Vol. 33 No. 8　　情　報　処　理　　Aug. 1992

解　説

ソフトウェアのコスト見積り技術†

大　筆　　豊††

1. はじめに

ソフトウェア開発管理の対象として，QCD (Quality：品質，Cost：コスト，Delivery：納期) の三要素がある．Q（品質）については，すでに本学会誌にも何度か解説されており，またD（納期）については，本特集の別の解説で述べられている．本稿では，C（コスト）管理の拠り所となる，見積り技術について解説する．

読者対象として，ソフトウェア開発に興味をもっている学生あるいはソフトウェア開発に従事している技術者を想定する．

まず本稿で用いる「見積り：Estimation」の定義を明確にしたい．一般に「見積り（書）」は，（サービスを含む広義の）商品売買契約のとき用いられ，商品「価格」の裏付けを表わす．ここでは「見積り」は，あくまでソフトウェア開発に必要なコストの見積りであって，製品の価格とは切り離して考えることとする．開発コストは，価格設定の重要な要素であるが，価格設定には，そのほか，市場の動向，顧客との関係，製品戦略など多くの要素があり，コスト見積りと価格の直接的な相関は考えない．

2. 見積り技術

2.1 見積りの対象

ソフトウェア開発にかかるコストには，開発に必要な，コンピュータのコストや要員訓練などのコストなどもあるが，最大のものは開発要員に支払われる人件費である．このため，ソフトウェア開発コストの見積りは（以下，見積り），ほぼ，開発工数（人月）の見積りであると考えて良い．

2.2 見積りの観点

見積りに関して，見積りの目的，見積りの時期，見積りの担当者，見積りの技術など，いろんな側面から考えられるが，本稿では，技術的側面から考えることにする．

見積りの手法は，以下のように分類できる[19]．

（1） 計算式によるコストモデル：
主なコスト要因を変数とした計算式

（2） 見積りの専門家による判定：
Delphi 法などを用いた見積りの専門家による推定

（3） 類似システムからの推定：
過去に開発した類似システムからの類推による見積り

（4） パーキンソンの法則：
開発に使用可能なリソース（予算や技術者数）が見積りに等しい

（5） 契約価格：
(契約交渉などで)顧客がそのシステム開発に支払える価格が見積りに等しい

（6） トップダウン見積り：
まずシステム全体の概略見積りを行い，個別のサブシステムへ割り当てる

（7） ボトムアップ見積り：
個別のサブシステムの見積りを積み上げ，システム全体の見積りとする

いずれの手法も，利点と欠点がある．現実はともあれ，(4)と(5)は，見積り技術としては受け入れ難い．技術として完成させ，実際に適用していくには，(1)のコストモデルを中心に，(2)，(3)，(6)，(7)を補完的に組み合わせることになる．今まで提案されている手法のほとんどは，計算式によるコストモデルが中心となっている．

見積りは，開発プロジェクトの計画段階から始まり，ソフトウェア開発ライフサイクルの各時点で継続して行われる．開発が進むにつれて，見積

† Software Cost Estimation by Yutaka OHFUDE (3rd Research Department, Systems & Software Engineering Laboratory, TOSHIBA Corporation).

†† (株)東芝システム・ソフトウェア技術研究所研究第三部

りの目的が変化していく．計画段階の見積りは，開発計画や予算の策定，あるいは契約の基礎資料として使われ，開発プロジェクト後の見積りは，原価管理のためのデータとなる．開発が進むにつれて，見積りの精度はだんだんと向上する(図-1)．

2.3 見積りに影響する要素

見積りの難しさは，開発コストに影響する要素が多岐にわたり，しかも開発の初期段階では確定しない要素が多いことによる．

開発コストに影響する要素として考えられているものとして，システムの規模と複雑さ，開発メンバの能力と経験，開発に用いるハードウェアの制約，開発支援ツールの使用と経験，開発システムの変更要求などがある．表-1 は，どの要素が見積りに影響するかについての例で，管理職へのアンケート調査の結果を表わしている[1]．

コストモデルを用いて見積もる場合でも，過去から蓄積された実績データや，見積り担当者の経験と直感に基づくことになる．

見積り担当者の経験や立場の違いによる，見積り精度の相関に関して面白い報告がある[20]．

表-1 見積りの影響要素[1]

