき，単に欠陥混入の有無だけでなく，保守性や可読性の観点から，低品質モジュールを定義する．次に，低品質モジュールの定義，および，および，2012年度の本先導的研究支援事業で得られた知見に基づき，低品質モジュールの予測（判別）に用いるメトリクスを選定する．特に,「モジュール間の構造的関係」を表し，かつ，ソフトウェア静的解析ツール「Understand 2.5」等で自動計測可能なメトリクスを検討対象として重視する．なお，低品質モジュールの予測（判別）に有効であっても，計測コストが大きい，適用範囲が狭い，といったメトリクスに対しては，当該メトリクスと相関が高く，計測コストがより小さい，あるいは，適用範囲の広いメトリクス（代替メトリクス）を選定する．これらにより，より有用性の高い解析・可視化の実現を目指す．最後に，20～30の大規模OSSプロジェクトのデータに基づいて，低品質モジュールメトリクスそれぞれの標準値域を算出し，低品質モジュールの予測（判定）に組み入れる．具体的な作業項目は次の通り．

シナリオの策定
低品質モジュールの定義
メトリクスの選定
標準値域の導入
プロトタイプの概要設計

プロトタイプの詳細設計
プロトタイプの実装
実証実験の計画・準備
実証実験の結果評価

(2) 具体的な研究成果（結果）

① ソフトウェアモジュール品質の第三者評価に向けた標準値域の開発

欠陥管理システムが保存されている欠陥票，および，バージョン管理システムに保存されているソースコードに基づき低品質モジュールを特定し，低品質モジュールの特徴量から標準値域を開発した．具体的には，モジュールをファイルレベルで解析し，欠陥を混入するモジュールを低品質モジュールとする．一次解析データとして，欠陥票，および，欠陥が混入していたソースコードとの紐付けを行った．これにより，各スナップショットにおける低品質モジュールの特定を行った．紐付けするためのアプローチは，Śliwerski らが提案した SZZ アルゴリズム[Śliwerski]を利用した．SZZ アルゴリズムは，版管理システムに記録されたコミットログと欠陥追跡システムに記録されたテキスト情報を用いる.SZZ アルゴリズムの概略は次の通り．

手順 1: コミットログメッセージに各欠陥票に付された ID と「ソースコードの修正」を示すキーワード（例えば，fixed や closed など）が記載されているか否かを確認する．含まれていない場合は，以降の手順をスキップし，次のコミットログメッセージを分析する．

手順 2: 手順 1 のコミットログメッセージに記載されていたバグ ID の修正進捗状況をバグ管理システムで確認し，既に修正済みか否かを確認する．修正が完了している場合，当該コミットログに対応するコミットが発生した時点で，その欠陥が修正されたとする．修正が完了していない場合は，以降の手順をスキップし，次のコミットログメッセージを分析する．

手順 3: 手順 2 で確認された「欠陥修正時」の直前のコミットにおいて欠陥が混入したとする．

本手法から低品質モジュールと欠陥をモジュールと，欠陥を含まないモジュールに区別し，それぞれのモジュール群で計測したメトリクス値を用いて標準値域を算出する．図 3-5-1 に示すように，左側に欠陥を含むモジュールの属性値（メトリクス値）の分布を，右側に欠陥を含まないモジュールの属性値（メトリクス値）の分布，をそれぞれ人口ピラミッド形式で示す．対象となるメトリクスは，ソフトウェア静的解析ツール「Understand 2.5」で自動計測可能な 13 個のメトリクスである（表 3-5-1 参照）．

表 3-5-1 標準値域を示すメトリクス（可視化対象メトリクス）

	名称	定義
プロダクトメトリクス	AvgCyclomatic (AC)	サイクロマティック複雑度の平均
	CountLineCodeExe (CLCE)	実行可能コード行数
	CountDeclClass (CDC)	クラス数
	CountDeclFunction (CDF)	関数の数
	CountLineBlank (CLB)	空白行数
	CountLineCode (CLC)	コード行数
	CountLineCodeDecl (CLCD)	宣言コード行数
	CountStmtExe (CSE)	実行可能ステートメント行数
	CountStmtDecl (CTD)	宣言ステートメント行数
	CountLineComment (CLCM)	コメント行数
プロセスメトリクス	NumberOfRevisions (NR)	変更行数
	AddedLines (AL)	追加行数
	RemovedLines (RL)	削除行数

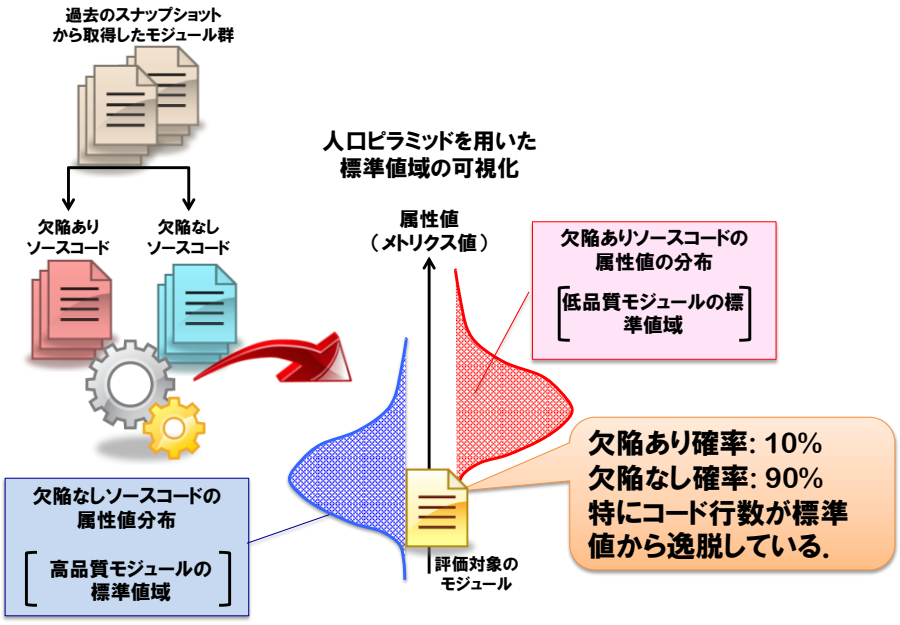


図 3-5-1 標準値域を用いた低品質モジュール評価

ここで対象とするのは，ソフトウェア開発者が作成したソースコードの品質（欠陥の有無）の第三者評価を求める者である．ソースコードを入力することで，欠陥混入確率，および，表 3-5-1 に示すメトリクスの中で特に標準値域から逸脱しているメトリクスを提示する．なお，提示する標準値域は次の 2 つである．

- プロジェクト内標準値域:プロジェクトの特徴を考慮するために，欠陥混入していた/ないソースコードのメトリクス値を計測し，開発した標準値域
- 一般標準値域:一般的なソフトウェアと比較するために，数十の OSS プロジェクトの開発データから欠陥混入していた/ないソースコードのメトリクス値を計測し，開発した標準値域

プロジェクト内標準値域は，プロジェクト特有の傾向を示す値域を作成するため，ソフトウェア品質の内的妥当性評価の向上に貢献する．一般標準値域は，ある特定のプロジェクトのみならず，社会で運用されている複数のソフトウェアのモジュールから標準値域を作成しているため，ソフトウェア品質の外的妥当性評価の向上に貢献する．ここでは，10 件の大規模 OSS (Eclipse, hibernate, openms, sipxecs, spring, wine, GIMP, X.org, samba, digikam) プロジェクトを対象にプロジェクト内標準値域と一般標準値域を作成した．各プロジェクトでは，欠陥追跡システムとして Bugzilla か JIRA が使用されており，バージョン管理システムには Github が使用されている．それぞれ 2014 年 4 月 1 日の時点で最新のソースコードを Github から取得し,取得時点から 3 ヶ月以内に発見されたソースコードを低品質モジュールとした．

評価対象ソースコードの標準値域からの逸脱度合いを示すために，メトリクス値を正規化することとした，具体的には，プロジェクト間で単純に比較するために，各標準値域から各メトリクスを抽出し，欠陥あり，欠陥なしのモジュール群において，それぞれメトリクス偏差値を求める．メトリクス偏差値の算出式は次のとおり．

$$メトリクス偏差値 = \frac{メトリクス値 - 各低品質モジュール群の平均値}{低品質モジュール群の標準偏差}$$

表 3-5-1 に示した 13 個の属性値（メトリクス値）について標準値域を算出し，可視化した例を図 3-5-2 に示す．同図では，6 つのプロジェクト Eclipse, hibernate, openms, sipxecs, spring, wine それぞれにおけるプロジェクト内標準値域，および，それらに基づく一般標準値域を示している．プロジェクト内標準値域は，プロジェクトの特徴，規模に応じてメトリクス値の分散が大きい場合と小さい場合があるが，一般標準値域において極端な分散の広がりを抑制していることがわかった．従って，一般標準値域は，社会に流通するソフトウェアの品質を示す標準値域を実現していることが分かる．

