ソフトウェアに関する論文の書き方

主に開発現場に携わる人のために

大阪大学大学院情報科学研究科　教授
井上 克郎

ソフトウェア開発の現場に携わっている技術者がソフトウェア工学やソフトウェアシステムに関する論文を書くための要点をまとめる。論文を書いて、それが採録され、論文誌に掲載されるのは簡単ではないが、大きな価値がある。本稿を参考に、多くの論文が書かれ、発表されることを期待する。

1 はじめに

ソフトウェア工学やソフトウェアシステムに関する論文は、非常に書きにくい、という声をよく耳にする[権藤2008]。これは他分野の論文に比べて相対的に、という意味であろうが、定量的な比較は困難である。ただ、筆者の主観としても、工学、理学の他の分野に比べて多少ハンディがあるように感じる。

これは、対象物のソフトウェアの可視化が困難である、絶対的な評価の基準がない等、種々の理由によるものと思われるが、だからと言って論文を書かなければどうなるであろうか。せっかく苦労して開発した技術や方法論を広く伝えることが出来ず、この分野の技術進歩が停滞してしまう。

本稿では、ソフトウェア開発に関わっている技術者を対象として、ソフトウェア関連の論文を書くためのいくつかのポイントを解説する。

自分自身、論文が自由に書けるとは思えず、まだまだ苦労をしているが、より多くのソフトウェア関係の論文が世の中に出て、この分野の技術進歩の一助になることを願って、思いつくままに記す。

2 論文を書く目的

科学技術分野に限らず、いろいろな分野で数多くの論文が執筆され、公表されている。

ソフトウェア分野では、このSEC journalを始め、情報処理学会論文誌、電子情報通信学会論文誌D分冊、コンピュータソフトウェア（ソフトウェア科学会）等の和文論文誌がある（英文でも受付あり）。また、IEEE Transactions on Software Engineering （TSE）、ACM Transactions on Software Engineering and Methodology（TOSEM）、Automated Software Engineering等を代表として、数多くのソフトウェア工学全般やソフトウェアシステム、プロセス、メトリクス等の特定のテーマを扱う英文論文誌がある。また、学会が主催する各種会議や研究会での発表もある（口頭のプレゼンテーションと議事録論文）。自分の研究成果をこれらの論文として掲載することによって以下のような効果があると考えられる。

・会社や組織の技術開発力の優位性をPR出来る。
・研究開発の1つの業績やマイルストーンとして認識出来る。

・論文として成果をまとめることにより、研究成果を整理し、その位置付けをはっきりさせることが出来る。

一方、論文を書くためには、仕事を振り返って整理し、時間をかけて文章化して、投稿をし、査読者のコメントに従って何度か修正をし、（論文誌によっては掲載料まで払って）やっと掲載される。査読者のコメントは強烈なことも多く、不採録（リジェクト）の場合もある。精神的にタフでなければやっていけない。

このような苦労が予想される論文について、執筆する価値があるかどうかは十分検討する必要がある。大学や研究機関では、論文が評価対象になっており、Publish or perish（論文出すか死か）の世界なので、書かざるを得ない。

逆に企業の開発現場では、論文を書くためのコストが予算として計上されていない場合が普通であるが、上記のようなメリットがあるので、トップの意思として積極的に取り組んでもらいたい。日本の開発現場では、先進的な取り組みを行ってきているが、論文がうまく書かれずに普遍化出来ない例が多々ある。単なる社内の一事例として世の中に埋もれては非常に残念である。

3 論文掲載までのプロセス

図1に典型的な論文の投稿から論文誌や雑誌に掲載されるまでのプロセスを示す。

まず、作成した論文を掲載したい雑誌、論文誌の投稿先に送付する。そこで、その論文を担当する編集委員（メタ査読者）が決められ、その編集委員と著者がやり取りをすることになる。

編集委員は、その論文にふさわしい査読者を選び、査読をしてもらう。査読者の数は、雑誌や論文の内容によって決まっており、通常2～3人である。また、1～3カ月程度が査読期間として割り当てられる。

