Vol.2012-SE-178 No.23
2012/11/2

Ajaxアプリケーションの保守容易性計測のための ソフトウェアメトリクス

平山 祐資[†1,a)]　小野 康一[†2]　深澤 良彰[†1]

概要：我々は Ajax アプリケーションの保守容易性を測るソフトウェアメトリクスを定義するための研究を行っている．通常の Web アプリケーションを対象とする保守容易性ソフトウェアメトリクスは既に存在するが．Ajax の特徴を考慮したものはまだない．我々は実際に Ajax アプリケーションを作成し．保守を繰り返すことで保守容易性が変化する経過を観察し具体的データを得た．このデータをもとに回帰分析により保守容易性を測るための指標を求めた．

キーワード：Ajax，ソフトウェアメトリクス，保守容易性

Towards software metrics to evaluate maintainability for measuring Ajax application software

YUSUKE HIRAYAMA[†1,a)]　KOUICHI ONO[†2]　YOSHIAKI FUKAZAWA[†1]

Abstract: We are working on software metrics to measure maintainability of Ajax application software. There are a number of previous studies about software maintainability metrics of classical Web application software. However, they do not consider characteristic properties of Ajax application software. We obtained concrete maintenance data by observing and recording the progress of iterative modifications on an actual Ajax application software through the software development process. It is expected that the software maintainability changes through the iterative modifications. We defined a suite of software maintainability metrics by a regression analysis based on those observed data.

Keywords: Ajax, softwaremetrics, maintainability

1. はじめに

今日 Web 技術の発展が目覚しい．簡単な HTML のみで静的に表現されていた文章も近年はさまざまな技術が使われるようになっており，Web アプリケーションは大規模化/複雑化の一途をたっている [1]．用いられるようになった技術として，動的に HTML を作成してクライアントに返す CGI，異なるプラットフォームで動作し CGI よりも高速になった Java Servlet，JavaServer Pages（JSP），近年ではさらにページの一部を非同期通信によって更新する Ajax(asynchronous JavaScript^{TM}+XML) が注目されている．この Ajax の台頭によりクライアント側で表現できる UI の幅が広がり RIA（Rich Internet Application）と呼ばれている．この RIA を実装するためのプラットフォームとして JavaFX，Adobe Flash，Microsoft Silverlight などもある．また将来的には HTML5 の勧告も予定されており，Web アプリケーションで表現できる幅はますます広がっている．

一方ユーザから求められる機能も多くなっており，稼働後において機能追加による保守は度々おこなわれる．一般的に保守のコストはソフトウェア開発/保守の大部分を占め，アプリケーションの複雑度が高いほど増す．また Web

[†1] 現在，早稲田大学
Presently with Waseda Uniersity
[†2] 現在，日本アイ・ビー・エム株式会社東京基礎研究所
Presently with IBM Research - Tokyo
[a)] yusuke-h.0808@ruri.waseda.jp

アプリケーションに限った話でないが，稼働直後は高い保守容易性（Maintainability）をもっていたアプリケーションも，度かさなる保守によって保守容易性が悪化することはよく知られている [2]．そのため，ある時点におけるアプリケーションが，どの程度の保守容易性を保っているかソフトウェアメトリクス（以下メトリクス）などを用いて示すことは重要である．保守容易性を測ることによって，そのアプリケーションが今後も保守が行う必要があるのか，それともこれ以上保守を続けた場合，開発コストよりも高くなり新規に開発しなおす必要があるのか測るための指標となるからだ．

メトリクスを用いて保守容易性を測る研究は昔から行われているが，Ajax を用いた Web アプリケーションを対象にしたものに関してはほとんど見られない．本研究ではメトリクスを用いて，Ajax アプリケーションの保守容易性計測の式を重回帰分析を用いることで提案する．なお本論文では保守容易性を実効行 100 行あたりの保守時間と定義した．

以下に本論の構成を述べる．2 章で関連研究について述べ，3 章では本稿で用いる Ajax とメトリクスの概説，4 章では本稿の対象とする保守及び Ajax との関連について述べ，5 章で Ajax アプリケーションに用いるメトリクスの提案及び保守容易性計測式の提案を行い，6 章では実際のアプリケーションに適用した結果及び考察を述べ，7 章でまとめと今後の展望について述べる．