影響要素	影響度 (1—5，平均値)
開発システムの複雑さ	4.28
既存システムとの統合の要求度合い	4.19
プログラムの複雑さ	3.96
機能の数でみたシステムの規模	3.80
プロジェクトチームメンバの能力	3.64
プログラムの数でみたシステムの規模	3.63
プロジェクトチームの応用分野についての経験	3.54
要求の潜在的な変更についての予測される頻度と程度	3.48
プログラミング言語についてのチームの経験	3.43
データベース	3.39
プロジェクトチームメンバの数	3.27
プログラミング標準や文書化標準の範囲	3.14
ソフトウェア生産支援ツールの使用可能性	3.13
開発モード（バッチまたはオンライン）	3.11
使用言語	3.08
プロジェクトチームのハードウェアの経験	3.08
テスト支援の使用可能性	2.84
実機上のテスト時間	2.71
メインメモリや二次記憶の制約	2.66
コード数でみたシステムの規模	2.26

表-1 の例でも分かるように，見積りに影響する要素は非常に多い．これらは，

（1） 開発規模や複雑さなどの，開発されるソフトウェアの性質と，

（2） 使用言語や開発チームの経験や能力，開発環境などの，開発の制約条件に大別できる．

見積りは，まず開発されるソフトウェアの性質を見極めることから始めるのが自然であろう．

ソフトウェアの規模や複雑さを定量化する尺度として，よく用いられるものとして，ソースライン数 (Source Line of Code，以下 SLOC と省略) がある．厳密には，コメントを含む総ライン数，コメントを除く総ライン数，実行コードのライン数，総開発ライン数，あるいは，テストのためなどのプログラミングを除く出荷ライン数などのうち，何を表わすかを明確にする必要がある．

ソフトウェアの複雑さを，SLOC より厳密に表わそうとする尺度として，Halstead のプログラム長[21]と McCabe の複雑度[22]がよく知られている．概略，Halstead のプログラム長は，プログラム中

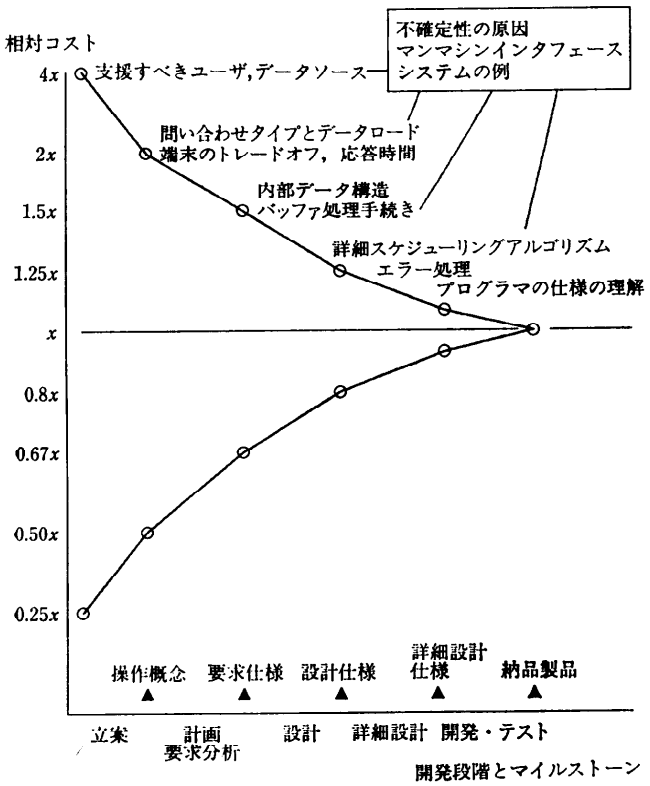

図-1 ソフトウェアコスト見積りの精度と開発段階[18]

のオペランドとオペレータの総数，MaCabe の複雑度は，プログラム中の条件判定の総数で求められる．しかし，これらの尺度も，開発工数との関係では，SLOC と相関することが分かっており[6]，多くのコストモデルの基本的な尺度として,SLOC が用いられている．

SLOC は，当然，使用する言語の違いにより，たとえば，アセンブラと FORTRAN では同じ仕様のソフトウェアを開発しても，値が異なることになる．このため，言語間の換算係数を掛け，基準とする言語で比較をするといったことが行われている．