返ってきた査読結果を見て、編集委員及び編集委員長は、「そのまま採録する（無条件採録）」、「不採録（返戻、リジェクト）」、「条件付採録」のいずれかの決定を下し、著者に連絡する。

条件付採録は、査読者のコメントに従って修正すれば採録の可能性があることを示すもので、通常、2～3カ月以内に改訂版を作って送付し、再度査読者によるチェックを受け、判定される。

一般に査読者のコメントは非常に厳しく感じられ、なかなかそれに従って論文の修正をするのは厄介であるが、

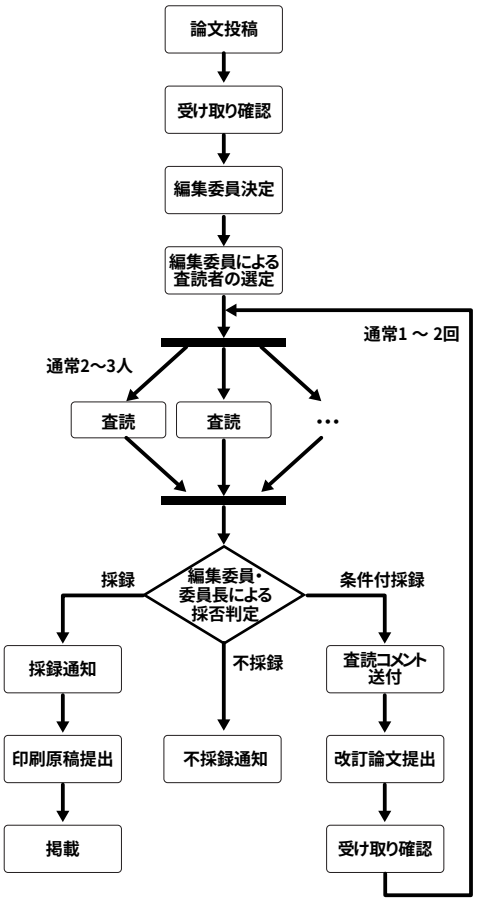

図1　典型的な論文の査読プロセス

専門家による指摘は重要であり、よく検討して、修正を行う（すべて査読者の言いなりというわけではないが）。

また、コメントに従ってどこをどのように修正したかを記した文章を付けるのが通常で、そこでは、ボランティアで査読を行ってくれている査読者に謝意を示した文言を入れるとよい。

4 論文の主テーマについて

会社の現場で工夫し、努力した開発案件を論文にしようとしたときに、まず考えなければならないのは、どのようなテーマで論文を書くかである。以下は、筆者が考えるポイントである。

① 自分にとっても読者にとっても面白いテーマを選ぶ

よく“論文の書き方”等で、「新規性」、「有効性」、「信頼性」等とこの価値をブレークダウンして表現されているが、非常に抽象的で分かりづらい[電子情報通信学会]。結局のところ、いかにして査読者や読者に面白い、読んで良かった、と思わせることが出来るかが重要である。そのためには、まず、書く本人が面白いと思えなければ

伝わるはずがない。開発現場で新たにこういう発見をして驚いた、開発方法を工夫したらびっくりするほど性能が改善した等、まず、自分が興奮し、それが他人にも共感を得るようなテーマでなければならない。

② 書くべき要素を絞る

ソフトウェア関係の論文として、対象のプロジェクトのライフサイクル全体を満遍なく書くことは、フォーカスがボケて良くない。査読者や読者にとって、例えばプロジェクトの中で使われた新しいアルゴリズムが重要なのか、定量的な評価が興味を引くのか、よく考えて、論文に書くものを厳選する。

③ 一般読者の知識を考える

投稿する論文誌をよく読み、対象とする読者の知識を推定してテーマ選定を行う。SEC journalの読者は、概ねソフトウェア技術者、とくにソフトウェアの信頼性や品質に非常に興味を持っている人々である。余りにも読者の知識とかけ離れた論文は、読者は理解出来ず、受け入れられない。一方、多くの読者が既に知っていたり簡単に想像出来たりするものでは、論文としての価値が無い（これは一般論で、実証の一方法として繰返実験等は価値がある）。

