不具合予測に関するメトリクスについての研究論文の系統的レビュー

畑 秀明　水野 修　菊野 亨

近年，ソフトウェアリポジトリのデータマイニングを利用した不具合予測の研究が活発に行われている．不具合予測に関連してどのようなメトリクスが研究されているかを明らかにするため，本稿では系統的レビューを行った．2 つの論文誌と 5 つの国際会議において，2000 年から 2010 年に発表された文献のうち 63 編を調査した．研究されたメトリクスを分析するため，測定対象と測定に必要な履歴情報に基づき 8 つの領域に分類した．レビューの結果，主に 2005 年以降コードやプロセスの履歴に基づくメトリクスが研究されるようになっており，また 2008 年以降には新たに開発組織や地理的位置関係に関連したメトリクスが提案されていることが明らかになった．

This paper provides a systematic review of studies related to software fault prediction with a specific focus on types of metrics. The research questions on this paper are: what types of metrics are used for studies related to faults, and is there trend in proposal of fault-related metrics. The review uses 63 papers in 2 journals and 5 conference proceedings published in 2000–2010. According to the review results, we found that code-related and process-related historical metrics had been used mainly since 2005, and organization-related and geographic metrics had been used since 2008.

1　はじめに

ソフトウェア開発やメインテナンスにおいて，テストやレビューといった品質確保の活動は，実用的な時間とコストで効果的に行うことが期待される．このため，重点的に品質確保の工数をかけるべき，不具合の潜在が疑われるモジュールを早期に予測することが求められる．こうしたソフトウェアの不具合予測は，ソフトウェア工学における重要なテーマの 1 つであり，活発に研究が行われている．実際のソフトウェアプロジェクトへ不具合予測の技術を適用し，効果的であったとの報告も行われている [8]．この論文では，Microsoft 社内で不具合予測を行う CRANE と呼ばれるシステムを構築し，Windows Vista の開発とメインテナンスで活用した様子が報告されている．

本稿は，こういった不具合予測と関連する研究について系統的レビューを行う．系統的レビュー (systematic review, systematic literature review) は，リサーチクエスチョンに対して文献選択の規準を明記し，網羅的で再現性のある文献調査を行う手法であり，その有用性が報告されている [34][35][45] ．本稿でも，この手法で文献調査を行い，本研究分野における近年の動向を明らかにする．

2009 年に発表された Catal らの不具合予測研究の系統的レビューは，本分野における最初の系統的レビューである [5]．彼らは不具合予測に関する研究について 1990 年から 2007 年までの文献を調査し，どのようなデータセットや手法が利用され，どの粒度のメトリクスが使われているかを明らかにしている．その結果，近年パブリックなデータセットの使用が大幅に増加し，手法においては機械学習が増加していることを報告している．またメトリクスの粒度は，メソッ

A Systematic Review of Software Fault Prediction Studies and Related Techniques.

Hideaki Hata, Tohru Kikuno, 大阪大学大学院情報科学研究科, Dept. of Information Systems Engineering, Osaka University.

Osamu Mizuno, 京都工芸繊維大学大学院工芸科学研究科, Dept. of Information Science, Kyoto Institute of Technology.

コンピュータソフトウェア, Vol.29, No.1 (2012), pp.106–117.
[解説論文] 2011 年 4 月 15 日受付.

ドレベルやクラスレベルが大半であり，近年になってファイルレベルの増加が見られることを示している．

近年，アクセス可能なパブリックなデータセットが増加し，また WEKA [18] や R [67] といったデータマイニングや統計処理のツールが容易に利用できるようになり，リポジトリからのデータマイニングが注目されている [80]．こうした研究動向は不具合予測の分野にも見られ，ソースコードファイルの開発履歴が不具合予測に有用であることが知られている．Catal らの報告に見られる，ファイルレベルのメトリクスを利用した研究の増加も，リポジトリマイニングを活用した不具合予測の研究増加が要因に考えられる．

本稿では，リポジトリマイニングの観点から不具合予測についての研究を概観するため，以下のリサーチクエスチョンを設定して，系統的レビューを行う．

RQ1: どんなメトリクスが提案，利用されているか

RQ2: メトリクスの提案にどのような動向が見られるか

本レビューによって近年の文献を系統的に整理することで，どのようなメトリクスが実証的に調査されているかを明らかにすることを目指す．これは，実務者や研究者にとって，本分野の関連研究調査の参考になるとともに，研究者にとっては今後の研究の方針を考える際の参考にもなる．

近年の研究動向を概観するため，2000 年から 2010 年に発表された文献を調査対象とする．Catal らは主に不具合予測の手法についての調査であったのに対して，本稿はメトリクスについての調査である．そのため Catal らの調査では除外されていた，メトリクスと不具合との相関関係を調査した研究も本稿では対象に含める．これによって，不具合予測モデルを構築しない研究であっても，不具合との相関関係が議論されたメトリクスについての研究を含むことができる．

以降，まず 2 節でレビュー手法として文献選択や分析方法を説明し，3 節で得られた結果からリサーチクエスチョンを議論する．最後に 4 節で本稿をまとめる．

2 レビュー手法

2.1 文献選択

網羅的な文献調査としては，検索キーワードを設定した検索システムの実行による，自動的な文献検索が考えられる．しかしソフトウェア工学の分野においては，自動的な文献選択は手動の文献選択に比べ低品質であることが報告されている [35]．そこで本稿では，表 1 に示した論文誌と国際会議において 2000 年から 2010 年に発表された文献に限定して手動で探索を行う．発表された文献の総数は 2,557 編である．

文献は，タイトルと概要を確認して調査する．不具合予測と明示していない文献もあるので，単純なキーワード設定だけでは網羅が難しい．また，不具合予測のモデルを構築した研究に限定せず，不具合との相関関係を議論する文献も選択したい．そこで，まず適格規準によっておおまかに不具合に関連する文献を選択する．次に，その中で不具合予測や不具合との相関関係を調査する研究でない，対象外とする研究を明示して除外するという方針をとる．

