混乱プロジェクトの予測へのベイジアンネットの適用

水野修 †, 天嵜聡介 †, 山之内太, 菊野亨 †, 高木徳生 †‡

† 大阪大学 大学院情報科学研究科 情報システム工学専攻
E-mail: {o-mizuno,amasaki,kikuno}@ist.osaka-u.ac.jp
‡ オムロン (株) ソーシアルシステムズ・ソリューション＆サービス・ビジネスカンパニー
公共ソリューション事業部事業企画部インテグレーション課
E-mail: yasunari_takagi@omron.co.jp

Abstract

ソフトウェア開発の現場では，プロジェクトが混乱状態に陥らないようにあらかじめ問題要因を探り，混乱の可能性を早期に予測することが望まれている．これまでに，プロジェクトマネージャに対してアンケートを実施することにより問題要因を特定し，混乱という事象に対処するために問題要因のパラメータの値に回帰モデルやクラスタ分析を適用して混乱を推定する手法の提案をしてきた．しかし，プロジェクトの早期段階でパラメータの値が全て判明していることはまれであり，それが原因となってプロジェクトの早期段階での上記モデルの適用は困難であった．

本研究ではソフトウェア開発プロジェクトが混乱するかどうかを，前もって収集されているリスク要因の分析結果も利用して予測するモデルの提案を行う．そのために，ベイジアンネットを用いた混乱予測モデルを作成する．ベイジアンネットを用いることにより，一部のパラメータの値が不明であっても事前に与えられた確率分布を利用して混乱の確率を算出できるようになる．次に，実際のソフトウェア開発現場から収集したデータを適用し，評価実験を行う．実験の結果，プロジェクトの早期段階でも非常に高い精度で混乱予測が可能となることを確認した．

1 はじめに

ソフトウェアを取り巻く利用要求の急速な高度化に従い，ソフトウェアの開発期間が短くなり，高信頼性も求められるようになってきている．そうした状況下では，ソフトウェア開発プロジェクトにおける問題点を早めに予測し，対応することが不可欠になってきている．ところが実際の開発現場では，プロジェクトの進行状況すら誰にも把握できなくなるような状況に陥る (それを混乱状態と呼ぶ) プロジェクトが発生することがある．このようなプロジェクトは多くの場合に「混乱状態」に陥ることを繰り返した後，最悪の事態に発展する [13]．混乱状態に陥ったプロジェクトが必ず最悪の事態に発展するわけではないが，混乱状態に陥るプロジェクトを事前に予測できれば，最悪の事態が起こることも回避できると考えられる．

そうした事態を回避するための一手法として，ソフトウェアプロジェクトのリスク管理に関する研究が盛んになされてきている．ソフトウェア開発プロジェクトにおけるリスクの分類は Boehm らによって行われたもの [2] をはじめ，多くの研究者たちによってなされてきている [8,12]．また，リスクを分析してプロジェクトの最終状態に関する予測を行う研究も行われている [1,6]．

我々はこれまでに，現場の開発者に対するアンケートに基づき問題要因を分析し，それとプロジェクトの混乱との関係を調査してきた．そして，プロジェクトマネージャへのアンケート調査に基づきプロジェクトの混乱を予測する手法を提案してきている [9,14]．

これまでの研究では主に回帰分析によるモデル作成を行ってきたが，回帰分析ではモデルを使用する際に全ての問題要因に対するアンケート結果が判明している必要があるため，現実の環境への適用が難しいという問題があった．つまり，事前に混乱を予測する場合，判明している問題要因の値が欠けていると予測ができなくなる恐れがあった．しかも，実際の開発現場では全ての問題要因の値が判明した後にプロジェクトの混乱予測をしていたのでは遅すぎるため，より実用的にするためにはもっと早い段階での混乱予測が求められていた．

そこで，本研究ではベイジアンネットを用いた混乱予測

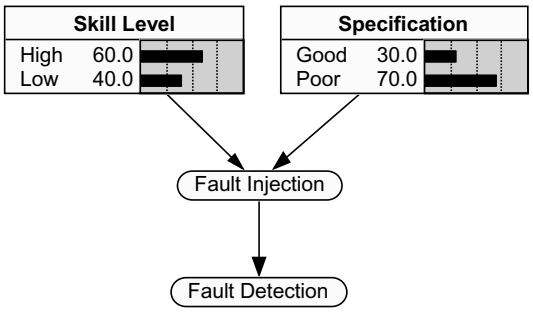

図 1. ベイジアンネットの例

表 1. 変数 SL の事前確率分布表

High(H)	Low(L)
0.6	0.4

モデルの作成を試みる．これにより問題要因の一部の値が分からない場合でも確率分布を用いてプロジェクトの混乱を予測することが可能になる．まず，過去のアンケートから問題要因を抽出し，ベイジアンネットを用いたプロジェクトの混乱予測モデルを構築する．これにモデル作成に用いたデータを適用してモデルの妥当性を確かめる．次に，実際に問題要因の値が未設定の場合の例として，早期に混乱プロジェクトかどうかを予測する場合を考える．そこでは比較的早期に判明する問題要因の値のみを抽出し，モデルに適用して混乱の予測を試みる．