④ 役に立つかどうかの視点を忘れずに

論文は、他人に読まれてその知識が広く伝わり、いろいろな形で利用されてこそ価値がある。掲載されることが目的になってしまってはいけない（学位取得のため等、大学的な事情はよくある話ではあるが…）。論文のテーマや要素を選ぶ際には、どのように役立つかを常に意識する必要がある。同業他社に知られてはまずい、という考えでは論文は書けない。オープンソースと同様、自分の知見をぜひ使って欲しい、という精神が重要である。とはいえ、所属企業の方針の確認も必要であるが、ソフトウェア関係の手法や技術は、論文に書く程度の情報だけでは簡単には真似出来ないので、どんどん公開すべきである。

5 論文の構成

ソフトウェア工学分野でよく見る論文のパターンは図2のような構造である。本稿では、このパターンに沿い、それぞれについて説明する。

5.1 タイトルについて

タイトルは非常に重要である。それを見て、何をテーマとして主張したい論文か、簡単に分かるようにすべきである。また、論文の内容に比べて広すぎず、狭すぎず適当な範囲をカバーしているかチェックする必要がある。

タイトル
著者、所属
概要
まえがき
本文1
本文2
…
　［本文列は例えば以下のような構成］
　モデル
　実装
　評価
　議論
　関連研究

まとめ
謝辞
参考文献
（付録）

図2　典型的なソフトウェア関係の論文の構成

タイトルは、過去の論文や関連する報告等のタイトルと同一であってはいけない。内容がほぼ同じであったとしても混同されないために、違いがある必要がある。

タイトルは、１つの日本語の文として、分かりやすくバランスの良いものであって欲しい。また、一文だけでは不十分な場合は、必要に応じて副題を付けて、主題を補足することもよく行う手法である。

タイトルを決めるのは簡単ではない。論文の主たるテーマを何にするか、どういう構成でストーリー展開するかが明確になっていないと、良いタイトルは付けられない。１つのタイトルを巡って、共著者が何時間にもわたって議論する、ということはよくあることで、この議論を通じてテーマやストーリーが整理されていく。

5.2 概要について

概要は、あらましや内容梗概、アブストラクトとも呼ばれ、その論文誌によって大体文量が指定されている。ポイントとしては、ここを読めば、背景、手法、結果、評価等の仕事の全体が分かると同時に、何を売りにした論文なのかが理解出来るようなものにする。そして、続く本文を読みたくなるような具体的なセールスポイント

(例えば「効率が30%向上した」等）をしっかり書く。

まえがきの一部を抜き出した概要がよくあるが、まえがきとの役割が違うので、新たな文章で書くのが美しい(読んでいると同じ文章の使いまわしは簡単に分かり、がっかりする)。

5.3 まえがきの書き方

まえがきは、その後に続く本文の導入である。従って、論文を書く背景や動機、目的や必要性、同様な研究技術の現状等を分かりやすく書く。新たな開発法に関しての論文であれば、なぜ今までの開発法では駄目か、どのような問題がクリアされなければならないか等である。ただし長い背景説明は読む気を削ぐので、出来るだけ簡潔にまとめる。重要な先行研究がある場合は、参照しておく。

一方まえがきは、それ自体で完結した短い読み物である必要がある。通常、査読者や読者が本文を丁寧に読むことは考えにくい。概要とまえがきだけで、評価されてしまう。従って、この論文の価値を分かってもらえるように、提案する手法、やったこと、評価結果、主張等も簡潔に書いておく。概要と重複する部分もあるが、一般にまえがきのほうが分量は多いので、より丁寧に書くことが出来る。また、以降の本文に書くべき内容であっても、セールスポイントに関しては、重複を恐れずに繰り返し書こう。

最後に、各章の簡単な説明を書く場合が多いが、必ず書かなければならないものではない。

5.4 本文について

一般に本文は複数の章から構成される。それにはいろいろなパターンがあるが、ここでは、ソフトウェア関係の論文によく見られるパターン（モデル、実装、評価、議論、関連研究）に沿って説明する。