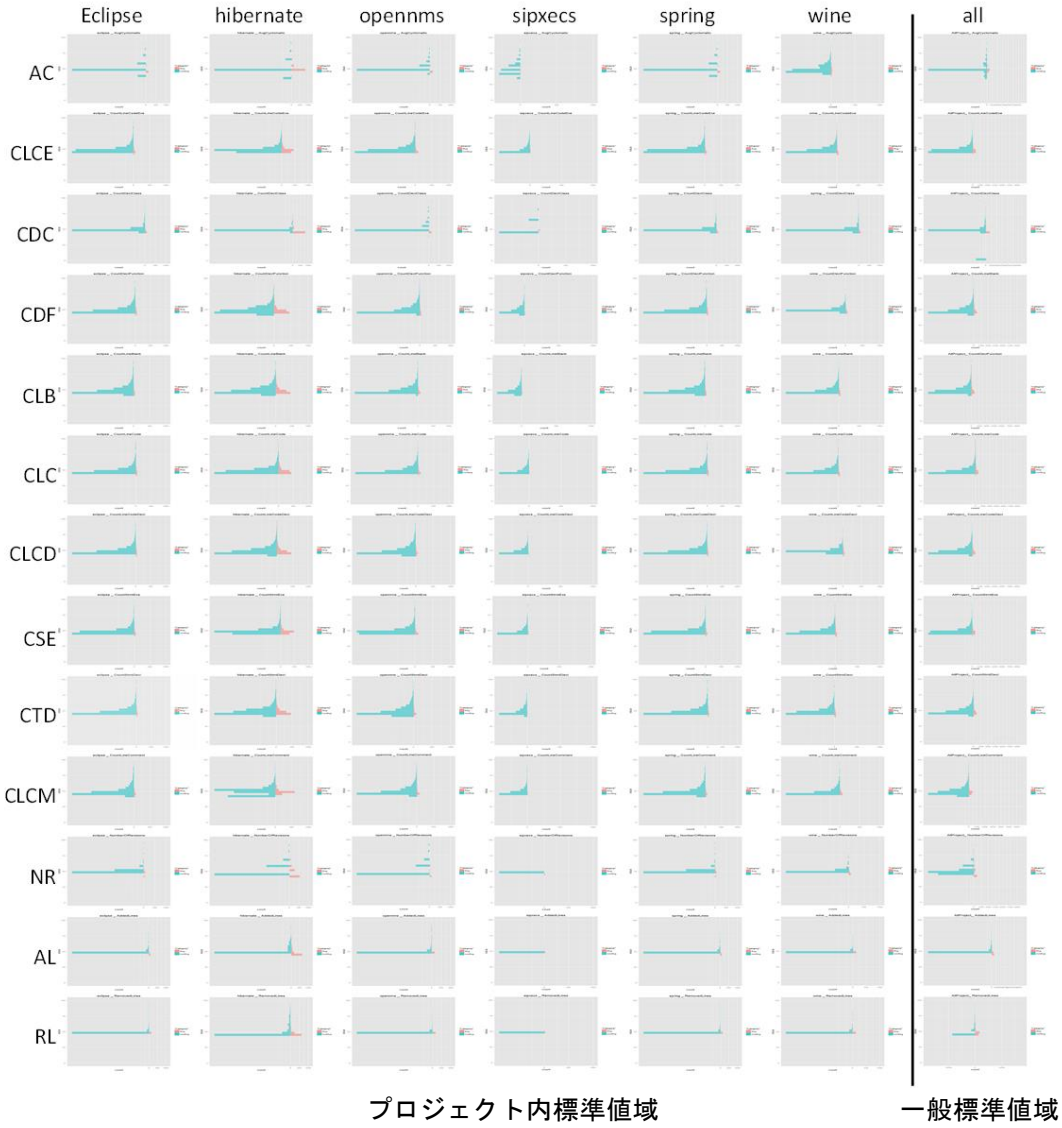


図 3-5-2 標準値域例

② 標準値域から算出した偏差値メトリクスを用いた低品質モジュールの予測

従来研究では，低品質モジュール予測モデルを構築するために，欠陥の有無を考慮せずにモジュールから計測されるメトリクスを使用していたが，提案する標準値域により，各メトリクスから低品質モジュールの確率を計測することができるようになった．当該確率は，一般的にモジュールから計測されるメトリクスに代わる，新たなメトリクスの計測方法となり得る．欠陥モジュール判別予測モデルの構築では，十分にモジュールが残されていない場合，他プロジェクトのモジュールからメトリクス値を計測，そして，欠陥が混入していたモジュールを特定し，予測モデルを構築していた．しかし，プロジェクトごとにソフトウェアの規模，および，特徴量が異なることから予測精度は決して高くない．一方で,構築した一般標準値域は,たとえソフトウェアの規模や特徴が異なっていたとしても，正規化されているためプロジェクトごとに異なるソフトウェア規模に左右されないという

利点があるため，異なるプロジェクトのモジュールから構築される予測モデルの精度向上が期待される．そこで，当初は，内的妥当性を強化するために予定していた，モジュール構造の解析は行わず，提案した標準値域の有用性を評価するために標準値域を用いた新たなメトリクス（偏差値メトリクス）提案し，予測モデルの開発を行った．本アプローチは，目的を変えるものではなく，他プロジェクトから見た一般標準値域との逸脱度合いを示し，第三者評価を求める産業界による導入を促すものである．本予測モデルは品質そのものの評価はもちろんのこと，外的妥当性の評価にもつながるものであり，「ソフトウェア品質の第三者評価」の普及と発展に大きく資する．

2012 年度の本先導的研究支援事業で開発した「低品質モジュール予測モデル」を改良し，同事業で開発した「低品質モジュール可視化システムのプロトタイプ」を改良モデルに基づき機能拡張することで，目標とする解析システムを実現した．モデル構築の概要を図 3-5-3 に示す．

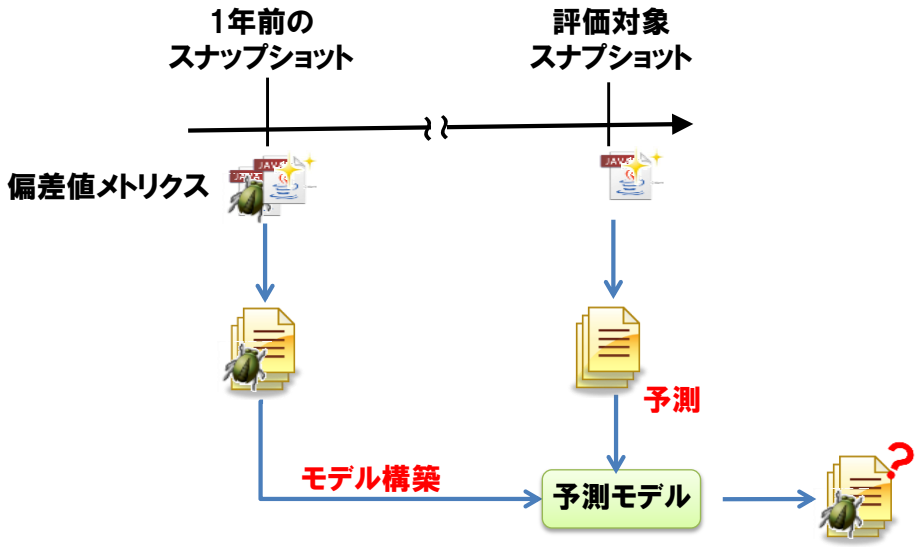


図 3-5-3 偏差値メトリクスを用いた低品質モジュール予測の概要

また，具体的な改良手順は次の通りである．

まず，一般標準値域から各メトリクスを抽出し，欠陥あり，欠陥なしのモジュール群において，それぞれメトリクス偏差値を求める．そして，各プロジェクトで求めたメトリクス偏差値を算出する．次に，メトリクス偏差値を用いて予測モデルを構築する．モデル構築にはランダムフォレスト法を用いた．ランダムフォレスト法とは，決定木（回帰木，分類木）を用いて集団学習を行う手法である．モデル構築用のデータセットに対して復元抽出を複数回行い，得られた各サンプル群に対して決定木を構築し，各決定木の多数決または平均により最終的な予測結果を得る．予測精度の評価には，Mende らが提案した *Popt* [Mende]を用いる．*Popt* は，横軸に欠陥密度が大きい順にモジュールを並べたときの累積行数，縦軸を累積欠陥数とし，欠陥密度の予測値と実測値が描く各 LOC-based Cumulative Lift Chart の AUC(曲線下面積)の差分 Δopt から求められる（図 3-5-4 参照）．

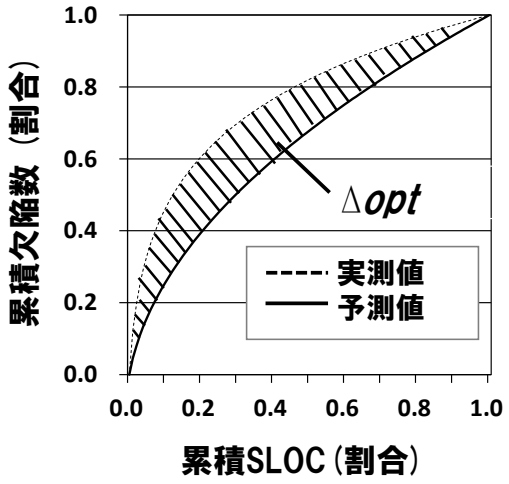


図 3-5-4　Δopt の定義

$Popt$ は欠陥モジュール予測後のテスト工数を考慮した尺度であり，欠陥モジュール予測モデルの評価に利用されている[Monden][Kamei]．Popt の値域はデータの欠陥数によって変動するため，ここでは次式により正規化する[Monden]．

$$\text{正規化された}Popt = \frac{Popt\ -\ MIN(Popt)}{MAX(Popt)\ -\ MIN(Popt)}$$

ここで MAX(Popt)，MIN(Popt)はそれぞれ $Popt$ の取り得る最大値，最小値とする．正規化により $Popt$ の値域は[0,1]となり，値が 1 に近づくほど予測精度が高いことを示す．以降では，$Popt$ はすべて正規化されているものとする．$Popt$ は，欠陥モジュール予測の評価指標として広く採用されている適合率や再現率，F1 値とは異なり，予測精度が，判別結果を 2 値に区切る際の閾値に影響されないという特徴を持つ．類似の評価指標として Alberg Diagram [Ohlsson]の AUC があるが，モジュールの規模は考慮されない．

プロジェクト wine，および，sipxecs における低品質モジュールの予測結果を表 3-5-2 に示す．同表は，次に示す 4 つのモデルによる予測結果を示している．

（ⅰ） メトリクス値を正規化せず，学習データを単一プロジェクトから得てモデル構築

（ⅱ） メトリクス値を行数で正規化し，学習データを単一プロジェクトから得てモデル構築

（ⅲ） メトリクス値を偏差値メトリクスで正規化し，学習データを単一プロジェクトから得てモデル構築

（ⅳ） メトリクス値を偏差値メトリクスで正規化し，学習データを複数のプロジェクトを混在させることで得てモデル構築

表 3-5-2 低品質モジュールの予測結果

評価データセット	学習データセット	(i) 正規化無し 単一プロジェクト	(ii) 行数で正規化 単一プロジェクト	(iii) 偏差値で正規化 単一プロジェクト	(iv) 偏差値で正規化 混在プロジェクト
wine	hibernate	0.5	0.62	0.5	0.22
	opennms	0.54	0.75	0.54	
	sipxecs	0.61	0.67	0.6	
	spring	0.54	0.69	0.54	
	wine	0.73	0.73	0.72	
	eclipse	0.39	0.75	0.37	

評価データセット	学習データセット	(i) 正規化無し 単一プロジェクト	(ii) 行数で正規化 単一プロジェクト	(iii) 偏差値で正規化 単一プロジェクト	(iv) 偏差値で正規化 混在プロジェクト
sipxecs	hibernate	0.37	0.28	0.37	0.26
	opennms	0.49	0.34	0.5	
	sipxecs	0.81	0.7	0.69	
	spring	0.58	0.47	0.62	
	wine	0.41	0.44	0.41	
	eclipse	0.21	0.27	0.23	

3.5.3 課題とその対応

当初，低品質モジュールとは，ソースコードの可読性が低いモジュールと定義していたが，可動性の自動計測が小規模なプロジェクトにのみ限られていたため，欠陥の有無のみ，つまり，欠陥があるモジュールを低品質とみなすこととした．