まず，以下の適格規準 (inclusion criteria) のいずれかを満たすものを選択する．

IC1: fault，defect，bug，failure，error といった不具合に関連した用語をタイトルか概要に含む文献であること．

表 1 対象とする論文誌と国際会議

論文誌	IEEE Trans. on Softw. Eng. Empirical Softw. Eng.
国際会議	Proc. Int. Conf. on Softw. Eng. Proc. Joint Meeting of European Softw. Eng. Conf. and ACM SIGSOFT Symp. on Found. of Softw. Eng. Proc. ACM SIGSOFT Int. Symp. on Found. of Softw. Eng. Proc. IEEE Int. Conf. on Softw. Maintenance Proc. IEEE Work. Conf. on Mining Softw. Repositories

IC2: ソフトウェアの品質や信頼性を議論している文献であること．

次に，以下の除外規準 (exclusion criteria) を適用して最終的に選択する文献を決定する．

EC1: テスト，インスペクションで不具合を検出する研究を除外する．

EC2: 不具合修正コミットを特定するなど，リポジトリマイニングの技術に関する研究を除外する．

EC3: 不具合修正プロセスを議論，予測する研究を除外する．

EC4: 実際の不具合についての実証的な調査研究を除外する．

選択規準を明らかにした通り，リポジトリを使っていない不具合予測など，特定の不具合予測の研究を対象外にはしていない．**EC1** の除外規準は，テストやインスペクションの技術を主題とした研究を除外するものであり，不具合との相関関係を調査する際のメトリクスとして，テストやインスペクションの結果を用いる研究を除外するものではない．

2.2 分析方法

リサーチクエスチョン **RQ1**，**RQ2** に基づきメトリクスの調査を行う．リポジトリマイニングの観点から概観するため，提案・利用されたメトリクスを分類して分析する．従来の複雑度メトリクスとリポジトリマイニングによるメトリクスを区別する呼び方として,「コードメトリクスと変更メトリクス」[52],「プロダクトメトリクスとプロセスメトリクス」[26] などがある．

従来の複雑度メトリクスは対象の版のコードから測定するのに対して，リポジトリマイニングによるメトリクスは測定対象をコードのみに限定せず，また，これまでの履歴からも測定する．本稿では，詳細な調査を行うため測定する対象と測定に必要な履歴のそれぞれに以下の項目を設定しメトリクスを分類する．

測定する対象: コード，プロセス，開発組織，地理的位置関係

測定に必要な履歴: 対象の版，これまでの履歴

測定対象としては，コード以外に，変更に関するプロセス，人に関する開発組織，分散開発に関する地理的位置関係に分類する．文献 [79] では対象別に，プロダクトメトリクス，プロセスメトリクス，リソースメトリクスという分類を紹介している．本稿の分類はコードとプロセスがそれぞれプロダクトメトリクスとプロセスメトリクスにあたり，開発組織と地理的位置関係がリソースメトリクスにあたると考えられる．メトリクス提案の動向についての分析を分かり易くするため，リソースメトリクスの 2 つを分けて分析する．また，この分類は調査した文献から設定したものであり，新しい分類のメトリクスが今後提案されることもあり得る．

またそれぞれの対象について，メトリクスの測定に必要な履歴を，対象の版のみか，これまでの履歴を利用するかで分類する．以上の分類項目によって，メトリクスを 8 領域 (4×2) に分類して調査する．

2.3 妥当性への脅威

妥当性に影響を及ぼす可能性のある問題について Yin の分類に基づいて議論する [61][71].

Construct validity（操作が適切か）

文献選択は手動で行ったので漏れが生じる可能性は否定できないが，2.1 節に示した文献の選択基準に基づき細心の注意を払って文献を選択した．

Internal validity（因果関係についての統計的推測が適切か）

本稿では因果関係を議論していない．

External validity（得られた知見は一般化可能か）

選択する文献を表 1 に示した論文誌と国際会議に定めた．これらは，ソフトウェア工学全般，実証的ソフトウェア工学，メインテナンス，リポジトリマイニングの分野から重要度の高い論文誌と国際会議である．

Reliablity（再現性があるか）

文献の選択基準を明記したので，本稿の分析は再現可能である．

3 レビュー結果

3.1 選択した文献

選択した文献は，論文誌から 28 編，国際会議から

35 編で合計 63 編であった．図 1 に文献数の推移を示す．近年発表数が増加していることが分かる．

3.1.1 手法による分類

データマイニングと統計的なアプローチで不具合を議論する 63 編の文献を，利用している手法によって分類した結果を表 2 に示す．

クラス分類による不具合予測は，ソフトウェアモジュールを不具合が含まれそうか否かに分類する手法である．計測したメトリクスを説明変数とした予測モデルを構築し，不具合が含まれそうなソフトウェアモジュールを予測する．パターンによる不具合予測が特定の不具合の有無を議論するのに対して，クラス分類は特定の不具合に限定せず有無を予測する．本手法は，Catal らが調査対象とした不具合予測の手法である [5]．Catal らは，利用される予測モデルの割合やデータセットの分類などを行っている．本稿ではメトリクスに着目した分析を行う．

不具合との相関関係を調査する研究は，直接不具合を予測するものではないが不具合予測に有効なメトリクスを議論している．Catal らは本手法を分析対象に入れていなかったが，表 2 に見られるように多くの文献がある．本稿では，リサーチクエスチョンに基づき不具合予測に関連したメトリクスの調査を行うため，クラス分類による不具合予測と不具合との相関関係の調査で対象となったメトリクスを合わせて，3.2 節と 3.3 節で議論する．

パターンに分類した手法は，開発履歴から何らかのパターンを抽出し，そのパターンに基づいた不具合を予測する．クラス分類とは異なる手法ながらデータマイニングに基づく不具合予測といえる．本手法には，特定のリビジョンにおけるメソッド呼び出し系列の例外を検出するもの [25][41][42]，開発履歴におけるメソッド呼び出しの共起パターンから例外を特定するもの [44]，不具合修正履歴のコードパターンから不具合を予測するもの [31]，不具合修正履歴の行パターンを活用して静的解析ツールの出力から優先すべき警告を示すもの [30] があった．