2. 関連研究

前述したようにメトリクスに関する研究は昔から行われているが，Web アプリケーションに特化したものは少なく，保守容易性を測る研究さらに限定される．

Ghosheh らは UML™を Web アプリケーションの特徴に着目し拡張したモデル（Conallen のモデル）からメトリクスを用いて保守容易性を測っている [3]．DiLucca らは oman のモデルというソフトウェアの特徴を階層構造にまとめたものを Web アプリケーションに拡張することで保守容易性を計測している [4]．また Cheh らは Web アプリケーションに CK メトリクス [5] を適用，保守コストと相関がないことを示した [6]．以上述べた研究は全て Ajax を考慮しない Web アプリケーションのため本研究の方向性と異なる．

加賀谷らは Ajax アプリケーション特有のメトリクスを提案した上で，バグの数を複雑度＝保守容易性とみなし，提案したメトリクスとの相関関係を出している [7]．本研究では保守容易性とメトリクスの関係式を導き出すことを目的とするため主旨が異なる．

3. Ajax とソフトウェアメトリクス

以下では本論文で対象とする Ajax アプリケーションの特徴について述べ，従来の Web アプリケーションとの違いを論ずる．その上で本論文で用いられるメトリクスについて説明する．

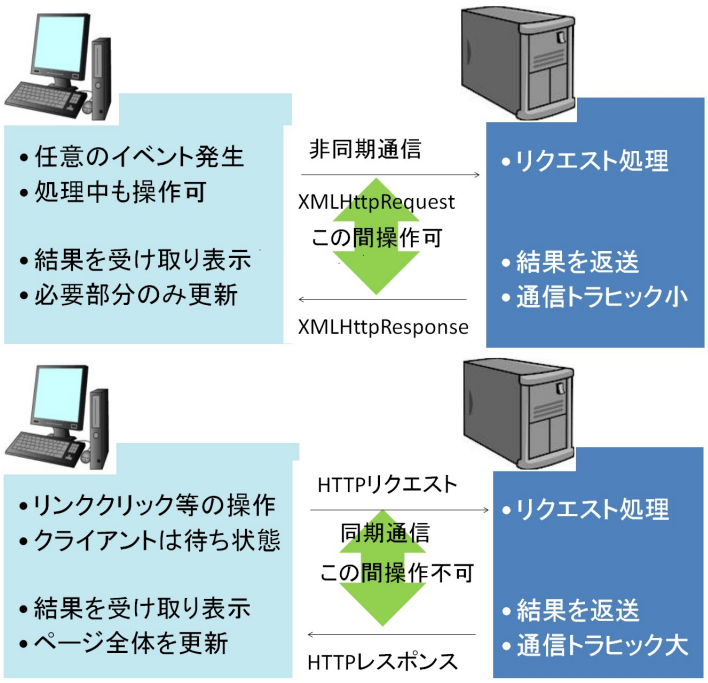

図 1 従来の Web アプリケーションと Ajax の違い
Fig. 1 Difference between classical Web application and Ajax.

3.1 Ajax

Ajax は JavaScript の HTTP 通信機能を使い，非同期でサーバーと XML 形式（プレーンテキストも可）のデータを通信できる．従来の Web アプリケーションはデータをサーバーに送り，その間 Web ブラウザ側は一切の操作ができなかった上，結果を得るためにはページ全体をリロードし直さなければならなかったが，非同期で通信をすることによって，シームレスな Web アプリケーションを作ることができるようになり，近年注目されている．

Ajax アプリケーションには以下のようなメリットがあげられる．

1）操作性の向上

通信ごとにページ全部を書き換える必要がないので，従来の Web アプリケーションに発生していたページのチラツキが発生しない．また通信中であっても操作が可能となる．

2）パフォーマンスの向上

ページ全体ではなく一部を更新するので相対的にデータ通信量も下がり，サーバーの負荷が軽減する．

3）視覚性の向上

従来の Web アプリケーションではできなかったリッチなユーザーインターフェイスを提供できる．

4）既存技術

従来の技術（JavaScript）を使っており，開発者が特別新しい環境用意する必要がない．またブラウザに標準で付いている機能のみを使うので使う側にも負担は少ない．