3.6 研究目標 6「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価」

3.6.1 当初の想定

(1) 研究内容

ソフトウェアプロジェクトトモグラフィで定義される 5 つの観点（要件，作業，組織，プロダクト，課題）すべてにおいてソフトウェアプロジェクトデータの解析・可視化が可能で，かつ，注目するデータグループへのズームイン，および，データグループから全体へのズームアウトが，それら観点を変更しながら再帰的に（階層数に制限なく）可能な方式の設計を行う．まず，ソフトウェアプロジェクトデータの量的属性を，ソフトウェア品質の第三者評価においてどのように活用するのか，その具体的手順を「第三者評価シナリオ」として策定する．次に，2012 年度の本先導的研究支援事業で定義した「プロジェクト構造モデル（ER 図）」を基に，再帰表現に適したデータ構造を定義する．更に，メニューバーや操作ボタン等を極力少なくし，Treemap 上での特徴的な操作である「ズームイン・ズームアウト」を直感的に行えるインタフェースについて検討し，解析・可視化のためのプロトタイプの設計を行う．

プロトタイプの実装には，電子アートとビジュアルデザインのためにオープンソースとして公開されているプログラミング言語・環境 Processing を用いる．また，データ入力・

操作装置として，従来から広く用いられているマウスだけでなく，タッチスクリーンやタブレット端末などについても検討する．実証実験は，本委託研究で策定した「(ソフトウェア品質の）第三者評価シナリオ」に基づいて行う．ソフトウェア品質の第三者評価やソフトウェアプロジェクトデータの分析に関する初心者と熟練者の双方を被験者とし，評価・分析のプロセスや結果を比較することで，提案する方式の妥当性，有用性，効率性などを評価する．

(2) 当初の到達目標と期待される効果

到達目標は,「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化技術の実現」である．具体的には，ソフトウェアプロジェクトトモグラフィで定義される 5 つの観点（要件，作業，組織，プロダクト，課題）すべてにおいてソフトウェアプロジェクトデータの解析・可視化が可能で，かつ，注目するデータグループへのズームイン，および,データグループから全体へのズームアウトが,それら観点を変更しながら再帰的に(階層数に制限なく）可能な Treemap システムを開発する．2012 年度の本先導的研究支援事業でも，ソースコード修正作業量を俯瞰するために Treemap を用いた．ただし，プロダクトと課題に関する一部のデータのみが表示可能であり，ズームイン・ズームアウト可能な階層数にも制限があった．ここでは特に,

① 5 つの観点でソフトウェアプロジェクトデータが扱え，再帰表現に適したデータ構造の定義.

② 探索的な操作が可能，かつ，習熟性や低作業負荷も重視したユーザインタフェースの実現.

を目指す.

ソースコードの行数・複雑さ，発見欠陥数，作業工数など，ソフトウェアプロジェクトは多くの量的属性を有している．Treemap は，そうした量的属性に基づいて対象を解析・可視化するのに適した手法である．5 つの観点でプロジェクトを俯瞰し，それら観点を変更しながら注目するデータへのズームイン・ズームアウトが可能になれば，これまで以上に探索的な解析・可視化が可能になることが期待される．

3.6.2 研究プロセスと成果

(1) 研究プロセス

まず,2012 年度の本先導的研究支援事業で定義した「プロジェクト構造モデル (ER 図)」を基に，ソフトウェア開発データの再帰表現に適したデータ構造を定義する．単に再帰表現が可能というだけでなく，試行錯誤を繰り返す探索型の解析・可視化に適したデータ構造を目指す．次に，メニューバーや操作ボタン等を極力少なくし，Treemap 上での特徴的な操作である「ズームイン・ズームアウト」を直感的に行える，習熟性や低作業負荷を重視したユーザインタフェースの実現を目指す．例えば，ズームインしたい領域をクリックすることで,ズームインの観点となる量的属性の候補を示すウインドウがポップアップし,フリック方式で属性を選択する方法，などが考えられる．また，操作履歴の保存・再現機能などを持たせることで，探索的な解析・可視化における作業負荷の低減を図る．更に,ズームイン・ズームアウトの観点となる量的属性をランダムに表示することで，偶発的発見（Serendipity）を促す．具体的な作業項目は次の通り.

シナリオの策定
データ構造の定義
プロトタイプシステムの概要設計
プロトタイプシステムの詳細設計
プロトタイプの実装
実証実験の計画・準備
実証実験の結果評価

(2) 具体的な研究成果（結果）

① 探索的 Treemap(Treemap Forest)による俯瞰と関係把握に適したデータ構造の定義

ソフトウェアプロジェクトデータを Treemap 上で表現するために定義したデータ構造を図 3-6-1 に示す．また，このデータ構造に基づき設計した，Treemap Forest のクラス図を図 3-6-2 に示す．

これらに基づき，プログラミング言語・開発環境である Processing を用いて Treemap Forest のプロトタイプシステムを実装した．実装規模はクラス数 6，コード行数 950 で．実証実験用に Eclipse JDT の開発履歴データを保持するデータベースも含まれている．なお,Treemap における並べ替えやデータ操作には,Ben Fry が開発した Processing 用 Treemap ライブラリを用いている．具体的には，同ライブラリで定義されているクラスとインタフェースを継承することで,図 3-6-2 中の SimpleMapItem クラスと MapModel インタフェースを実現している．同ライブラリは，Martin Wattenberg，Ben Bederson らが開発した Java 用 Treemap ライブラリを Processing 用に改造したものであり，MPL ライセンスの下で一般に配布されている．

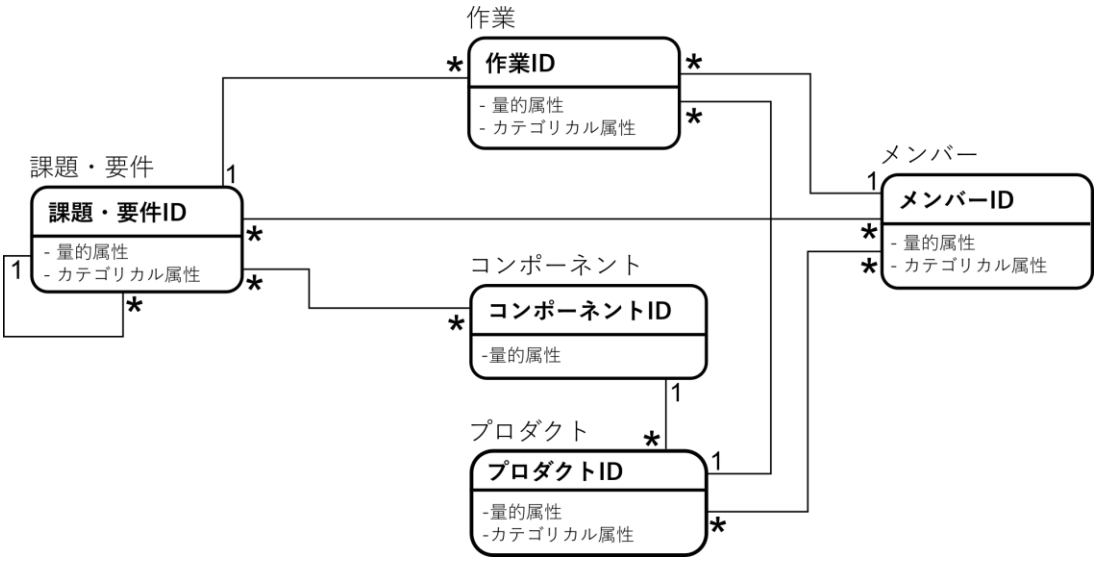


図 3-6-1　Treemap Forest におけるデータ構造

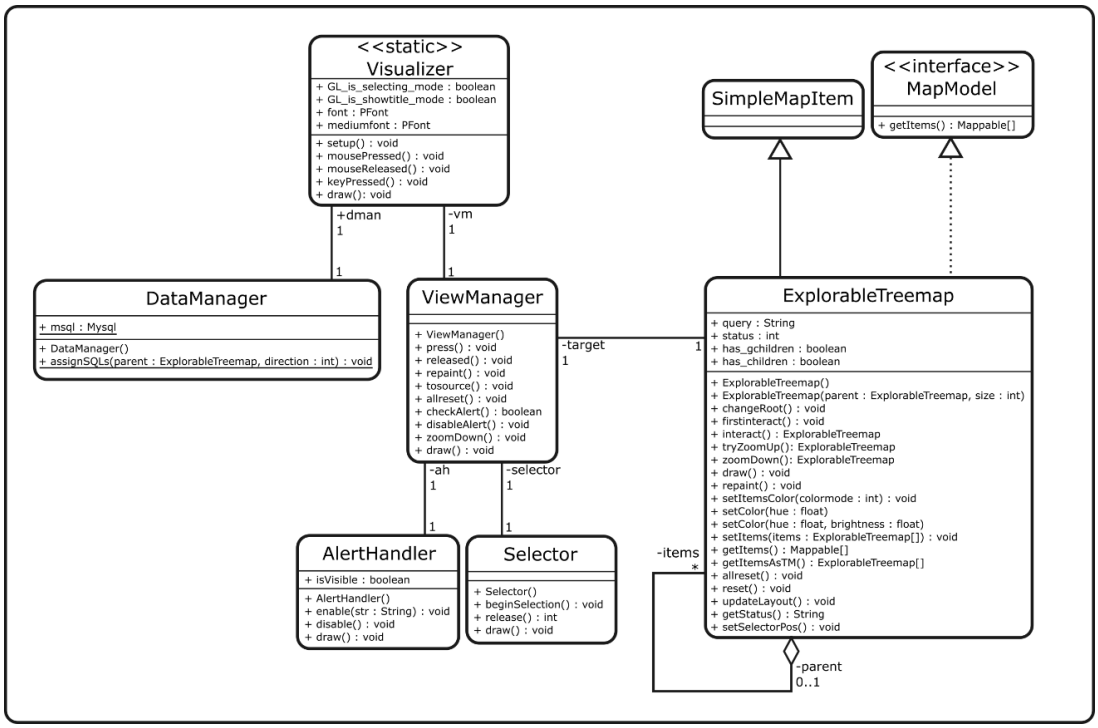
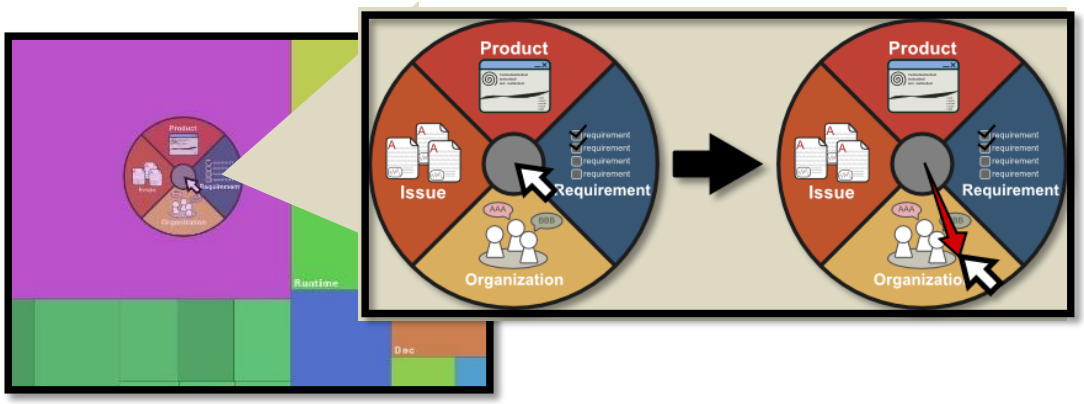