Song らは相関ルールのマイニングから同時に起こりそうな不具合 (不具合アソシエーション) を予測する手法を提案している [65]．

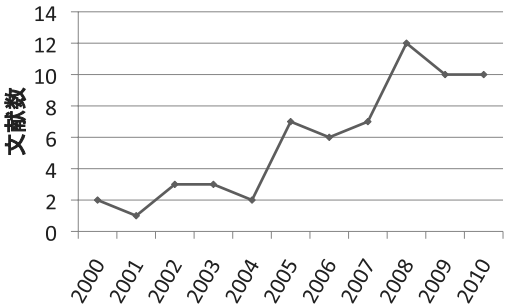

図 1 文献数の推移

表 2 利用している手法

手法	文献数
クラス分類による不具合予測	36
不具合との相関関係の調査	20
パターンによる不具合予測	6
不具合アソシエーション予測	1

本稿ではメトリクスに注目した分析を行うため，表 2 に示した 4 つの手法のうち，クラス分類と相関関係に該当する 56 編の文献を対象に，以降の分析を行う．

3.1.2 対象とする不具合の調査

不具合の表記には，fault，defect，bug，failure などがある．3.1.1 節で分析対象に定めた 56 編の文献が対象とした不具合は，以下のように 3 つに分類できる．

フォールト: 障害を引き起こすソースコードの一部．修正されることによって分かる．fault，defect，bug を含む．ただし，フォールトを対象とした文献においても，フォールトの有無を議論するもの以外に，単位サイズあたりのフォールト数 (フォールト密度) やフォールトに割り当てられた深刻度 (フォールト深刻度 [73]) を扱うものがある．フォールトの有無とフォールト密度をともに対象にした文献もある [10][24][57] ．主にフォールト密度を対象とした研究は，表 2 では不具合との相関関係の調査に含めた．

障害: フォールトに起因する障害 (failure)．リリース後に報告されることによって分かることから，障害を予測することはリリース後に残るフォール

表 3　対象とする不具合

不具合	文献数
フォールト	49
障害	6
ビルドエラー	1

トを予測することを意味する．

ビルドエラー: ビルドの失敗 [70].

不具合別にその不具合を対象とする文献数を表 3 にまとめた．1 つの文献で複数の不具合を対象としたものもある．多くの研究はフォールトを対象としているが，他にも異なる不具合を対象とした研究があることが分かる．不具合の種類によって分析に必要なデータが異なる．ただし用語を混同しているケースがあることが指摘されている [55].

3.2　メトリクスの種類

研究されたメトリクスを分類してまとめた結果を表 4 に示す．またそれぞれのメトリクスを提案，利用している文献番号も記す．メトリクスの 8 つの分類に，それぞれ領域 (1) から領域 (8) の領域番号を付す．表 4 に示したように様々なメトリクスが研究されていることが分かる．以降では，利用されることの多いメトリクスに関連する主要な文献を紹介する．

1. **Graves らの研究** [16]**：履歴を活用し，領域 (2)，(4)，(5)，(6) のメトリクスを利用した初期の文献．**
   Graves らは計測メトリクスとして，過去のフォールト数，変更回数，存在期間，開発した組織，開発者数などを利用している [16]．電話システムを対象としたケーススタディから，これらのメトリクスが従来の複雑度メトリクスと比べて不具合予測により有効であることを報告している．
2. **Nagappan らの研究** [54]**：領域 (2)．**
   Nagappan らは，変更行数に関連したメトリクスで不具合予測を行っている [54]．基本的なメトリクスとして，*Churned LOC*(最初のバージョンからの追加・修正行数の合計)，*Deleted LOC*(最初のバージョンからの削除行数の合計)，*Total LOC* (対象バージョンの行数) などを計測し，$ChurnedLOC/TotalLOC$ や $DeletedLOC/TotalLOC$ の値を説明変数とした予測モデルを構築し，Windows Server 2003 でのケーススタディから有用性を報告している．変更行数に関連したメトリクスは，表 4 に示すように，以降も基本的なメトリクスとして多くの文献で利用されている．
3. **Hassan らの研究** [20]**：領域 (4)．**
   Hassan らは，変更頻度，不具合修正頻度，変更後の経過日時，不具合修正後の経過日時といったプロセスの履歴情報を用いた不具合予測手法を提案している [20]．本手法は，計測が容易なメトリクスを用い，直ちに不具合予測ができるシステムで，特に早期の不具合予測を行うことを意図している．また前述の 2 つの研究が商用のソフトウェアを対象としたのに対して，本研究は 6 つの大規模なオープンソースソフトウェアを対象としている．コードの存在期間，変更回数，不具合修正回数のメトリクスも多くの研究で利用される基本的なメトリクスとなっている．
4. **Mizuno らの研究** [48]**：領域 (1) の新たなメトリクス．**
   Mizuno らは，ソースコードをテキストと見なし，スパムフィルタリングの技術で不具合有り，不具合無しに分類する手法を提案している [48]．本手法はソースコード中にある特定のトークンの出現回数をメトリクスとしたベイズ分類器である．トークンメトリクスはこれまでのメトリクスと比べ，計測するメトリクスの種類が非常に多くなり解釈が難しくなるが，テキスト分類の技術を活用できるというメリットがあり，また不具合予測精度も同等以上と報告されている．予測モデルとしては，サポートベクトルマシン [32] やロジスティック回帰 [22] が提案されている．
5. **Moser ら** [52]**，Kamei ら** [26] **の研究：領域 (1) の従来のメトリクスと領域 (2)，(4) や (6) のメトリクスとの比較．**
   コードの対象の版から計測する従来のメトリクスとリポジトリマイニングを活用したメトリクスとの比較調査も行われている [26][52]．いずれ