メリットがあればデメリットも当然ある．ここではデメリットについても述べる．

1）クロスブラウザ問題

Ajax の歴史的背景からブラウザ間で実装方法が若干異なり表現も異なる場合がある．

2）視覚性向上による問題

視覚性が向上したことによりページ全体のリフレッシュがなくなったため，ページが更新されたかわかりづらいという面がある．

3）ユーザへの負荷

サーバー側への負荷が少なくなったということは，それだけクライアント側での負荷が増えたということである．開発者はクライアント側の負荷を考慮しなければならない．

しかしいずれも解決可能な問題であり，リッチなインターフェースを提供する上で必要な技術の一つであることは間違いない．

Ajax を用いたアプリケーションの例として文字を入力することでそれに対応した検索語の候補を出す Google Suggest^{TM}やドラッグ等の操作で地図の読みこまれていない部分だけを非同期通信で表示する Google Maps. ^{TM}などがある．

これまでに述べてきたように，Ajax アプリケーションは従来の Web アプリケーションと違った仕組みで動作するため，当然実装方法も従来と異なる．よって Ajax アプリケーションの保守容易性を測る場合は，その従来との違いを考慮して測らなければならない．

3.2 ソフトウェアメトリクス

先にも述べたとおりソフトウェアメトリクスはソフトウェアを定量的に測る指標であり，保守容易性や工数予測の一手段として用いられる．有名な指標としてソフトウェアの規模を測る LOC(Lines of Code) や，ソフトウェアの機能を基準として処理の複雑さから点数を出し規模を測る Function Point Analysis（ファンクションポイント法/FP 法）[8] などがある．これらはソフトウェアの規模を測ることにより，ソフトフェア開発の工数を見積もるときに用いられることが多い．またプログラムの制御フローを図に書きノードとエッジの数からソフトウェアの複雑度を測る Cyclomatic Complexity（サイクロマティック複雑度）[9] や，オペランドとオペレータの種類数と出現数に基づいて複雑度を定義した Halstead のメトリクス [10] や，先にも述べたオブジェクト指向モデルの構造的複雑度を測る CK メトリクスなどがある．これらは主にソフトウェア開発後の保守容易性を測るのに用いられる．

また近年では Web アプリケーションのメトリクスも提案されており，Ghosheh らは Conallen の Web アプリケーションモデル [11] をもとにメトリクスを提案した [3]．**表 1** にその一部を示す．

表 1 Ghosheh らが提案したメトリクス（一部）
Table 1 A proposal metrics for Ghosheh.

メトリクス	メトリクスの説明
NserverP	サーバーページの数
NclientP	クライアントページの数
NformP	フォームの数
NformE	フォームの要素数
NlinkR	リンクの数
NsubmitR	サブミットの数
NincludeR	サーバーページ-サーバーページの関係

先に述べた分類方法以外にも，ソースコードメトリクスとデザインメトリクスという分け方もある．デザインメトリクスとは設計段階で測れるソフトウェアの側面で，FP 法やクラス間の依存関係などがある．

もう一つのソースコードメトリクスは開発の過程で得られるものであり，Cyclomatic Complexity や LOC，クラス内のメソッドの数，メソッドの行数などがあり，保守容易性を測るときに用いられることが多い．本研究においてもこちらのメトリクスを用いる．しかし上記あげたものだけでは Ajax の特徴をとらえきれておらず別途考慮する必要がある．

4. Ajax アプリケーションの保守

本章では保守について概説したうえで本稿の対象とする．Ajax アプリケーションの保守について述べる．

4.1 保守と保守容易性

ソフトウェアの保守とは，納入後のソフトウェアに対して加えられる，欠陥の修正，性能の改善，環境移行適応のための改訂のことである [12]．また保守は目的に応じて以下の 5 つの種類に分けられる [13]