① モデル

ソフトウェア開発に関するプロダクトやプロセスを分かりやすく説明するためには、対象を抽象化して図や式、(擬似）プログラム等で表現した「モデル」を用いることが多い。対象をモデルで表現することをモデル化という。例えば、「開発プロセスを状態遷移図でモデル化する」等が一例である。

論文の主張をうまく表現するために、適当なモデルを設計することは重要である。状態遷移図やフローチャート等はプロセス関係によく用いられるモデルである。既にあるモデルをそのまま利用するか、拡張して用いることもよく行われる。例えばWBSを拡張してプロジェクトを表現する等である。

モデルを用いないこともあるが、言葉での表現を用いるだけでは、問題を厳密に定義することが出来なかったり、誤解を生じさせたりすることがあるので気を付ける必要がある。

② 実　装

モデルで表現したことを実際にシステム化した場合、その実装について書くことが多い。論文のテーマと直接関係ない実装の詳細を書く必要はないが、少なくともどの程度のシステムを作成したかが分かるようなデータを入れると、提案しているテーマの現実味が増し、査読者や読者を説得するのに都合が良い。

もし、システムの機能や性能が主たるテーマの場合は、より詳しく実装について書く必要がある。しかし、システムのすべてについて同じレベルで記述する必要はない。システム中の構成要素ごとに同じような説明を行っている論文があるが、それは良くない。関心事である機能や性能に直接関連する部分をより詳しく書き、関連の無い部分は大幅に省略すべきである。

ソフトウェア開発に関する提案であれば、その提案手法に沿った開発の実行がこれに当たる。

③ 評　価

実験やケーススタディ等いろいろな評価方法があるが、ソフトウェア関係の論文としては、評価が無ければあまり価値が無い。論文で著者が提案しようとしていることが本当に有効で有益なものかを、読者や査読者が判断出来るように、出来るだけ、定量的なデータを示す必要がある。これを行うために、実装したシステムの実行データや、提案した手法の種々の実験データの収集が重要である。また、通常、ソフトウェア関連のデータは複雑な要因が絡んで結果を簡単に論じることが出来ない場合が多く、データの処理や評価のためには統計的処理が必要である[豊田1994]。これに関しては、「7.3　ソフトウェア技術の実証と評価」でも詳しく述べる。

④ 議　論

得られた結果や成果に関して、主に定性的な議論を記述する。何が分かったか、何が良くなったか、どういう効果があったか等のポジティブな面だけではなく、どういう限界があるか、まだ解決されていない問題は何か等のネガティブな面も書くべきである。そうすることによって、公正な立場で書かれていることが示せ、論文全体の説得力が向上する。

ここで書くことは、選定したテーマに沿っており、読者にとって「なるほど」と思わせる必要があろう。反発を招くような独善的な解釈や主張の展開は、他の部分の

論旨の展開も駄目にする。慎重な論理展開が必要である。

⑤ 関連研究

先行する研究との違いやアドバンテージについて議論する。まえがきのすぐ後の本文の先頭に書くこともある。また、上記「④ 議論」の議論の一部として書かれることも多い。

先行研究すべてを網羅することは容易ではないが、重要な論文は必ず引用し、議論しておくことは、著者らの研究が独善的ではないことを示す意味で大切である。通常、先行研究をいくつかのグループに分け、それらごとに、違い、特徴等を書いていく。ただし、違いを強調するために、不当に先行研究をけなすことは良くない。

5.5 あとがきについて

「あとがき」、「おわりに」、「まとめ」等、いろいろな書き方がある。論文の主張を強調するために、成果や結果を再び書く。それほど時間が無い読者は、まえがきの後、本文を飛ばして、ここだけを読む場合も多い。また、本文で書ききれなかった少し主題とは外れた議論や主張を書く場合もある。