図 3-6-2　Treemap Forest のクラス図

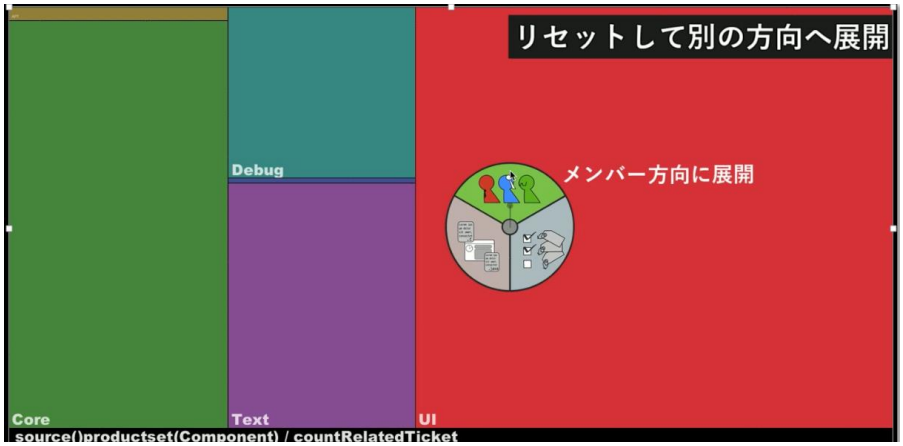
② 探索的な操作が可能，かつ，習熟性や低作業負荷も重視したユーザインタフェースの実装

Treemap Forest におけるユーザインタフェースの概念図と実際の表示例を図 3-6-3 に示す．また，その実装のために設計した動作モデルを図 3-6-4 に示す．これらに基づき，Treemap Forest は，次のような機能を提供する．

- ズームインによる観点間の移動
- Treemap 表示における観点のランダム設定
- 異なる観点間の関連を表現するための，Treemap の再帰展開・表示
- 偶発的発見を支援するための，観点のみを指定した Treemap 展開・表示
- ズームイン操作履歴の記録，遡及
- 展開・ズームアップに関する文脈情報の破棄，初期状態へのリセット
- Treemap 表示におけるカラーリングの変更



(a) 概念図



(b) 実際の表示例

図 3-6-3　Treemap Forest のユーザインタフェース

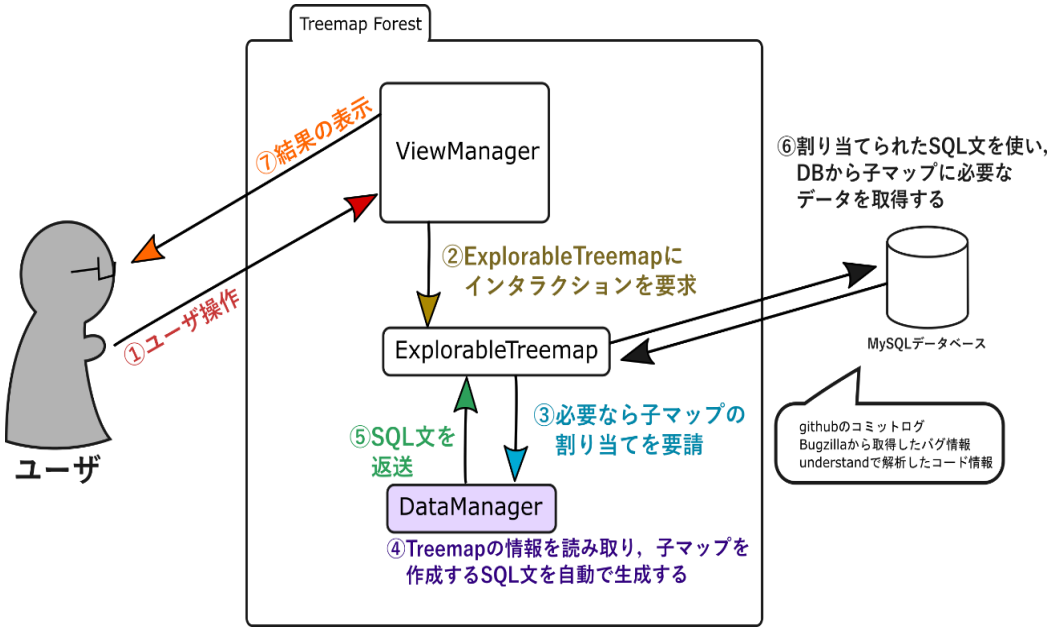


図 3-6-4　Treemap Forest の作モデル

③ 実証実験の計画・準備

プロトタイプシステムの評価には，実際に開発され，広く利用されているソフトウェアのソースコード，および，その開発データを用いることとした．そこで，Eclipse JDT プロジェクトで開発されているソースコードを入手すると共に，同プロジェクトで用いられているバグ管理システム，および，版管理システムが蓄積，管理している開発データを取得し，静的解析，および，プロトタイプシステムへの読み込みが可能であることを確認した．

④ 実証実験

Treemap Forest が，ソフトウェア品質の第三者評価に役立つのかを，実験準備において入手，取得した Eclipse JDT のソースコードとその開発データを用いて確認した．ソフトウェア品質の第三者評価では，品質そのものの評価に加えて，対象ソフトウェアの品質が低い場合，あるいは，価により対象ソフトウェアが低品質と判明した場合には，品質低下の原因究明と再発防止策の検討が行われることになる．原因究明と再発防止には，要素作業として，

1) プロジェクト全体の俯瞰
2) プロジェクト構成要素（間）の傾向や関係性の顕在化・評価
3) 仮説の生成・検証

が不可欠となる．以下では，これら３つの要素作業のプロトタイプシステム上での実行例を示す．

1) プロジェクト全体の俯瞰

ソフトウェア開発プロジェクト進行中のある時点における状況を，第三者が大まかに把握し，理解しようとする場合を想定する．第三者評価の初期段階において，評価者は，（現実世界に存在する）評価対象プロジェクトそのものはもちろんのこと，そのプロジェクトから収集されたデータ（開発データ）の詳細について知る由もない．また，一般に，プロジェクトの全体像を大まかに把握や理解する際に，プロジェクトをどの観点でまず可視化すべきかという基準はなく，知見も得られていない．評価者がプロジェクトの把握や理解を始めようにも，その手がかり，とっかかりがないのが現状である．Treemap Forest では，この問題を解決するため，起動時（作業開始時）には，可視化の観点がランダムに設定されて Treemap が表示される．

Treemap Forest 起動時の表示例を図 3-6-5 に示す．この例では，観点として「コンポーネント（プロダクトの一種）」がランダムに設定され，各コンポーネントを表す矩形領域の面積は，（ランダムに設定された）属性「コンポーネントに関係するチケット数」の値で按分されている．同図を見ると，コンポーネント UI，Core，Text，Debug などが存在し，コンポーネント UI と Core に関係するチケットが特に多いことが分かる．

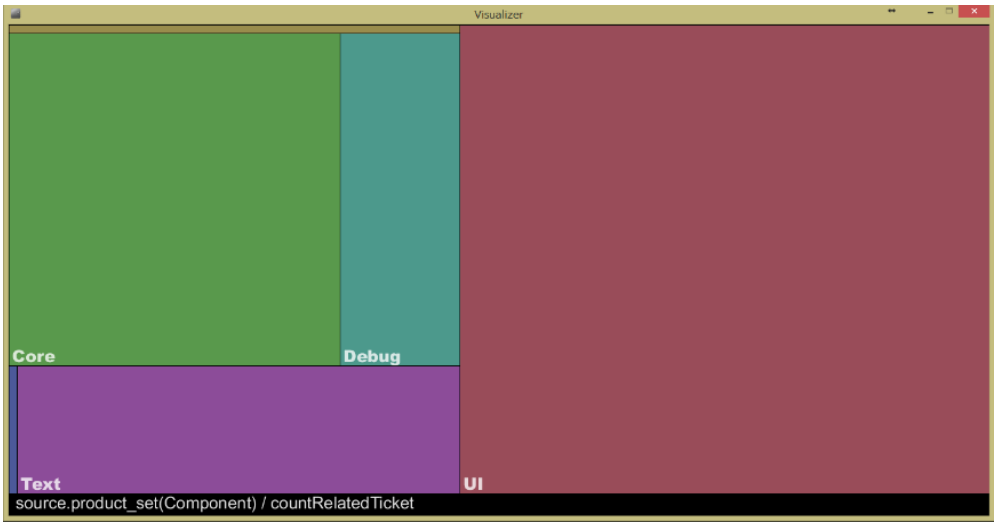


図 3-6-5　プロジェクト俯瞰例：観点「コンポーネント」,「関連するチケット数」

より詳細な分析へと進むことも可能であるが，プロジェクト全体の大まかな把握と理解が目的なので，観点を変えて俯瞰してみることとする．プロトタイプシステムの「観点切り替え」機能を用いることで，異なる観点で作成された Treemap を表示させることが出来る．図 3-6-6 では，観点として「メンバー」が（ランダムに）設定され，各メンバーを表す矩形領域の面積は，（ランダムに設定された）属性「メンバーに割り当てられたチケット数」の値で按分されている．同図を見ると，割り当てられているチケット数はメンバーによってばらついており，メンバーaeschli，akiezun に数多くのチケットが割り当てられていることが分かる．なお，メンバー名「jdt-****-inbox」は，メンバーが割り当てられていないチケットに対して付与される既定値であり，”****”には，対応するコンポーネント名が記される．約半分のチケットにメンバーが割り当てられておらず，また，いずれのコンポーネントにも，メンバーが割り当てられていない関連チケットがあることが分かる．