1）適応保守（adaptive maintenance）

変化した又は変化している環境において，ソフトウェア製品を使用できるように保ち続けるために実施するソフトウェア製品の引渡し後の修正．

2）是正保守（corrective maintenance）

発見された問題を訂正するために行うソフトウェア製品の引渡し後の受身の修正．いわゆるバグ修正．

3）緊急保守（emergency maintenance）

是正保守実施までシステム運用を確保するための，計画外で行われる一時的な修正．

4）完全化保守（perfective maintenance）

潜在的な欠陥が故障として現れる前に検出し訂正するた

めに行う，引渡し後のソフトウェア製品の修正．

5）予防保守（preventive maintenance）

引渡し後のソフトウェア製品の潜在的な障害が運用障害になる前に発見し，是正を行うための修正．

本研究においては「適応保守」及び「是正保守」を対象とする．理由としてこれらの保守が最も一般的に行われるものだからである．また本研究が測る品質である保守容易性は「解析容易性（analyzability）」「変更容易性（changeability）」「安定性（stability）」「試験性（testability）」「標準適合性（compliance）」[14] にわけられるが本稿では「解析容易性」＋「変更容易性」＋「試験容易性」の 3 つに焦点を当てる．これらを組み合わせることで，保守全体を見渡せると考えたためである．

4.2 保守容易性メトリクスの要件

Ajax は従来と違う機能を持つため実装方法も異なる．従来の実装方法ではサーバー側で記述する量が多く，保守もサーバー側に変更の重きが置かれる．対して Ajax はクライアント側で実装する量が増え，必然的に保守もクライアント側で記述される量が増える．このため Ajax アプリケーションの保守容易性を測るには，従来から用いられていたメトリクスのみでは不十分であり，Ajax の特性を考慮する必要がある．具体的には，サーバー側のメトリクスとは別にクライアント側のメトリクスを考える必要がある．また Ajax は通信データの形式が XML かプレーンテキストにわかれる．本研究では，まず基本的なプレーンテキスト形式のデータ通信を実装した Ajax アプリケーションに，焦点を当て保守容易性を計測する．

5. 保守容易性メトリクスの提案

本章では 2 つの事項について述べる．まず Ajax の特性を考慮したメトリクス及び従来から使われていたメトリクスのうち，本研究で用いるメトリクスについて述べる．次に実験として Ajax アプリケーションに保守を施して保守容易性が変化する様子を観察し，本研究で用いるメトリクスを適用して，その結果から重回帰分析を行って，保守容易性計測の式を導く．

5.1 考慮する指標

表 2 に Ajax の特性を考慮したメトリクスを示す．

本研究では Ajax の特性を考慮するために保守容易性は「XMLHttpRequest オブジェクトの数」「JavaScript ファイル全体のオブジェクト数」「非同期通信によって返ってきたデータ数」「イベントリスナの数」「DOM の変更度合い」に依存するものとする．これらを選んだ理由として，Ajax は JavaScript で実装されており，まず JavaScript の特徴をとらえたメトリクスを用いる必要があったため，また Ajax の処理の特徴は非同期通信によってページの一部を書き換えることからその処理の中でのデータの流れを追うことで Ajax の特徴をとらえることができると考えたためである．このうち「XHRobj」及び「cDOM」は加賀谷らが既に提案している [6]．本研究ではこれに加えて JavaScript の特徴も加えた．またこれらのメトリクスのみで保守容易性を測ることは不十分であるため，従来から用いられているメトリクスの中で**表 3** に示すメトリクスを用いることにした．

表 2 Ajax の特性を考慮したメトリクス
Table 2 metrics for Ajax.

メトリクス	メトリクスの説明
XHRobj	XMLHttpRequest オブジェクト数
jSobl	javascript 側のオブジェクト数
ResDataNum	返ってきたデータ数
EventNum	イベントリスナの数
SendDataNum	Ajax 通信で送信されるデータ数
cDOM	DOM 構造の変更度合い

表 3 その他のメトリクス
Table 3 metrics for others

メトリクス	メトリクスの説明
ClientLocalNum	クライアントのローカル変数
ClientGlobalNum	クライアントのグローバル変数
ClientM	クライアント側のメソッド数
CliemtLOC	クライアントの LOC
Wnum	ウィジェットの数
ClientCC	クライアント側のサイクロマティック複雑度
ServerLOC	サーバー側の LOC
ServerLocalNum	サーバー側の変数
MethodNum	サーバー側のメソッド数
ServerCC	サーバー側のサイクロマティック複雑度
NestNunm	ネストの数