更に、今後の研究や開発の進め方の方向性を示して終わるのが一般的である。

5.6 謝辞について

論文の執筆者として名前を載せなかった関係者に感謝の意を表する記述を行う。また、研究や開発を支援してもらったスポンサーを明記する必要がある場合は、ここに書く。一般に、謝辞に名前を載せることで生じる問題は、載せないで生じる問題よりは少ないので、出来るだけ広く載せる。

5.7 参考文献の書き方

論文中で引用する参考文献は、正確にその著者や著書・論文名、出典、年等を書く。書き方は、その論文誌の決まりがある場合はそれに従う。書き方が統一されていなかったり、一部の情報が欠落していたりする参考文献のリストは非常に見苦しいので、細心の注意を払う必要がある。昔は、完全なリストを作るのは大変な作業だったが、今はインターネットで容易に情報収集出来、作ることが出来るようになったので、丁寧に取り組みさえすればよい。

査読者は、その論文と他の論文との関連を知るために、参考文献リストをよく見る。重要な参考文献が抜けている場合は、その論文のレベルを疑う。また、査読者は同じような開発、研究を行っている場合が多いが、査読者が行った先行論文が参考文献リストに載っていない場合も、悪印象を与える。

そういう観点で、同様な先行研究の調査は重要で、ある程度網羅的に収集を行い、取捨選択して参考文献のリストを作る。

5.8 付録について

付録はオプションであり、多くの論文では付録は無い。本文で書くと、読者にとって論旨の理解を妨げるような詳細なデータや手法、証明等がある場合は、付録を設け、そこに書くとよい。一部の興味ある読者や査読者が読む。ただ、あまりにも長大な付録は、ページ数が増える要因にもなるため、避けるべきである。その場合、一部の読者にとって必要と思われる情報は、別途、技術レポート等の形で公開し、必要に応じて、付録への参照を誘導するような文言を書く。

6 分かりやすい文章について

論文の文章は、読んで理解出来なければ、査読者は×をつけるし、読者は途中で読むのを止める。出来るかぎり、平易で理解しやすい文章を書くべきである。以下いくつかのポイントを説明する。

① 用語の定義

著者にとっては当たり前の用語でも、読者にとってはなじみの無いものであったり、理解が異なっていたりする場合もある。当たり前と思っても、予備知識の無い読者に向けて書くつもりで、丁寧な解説が必要である。出来るだけその用語が出現する最初のところで、1〜2行の簡単な説明をつけるべきである。新たな用語を使う際には、その説明は必要かどうか、謙虚になって検討すべきである。

② 自己完結性

論文は、ある程度の背景知識を持った同業者が読めば、内容が理解出来るように、必要な説明を全部含んでおく必要がある。企業特有の知識やツール等を前提にした説明では、他者は理解出来ない。また、先行研究の論文があってそれに基づいて論文を書く場合も、その概要がある程度分かるように、説明を書くべきである。その論文単独で、テーマが理解出来るように工夫する必要がある。

③ 分かりやすい日本語文

日本語として分かりやすい文になるよう努力する。主語、述語がきちんとあるか、論理的な構造になっているか、長すぎないか、用語の意味や使い方にブレは無いか、等いろいろなチェックポイントがある[谷口2008]。

書くべき事柄が論理的に整理されていないと、良い日

本語にはならない。長い文章は、論理が途中で分からなくなってしまうので、短い文章に分解するとよい。

④ 構造的に書く

長い段落の文章を、最初から最後まで読みきるのは大変である。出来るだけ論旨を整理し、文章を構造化する。箇条書きを利用して、議論のポイントが視覚的に理解出来るように工夫する。

⑤ 例の利用

難しいアルゴリズムや手法等の説明には苦労する。擬似プログラムやフローチャートを用いても読者には容易には理解されない。そういう場合は、出来るだけ単純な例を用いて説明すると理解しやすい。ただし論文としては例だけではなく、アルゴリズムや手法の一般的な説明も必要である。