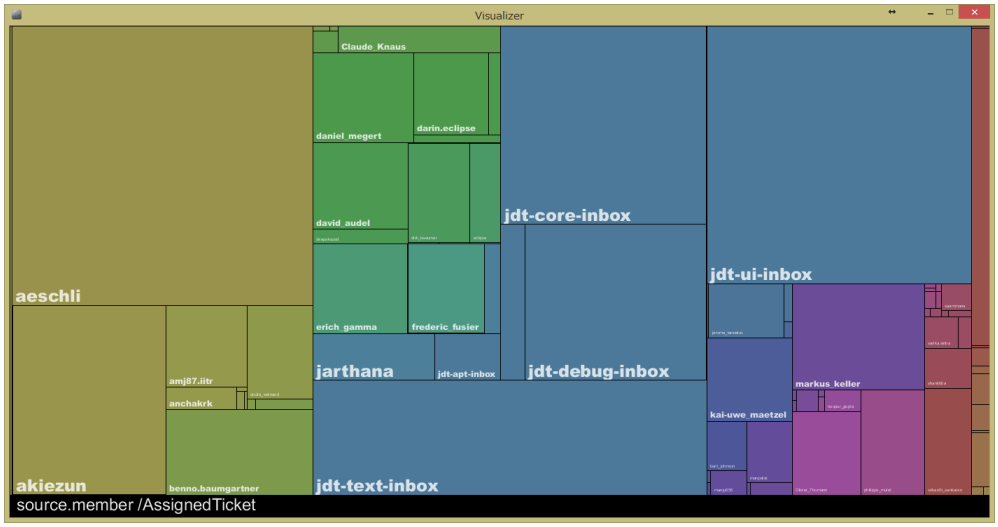


図 3-6-6　プロジェクト俯瞰例：観点「メンバー」,「割り当てられたチケット数」

再び「観点切り替え」機能を用い，観点を変えて俯瞰してみることとする．図 3-6-7 では，観点として「チケット」が（やはりランダムに）設定され，各チケット表す矩形領域の面積は，（やはりランダムに設定された）属性「確認状況」の値で按分されている．ただし，属性「確認状況」は，いわゆるカテゴリカル変数であり，より正確には，矩形領域の面積は，「確認状況」を表す各カテゴリに属するチケット数に比例して按分されていることになる．同図を見ると，半数以上のチケットが「未割当（---)」として残っており，また，「重複（DUPLICATE)」，「修正されていない(WON'T FIX)」，「確認した環境では動作している（WORKS FOR ME)」といったカテゴリがあることも分かる．

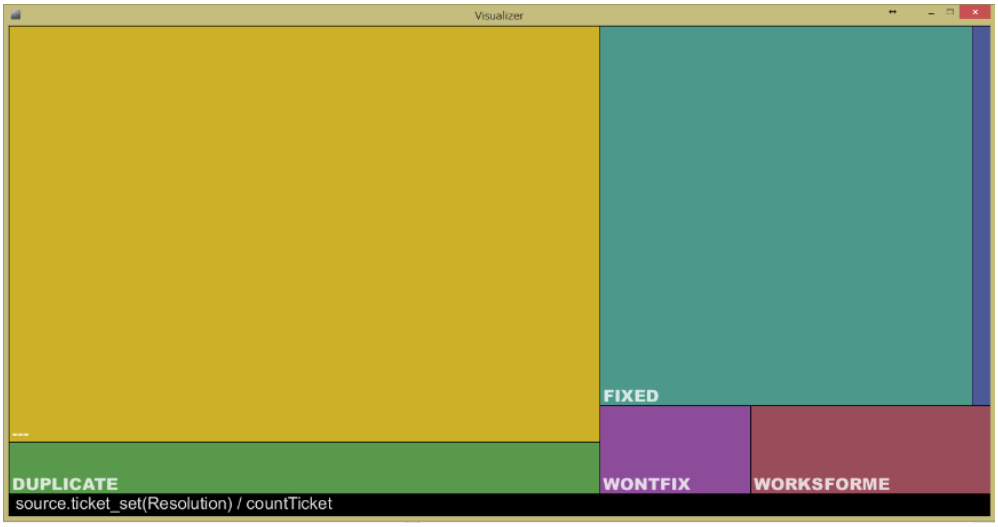


図 3-6-7　プロジェクト俯瞰例：観点「チケット」，「確認状況」

以降も，同様に観点を変えての俯瞰を続けることが出来る．詳細は割愛するが，観点として「コンポーネント」とその「サイクロマティック複雑度の最大値(SumCyclomatic)」が設定された表示例を図 3-6-8 に，観点として「コンポーネント」とその「割当・解決状況」が設定された表示例を図 3-6-9 に，それぞれ示す．図 3-6-8 からは，ソースコードの複雑さが，コンポーネント間で大きくばらついており，コンポーネント Core には，極端に複雑なソースコードが含まれていることが分かる．また，図 3-6-9 からは，「解決済み(RESOLVED)」のチケットは全体の約半数が未解決であり，残る半数(の未解決チケット)は「未割当(NEW)」と「割当済み（ASSIGNED)」に二分されることが分かる．

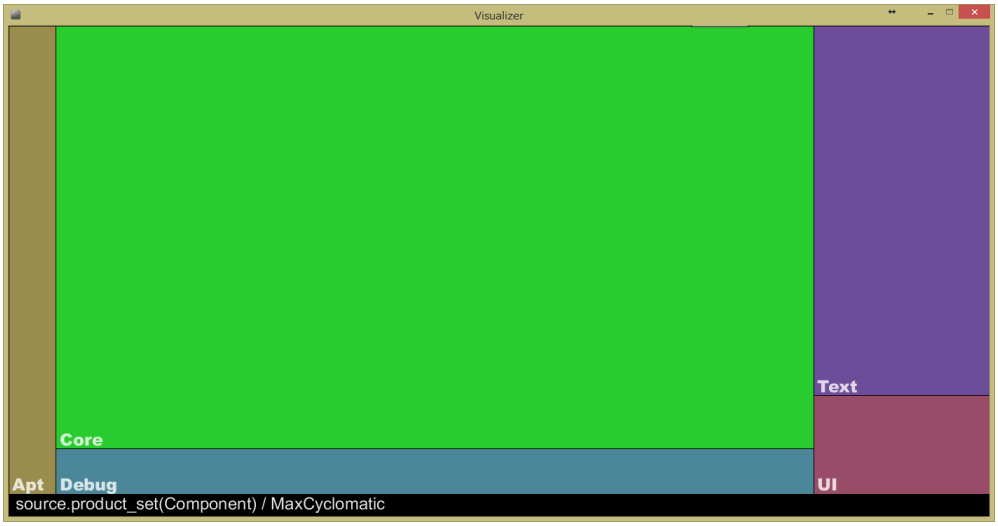


図 3-6-8　プロジェクト俯瞰例：観点「コンポーネント」,「サイクロマティック複雑度の最大値(SumCyclomatic)」

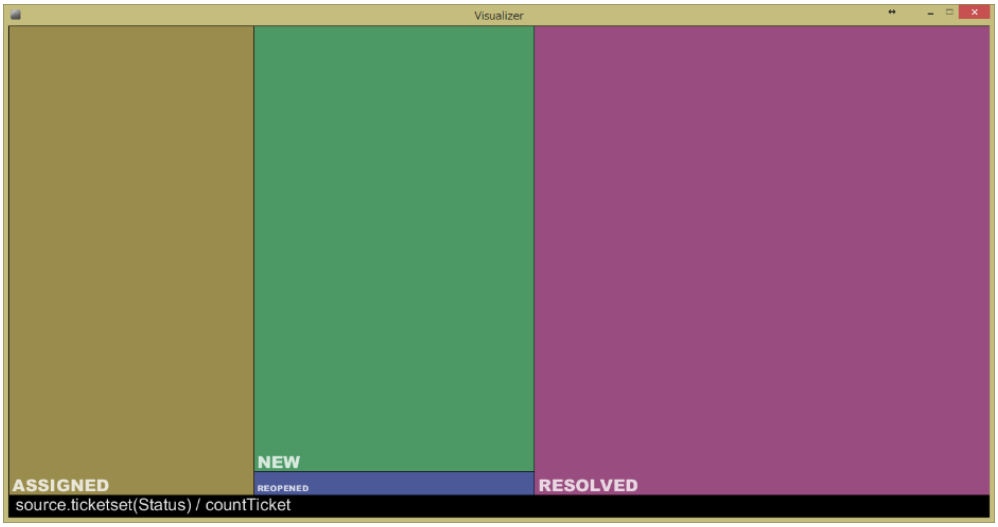


図 3-6-9　プロジェクト俯瞰例：観点「チケット」,「割当・解決状況」

2) プロジェクト構成要素（間）の傾向や関係性の顕在化・評価

プロジェクト全体の俯瞰によって，プロジェクトの全体像や状況の大まかな把握，理解を進める中で，評価者は，対象ソフトウェアにおける品質低下の原因究明につながる糸口を見つけようと，いわゆる，探索的な可視化，解析が行われることになる．例えば，ソフトウェア品質との関連性が指摘されている属性の値分布や外れ値の存在は，そうした糸口の一つとなる．

図 3-6-8 で観点として用いた「サイクロマティック複雑度」は，ソフトウェア品質との関連性が指摘されている属性（メトリクス）の一つである．サイクロマティック複雑度は，ソースコード中の分岐数に 1 を加えた値であり，ソースコード中のすべての経路を最低 1 回実行するために必要なテスト回数（テストデータ数）の下限に一致する．複雑度の大小が品質の高低を直接示すものではないが，複雑度に見合ったテスト工数が，コンポーネント（やその構成要素であるファイル）に割り当てられないと，十分なテストが行われず，結果として，当該部分のソースコードが低品質となる可能性が高い．そこで，コンポーネントの構成要素であるファイル単位で「サイクロマティック複雑度」を確認することとする．実装したプロトタイプシステムは，Treemap 形式で分割表示されている構成要素（子要素）を，更にその構成要素（孫要素）によって Treemap 形式で分割表示することができる．各孫要素の表示面積は，その属性値に比例して按分される．また，同システムは，表示履歴を記録しその履歴をたどる機能を有しており，こうした探索的な可視化，解析が容易となっている．

図 3-6-10 は，履歴をたどる機能によって「図 3-6-8 で示した表示」に戻り，各コンポーネントをその構成要素（孫要素）である「ファイル」の属性値に基づいて Treemap 形式で表示するために，ポップアップメニューで観点「プロダクト」を選択している様子を示している．また，図 3-6-11 は，そうした操作の結果として得られた表示であり，孫要素である各ファイルを表す矩形領域の面積は，属性「サイクロマティック複雑度」で按分されている．同図を見ると，ファイル Parser.java のサイクロマティック複雑度が，コンポーネント Core の中でひときわ大きいことが分かる．