5.2 保守容易性計測指標の提案

5.1 で列挙した指標から相互に独立しているものだけを選別し，メトリクスとして計測すべき指標群のみ定義し直した．この後に定義したメトリクスを実際に行った保守に対して適用し，得られたデータをもとに重回帰分析を行うが，重回帰分析は説明変数が互いに独立であるという前提がある．ある説明変数間同士に相関関係があった場合，その両方を用いて式を作ると説明変数が互いに影響し，精度の低い式になるためである．5.1 で述べたメトリクスは他のメトリクスとの相関関係が考慮されていないため，相関を排除したメトリクスを選び直す必要がある．定義しなおしたメトリクスを**表 4** に示す

5.3 重回帰分析による係数決定

保守容易性計測指標の式を定義するため，具体的な Ajax アプリケーションソフトウェアを用いて解析することとした．事例として社員情報管理アプリケーションを想定し

表 4 保守容易性計測指標一覧
Table 4 List of maintenance measurement indexes

メトリクス	メトリクスの説明
XHRobj	XMLHttpRequest オブジェクト数
jSobl	javascript 側のオブジェクト数
SendDataNum	Ajax 通信で送信されるデータ数
cDOM	DOM 構造の変更度合い
ClientM	クライアント側のメソッド数
ClientCC	クライアント側のサイクロマティック複雑度
ServerLocalNum	サーバー側の変数
MethodNum	サーバー側のメソッド数
ServerCC	サーバー側のサイクロマティック複雑度

た．まず最初に社員情報の登録機能のみを持つ Ajax アプリケーションを作り，それに対して繰り返し保守を行い，機能追加や変更を与えた．その開発過程を通じて，5.2 で定義した指標でデータを取得した．**図 2** と**表 5** に保守を行ったソフトウェアの概要を述べる

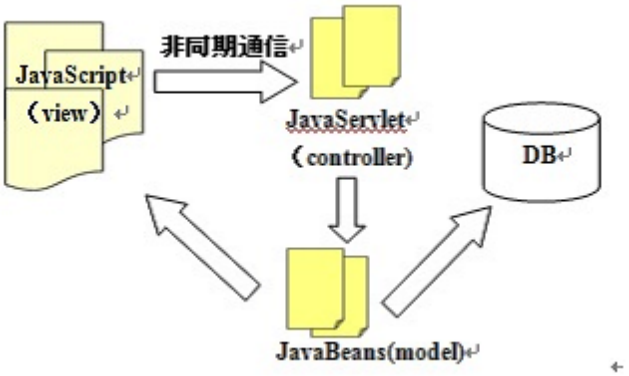

図 2 ソフトウェア概要図
Fig. 2 Overview of software.

MVC 構造にのっとり構築した．基本的にクライアント側から非同期通信でサーバーサイドにデータが行き，データの種類によって処理を変え DB を更新，もしくは DB からデータを取得結果をクライアント側に返す．データ処理の基本である CRUD 処理を実装している．

表 5 実験に用いたアプリケーションソフトウェア概要
Table 5 overview of developed application software for experiment

	社員情報管理アプリケーション
LOC	1755 クライアント 1145　サーバー 610
クライアントページ数	6
クラス数	9
構成	JavaScript+HTML+Java+MySQL
機能概要	社員の登録処理及び参照削除更新機能

2000 行弱と規模としてはかなり小さめであるが，Ajax で実際に使われるであろう機能はある程度満たしているため，また学生一人で設計からすべて行っているため，この規模に落ち着いた．次に**表 6** に行った保守の概要について述べる．

表 6 社員情報管理アプリケーションソフトウェアに実際に適用した保守題目
Table 6 maintenance items applied to the employee information management application software

行った保守	保守時間
入力項目の追加・DB のデータ構造変更	1:29:17
所属を動的に取得するように変更	2:10:36
登録する名前で同姓同名を動的に確認	1:58:52
DB から所属の情報を動的に取得	2:04:26
データの追加	2:21:49
台帳データ追加	3:15:07
台帳参照機能追加	3:04:34
郵便番号から住所検索する機能追加	2:44:04
データ追加	1:44:08
データの追加	1:19:48
入力項目の妥当性を動的に確認する機能追加	4:28:27
予測検索機能追加	3:56:13
更新チェック機能追加	4:44:59