また，単純な部分と複雑な部分では，同じステップ数でも開発に要する工数が異なる．これらの矛盾を除くため，開発すべきシステムの機能に着目した尺度を見積りの基本的な尺度とする手法（ファンクションポイント法）が注目されている．

3. 代表的な見積りモデル

本章では，代表的なコストモデルとして，SLOC に基づいた，Putnam モデルおよび COCOMO モデル，ソフトウェアの機能に着目したファンクションポイント法を紹介する．

3.1 Putnam モデル

Putnam[12] は，経験的な観測により，ソフトウェア開発の工数が次式に示す Rayleigh 分布に従うとの仮説をたてモデルを開発した．

$$m(t)=\frac{2Kt}{t_d^2}\exp\left[-\frac{t^2}{2t_d^2}\right]$$

ここで，$m(t)$ は時刻 t における要員数，K は開発プロジェクトの総工数，t_d は，要員数が最大になる時刻を表わす．また，困難度という尺度を

$$\text{困難度}\quad D=\frac{K}{t_d^2}$$

で定義した．たとえば，$K=30$（人年），$t_d=1$（年）のとき，このプロジェクトの困難度は，30 である．そして，生産性を

$$\text{生産性}\quad P=\frac{\text{総 SLOC}}{\text{総工数}}\left(\frac{S}{K}\right)$$

で定義すると，

$$P=C\times D^{-2/3}\quad (C\text{ は常数})$$

あるいは，代入により

$$S=c_tK^{2/3}t_d^{4/3}$$

を得る．この式により，投入コストおよび要員計画から総 SLOC を計算できる．あるいは，要員計画と見積り SLOC から，コストの見積りが得られる．

c_t は，技術定数と呼び，使用するプログラミング技法，プロジェクトチームや要員の経験と能力，開発環境などできまる．具体的には，過去の類似プロジェクトのデータの曲線上から算出する．

Putnam モデルは，SLOC と総工数の関係とともに，要員配分のモデルとしても使え，有効なモデルである[8]．しかし，見積りの基本となる SLOC をいかに得るかについては，経験者が Delph 法などで決めることとしており，根本的な課題は解決していない．

3.2 COCOMO モデル

Boehm が提唱した COCOMO モデル (COnstructive COst MOdel)[18),19)]は，実際によく使われており，適用事例に関する多くの研究報告がある．

開発すべきソフトウェアシステムの規模を SLOC で表わし，それと，プロジェクトの困難度や，開発に影響する他の要因を組み合わせ，コストや開発期間を見積もることになる．

COCOMO モデルには，BASIC モデル，INTERMEDIATE モデルおよび DETAILED モデルの，3種類のモデルがある．

開発プロジェクトの困難度として

（1） ORGANIC モード：比較的小規模のチームで手慣れた開発環境や手法があり，開発ソフトウェアの分野にも経験がある

（2） SEMI-DETACHED モード：経験者と未経験者の混成で，開発システムの経験も限定されている

（3） EMBEDDED モード：ハードウェアとソフトウェアが密に関連し，要求仕様の変更がたびたび起こるような開発で，プロジェクトチームも開発システムの経験が少ない

の3種類が考えられている．

[BASIC モデル]

MM (Man-Months：開発工数) と $KDSI$ (Kilostep of Delivered Source Instructions：出荷 SLOC) の関係は，次の式で表わされる．

$$MM=a\times(KDSI)^b$$

ここで，a,b はプロジェクトの困難度で決まる定数で，Boehm による困難度は，

ORGANIC　　モード：$a=2.4$, $b=1.05$

SEMIDETACHED モード：$a=3.0$, $b=1.12$
EMBEDDED モード：$a=3.6$, $b=1.20$
である．

BASIC モデルでは，以下のように，$TDEV$ (Time to DEVelop：開発期間) の予測を開発工数の見積りと同様の計算式で与える．

$TDEV=a\times(MM)^b$
ORGANIC モード：$a=2.5$, $b=0.38$
SEMIDETACHED モード：$a=2.5$, $b=0.35$
EMBEDDED モード：$a=2.5$, $b=0.32$

たとえば，32 $KDSI$ と見積もられた，ORGANIC モードの開発プロジェクトは

$MM=2.4\ (32)^{1.05}=91$ 人月
$TDEV=2.5\ (91)^{0.38}=14$ カ月

となる．

[INTERMEDIATE モデル]