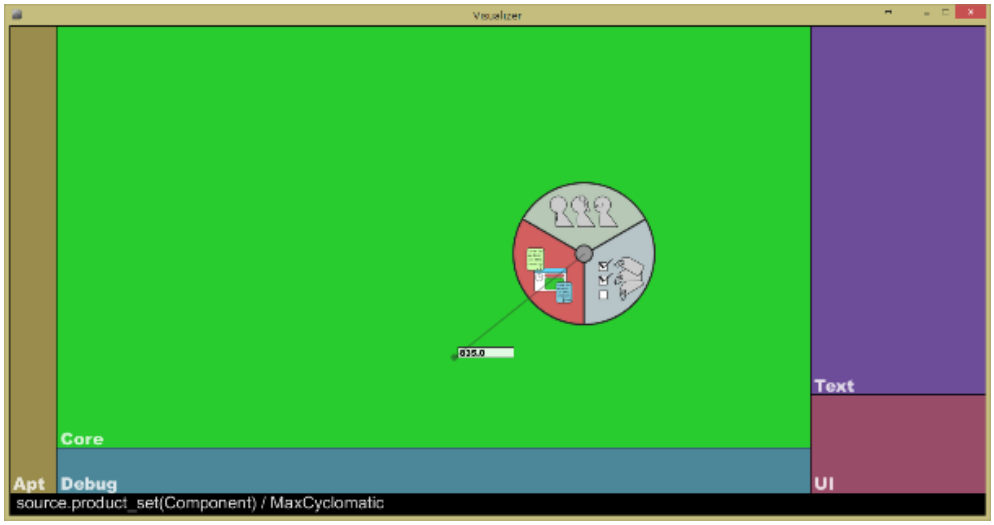


図 3-6-10　「図 3-6-8 で示した表示」からプロダクト方向に展開

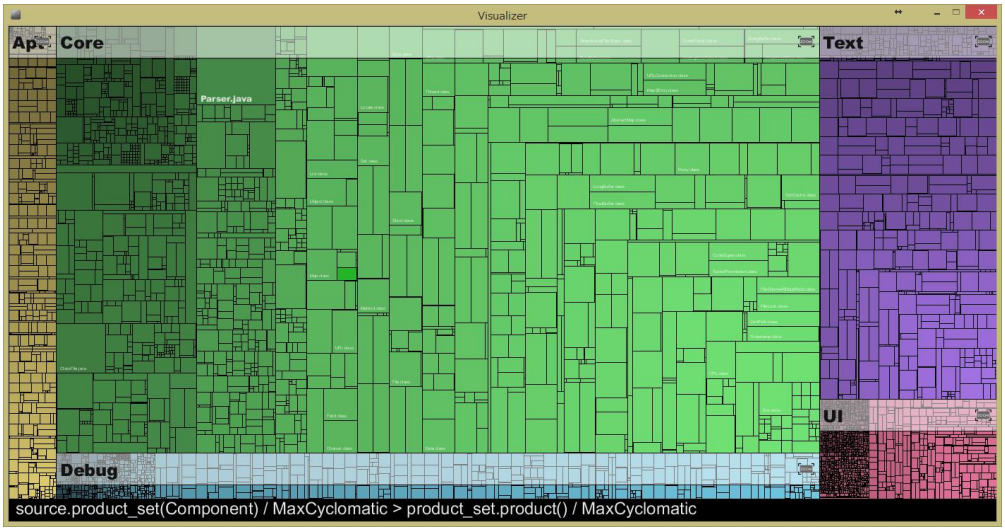


図 3-6-11　詳細表示：観点「コンポーネント」→「ファイル」,「サイクロマティック複雑度」

ただし,「サイクロマティック複雑度」は,「ソースコード中の分岐数に 1 を加えた値」であり，一般に，ファイルサイズ（コード行数）が大きくなると，同複雑度も大きくなる傾向にある.（ファイルサイズが極端に大きいことも，ソフトウェア品質に影響するとの指摘もあるが,）ファイル Parser. java は，そのサイズが大きいために，他のファイルと比べてサイクロマティック複雑度が大きくなっているだけとも考えられる．そこで，同様にして，孫要素である各ファイルをその属性「コード行数」に基づいて Treemap 形式で表示してみる.得られた結果を図 3-6-12 に示す.同図を見ると,ファイル Parser. java 以外にも,ファイル ClassFile. java など,コード行数が比較的大きいファイルが多数存在することが分かる．コンポーネント Core においては，ファイルのサイズが大きいからといってそのサイクロマティック複雑度が大きいとは限らないことになる.

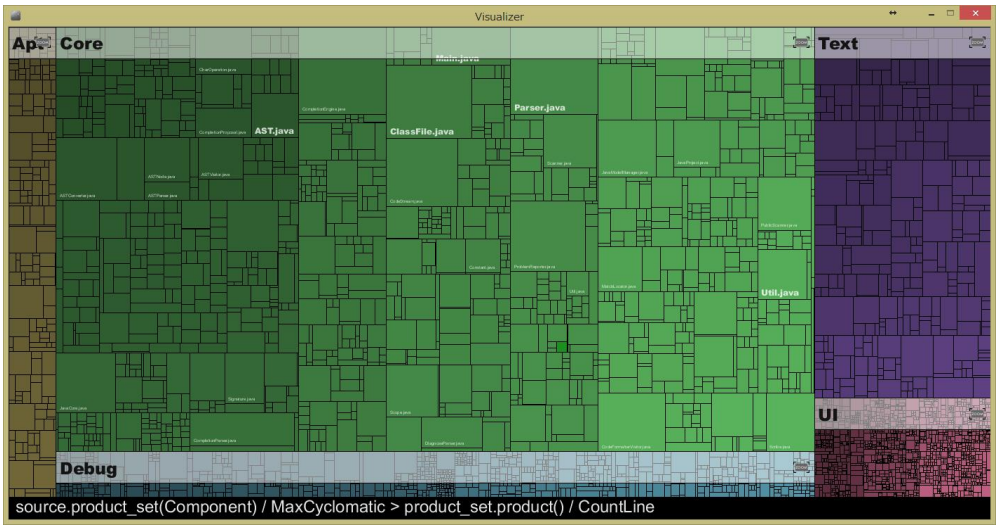


図 3-6-12　詳細表示：観点「コンポーネント」→「ファイル」,「コード行数」

以上により，コンポーネントCoreのファイルParser.javaは，サイクロマティック複雑度の観点からは特異な存在であり，対象ソフトウェアにおいて品質低下が見られるのであれば，その原因究明に向けた詳細分析の対象候補の一つとなる．また，対象ソフトウェアのごく一部でではあるが，サイクロマティック複雑度とコード行数の間で，必ずしも高い相関が見られないことも分かった．この傾向が対象ソフトウェア全体で見られるものであり，かつ，コンポーネントやファイルのテスト工数が，それらのコード行数に基づいて割り当てられているのであれば，十分にテストされていないコンポーネントやファイルが存在することになる．テスト計画やテストプロセスの妥当性評価の端緒となることも考えられる．

3) 仮説の生成・検証

「2)プロジェクト構成要素（間）の傾向や関係性の顕在化・評価」においては，対象ソフトウェアにおける品質低下の原因究明につながる糸口を見つけるところまでがTreemap Forest上での作業であり，仮説の生成や検証には，より詳細な分析が別途必要となる．ただし，比較的単純な仮説であれば，Treemap Forestによる表示に基づいて生成し，その検証を行うことも可能である．

例えば， 先述の通り，対象ソフトウェアにおいては，

- 「解決済み（RESOLVED)」のチケットは全体の約半数が未解決であり，残る半数（の未解決チケット）は「未割当（NEW)」と「割当済み（ASSIGNED)」に二分される（図3-6-9より）
- 関連するチケット数は，コンポーネントによって大きくばらついており，チケットの約半数はコンポーネントUIに関連している（図3-6-5より）．

ということが分かっている．このことから，未解決チケットが半数にものぼるのは，特定のコンポーネント（具体的には，コンポーネントUI）に多くのチケットが関係づけられている結果ではないか，と推察することが出来る．すなわち，この２つから

仮説１：チケットの多くが特定のコンポーネントに関連づけられ，その対応が遅れているため，未解決のチケットが多数存在する．

あるいは

仮説２：各コンポーネントにおけるチケットの未解決率は一定ではなく，コンポーネントに関連づけられているチケットが増えると未解決率は大きくなる．

といった仮説を生成することが出来る．

この仮説を検証するため，「図3-6-7で示した表示」に戻り，チケットの「割当・解決状況」それぞれを，関連するコンポーネント数で更に按分した結果を図3-6-13に示す．同図を見ると，「解決済み（RESOLVED)」となっているチケットの大半は，コンポーネントUIに関連するチケットであり，逆に，「未割当（NEW)」となっているチケットに限れば，その割合は,コンポーネントUIよりもコンポーネントCoreが大きいことが分かる.詳細なデー

タや数値を確認するまでもなく，上記の仮説１，および，仮説２は棄却される．

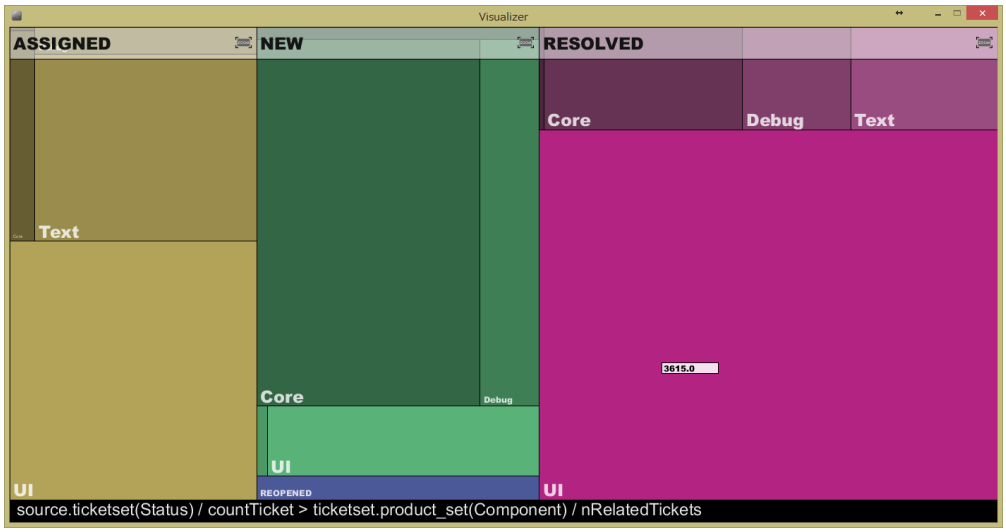


図 3-6-13　詳細表示：観点「チケット」,「割当・解決状況」→「コンポーネント数」

3.6.3 課題とその対応

Treemap では，対象データの全体を示す矩形領域が，その構成要素（子要素）によって分割表示され，各子要素の表示面積は，利用者が指定した属性値に比例して按分される．面積の按分に用いられる属性値が極端に大きい場合は，「外れ値」として発見しやすいが，属性値が極体に小さい場合は発見が難しい．そこで，必要があれば，属性値の逆数で面積をを按分する機能を考案した．

4 考察

4.1 判明した効果や課題と今後の研究予定

《研究目標 1「欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」》

本研究目標では，欠陥レポートの記述から欠陥の内容によって識別するアプローチを取った．一方，学会での情報収集から，欠陥の及ぼすインパクトによって識別するアプローチも活発に研究されていることがわかった．Kashiwa らは，これまでに研究されている，プロセスに関する 3 つの欠陥とプロダクトに関する 3 つの欠陥をハイインパクトな欠陥としてまとめている[Kashiwa]．

- プロセス関連

  Surprise bugs：予期していな時期や位置に発生し，ワークフローやタスクスケジュールの変更が必要となる欠陥[Shihab]．

  Dormant bugs：休眠バグ．発見時よりも何バージョンか以前に既に混入されていた欠陥．優先的に修正する際は，予定のワークフローを変更する必要がある[Chen]．