行った保守はデータ構造の変化，データの追加，機能追加などよくある保守を加えた．保守時間は変更 100 行あたりにかかった時間であり変更部の特定及び理解，変更，テストまでの時間を含める．しかしながら，ここの保守時間は先に述べた時間から若干の補正をかけている．理由として繰り返される保守の間で慣れが生じてしまい．純粋に保守を繰り返すことで生じる保守容易性の悪化が見れないと考えたためである．そのため初期のころかかっていたであろう学習にかかる時間（仮に学習コスト）を排除することで，純粋に保守にかかった時間を導き出した．具体的な排除の方法に関しては紙面の都合上省く．

次に 5.2 で定義したメトリクスを用いて重回帰分析を行い，式を導出するが，導かれた式が有意水準を満たさなければ回帰式が受け入れられない，また式が受け入れられてもその変数がその式を説明するのに必要かどうかを見る必要がある．そのため変数を選択し，その都度変数を調整する必要がある．

表 7 式の有意性
Table 7 Significance of the expression

	回帰統計
重相関係数	0.9991
決定係数	0.998
修正済決定係数	0.796
観測数	12
F 値	1.5E-05

最終的に**表 7** と**表 8** の結果となった．修正済決定係数は回帰式の確からしさを示す．1 に近いほど確からしい．一

表 8　各変数の係数
Table 8　Coefficient of each variable

変数	係数
jSobl	0.057
EventNum	1.303
cDOM	1.404
ClientCC	72.534
ServerLocalNum	0.26
MethodNum	1.763
ServerCC	-41.685

般的に 0.7 以上で確からしいと言えるので，この式はある程度確からしいと言える．またＦ値は有意水準（$\alpha = 0.05$ 未満）を満たすことによって，この式が受け入れられることがいえる．Ｆ値は（1.5E-05）なのでこの式は受け入れられる．表 8 はそれぞれの変数の係数を示す．よって保守容易性計測指標は，（1）のようにあらわせる

$$M = 0.057 \times jSol + 1.303 \times EventNum + 1.404 \times cDOM + 72.534 \times ClientCC + 0.26 \times ServerLocalNum + 1.763 \times MethodNum - 41.685 \times ServerCC \quad (1)$$

6. 評価考察

本章では 5.3 で出した式をいくつかの別のアプリケーションに適用し式の確からしさをを評価し，その上でその結果について考察を行う．

6.1 評価

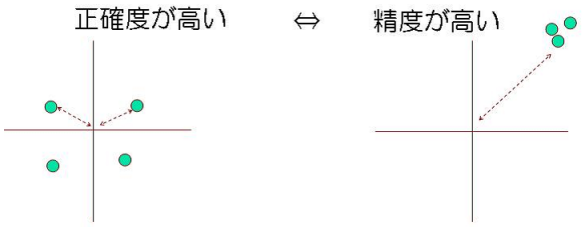

図 3　精度と正確度の違い
Fig. 3　Precisiont and Accuracy.

保守容易性を精度（Precision）重視で測るか正確度（Accuracy）重視で測るかは重要である．精度が高いとは似た傾向のものを測ればそれらは近い位置に配置される（測定によって生じうる偶然誤差が小さい）ことで，正確度は高いとは測りたいものとその実際のものの誤差ができるだけ少ない（測定が現実を反映する度合いが高い）ことである．測定の結果は正確度は高くても精度が低いこともある，また精度が高いが正確度が低いこともある．**図 3** に両者の違いを表す．正確かつ高精度な「結果」を有効あるいは妥当であるという．

本研究では保守にかかる時間を測ることから，実際の保守時間から前後にぶれるたとしても，その誤差が少ないほうがいいと考えた．そのため可能な限り現実に対して誤差が低い正確性の高さに重き置く．よって本評価では 5.2 で出した式を別途作成し，保守を行った幾つかの Ajax アプリケーションに対して適用し実際の保守時間との相関係数を出し式を評価することにした．評価のために作成した社員情報検索アプリケーションの概要を**表 9** 及び**表 10** に示す．

表 9　社員情報検索アプリケーションソフトウェアの概要
Table 9　overview of employee information retrieval application software