表 4 提案・利用されたメトリクスの分類

	対象の版	これまでの履歴
コード	**領域 (1)** 行数 [4][10][14][24][26]–[29][32][37]–[40][43][47][55][57][68][69][72] オペレータ数 [9][10][26]–[29][37][40][43][47][55][68] McCabe メトリクス [26][32][40][43][55][58][68][75] Halstead メトリクス [10][40][43][47][68] CK メトリクス [9][12][17][24][26][63][66][73] メソッド呼び出し関係 [55][64] リユースコード [51] 静的解析ツールの出力 [53][62] ディレクトリ [32] 横断的関心事 [11][15] コーディング規約違反 [3] コードクローン [60] トークン [21][22][32][48][49] 依存グラフ [74]	**領域 (2)** 変更行数 [9][26][32][36][46][52][54][62][75] メソッド呼び出し関係の履歴 [64] 静的解析ツール出力履歴 [62]
プロセス	**領域 (3)** プログラミング言語 [57][69] コミットログ [32] コミット日時 [32]	**領域 (4)** 過去のフォールト [13][16][36][57][75] コードの存在期間 [9][16][20][26][33][57][62][69] 変更回数 [9][16][20][26][33][46][52][56][57][59][69] 不具合修正回数 [9][20][26][33][37][52][69] リファクタリング回数 [9][26][52] ロジカルカップリング [33][50][52] 変更の複雑度 [9][19][50] テストケース数 [36]
開発組織	**領域 (5)** 開発した組織 [16] 開発者 [32]	**領域 (6)** 開発者数 [9][16][46][52][56][59][69][75] 新規・脱退開発者数 [50][56][69] 開発者とコードの関係 [46][56][59][70] 開発者間の関係 [6][46][50][56][59][70] 開発した組織数 [56]
地理的位置関係	**領域 (7)** 対象の版が開発された位置に関するメトリクスが考えられるが，利用した文献はない	**領域 (8)** 関わった開発者の位置 [1][46][50]

の文献も変更回数や変更行数といった履歴情報に基づくメトリクスが従来からのメトリクスと比べて有用であると報告している．

6. **Nagappan らの研究** [56]**：主に領域 (6)．**
Nagappan らは，「ソフトウェアの構造は開発する組織を反映する」という Conway の法則 [7] を実証的に分析するため，以下に示す組織メトリクスを提案している [56]．
   - 開発者数
   - 脱退した開発者数
   - 開発した組織数
   - オーナーシップの組織階層
   - 組織の凝集度
   - 開発者の凝集度
   - 組織の編集割合

   Windows Vista を対象とした適用実験から，変更行数メトリクスや複雑度メトリクスと比べても組織メトリクスは高い予測精度が得られることを報告している．

7. **ネットワークメトリクス** [46][59][70][74]**：領域 (1) または領域 (6)．**
コード間や開発者間のネットワークに関するメトリクスの不具合予測への適用が，同時期に複数の文献で提案されている [46][59][70][74]．Zimmermann らは，ソフトウェアモジュール間のデータ依存やメソッド呼び出しのグラフに対して連

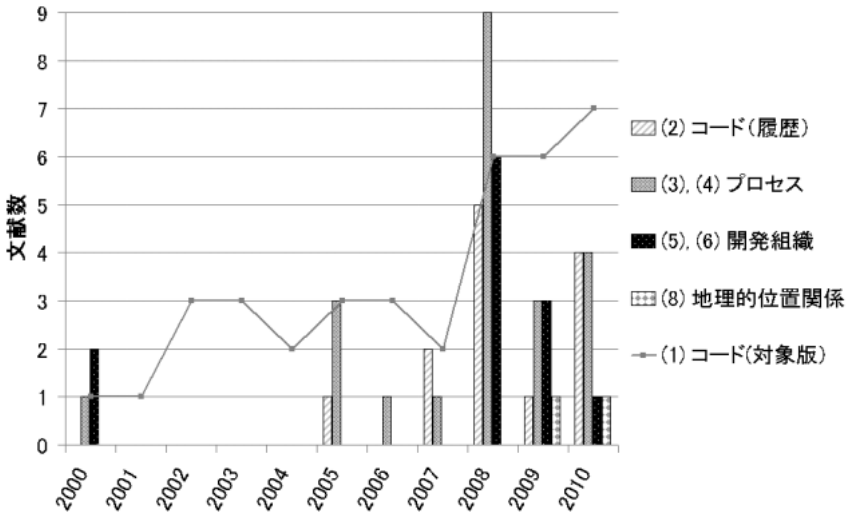

図 2　研究対象となるメトリクス領域の推移

結性や近接性に関するメトリクスを提案している [74]．Windows Server 2003 への適用実験で複雑度メトリクスと比べて高いフォールト予測精度が得られたことを報告している．また同様にネットワークに関連したメトリクスを開発者ネットワークに対して提案し，障害 [46][59] やビルドエラーの予測 [70] を行った文献もある．

8. **Bird らの研究** [1]**：領域 (8)．**
地理的に分散した開発と不具合への調査も行われている [1][50]．Bird らは，ビル，カフェテリア (ビルが異なっても一緒に食事が可能な範囲)，キャンパス，都市，大陸，世界を開発の地理的位置として分類し，位置の違いと障害との相関を調査している [1]．Windows Vista でのケーススタディではこれらに相関が見られないことを報告している．一方，電話システムでの調査から，Mockus は分散開発がソフトウェアの品質に影響すると報告している [50]．

3.3　時系列の推移

メトリクス提案の動向を分析するため，表 4 に示す 8 つの領域に含まれるメトリクスを研究した文献数を 1 年ごとにまとめた結果を図 2 に示す．1 つの文献で複数のメトリクス領域を利用している場合は，各領域ごとに文献数をカウントしている．領域 (3) と (4)，(5) と (6) はそれぞれ合算した．領域 (1) は継続して利用されているので折れ線グラフで推移を表し，他の領域は棒グラフで表した．

不具合予測に関連した研究の動向に関して，発表文献数の動向，リポジトリマイニングの観点からのメトリクスの動向，新しく提案されたメトリクスの動向に着目して分析を行う．新しく提案されたメトリクスに関しては，領域 (1) とその他の新領域とを分けて議論する．

発表文献数の動向　発表文献数の推移は図 1 に見られる．図 1 から，2005 年以降文献数が増加していることが分かる．Catal らは，2005 年以降にパブリックなデータの利用が増えていることを報告している [5]．要因の 1 つとして，再利用可能な実験データを共有する PROMISE リポジトリ [2] が 2005 年以降に運営されていることを挙げている．このような 2 次利用可能なデータやオープンソースソフトウェアからのデータ取得が容易になることにより，実証的なデータを利用した研究が増加したと考えられる．