  Blocking bugs：依存関係から，他の欠陥修正より優先的に修正する必要がある欠陥．タスクスケジュールに影響がある[Valdivia]．

- プロダクト関連

  Security bugs：セキュリティに関連する欠陥．ユーザに直接影響があるハイインパクトな欠陥である[Gegick]．

  Performance bugs：プログラミングエラーなどによって深刻なパフォーマンス低下をもたらす欠陥．ユーザに大きな影響がある[Nistor]．

  Breakage bugs：以前は使えていた機能が，別の機能追加などの変更によって使えない状態になる欠陥[Shihab]．

今後，欠陥の内容による識別とインパクトによる識別がどのような関係になっているか，実用上どのような利点や欠点があるかなどを調査，研究することが必要である．既に数千単位のバグレポートを目視で丹念に調べ，上記のプロセス関連とプロダクト関連の欠陥の有無を明らかにしつつある．今後は，そうしたデータや調査結果に基づいて，欠陥の内容による識別との関連性や修正時間・修正割り当て状況などを統計的に分析していく．

《研究目標 2「ソフトウェア障害の波及度を定量的に評価する方式の設計と実証的評価」》

設定した到達目標をおおむね達成することができた．具体的な成果は，1)第三者評価，2)波及度の定量化，および，自動計測ツールの開発，3)モデル化，である．波及度を自動的に計測する枠組みを構築した点は，他の類似研究と比べ優れている．

基準（ランダムに予測する場合）と比べて，適合率を 100%向上することができた．その一方で，適合率 80%を達成することが，新たに見出された課題である．その解決のために，精度向上に寄与するメトリクス（単位期間内のクラッシュの増加率など）の調査と，評価実験の繰り返しを行う．また，実開発プロジェクトにおいて，波及度をどのように扱うべきかを実証的に評価することも，新たに見出された課題である．この点は，実開発プロジェクトへの参画，及び，クラッシュレポートシステムの導入を実施し，データを蓄積することで実施可能であると考える．

《研究目標 3「非機能要件の自動評価方式の設計と実証的評価」》

非機能要件を 3 段階で評価した場合，人手のよる評価との一致率が約 70%，±一致率が約 97%と高い値となっており，実用に耐える成果といえる．なお，5 段階評価としても，人手のよる評価との一致率は 62%で，一致率の低下は 7.9%に留まる．

新たに見出された課題としては，自動評価のためのモデル構築に一定のコストを要する点がある．この点は，学習データを必要としない教師なしの自動評価により解決されると考える．具体的には，それぞれの非機能要件キーワードの出現回数に TF-IDF 等による重みづけを行って加算したものを非機能要件の評価スコアとして用いることが考えられる．文書中に非機能要件キーワードが出現しているということは非機能要件について何らかの記述がなされているということであるため，本アプローチは，自動評価の一手法として一定の効果が期待される．研究計画としては，重みづけ方法には TF-IDF 以外にも様々な方法が考えられるため，今後，それらを実験的に比較することと，教師ありの自動評価との精度比較を行うことが挙げられる．

論文発表等による外部の客観的評価としては，本成果の一部が，博士学位論文「齊藤康廣，ソフトウェア開発の超上流工程における非機能要件の定量的評価，奈良先端科学技術大学院大学」に収録される予定であり，同論文が審査委員から実用性の高い研究として高い評価を得ている点を挙げることができる．

《研究目標 4「多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価」》

利用アプリ履歴に対する改ざんを検出する手法と，利用アプリと開発作業の自動マッピングを 97%という高い精度で実現する手法によって，第三者評価に耐えうる正当なデータに基づいた開発作業の解析が可能になると考える．同時に，改ざんの恐れがないデータをソフトウェアの発注者やユーザに提供することで，今まで知ることのできなかった開発過程の情報が得られるようになり，開発ソフトウェア品質やその実現プロセスに対する「情報の非対称性」を解消する．更に，アプリケーション実行履歴は，分単位の細粒度で収集，記録され，開発作業との自動マッピングも高い精度で実現されていることから，開発組織の労務管理データを補完し，従来よりも詳細な工数管理が可能となる．

本研究目標において開発した自動マッピングの手法は前後の実行履歴を用いない予測（従来法に相当）と比べて精度が 25～35%向上した一方，処理時間が 10～30 倍（数分程度）に増加した．自動マッピングは評価の際に一度だけ行うため，大きな障害にはならないと考えるが，今後の研究で高速化の方法について検討したい．具体的には，マッピングに用いているランダムフォレストについて，作成する決定木の深さを制限する方法や，派生アルゴリズムのうち高速化が見込める Extremely Randomized Trees や Random Ferns を用いる方法が考えられる．これらの手法を用いた自動マッピングの精度と速度をそれぞれ実験で評価することで，速度と精度の両方に優れた手法を検討する．

利用アプリ履歴の改ざん検出について，開発組織内での改ざんの可能性に着目してシステムを構築した．

今後，情報セキュリティ分野における研究成果を調査し，外部組織からの攻撃によって

第三者評価の正当性を損なう可能性がある通信部分や，情報漏洩による利用者への損失の可能性がある利用プロセスを整理・検証することで，第三者評価の信頼性をより高めることができると考えられる．今後の研究計画としては，外部からの攻撃や情報漏洩の可能性がある箇所に対するアクセスを意図した攻撃者モデルを複数構築し，システムが対処できない箇所やその対応策を明らかにした上でシステムを改善する予定である．

《研究目標 5「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」》

2012 年度の本先導的研究支援事業では，ソフトウェア品質の外的妥当性評価のためにメトリクス基準値を作成していたが，メトリクス値が欠陥にどのように影響しているのか明らかにしていなかった．たとえメトリクスが異常値を示したとしても，その値は一般的なソフトウェアと比べて異常値なだけであり，低品質モジュールであるかまでは判定できなかった．「欠陥を含むモジュール」と「欠陥を含まないモジュール」に分類し，それぞれから標準値域を作成することで，欠陥が混入している可能性を定量的に判定することを可能にした．

偏差値メトリクスは，特定のプロジェクトにおける標準値域だけでなく，社会で利用されている他の OSS プロジェクトの標準値域と比較する点において品質の第三者評価に有用である可能性が高い，とのコメントを，ウィンターワークショップ 2015 での発表において得ている．実験結果でも，低品質モジュールと高品質モジュールとで標準値域分布に差のあることが確認されており，低品質モジュールを判別する一手法として一定の効果が期待される．ただし，偏差値メトリクスを用いた低品質モジュール予測手法は，十分な予測精度を得ることができず，低品質モジュール特定の自動化を実現することは未だ難しいと考える．原因の一つとして，多様なソフトウェア開発プロジェクトのデータから標準値域を作成したことが考えられる．工数予測の研究においては，モデル構築に使用するプロジェクトを選別することも試みられている[Kakimoto]．偏差値メトリクスの作成においても，規模やドメインなどに基づいて対象プロジェクトを層別することで，予測精度の向上が期待される．一般に，層別すると，メトリクス作成に使用できるデータセット数は少なくなるので，GitHub などの大規模プロジェクト管理サービスから，ドメインや規模などが同じと見なせるデータセットを自動抽出するシステムの開発が必要となる．

《研究目標 6「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価」》

スナップショット内のデータについて，その量的属性や関係を可視化できることをプロトタイプの実装を通じで示すことができた．プロジェクト全体の俯瞰，プロジェクト構成要素（間）の傾向や関係性の顕在化・評価，仮説の生成・検証といった，ソフトウェア品質の第三者評価で必要となる作業の効率化に貢献する度合いを，より多様なデータとシナリオのもとに明らかにする．また，プロトタイプ実装の開発を続け，解析・可視化機能の強化，高度化を目指す．

解析機能の具体的な強化策としては，Treemap Forest に対するユーザ操作に基づいてデータベース問い合わせを生成するモジュールの改良があげられる．現状では，対象プロ

ジェクトにあわせてハードコーディングされている「データベース内に格納された要素間の関係」を外部ファイルから読み取るようにすることで，より多種多様なデータセットに対して，素早く適応できる状態にする．

可視化機能の具体的な強化策として，特定の矩形を選択した際に，対応するより詳細な情報を表示するデータティップ機能の実装が考えられる．また，Treemap Forest 上で可視化する量的属性を明示的に選択可能とすることで，特にデータ解析の熟練者にとって，仮説検証を容易にすることが考えられる．

4.2 研究成果の産業界への展開について

ソフトウェア品質の第三者評価はもとより，ソフトウェアの品質や生産性の向上を目指した技術の研究開発においては，その技術の妥当性や有用性を実証実験等により評価することが不可欠である．ただし，ソフトウェア，および，その開発・利用プロセスは多様であり,それらを広く対象とする実験等を行うことは容易でない.現実的には,(いくつかの)具体的な対象において実験を行い，得られた評価結果や知見を対象や目的に応じて解釈・援用することになる．

なお，研究成果の産業界への展開を考えるのであれば，そもそもその展開先である産業界（企業）において開発されているソフトウェアを研究対象とし，その企業における具体的な技術的課題の解決を試みるべきである，という考えもある．確かに，対象や目的が明確であるため，より具体的で実用性の高い結果が得られると思われる．しかし，その裏返しとして，対象についての情報が広く開示されなければ，評価結果や知見の解釈や援用は難しくなり，広く利用することはできない．

本委託研究における実証的評価は，限られた数のソフトウェアへの適用を通じて行われている．あらゆるソフトウェアをカバーするには決して十分な数とは言えないが，評価に用いたそれらソフトウェアは，いずれも，オープンソースソフトウェアかそれに準ずるソフトウェアである．ソースコードはもちろんのこと，開発データも公開され，容易に入手可能である．それら公開情報は，評価結果や知見の解釈や援用に大いに役立つと考える．加えて，開発した技術の多くについてはツール化やウェブサービス化を通じて産業界に展開予定である.産業界への展開に向けた研究目標それぞれの取組や現状は次の通りである.