INTERMEDIATE モデルでは次の手順で，開発コストを見積もる．

(1) BASIC モデルと同様の式から，出荷 SLOC ($KDSI$) とプロジェクトの困難度により，見かけ上の開発工数 (MM_{NOM}) を求める．

$MM_{NOM}=a\times(KDSI)^b$
ORGANIC モード：$a=3.2$, $b=1.05$
SEMIDETACHED モード：$a=3.0$, $b=1.12$
EMBEDDED モード：$a=2.8$, $b=1.20$

(2) 15 のコスト決定要因 (表-2) の程度から，要因乗数を決定する．

(3) 見積り開発工数 (MM_{DEV}) は，見かけ上の工数に，すべての要因乗数を掛け合わせたものである．

$MM_{DEV}=MM_{NOM}\times c(1)\times c(2)\cdots\times c(15)$

(4) 開発工数と他の要素から，出荷時期，要員分布，コンピュータコスト，保守コストなどが決定される．たとえば，開発期間 ($TDEV$) はつぎの式で表わされる．

$TDEV=2.5\times(MM_{DEV})^b$
ORGANIC モード：$b=0.38$
SEMIDETACHED モード：$b=0.35$
EMBEDDED モード：$b=0.32$

DETAILED モデルも INTERMEDIATE モデルとほぼ同様であるが，コスト要因の検討がより詳細になり，工程ごとの工数見積りが可能になる．

Boehm も指摘しているが，このモデルを適用するにしても，そのまま当てはまるとは限らない．組織や開発分野に適合するように，コスト決定要因を取捨選択し，要因乗数などのパラメータ値は，経験に基づき決定する必要がある[15]．

3.3 ファンクションポイント法

ファンクションポイント法 (Function Point：以下 FP と省略) は，Albrecht[3] により開発された手法である．当初は，ソフトウェア開発の生産性の尺度として提案されたが，見積り手法として改良され，現在最も注目されている手法である[2),4),10)]．

基本的な特徴は，ソフトウェアの機能に着目し，機能を数え上げることでその規模を評価することである．SLOC では，要求された機能を実現するための，結果として表わされるコードを数え上げるのに比べ，より，直接的な評価方法といえる．すなわち，開発する対象 (What) と，開発手段 (How) を分離する手法といえる．

FP 法では，ソフトウェアの機能として入力，出力，問合せ処理，マスタファイル，および，

表-2 INTERMEDIATE COCOMO モデルの開発工数乗数[19]

コスト決定要因	程度					
	非常に低い	低い	普通	高い	非常に高い	極端に高い
[製品の特性]						
RELY 信頼性の要求	.75	.88	1.00	1.15	1.40	
DATA データベースのサイズ		.94	1.00	1.08	1.18	
CPLX 製品の複雑さ	.70	.85	1.00	1.15	1.30	1.85
[コンピュータの特性]						
TIME 実行時間の制約			1.00	1.11	1.30	1.66
STOR メイン・メモリの制約			1.00	1.06	1.21	1.56
VIRT 仮想マシンの可能性		.87	1.00	1.15	1.30	
TURN コンピュータの応答性		.87	1.00	1.07	1.15	
[技術者の特性]						
ACAP アナリストの能力	1.46	1.19	1.00	.86	.71	
AEXP アプリケーションの経験	1.29	1.13	1.00	.91	.82	
PCAP プログラマの能力	1.42	1.17	1.00	.86	.70	
VEXP 仮想マシンの経験	1.21	1.10	1.00	.90		
LEXP プログラミング言語の経験	1.14	1.07	1.00	.95		
[プロジェクトの特性]						
MODP 最新プログラミング手法の使用	1.24	1.10	1.00	.91	.82	
TOOL ソフトウェアツールの使用	1.24	1.10	1.00	.91	.83	
SCED 開発スケジュールの要求	1.23	1.08	1.00	1.04	1.10	

インタフェースに注目し，それぞれの個数に対して，重みを掛け，加え合わせた後，それぞれの要素の複雑度を考慮して調整する．

重みを決めるのに，Albrecht は，「議論と試行」によるとしているが，類似プロジェクトの経験やデータの蓄積が重要である．Albrecht による重みは次のとおりである．