リポジトリマイニングの観点からの動向　領域 (2) から (8) は，主にリポジトリマイニングに基づくメトリクスである．これらのメトリクスは特に

2005 年以降によく研究されていることが，図 2 から分かる．リポジトリから得られる履歴情報に基づくメトリクスは 2000 年に Graves らが提案し，電話システムに対してケーススタディを行っている [16]．2005 年以降，複数のオープンソースソフトウェアへの適用に用いられ，広くその有用性が認められたと考えられる．

領域 (1) の新しいメトリクス 対象の版のコードから得られるメトリクスである領域 (1) は，表 4 に見られるように最もメトリクスの種類が多い．図 2 に示すように，これらのメトリクスを利用した文献は毎年発表されていることが分かる．詳しく見ると，行数メトリクスのように継続して計測されるメトリクス以外に，静的解析ツールの出力やコードクローン，トークン，依存グラフのネットワークメトリクスなどのように，新しく様々なメトリクスが提案されている．

新領域の新しいメトリクス コードの履歴情報やプロセスに関する領域 (2)，(3)，(4) は 2005 年以降に多くの研究で利用されている．また，開発組織に関連する領域 (5) と (6) のメトリクスは 2008 年以降に研究されていることが分かる．さらに地理的位置関係に関連した新しいメトリクスの研究が 2009 年から行われている．

4 まとめ

本稿では，リポジトリマイニングの観点から近年の不具合予測と関連研究の調査を行った．どんなメトリクスが提案，利用されているか (**RQ1**)，またメトリクスの提案にどのような動向が見られるか (**RQ2**) というリサーチクエスチョンを設定して系統的レビューを行った．2 つの論文誌と 5 つの国際会議において，2000 年から 2010 年に発表された文献から 63 編を選択し，分析した．

利用されるメトリクスを年ごとに調査した結果，2005 年以降にコードの履歴情報やプロセスに関するメトリクスが広く使われるようになっていることが分かった．従来の複雑度メトリクスに基づく予測モデルと比べても高い予測精度が得られるという報告もあり，有用性が認知され幅広く利用されている．

また 2008 年以降は開発組織や地理的位置関係に関するメトリクスが研究されるようになっていることを明らかにした．開発組織に関するメトリクスも複雑度メトリクスと比べても有用であることが複数の文献で報告されている．ただし，分散開発の品質への影響を考慮した地理的位置関係に関するメトリクスについては，不具合との相関が見られないという文献，不具合予測に有用だと主張する文献もあり，さらなるケーススタディが必要と思われる．

最後に，今後の不具合予測に関連して研究が求められる点を挙げる．

- **新たなメトリクスの提案．** 本稿の調査で見られたように不具合予測に有用なメトリクスが新たに提案されてきた．測定対象として考えられるものとしては，コメント文記述 [76]–[78] を含むドキュメントや版管理されたテストなどが考えられる．
- **細粒度なモジュールでの履歴情報取得．** 履歴に基づくメトリクスは版管理リポジトリからデータを取得するため，主にファイルレベルのメトリクスとなり，不具合予測を行う対象モジュールの粒度もファイルレベルとなる．関数レベルなどの細粒度な履歴に基づくメトリクスによる細粒度な不具合予測 [23] が有用と考えられる．
- **多くのソフトウェアを対象としたケーススタディ．** 開発組織や地理的位置関係に関するメトリクスなどの，新たに提案されたメトリクスは適用対象が少数の商用のソフトウェアのみというものも多い．オープンソースソフトウェアを含む，多くのソフトウェアを対象としたケーススタディを行うことで実証的に有用性を調査する必要がある．

謝辞 本論文の初版に対して有益なコメントをいただいた査読者の皆様に感謝する．本研究は，日本学術振興会 科学技術研究費補助金特別研究員奨励費 (課題番号：23・4335) の助成を受けた．

参 考 文 献

[1] Bird, C., Nagappan, N., Devanbu, P., Gall, H. and Murphy, B.: Does distributed development affect software quality? An empirical case study of Windows Vista, in *Proc. of 31st Int. Conf. on*

Softw. Eng. (ICSE '09), IEEE Computer Society, 2009, pp. 518–528.

[2] Boetticher, G., Menzies, T. and Ostrand, T.: PROMISE Repository of empirical software engineering data, http://promisedata.org/ repository, West Virginia University, Department of Computer Science, 2007.

[3] Boogerd, C. and Moonen, L.: Evaluating the relation between coding standard violations and faultswithin and across software versions, in *Proc. of 6th IEEE Work. Conf. on Mining Softw. Repositories* (*MSR '09*), 2009, IEEE Computer Society, pp. 41–50.

[4] Briand, L. C., Melo, W. L. and Wüst, J.: Assessing the applicability of fault-proneness models across object-oriented software projects, *IEEE Trans. Softw. Eng.*, Vol. 28, No. 7(2002), pp. 706–720.

[5] Catal, C. and Diri, B.: Review: A systematic review of software fault prediction studies, *Expert Syst. Appl.*, Vol. 36, No. 4(2009), pp. 7346–7354.

[6] Cataldo, M., Mockus, A., Roberts, J. A. and Herbsleb, J. D.: Software Dependencies, Work Dependencies, and Their Impact on Failures, *IEEE Trans. Softw. Eng.*, Vol. 35, No. 6(2009), pp. 864–878.

[7] Conway, M.: How do committees invent, *Datamation magazine*, Vol. 14, No. 4(1968), pp. 28–31.

[8] Czerwonka, J., Das, R., Nagappan, N., Tarvo, A. and Teterev, A.: CRANE: Failure Prediction, Change Analysis and Test Prioritization in Practice – Experiences from Windows, in *Proc. of 4th IEEE Int. Conf. on Softw. Testing, Verification and Validation* (*ICST '11*), IEEE Computer Society, 2011, pp. 357–366.

[9] D'Ambros, M., Lanza, M. and Robbes, R.: An extensive comparison of bug prediction approaches, in *Proc. of 7th IEEE Work. Conf. on Mining Softw. Repositories* (*MSR '10*), IEEE Computer Society, 2010, pp. 31–41.