《研究目標 1「欠陥の収束・発散プロセスをトピック別に評価するモデルの構築と実証的評価」》

欠陥トピックの自動特定技術について，ネットワークセキュリティの研究者も含むいくつかの研究者からコンタクトがあった．この技術は，ソフトウェア開発に関わらず，他の分野での欠陥自動分類に応用できると考えられるので，プロトタイプの公開を目指す．

《研究目標 2「ソフトウェア障害の波及度を定量的に評価する方式の設計と実証的評価」》

トピック別の波及度計測・表示を Web サービス化する予定である．そのサービスを OSS 開発プロジェクトに適用し，障害報告時の対応に関する改善点を指摘する．

《研究目標 3「非機能要件の自動評価方式の設計と実証的評価」》

成果を産業界に導入しやすくする工夫として，自動評価に不可欠なモデル構築用データを http://se-naist.jp/NFR/にて公開している．

《研究目標 4「多人数の開発作業を正当なデータに基づいて解析・可視化する方式の設計と実証的評価」》

産業界への成果導入を念頭に，作成したシステムのインストーラおよび導入手順について Taskpit の Web サイト（http://taskpit.jpn.org/）などで公開予定である．

《研究目標 5「低品質モジュールの構造的・時間的特性を解析・可視化する方式の設計と実証的評価」》

一般標準値域は，第三者評価における一つの基準となる．OSS 開発プロジェクトから得たソースコードの標準値域をウェブ上で公開予定である．なお，開発手法やソフトウェアそのものの変化とともに，標準値域も変化していくと考えられることから，公開する標準値域は適宜更新していく予定である．

《研究目標 6「ソフトウェアプロジェクトデータの量的属性の再帰的な解析・可視化方式の設計と実証的評価」》

Treemap Forest の実装には，Java の派生言語である Processing を用いており，Java からの直接利用，また，javascript 用ライブラリを利用すればウェブアプリ化も可能である．ただし，現状では，データの読み込み部をデータセットに合わせてカスタマイズする必要であり，ウェブアプリ化したとしても，その利用コストは小さくない．そこで，すこしでも利用コストを下げるため，先述のように，解析・可視化機能を強化した上で，Treemap Forest の利用やカスタマイズのデモ，そのソースコードや設計情報などもウェブ上で公開することを検討する．

先に述べたとおり，本委託研究で開発された技術，得られた評価結果や知見は，あらゆるソフトウェアにそのまま適用できるのもではない．ただし，単なるアイディアの提案に留まらず，オープンソースソフトウェアを利用してその実証的評価を行っている．ソフトウェア品質の第三者評価を検討，実施しようとするソフトウェア利用者，そして，ソフトウェア開発者自身にとっても，本報告書はひとつのリファレンスになると考えている．もちろん，ソースコードや開発データが公開され，容易に入手可能であるからといって，本報告書で示された技術，評価結果，そして，知見を容易に解釈，活用できるとは限らない．ソフトウェア利用者，開発者，そして産業界に望むのは，そうした解釈や活用によっておのおのの技術的課題の解決につながるのか否かを実際に試し，検討することである．そして，課題解決につながるのであれば，そうした試みをベストプラクティスとして広く公開し，課題解決につながらないのであれば，何が足りないのか，課題解決のためにどのようなアプローチが考えられるのか，等を研究者にフィードバックすることで，産学連携は次のステップに進むことになる．

また，より直接的なアプローチとしては，本委託研究で開発された技術を，研究者主導で，商用ソフトウェア開発に適用することが考えられる．商用ソフトウェアの開発データ

は，ほとんどの場合，公開されることがないので，共同研究契約などに基づき，企業から特別に提供を受ける必要がある．また，研究者やその所属組織が発注者となるソフトウェアの開発において，開発費の 10〜20%を開発データ収集費として上積みし，開発データを得る方法もある．本委託研究の研究責任者らは，どちらの方法も実施した経験を有しているが，いずれにしても，非常に限られた数の開発データしか入手できず，結果の有意性を示したり，一般化したりすることは，現実的には不可能に近い．

商用ソフトウェアの開発データを企業から得やすくする手法としては，データスクランブル技術の応用が考えられる．データスクランブルとは，アクセス権限を持たない者から重要なデータを保護するために，データ間の関係など，ある種の性質を保持したまま，データを意味不明な文字列などに変換する操作である．ソフトウェアの開発データに含まれる固有名詞だけでなく，数値データも変換することで，どの企業のどのプロジェクトの開発データかの特定を難しくすることができる．平均や分散に基づく数値データの正規化もその一種と言えるが，変換によって分析結果が変わってはいけないし，逆変換等によって元の数値データが容易に得られても困る．オリジナルとなる開発データを，分析目的毎に変換するといったことが必要になるかもしれないが，分析目的別に開発データが複製され，集積されることに不都合はないし，正規化，と捉えれば，より多くの開発データが比較可能となり，分析対象になるとも考えられる．技術開発はこれからであるが，産業界には，分析目的の明確化と優先順位付け，そして何よりも，オリジナルとなる開発データの幅広い提供を期待する．

参考文献

[Bernstein] M. Bernstein, J. Shrager, and T. Winograd, “Taskpose: Exploring Fluid Boundaries in an Associative Window Visualization,” *Proc. of* 21st *Annual* ACM *Symposium on User Interface Software and Technology* (UIST), pp.231-234, 2008.

[Bowring] J. Bowring, A. Orso, and M. J. Harrold, “Monitoring deployed software using software tomography,” *Proc. of* ACM *Workshop on Program Analysis for Software Tools and Engineering*, pp.2-9, 2002.

[Buse] R. P.L. Buse, and W. Weimer, “Learning a Metric for Code Readability,” IEEE *Transactions on Software Engineering*, Vol.36, No.4, pp.546-558, 2010.

[Chen] T.-H. Chen, M. Nagappan, E. Shihab, and A. E. Hassan, “An empirical study of dormant bugs,” *Proc. of* 11th *Working Conference on Mining Software Repositories* (MSR 2014), pp.82-91, 2014.

[Dhaliwal] T. Dhaliwal, F. Khomh, and Y. Zou: Classifying field crash reports for fixing bugs: A case study of Mozilla Firefox, *Proc. of International Conference on Software Maintenance* (ICSM 2011), pp.333-342 (2011).

[Dang] Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel, “ReBucket: a method for clustering duplicate crash re- ports based on call stack similarity,” *Proc. of International Conference on Software Engineering*, pp.1084-1093 (2012).

[Granitzer] M. Granitzer, A. S. Rath, M. Kroll, C. Seifert, D. Ipsmiller, D. Devaurs, N. Weber, and S. N. Lindstaedt, ”Machine Learning based Work Task Classification,” *Journal of Digital Information Management*, pp.306-313, 2009.

[Gegick] M. Gegick, P. Rotella, and T. Xie, “Identifying security bug reports via text mining: An industrial case study,” *Proc. of* 7th *Working Conference on Mining Software Repositories* (MSR 2010), pp.11-20, 2010.

[Graves] T. L. Graves,A. F. Karr, J. S. Marron, and H. Siy, “Predicting fault incidence using software change history,” IEEE *Transactions on Software Engineering*, Vol.26, pp.653-661, 2000.

[Hassan] A. E. Hassan, “Predicting faults using the complexity of code changes,” *Proc. of* 31st *International Conference on Software Engieering*, pp.78-88, 2009.

[Hata] 畑秀明, 松本健一, トピックを識別したバグ予測モデルの構築, ソフトウェアエンジニアリングシンポジウム 2013 論文集, pp.1-8, 2013.

[Herraiz] I. Herraiz, D. M. German, J. M. Gonzalez-Barahona, and G. Robles, “Towards a Simplification of the Bug Report form in Eclipse,” *Proc. of Internantional Working Conference on Mining Software Repositories*,

pp.145-148, 2008.

[IPA1] 情報処理推進機構, ソフトウェアの品質説明力強化のための制度フレームワークに関する提案（中間報告), 2011年9月.

[IPA2] 情報処理推進機構,「ソフトウェア品質説明力強化の普及・推進のための調査」報告書, 2013年2月.

[IPA3] 情報処理推進機構,ソフトウェア品質説明力強化に向けた実験報告書,2013年2月.

[IPA4] 情報処理推進機構,2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書, 2013年4月.

[IPA5] 情報処理推進機構,「タイムスタンプ技術に関する調査」報告書, 2004年4月.

[Kakimoto] 柿元健, 渡辺竜,“統合ソフトウェア開発データにおけるプロジェクト欠損率を基にしたコスト予測方法の検討”, ウィンターワークショップ2015・イン・宜野湾論文集, pp.33-34, 2015.

[Kamei] Y. Kamei, S. Matsumoto, A. Monden, K. Matsumoto, B. Adams, and A.E. Hassan,“Revisiting common bug prediction findings using effort-aware models,”*Proc. of International Conference on Software Maintenance* (ICSM 2010), pp.1-10, 2010.

[Kashiwa] Y. Kashiwa, H. Yoshiyuki, Y. Kukita, and M. Ohira,“A Pilot Study of Diversity in High Impact Bugs,”*Proc. of* 30*th International Conference on Software Maintenance and Evolution* (ICSME2014), pp.536-540, 2014.

[Khomh] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan: An Entropy Evaluation Approach for Triaging Field Crashes: A Case Study of Mozilla Firefox, *Proc. Working Conference on Reverse Engineering* (WCRE'11), pp.261-270 (2011).

[Kim] S. Kim,T. Zimmermann, E. J. Whitehead Jr, and A. Zeller,“Predicting Faults from Cached History,”*Proc. of* 29*th International Conference on Software Engieering*, pp.489-498, 2007.

[Marco] M. D'Ambros, M. Lanza, and R. Robbes,“Evaluating defect prediction approaches: a benchmark and an extensive comparison,”*Empirical Software Engineering*, Vol.17, Issue 4-5, pp.531-577, 2012.

[Mende] T. Mende and R. Koschke,“Revisiting the evaluation of defect prediction models,”*Proc. of International Conference on Predictor Models in Software Engineering* (PROMISE2009), pp.1-10, 2009.

[Mockus] A. Mockus, R. T. Fielding, and J. D. Herbsleb,“Two Case Studies of Open Source Software Development: Apache and Mozilla,” ACM *Transactions on Software Engineering Methodology*, Vol.11, No.3, pp.309-346, 2002.

[Monden] A. Monden, T. Hayashi, S. Shinoda, K. Shirai, J. Yoshida, M. Barker, and K. Matsumoto,“Assessing the cost effectiveness of fault prediction

in acceptance testing," IEEE *Transactions on Software Engineering*, Vol.39, No.10, pp.1345-1357, 2013.

[Nagappan] N. Nagappan, and T. Ball, "Use of relative code churn measures to predict system defect density," *Proc. of* 27th *International Conference on Software Engineering*, pp.284–292, 2005.

[Nistor] A. Nistor, T. Jiang, and L. Tan, "Discovering, reporting, and fixing performance bugs," *Proc. of* 10th *Working Conference on Mining Software Repositories* (MSR 2013), pp.237-246, 2013.

[Ohlsson] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," IEEE *Transactions on Software Engineering*, Vol.22, No.12, pp.886-894, 1996.

[Sarma] A. Sarma, Larry Maccherone, Patrick Wagstrom, and Jim Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," *Proc. of* 31st *International Conference on Software Engineering*, pp.22-33, 2009.

[Shihab] E. Shihab, A. Mockus, Y. Kamei, B. Adams, and A. E. Hassan, "Highimpact defects: A study of breakage and surprise defects," *Proc. of* 19th ACM SIGSOFT *Symposium and the* 13th *European Conference on Foundations of Software Engineering* (ESEC/FSE 2011), pp.300-310, 2011.

[Śliwerski] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" *Proc. of* 3rd *International Workshop on Mining Software Repositories* (MSR 2005), pp.1-5, 2005.

[Valdivia] H. Valdivia Garcia and E. Shihab, "Characterizing and predicting blocking bugs in open source projects," *Proc. of* 11th *Working Conference on Mining Software Repositories* (MSR 2014), pp.72-81, 2014.

[Yasuda] 保田勝通，ソフトウェア品質保証の考え方と実際，実践ソフトウェア開発工学シリーズ１３，日科技連，1995.