- 入力の数　×4
- 出力の数　×5
- 問合せ処理の数　×4
- マスタファイルの数　×10
- インタフェースの数　×7

重み付けした合計に対し，次のような調整を行う．入力，出力あるいはファイルが特に複雑な場合，5% を加える．内部処理が複雑な場合，さらに5% を加える．オンライン処理や性能について考慮して，同様の追加を行う．この調整は，±25%を越えない範囲で行う．ここで計算された結果が，顧客に提供されるソフトウェアの FP となる．

FP 法は，提案されてから後，継続して改良されており，たとえば，FP を求めるためのワークシート（図-2）ではかなり厳密に記述するようになっている．

FP 法は，SLOC に基づく他のモデルに比べ

- 開発すべきソフトウェアについて，比較的早い段階からユーザと議論できる
- ユーザの観点から，入力や出力など，システムの外部仕様に基づき，FP を計算できるので，開発の早い段階にその妥当性が確認できる
- 開発に用いる技術や言語からの影響から切り離して考えられる

などの利点がある．

欧米では，FP に関する研究組織が活動しており，ソフトウェア見積りの国際標準としようとする動きもある．

FP 法は，当初，事務処理ソフトウェア開発を対象として開発されてきたため，FP の計算方法も，入出力やファイルを重視するなど，かなりの影響を受けている．このため，科学技術計算，リアルタイムシステムあるいは組込みソフトウェアの開発などの分野でこのまま使うことは難しい．最近，ファンクションポイント法を拡張しこれらの分野にも適用できるよう，アルゴリズムの複雑さも取り入れるようにした手法の提案がある．

・ファンクションカウント

タイプ ID	記　述	複雑度 単純	普通	複雑	合計
IT	外部入力	_× 3=_	_× 4=_	_× 6=_	___
OT	外部出力	_× 4=_	_× 5=_	_× 7=_	___
FT	論理内部ファイル	_× 7=_	_×10=_	_×15=_	___
EI	外部インタフェースファイル	_× 5=_	_× 7=_	_×10=_	___
QT	外部問合せ処理	_× 3=_	_× 4=_	_× 6=_	___
FC	調整前総ファンクションポイント				___

・処理複雑度

ID	特　性	DI	ID	特　性	DI
C1	データ通信	_	C 8	オンライン更新	_
C2	分散機能	_	C 9	複雑処理	_
C3	性　能	_	C10	再利用性	_
C4	運用負荷	_	C11	設置容易性	_
C5	トランザクション密度	_	C12	運用容易性	_
C6	オンラインデータエントリ	_	C13	複数サイト	_
C7	エンドユーザ能力	_	C14	設備変更	_
PC	総影響度				_

・DI（影響程度）値
・現在は関係なし，あるいは影響なし=0　・影響あり　=3
・たいして影響なし　=1　・かなり影響あり=4
・少し影響あり　=2　・強く影響あり　=5

PCA 処理複雑度調整値　$=0.65+(0.01\times PC)$　=____

FP　ファンクションポイントメジャ　$=FC\times PCA$　=____

図-2　ファンクションポイント計算ワークシート[1]

4. おわりに

ソフトウェア開発工数の見積りは，開発計画や原価管理の基礎となるもので，その重要性については，議論の余地がない．しかし，開発の早い段階で正確に見積もることの難しさは，提案されているいずれの手法を用いても変わりがない．

いずれの手法を用いるにしても，SLOC あるいは FP のように，基本となる値をいずれかの手段で予測する必要があり，これを決めるのは，結局のところ開発に関与する人である．また，いずれのモデルを適用しようとしても，重みやパラメータ

の値は，開発分野や，組織の文化に応じて適合させる必要がある．類似システムの経験などを蓄積し，組織として成熟度を図る必要がある．すなわち，実務的な観点からいえば，どの手法を選ぶかより，いずれの手法を採ったとしても，継続してデータを蓄積し，組織や製品分野に適合するように改良していく組織的な活動と組織の成熟度のほうが重要である[10),11),17),23)]．いささか非技術的な結論になったが，本稿が会員諸氏の参考になれば幸いである．