[10] Denaro, G. and Pezzè, M.: An empirical evaluation of fault-proneness models, in *Proc. of 24th Int. Conf. on Softw. Eng.* (*ICSE '02*), ACM, 2002, pp. 241–251.

[11] Eaddy, M., Zimmermann, T., Sherwood, K. D., Garg, V., Murphy, G. C., Nagappan, N. and Aho, A. V.: Do Crosscutting Concerns Cause Defects?, *IEEE Trans. Softw. Eng.*, Vol. 34, No. 4(2008), pp. 497–515.

[12] El Emam, K., Benlarbi, S., Goel, N. and Rai, S. N.: The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics, *IEEE Trans. Softw. Eng.*, Vol. 27(2001), pp. 630–650.

[13] Fenton, N., Neil, M., Marsh, W., Hearty, P., Radliński, L. and Krause, P.: On the effectiveness of early life cycle defect prediction with Bayesian Nets, *Empirical Softw. Eng.*, Vol. 13, No. 5(2008), pp. 499–537.

[14] Fenton, N. E. and Ohlsson, N.: Quantitative Analysis of Faults and Failures in a Complex Software System, *IEEE Trans. Softw. Eng.*, Vol. 26, No. 8(2000), pp. 797–814.

[15] Ferrari, F., Burrows, R., Lemos, O., Garcia, A., Figueiredo, E., Cacho, N., Lopes, F., Temudo, N., Silva, L., Soares, S., Rashid, A., Masiero, P., Batista, T. and Maldonado, J.: An exploratory study of fault-proneness in evolving aspect-oriented programs, in *Proc. of 32nd Int. Conf. on Softw. Eng.* (*ICSE '10*), ACM, 2010, pp. 65–74.

[16] Graves, T. L., Karr, A. F., Marron, J. S. and Siy, H.: Predicting Fault Incidence Using Software Change History, *IEEE Trans. Softw. Eng.*, Vol. 26, No. 7(2000), pp. 653–661.

[17] Gyimothy, T., Ferenc, R. and Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction, *IEEE Trans. Softw. Eng.*, Vol. 31, No. 10(2005), pp. 897–910.

[18] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H.: The WEKA data mining software: an update, *SIGKDD Explor. Newsl.*, Vol. 11, No. 1(2009), pp. 10–18.

[19] Hassan, A. E.: Predicting faults using the complexity of code changes, in *Proc. of 31st Int. Conf. on Softw. Eng.* (*ICSE '09*), IEEE Computer Society, 2009, pp. 78–88.

[20] Hassan, A. E. and Holt, R. C.: The Top Ten List: Dynamic Fault Prediction, in *Proc. of 21st IEEE Int. Conf. on Softw. Maintenance* (*ICSM '05*), IEEE Computer Society, 2005, pp. 263–272.

[21] Hata, H., Mizuno, O. and Kikuno, T.: An extension of fault-prone filtering using precise training and a dynamic threshold, in *Proc. of 5th Work. Conf. on Mining Softw. Repositories* (*MSR '08*), ACM, 2008, pp. 89–98.

[22] Hata, H., Mizuno, O. and Kikuno, T.: Fault-prone module detection using large-scale text features based on spam filtering, *Empirical Softw. Eng.*, Vol. 15, No. 2(2010), pp. 147–165.

[23] Hata, H., Mizuno, O. and Kikuno, T.: Reconstructing Fine-Grained Versioning Repositories with Git for Method-Level Bug Prediction, in *Proc. of 2nd Int. Workshop on Empirical Softw. Eng. in Practice* (*IWESEP '10*), 2010, pp. 27–32.

[24] J. Pai, G. and Bechta Dugan, J.: Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods, *IEEE Trans. Softw. Eng.*, Vol. 33, No. 10(2007), pp. 675–686.

[25] Jiang, L., Su, Z. and Chiu, E.: Context-based detection of clone-related bugs, in *Proc. of 6th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.* (*ESEC-FSE '07*), ACM, 2007, pp. 55–64.

[26] Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K., Adams, B. and Hassan, A. E.: Revisiting common bug prediction findings using effort-aware models, in *Proc. of 26th IEEE Int. Conf. on*

Softw. Maintenance (*ICSM '10*), IEEE Computer Society, 2010, pp. 1–10.

[27] Khoshgoftaar, T. M. and Seliya, N.: Fault Prediction Modeling for Software Quality Estimation: Comparing Commonly Used Techniques, *Empirical Softw. Eng.*, Vol. 8, No. 3(2003), pp. 255–283.

[28] Khoshgoftaar, T. M. and Seliya, N.: Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study, *Empirical Softw. Eng.*, Vol. 9, No. 3(2004), pp. 229–257.

[29] Khoshgoftaar, T. M., Yuan, X., Allen, E. B., Jones, W. D. and Hudepohl, J. P.: Uncertain Classification of Fault-Prone Software Modules, *Empirical Softw. Eng.*, Vol. 7, No. 4(2002), pp. 297–318.

[30] Kim, S. and Ernst, M. D.: Which warnings should I fix first?, in *Proc. of 6th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.* (*ESEC-FSE '07*), ACM, 2007, pp. 45–54.

[31] Kim, S., Pan, K. and Whitehead, Jr., E. E. J.: Memories of bug fixes, in *Proc. of 14th ACM SIGSOFT Int. Symp. on Found. of Softw. Eng.* (*SIGSOFT '06/FSE-14*), ACM, 2006, pp. 35–45.

[32] Kim, S., Whitehead, Jr., E. J. and Zhang, Y.: Classifying Software Changes: Clean or Buggy?, *IEEE Trans. Softw. Eng.*, Vol. 34, No. 2(2008), pp. 181–196.

[33] Kim, S., Zimmermann, T., Whitehead Jr., E. J. and Zeller, A.: Predicting Faults from Cached History, in *Proc. of 29th Int. Conf. on Softw. Eng.* (*ICSE '07*), IEEE Computer Society, 2007, pp. 489–498.

[34] Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J. and Linkman, S.: Systematic literature reviews in software engineering - A systematic literature review, *Inf. Softw. Technol.*, Vol. 51, No. 1(2009), pp. 7–15.