⑥ 図や表の利用

手順や方式、システム構成等、ソフトウェア関連の論文では多くの図や表が用いられる。これは、文章だけの説明よりずっと理解しやすいので、積極的に活用しよう。

図や表には、本文を読まなくてもその意味が分かるように、丁寧なキャプション（説明文）を書く。また、線や点、スケールが分かるように、単位や凡例を記載する。複雑な図や表は、読者が解読するのが困難になるので、表現する対象を絞るか、または、表現したい部分を強調するとよい。

7 ソフトウェア技術者にとっての論文作成のポイント

企業の実務家が書く論文を読んで感じるのは、開発や研究に非常に大きな努力をしており、大学や研究所では出来ない規模や品質の対象を扱っている割に、研究の位置付けやデータ処理、評価、関連研究との違い等が弱く、論文として非常に損をしていることである。それらを克服するためのいくつかのポイントを示す。

7.1 社内向けの発表と論文の違い

社内発表では、背景知識として社内共通のものが前提となって、それに基づいて論文や発表を組み立てていく。論文は社外の多数の一般技術者が読者である。まず、テーマとして内部受けするだけのものか、社外の一般読者に耐え得るものか、よく考える必要がある。社内では評価されないものでも、対外的には評価され得るものもあり、よく検討して積極的に論文としてまとめて欲しい。

そのためには、関連研究の文献を探し、世の中のレベルはどうなっているか、それに対して開発したものがどの程度のものか冷静に分析する必要がある。先行事例に対して、すべて勝っている必要はない。あるポイントで優れているならば、そこだけをテーマに絞り論文を構成するのも良い考えである。

また、論文を書く上で、社内発表では簡単に伝わる内容も、一般読者には理解不能な用語や概念がある。新入社員に説明するようなつもりで、慎重な用語や概念の使い方が必要である。一般の読者が読んで分からない、というのは読者の責任ではなく、著者の責任である、ということを肝に銘じて執筆をする必要がある。

7.2 データの表現について

提案する手法やツールの評価に関して、定量的な議論をすることが重要である。そのためには、社内で収集したデータを公表する必要がある。いろいろな制約があり、集めたデータをそのまま論文として公表することは出来ない場合がほとんどであろう。そのためには、生データを直接論文に書くのではなく、加工した形で書く。

例えば、ある手法Mを用いて、バグの検出を効率化する、という論文を書くとする。生データとしては、X月Y日にバグがZ個検出された、というようなものであろう。論文としては、どのように効率化されたか、ということが知りたいわけで、データの絶対値には興味はない。従って、作業時間全体のa%の時点で、全バグ中のb%のバグが検出された、で十分である。作業量も同様に、例えばテスト工程のc%の工数をこのMのために用いた、という表現でよい。

また、プロジェクト自身の定性的な性質を記述する場合でも、顧客が推測されないように心がける。例えば、顧客の業種を示す場合も、金融系、流通系等、抽象的表現を用いればよい。プロジェクトの規模を示すためには、人月やコード量、FP値等を明示的に示すことが出来ればよいが、それが無理な場合は、100人月程度、数MLOC規模等とそれらのおおまかな値を示すことで読者に規模の印象を与えることが出来る。

7.3 ソフトウェア技術の実証と評価

ソフトウェアに関する論文では、提案する手法（アルゴリズム、ツール、システム等）の評価が重要である。ソフトウェア工学の論文は他の工学や科学の分野に比べて、提案する手法の評価が甘い、という指摘もある[ZELKOWITZ1998]。それは、ソフトウェアに関する評価、とくに定量的な評価を行うことが困難である場合が多いからである。理由としては次のようなものがある。

・評価するための手間や時間が大きく、経済的に成り立ちにくい。簡単に計測出来る物理量等に比べて、ソフ

トウェアに関する種々の特性を計るのは容易ではない。とくにソフトウェア開発に関する技法等に関して、その効果を計測するためには、何度も同じような開発を違った環境で繰り返し行う必要があるが、そのような実験の実行は、経済的に非現実的である。

・絶対的な計測基準が定めにくい。物理量と異なり、人造物であるソフトウェアプロダクトやプロセスは、定量的な尺度を広く合意の下で決めることは困難である。行数（LOC）1つを取っても、空行やコメント行の扱いをどうするか等、議論は多い。