参考文献

1) Lederer, A. L. and Prasad, J.: Nine Management Guidelines for Better Cost Estimating, Communi. ACM, Vol. 35, No. 2, pp. 51-59 (1992).
2) Albrecht, A. J. and Gaffney Jr., J. E.: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Trans. on SE, Vol. SE-9, No. 6, pp. 639-648 (1983).
3) Albrecht, A. J.: Measuring Application Development Productivity, Proc. of the Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83-92 (1979).
4) Behrens, C. A.: Measuring the Productivity of Computer Systems Development Activities with Function Points, IEEE Trans. on SE, Vol. SE-9, No. 6, pp. 648-652 (1983).
5) Laranjeira, L. A.: Software Size Estimation of Object-Oriented Systems, IEEE Trans. on SE, Vol. 16, No. 5, pp. 510-521 (1990).
6) Lind, R. K. and Vairavan, K.: An Experimental Investigation of Software Metrics and Their Relationship to Software Development Effort, IEEE Trans. on SE, Vol. 15, No. 5, pp. 649-653 (1989).
7) Low, G. C. and Jeffery, D. R.: Function Points in Estimation and Evaluation of Software Process, IEEE Trans. on SE, Vol. 16, No. 1, pp. 64-71 (1990).
8) Warburton, R. D. H.: Management and Predicting the Costs of Real-Time Software, IEEE Trans. on SE, Vol. SE-9, No. 5, pp. 562-569 (1983).
9) Boydston, R. E.: Programming Cost Estimation: Is It Reasonable?, Proc. IEEE 7th ICSE, pp. 153-159 (1985).
10) Sunohara, T., Takano, A., Uehara, K. and Ohkawa, T.: Program Complexity for Software Development Management, Proc. IEEE 5th ICSE, pp. 100-106 (1981).
11) Itakura, M. and Takayanagi, A.: A Model for Estimating Program Size and Its Evaluation, Proc. IEEE 6th ICSE, pp. 104-109 (1982).
12) Putnam, L. H. and Fitzsimmons, A.: Estimating Software Cost, Datamation, pp. 189-198 (Sep.), pp. 171-178 (Oct.) and pp. 137-140 (Nov. 1979).
13) Sommerville, I.: Software Engineering (Third Edition), Chap. 26 Software Cost Estimation, Addison-Wesley, pp. 513-531 (1989).
14) Reifer, D. J.: A Poor Man's Guide to Estimating Software Costs, Texas Instruments, Inc. Tech. Report RC 1-TR-012 (1985).
15) Miyazaki, Y. and Mori, K.: COCOMO Evaluation and Tailoring, Proc. IEEE 8th ICSE, pp. 292-299 (1985).
16) Gaffney Jr., J. E.: The Impact on Software Development Cost of Using HOL's, IEEE Trans. on SE, Vol. SE-12, No. 3, pp. 496-499 (1986).
17) Okada, M. and Azuma, M.: Software Development Effort Estimation Study—A Model from CAD/CAM System Development Experiences, Proc. IEEE COMPSAC '82, pp. 555-564 (1982).
18) Boehm, B. W.: Software Engineering Economics, IEEE Trans. on SE, Vol. SE-10, No. 1, pp. 4-21 (1984).
19) Boehm, B. W.: Software Engineering Economics, Englewood Cliffs, NJ, Prentice-Hall Inc, (1981).
20) Hihn, J. and Habib-agahi, H.: Cost Estimation of Software Intensive Project: A Survey of Current Practices, Proc. IEEE 17th ICSE, pp. 276-287 (1991).
21) Halstead, M.: Elements of Software Science, New York: Elsevier (1977).
22) McCabe, T.: A Complexity Measure, IEEE Trans. ES, Vol. SE-2 (1976).
23) 小林，栃尾：YPS 適用における最適モジュールサイズと作業時間の見積技法，FUJITSU, 42, 4, pp. 381-389 (1991).
24) 上条：ソフトウェアのコストについて—見積りと実績，情報処理，Vol. 21, No. 10, pp. 1050-1056 (1980).

(平成4年6月8日受付)

大篠　豊（正会員）

1967年京都大学工学部数理工学科卒業．1969年同修士課程修了．同年(株)東芝総合研究所情報システム研究所に入所．以来，CADシステム，言語コンパイラ，ソフトウェア開発支援システムの研究開発に従事．現在，同社システム・ソフトウェア技術研究所研究第三部部長．電子情報通信学会，ソフトウェア科学会各会員．