[35] Kitchenham, B. A., Brereton, P., Turner, M., Niazi, M. K., Linkman, S., Pretorius, R. and Budgen, D.: Refining the systematic literature review process–two participant-observer case studies, *Empirical Softw. Eng.*, Vol. 15, No. 6(2010), pp. 618–653.

[36] Kläs, M., Elberzhager, F., Münch, J., Hartjes, K. and von Graevemeyer, O.: Transparent combination of expert and measurement data for defect prediction: an industrial case study, in *Proc. of 32nd Int. Conf. on Softw. Eng.* (*ICSE '10*), ACM, 2010, pp. 119–128.

[37] Knab, P., Pinzger, M. and Bernstein, A.: Predicting defect densities in source code files with decision tree learners, *Proc. of 3rd Int. Workshop on Mining Softw. Repositories* (*MSR '06*), ACM, 2006, pp. 119–125.

[38] Koru, A. G., Zhang, D., El Emam, K. and Liu, H.: An Investigation into the Functional Form of the Size-Defect Relationship for Software Modules, *IEEE Trans. Softw. Eng.*, Vol. 35, No. 2(2009), pp. 293–304.

[39] Koru, G., Liu, H., Zhang, D. and Emam, K.: Testing the theory of relative defect proneness for closed-source software, *Empirical Softw. Eng.*, Vol. 15, No. 6(2010), pp. 577–598.

[40] Lessmann, S., Baesens, B., Mues, C. and Pietsch, S.: Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings, *IEEE Trans. Softw. Eng.*, Vol. 34, No. 4(2008), pp. 485–496.

[41] Li, Z., Lu, S., Myagmar, S. and Zhou, Y.: CP-Miner: Finding Copy-Paste and Related Bugs in Large-Scale Software Code, *IEEE Trans. Softw. Eng.*, Vol. 32, No. 3(2006), pp. 176–192.

[42] Li, Z. and Zhou, Y.: PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code, in *Proc. of 5th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.* (*ESEC/FSE-13*), ACM, 2005, pp. 306–315.

[43] Liu, Y., Khoshgoftaar, T. M. and Seliya, N.: Evolutionary Optimization of Software Quality Modeling with Multiple Repositories, *IEEE Trans. Softw. Eng.*, Vol. 36, No. 6(2010), pp. 852–864.

[44] Livshits, B. and Zimmermann, T.: DynaMine: finding common error patterns by mining software revision histories, in *Proc. of 5th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.* (*ESEC/FSE-13*), ACM, 2005, pp. 296–305.

[45] MacDonell, S., Shepperd, M., Kitchenham, B. and Mendes, E.: How Reliable Are Systematic Reviews in Empirical Software Engineering?, *IEEE Trans. Softw. Eng.*, Vol. 36, No.5(2010), pp. 676–687.

[46] Meneely, A., Williams, L., Snipes, W. and Osborne, J.: Predicting failures with developer networks and social network analysis, in *Proc. of 16th ACM SIGSOFT Int. Symp. on Found. of Softw. Eng.* (*SIGSOFT '08/FSE-16*), ACM, 2008, pp. 13–23.

[47] Menzies, T., Greenwald, J. and Frank, A.: Data Mining Static Code Attributes to Learn Defect Predictors, *IEEE Trans. Softw. Eng.*, Vol. 33, No. 1(2007), pp. 2–13.

[48] Mizuno, O., Ikami, S., Nakaichi, S. and Kikuno, T.: Spam Filter Based Approach for Finding Fault-Prone Software Modules, in *Proc. of 4th Int. Workshop on Mining Softw. Repositories* (*MSR '07*), IEEE Computer Society, 2007, No. 4.

[49] Mizuno, O. and Kikuno, T.: Training on errors experiment to detect fault-prone software modules by spam filter, in *Proc. of 6th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.* (*ESEC-FSE '07*), ACM, 2007, pp. 405–414.

[50] Mockus, A.: Organizational volatility and its ef-

fects on software defects, in *Proc. of 18th ACM SIGSOFT Int. Symp. on Found. of Softw. Eng.* (*FSE '10*), ACM, 2010, pp. 117–126.

[51] Mohagheghi, P., Conradi, R., Killi, O. M. and Schwarz, H.: An Empirical Study of Software Reuse vs. Defect-Density and Stability, in *Proc. of 26th Int. Conf. on Softw. Eng.* (*ICSE '04*), IEEE Computer Society, 2004, pp. 282–292.

[52] Moser, R., Pedrycz, W. and Succi, G.: A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, in *Proc. of 30th Int. Conf. on Softw. Eng.* (*ICSE '08*), ACM, 2008, pp. 181–190.

[53] Nagappan, N. and Ball, T.: Static analysis tools as early indicators of pre-release defect density, in *Proc. of 27th Int. Conf. on Softw. Eng.* (*ICSE '05*), ACM, 2005, pp. 580–586.

[54] Nagappan, N. and Ball, T.: Use of relative code churn measures to predict system defect density, in *Proc. of 27th Int. Conf. on Softw. Eng.* (*ICSE '05*), ACM, 2005, pp. 284–292.

[55] Nagappan, N., Ball, T. and Zeller, A.: Mining metrics to predict component failures, in *Proc. of 28th Int. Conf. on Softw. Eng.* (*ICSE '06*), ACM, 2006, pp. 452–461.

[56] Nagappan, N., Murphy, B. and Basili, V.: The influence of organizational structure on software quality: an empirical case study, in *Proc. of 30th Int. Conf. on Softw. Eng.* (*ICSE '08*), ACM, 2008, pp. 521–530.

[57] Ostrand, T. J., Weyuker, E. J. and Bell, R. M.: Predicting the Location and Number of Faults in Large Software Systems, *IEEE Trans. Softw. Eng.*, Vol. 31, No. 4(2005), pp. 340–355.

[58] Pighin, M., Podgorelec, V. and Kokol, P.: Fault-Threshold Prediction with Linear Programming Methodologies, *Empirical Softw. Eng.*, Vol. 8, No. 2(2003), pp. 117–138.

[59] Pinzger, M., Nagappan, N. and Murphy, B.: Can developer-module networks predict failures?, in *Proc. of 16th ACM SIGSOFT Int. Symp. on Found. of Softw. Eng.* (*SIGSOFT '08/FSE-16*), ACM, 2008, pp. 2–12.