	検索システム
LOC	2032 クライアント 1451　サーバー 581
クライアントページ数	3
クラス数	4
構成	JavaScript+HTML+Java+MySQL
機能概要	社員の検索機能

表 10　社員情報検索アプリケーションソフトウェアに実際に適用した保守項目
Table 10　maintenance items applied to the employee information retrieval application software

行った保守 式で出した時間	実際の保守時間	
検索データ項目追加	1:48:00	1:57:18
検索データ項目追加	1:51:00	1:57:08
ソート条件の追加	2:47:00	2:05:13
次へ，戻る機能追加	3:16:00	2:14:00
ソート条件の追加	3:26:00	2:17:04
ソート条件の追加	4:13:00	3:08:34
データの追加	4:21:00	3:35:47
相関関係	0.764	

また Java 以外の言語にも適用できるか確認するため，オープンソースの投票情報管理アプリケーションの [16] に保守を行った．概要を**表 11** 及び**表 12** に示す．

表 11　投票情報管理アプリケーションソフトウェアの概要
Table 11　overview of vote information management application software

	投票システム
LOC	1058 クライアント 957　サーバー 301
クライアントページ数	4
サーバー側ページ数	4
構成	JavaScript+HTML+PHPa+MySQL
機能概要	社員の検索機能

表 12 投票情報管理アプリケーションソフトウェアに実際に適用した保守項目

Table 12 maintenance items applied to the employee information management application software

行った保守	式で出した時間	実際の保守時間
データ追加	2:01:01 1:42:31	
DB の構造変更	1:48:21 2:01:45	
データの変更	2:11:11 2:22:48	
相関関係		0.468

6.2 考察

表 9 に示すように検索システムに対して，適用した場合強い相関を示した．確かに初期に行なっていた保守に関してはかなり近い値を示している．しかし後になると実際の値との差が大きく，この相関係数が示すほど保守容易性を測れているとはいえない．この検索システムはあえてコードクローンを含むように実装されており，社員情報検索アプリケーションソフトウェアの保守時間が，社員情報管理アプリケーションソフトウェアの保守時間に比べて短いことからもそれが伺える．社員情報管理アプリケーションソフトウェアには，コードクローンをほとんど含まなかったため，重回帰分析を行うときにコードクローンを考慮しなかった．そのため，実際の保守時間と式で導き出した保守時間との間に，ずれが生じてしまったと考えられる．また重回帰分析によって導かれた式は，サイクロマティック複雑度に大部分の重きが置かれており，その他のメトリクスの重みが軽くなっている．確かにサイクロマティック複雑度は保守容易性に関わる大事な要素だが，サイクロマティック複雑度の重みがあまりに強いため，今回選んだその他のメトリクスがほとんど影響を及ぼしていない．結果サイクロマティック複雑度のみによるところが大きく，重回帰分析に用いるメトリクスを再考する必要があると考える．

一方オープンソースの方はそこまで高い相関を示さなかった．規模が小さく，機能自体が単一の質問に対して幾つかの項目から一つを選び，非同期通信でこれまで選んだ項目の割合を示す単純な機能だったため，拡張性に優れない．そのため保守が少くなりデータの数が足りておらず，一つのデータの影響が大きいことが原因にあげられる．またこのアプリケーションはサーバー側が PHP で書かれている．式を導き出す要素は言語によらないものだが言語によって保守にかかる時間に対して補正をかける必要があるかもしれない．

7. おわりに

Web アプリケーションは近年大きく進歩し，多機能化/複雑化を果たした結果保守のコストが増大した．Web アプリケーションに限った話ではないが，この増大するコストを少しでも削減するために，現在の保守容易性をソフトウェアメトリクスなどを用いて知ることは重要である．

本研究では，近年 Web アプリケーションの中でも特に注目されている技術の一つである Ajax に注目し，保守容易性計測式を提案した．Ajax アプリケーションの保守容易性を計測するために，Ajax の特性に特化したメトリクスを用いて式を作成し，保守容易性を保守時間で測ったこと本研究の新規性である．