・一般に評価結果に影響を与える環境の要素が多く、提案するものが本当に有効なのか、たまたま外の要因でそうなったのか判別が難しい。そのような問題を排除するためには、人為的に制御された実験環境を整備し、提案手法を使う場合と使わない場合で比較すればよい（新薬の治験と同様な手法）。しかし、そのような制御された実験環境、とくに大規模ソフトウェア開発に関する環境を整備することは、物理的、経済的に困難である。

しかしながら、難しいからといって評価を行わず提案するだけでは、科学的な態度ではない。完璧に制御された環境での対比実験で評価を行うことは出来なくとも、いろいろ工夫の余地はある。

・対比実験が出来ない場合、単独の実証実験や適用事例の結果について細かく分析し、他の因子ではなく提案手法の効果によって、有効な結果が得られていることを定性的に議論する。

・小規模な対象を厳密に評価し、それが実用的な規模まで拡大出来る（スケールアップ）ことを議論する。

・環境は整っていないが、提案する手法を利用した事例と利用しない事例を集め、統計的処理を用いて、有意差を議論する。

これらを随時組み合わせて、効率的に評価を行い、読者に提案手法の有効性を訴える。

7.4　より良い論文を作るために

より良い論文を作成するためには、既に出版され、良い論文という評価を受けているものを読み、そのパターンを真似るのが近道である。分かりやすく、説得力のある論文を見つけたら、その論理展開の仕方を分析し、自分のテーマに関して、同様な論理展開が出来るか検討してみよう。手法や評価方法も、良い論文の使える部分を利用してみよう。

ただし、文章のコピーや図のそのままの利用は（参考文献として明示した小規模な場合を除き）してはならない。

一度書いた論文は、自分で何度も読み返し、論旨が一貫しているか、読者がつまづくところはないか、用語の統一が出来ているか、日本語としておかしくないか等を慎重に調べる。

さらに、第三者に読んでもらい、コメントをもらうことも重要である。自分1人では気づかない種々の問題点の発見に役立つ。書いた論文の初めての読者が査読者というのは危険すぎる。査読者に誤字脱字を発見されるようでは、論文の採録は程遠い。

8　おわりに

企業の人にとって論文を書くことは容易ではない。論文を書く直接的な利益も見えにくい。しかし、「2　論文を書く目的」にも書いたように、論文を作成することは、個人的にも組織的にも多くのメリットがある。

今回は、国内の論文誌に日本語で論文を書くことを前提として本稿の論を進めてきたが、ぜひ、英文の論文作成も試みて欲しい。国際的な評価を得るためには、英文での国際会議や論文誌、雑誌への発表が非常に有効である。日本語の論文よりさらに手間を要するが、その価値は高い。

最後に、本稿によって、ソフトウェアに関する論文投稿、とくにソフトウェア開発に従事する現場の技術者からの投稿が増えることを期待する。

謝辞

本稿に対して、ご意見をいただいた大阪大学楠本真二教授に感謝する。また、執筆のご支援をいただいたSEC journalの神谷芳樹編集長及び矢野亜希編集委員に深謝する。

参考文献

[ZELKOWITZ1998] M. Zelkowitz, and D. Wallance, Experimental Models for Validating Technology, IEEE Computer, Vol.31, No.5, pp.23-31, May 1998

[権藤2008] 権藤克彦：なぜソフトウェア論文を書くのは難しいか？，第15回ソフトウェア工学基礎ワークショップポスターセッション，2008-11

[電子情報通信学会] 電子情報通信学会：電子情報通信学会和文論文誌投稿のしおり（情報・システムソサエティ），http://www.ieice.org/jpn/shiori/iss_4.html#4.1

[豊田1994] 豊田秀樹：違いを見ぬく統計学―実験計画と分散分析入門―，ブルーバックスB1013，講談社，1994

[谷口2008] 谷口健一：情報科学における論文の書き方，大阪大学大学院情報科学研究科コンピュータサイエンスセミナー資料，2008-6