[60] Rahman, F., Bird, C. and Devanbu, P.: Clones: What is that smell?, in *Proc. of 7th IEEE Work. Conf. on Mining Softw. Repositories* (*MSR '10*), IEEE Computer Society, 2010, pp. 72–81.

[61] Runeson, P. and Höst, M.: Guidelines for conducting and reporting case study research in software engineering, *Empirical Softw. Eng.*, Vol. 14, No. 2(2009), pp. 131–164.

[62] Ruthruff, J. R., Penix, J., Morgenthaler, J. D., Elbaum, S. and Rothermel, G.: Predicting accurate and actionable static analysis warnings: an experimental approach, in *Proc. of 30th Int. Conf. on Softw. Eng.* (*ICSE '08*), ACM, 2008, pp. 341–350.

[63] Shatnawi, R.: A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems, *IEEE Trans. Softw. Eng.*, Vol. 36, No. 2(2010), pp. 216–225.

[64] Shin, Y., Bell, R., Ostrand, T. and Weyuker, E.: Does calling structure information improve the accuracy of fault prediction?, in *Proc. of 6th IEEE Work. Conf. on Mining Softw. Repositories* (*MSR '09*), IEEE Computer Society, 2009, pp. 61–70.

[65] Song, Q., Shepperd, M., Cartwright, M. and Mair, C.: Software Defect Association Mining and Defect Correction Effort Prediction, *IEEE Trans. Softw. Eng.*, Vol. 32, No. 2(2006), pp. 69–82.

[66] Subramanyam, R. and Krishnan, M. S.: Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects, *IEEE Trans. Softw. Eng.*, Vol. 29, No. 4(2003), pp. 297–310.

[67] The R Project for Statistical Computing: R, http://www.r-project.org/.

[68] Turhan, B., Menzies, T., Bener, A. B. and Di Stefano, J.: On the relative value of cross-company and within-company data for defect prediction, *Empirical Softw. Eng.*, Vol. 14, No. 5(2009), pp. 540–578.

[69] Weyuker, E. J., Ostrand, T. J. and Bell, R. M.: Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models, *Empirical Softw. Eng.*, Vol. 13, No. 5(2008), pp. 539–559.

[70] Wolf, T., Schroter, A., Damian, D. and Nguyen, T.: Predicting build failures using social network analysis on developer communication, in *Proc. of 31st Int. Conf. on Softw. Eng.* (*ICSE '09*), IEEE Computer Society, 2009, pp. 1–11.

[71] Yin, R. K.: *Case Study Research: Design and Methods*, Applied Social Research Methods, Sage Publications, 4th edition, 2008.

[72] Zhang, H.: An investigation of the relationships between lines of code and defects, in *Proc. of 25th IEEE Int. Conf. on Softw. Maintenance* (*ICSM '09*), 2009, pp. 274–283.

[73] Zhou, Y. and Leung, H.: Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults, *IEEE Trans. Softw. Eng.*, Vol. 32, No. 10(2006), pp. 771–789.

[74] Zimmermann, T. and Nagappan, N.: Predicting defects using network analysis on dependency graphs, in *Proc. of 30th Int. Conf. on Softw. Eng.* (*ICSE '08*), ACM, 2008, pp. 531–540.

[75] Zimmermann, T., Nagappan, N., Gall, H., Giger, E. and Murphy, B.: Cross-project defect prediction: a large scale experiment on data vs. domain vs. process, in *Proc. of 7th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.* (*ESEC/FSE '09*), ACM, 2009, pp. 91–100.

[76] 平田幸直, 水野修: テキスト分類に基づく Fault-prone モジュール検出法におけるコメント行の影響の分析, 情報処理学会研究報告 ソフトウェア工学 (SE),

2010, pp. 1–8.
[77] 畑秀明, 水野修, 菊野亨: メソッドに対するコメント文記述の変更履歴とメソッドの不具合との関係に関する実証的考察, 電子情報通信学会技術研究報告, Vol. IEICE-110, No. IEICE-SS-336, 電子情報通信学会, 2010, pp. 13–18.
[78] 阿萬裕久: オープンソースソフトウェアにおけるコメント文記述とフォールト潜在率との関係に関する実証的考察, ソフトウェアエンジニアリング最前線 2010, 2010, pp. 97–100.
[79] 阿萬裕久, 野中誠, 水野修: ソフトウェアメトリクスとデータ分析の基礎, コンピュータソフトウェア, Vol. 28, No. 3(2011), pp. 12–28.
[80] 小林隆志, 林晋平: データマイニング技術を応用したソフトウェア構築・保守支援の研究動向, コンピュータソフトウェア, Vol. 27, No. 3(2010), pp. 13–23.

畑　秀明

2007 年大阪大学工学部電子情報エネルギー工学科卒業．2009 年同大学大学院博士前期課程修了．現在，同大学院博士後期課程に在学．ソフトウェア不具合予測手法，ソフトウェアリポジトリのマイニングに関する研究に従事．情報処理学会，電子情報通信学会，日本信頼性学会，ACM，IEEE 各会員．

水野　修

1996 年大阪大学基礎工学部情報工学科卒．1999 年同大学大学院基礎工学研究科情報数理系専攻博士後期課程退学．同年同大学院基礎工学研究科助手．同大学院情報科学研究科助教を経て，2009 年京都工芸繊維大学大学院工芸科学研究科准教授．博士(工学)．エンピリカルソフトウェア工学，ソフトウェア不具合予測手法に関する研究に従事．情報処理学会，電子情報通信学会，IEEE 各会員．

菊野　亨

1975 年大阪大学大学院博士課程修了．工学博士．同年広島大学工学部講師．同大学助教授を経て，1987 年大阪大学基礎工学部情報工学科助教授．1990 年同大学教授．現在，大阪大学大学院情報科学研究科教授．大阪大学国際交流センター・センター長．主にフォールトトレラントシステム，ソフトウェア開発プロセスの定量的評価に関する研究に従事．情報処理学会，電子情報通信学会，情報処理学会各フェロー．ACM，IEEE 各会員．日本信頼性学会前会長．