しかし式の正確性としてはまだ不十分なところも多かった．例えばコードクローンについて考慮してなかったため正確な評価が出なかったことや，保守容易性計測式を導くために，用いられたアプリケーションソフトウェアの規模が小さく，より大きななシステムに対して，適用できるようなモデルではなかったことがあげられる．また Ajax だからこそ起こりえる保守を考慮して行ったが，考慮から漏れてしまった部分がないとも言い切れないため、カバーできていない部分もあるかもしれない．特に今回の Ajax のデータのやり取りはプレーンテキストのみで行われており，よく使われる形式の JSON 形式のデータのやり取りには対応できていない．また本研究における保守は著者が単独で行っており属人性の問題があげられる．

今後は今回の式で考慮されなかった部分について見直し，式の精度を上げていきたいと考えている．特に今回重回帰分析で選んだ変数間の相関関係はできる限り低いものを選んだが，排除しきれてない部分もありそれが式の精度を下げている可能性がある．この問題を解決するため PLS 回帰分析を行うことも視野を入れたい．今回はデータのやり取りを簡単なプレーンテキスト形式で行ったが，実際に最もよく使われる形式は JSON のため JSON 形式対応する様式を拡張することで適応範囲を広げていきたい．

また今回の実験においてクライアント側のメトリクスはすべて手作業で測り．式の計算も手動で行っていたためこれらをツールで測れるようにすることも今後の目標としたい．

参考文献

[1] Abraho, S. Condori-Fernndez, N. Olsina, L. and Pastor, O. : *Defining and validating metrics for navigational models*, Proc. of the 9th International Software Metrics Symposium（METRICS2003), IEEE Computer Society , pp. 200-210, （2003）.

[2] John, B. Michael, W. T. Eric, H. G. Lee, S. and Richard, C. H. : *Architectural repair of open source software*, Proc. of International Workshop on Program Comprehension (IWPC2000).

[3] Ghosheh,E. Black, S. and Qaddour, J. : *Design metrics for Web application maintainability measurement*, Proc. of ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), IEEE Computer Society Press, pp. 778?784. (2008).

[4] DiLucca, G. Fasolino, A. Tramontana, P. and Visaggio,C. : *Towards the definition of a maintainability model for Web applications*, Proc. of the 8th Euro-

pean Conference on Software Maintenance and Reengineering, pages 279-287, IEEE Computer Society Press, (2004).
[5] Chidamber,S.R. and Kemerer,C.F. : *A Metrics Suite for Object-Oriented Design*, Proc. of IEEE Trans. Software Eng., vol. 20, no. 6, pages 476-493, June (1994).
[6] Chae,S. H. Kim,Y. T. and Jung, W.Sung. Lee,J. S. : *Using metrics for estimating maintainability of Web applications: An empirical study*, Proc. of ICIS, pages 1053-1059. IEEE, (2007).
[7] 加賀谷有紀，海尻賢二，海谷治彦．: Ajax のための Web 保守性メトリクスの提案, 電子情報通信学会技術研究報告， Vol. 111， No. 282， pp. 91-96（2011）.
[8] Albrecht, J. Gaffney, J. R. : *Software function, source lines of code and development effort prediction*, Proc of. a software science validation, IEEE Transactions on Software Engineering, vol. 9, no. 6, pages 639-648, （1983）.
[9] Watson, A. and McCabe, T. : *Structured Testing: A Teting Methodology Using the Cyclomatic Complexity Metric*, Proc of. NIST Special Publication, Vol.500-235, Aug. 1996..
[10] Halstead, M. : *Elements of Software Science (Operating and programming systems series)*, Proc. of Elsevier Science Inc., New York, NY, USA, (1977).
[11] Conallen, J. : *Building Web Applications with UML, 2nd edition*, Addison-Wesley（2002）.
[12] IEEE Std 1219: Standard for Software Maintenance. 1997.
[13] ソフトウェア技術ーソフトウェアライフサイクルー保守. 東京:財団法人 日本規格協会. JIS X 0161. 日本工業標準調査会 (2008).
[14] ISO/IEC 9126-1:2001, Software engineering - Product quality - Part 1:Quality model.
[15] http://www.stateofflow.com/UpdateSite/(EclipseMetricsPlugin)
[16] http://www.dhtmlgoodies.com/index.htm/(AJAX Poll)

本稿中の会社名．製品名およびサービス名等はそれぞれ各社の商標である．