2 ベイジアンネット

ベイジアンネット (Bayesian Brief Network: BBN) とは，得られた情報を利用することで不確定な要因を含む場合にその要因に対応付けられた変数の確率分布の評価を可能とする確率モデルである [4, 11]．BBN では対象の変数間の依存関係を表すために，各変数をノードで表す．さらに変数間を有向のリンクで結び，それらのパスが循環しないようにした非循環有向グラフとして表す．

確率変数 X_i, X_j の間で $X_i \rightarrow X_j$ という依存関係が存在する時，X_i を親ノード，X_j を子ノードと呼び，この依存関係を条件付確率 $P(X_j|X_i)$ で表す．BBN では全ての確率変数 $X_1, \cdots, X_n$ 間の依存関係を結んだネットワークを構築する．この時，確率変数が離散値をとるなら，条件

表 2. 変数 SP の事前確率分布表

Good(G)	Poor(P)
0.3	0.7

表 3. 変数 FI の条件付き確率分布表

SL	SP	True(T)	False(F)
H	G	0.1	0.9
H	P	0.2	0.8
L	G	0.4	0.6
L	P	0.8	0.2

表 4. 変数 FD の条件付き確率分布表

FI	True(T)	False(F)
T	0.7	0.3
F	0.1	0.9

付確率は全ての場合の確率の値を書いた表 (条件付確率分布表: Conditional Probability Table(CPT)) で表すことができる．

BBN による確率推論とは観測された変数の値から，知りたい確率変数の確率を求めることである．具体的には以下の手順で計算を行う：(1) 観測された変数の値をノードに与える．(2) 親ノードも観測値も持たないノードに事前確率分布を与える．(3) 知りたい対象の変数の事後確率を求める．事後確率を求めるために既知の値からの確率伝播 (変数間の局所計算) によって各変数の確率分布を更新していく計算が，BBN 上の計算アルゴリズムの特徴である．

例 図 1 はフォールトの混入，検出の単純なモデルを BBN を用いて表したものである．Fault Injection (変数 FI) の親ノードが Skill Level (変数 SL) と Specification (変数 SP) であり，また子ノードが Fault Detection (変数 FD) となっている．つまり，不具合の混入はスキルレベルと仕様の出来に依存し，不具合の発見は不具合の混入にのみ依存するというモデルである．各ノードの確率変数は 2 値の値をとり，それぞれ表 1 から表 4 に示す確率分布が与えられているものとする．

ここでは図 1 を用いて簡単な計算の例を示す．開発者のスキルレベルが低いと分かっており ($SL = L$)，かつ，不具合が発見されない ($FD = F$) 時に，不具合が混入している確率 (FI) の分布を求めることを考える．

これは，変数 FI に対する $P(FI|SL = L, FD = F)$ という確率分布を求めることになる．ベイズの定理よりノードの確率は，

$$P(FI|SL = L, FD = F) = \frac{P(FI|SL = L)P(FD = F|FI)}{P(FD = F|SL = L)}$$

ここで，$\alpha = \frac{1}{P(FD=F|SL=L)}$ を FI の値によらない正規化定数とすると，$SL = L$ と $FD = F$ は FI が与えられると条

件付独立になるので,

$$P(FI|SL = L, FD = F) = \alpha P(FI|SL = L)P(FD = F|FI)$$

となる．このうち $P(FI|SL = L)$ は次の計算によって求めることができる．

$$\begin{aligned} P(FI = T|SL = L) &= P(FI = T|SL = L, SP = G)P(SP = G) \\ &+ P(FI = T|SL = L, SP = P)P(SP = P) \\ &= 0.4 \times 0.3 + 0.8 \times 0.7 \\ &= 0.68 \end{aligned}$$

同様に,

$$P(FI = T|SL = L) = 0.32$$

となる．また，$P(FD = F|FI)$ については，表 4 の CPT より,

$$\begin{aligned} P(FD = F|FI = T) &= 0.3 \\ P(FD = F|FI = F) &= 0.9 \end{aligned}$$

となる．ここで α について $P(FD = F|SL = L)$ を求めると,

$$\begin{aligned} P(FD = F|SL = L) &= P(FD = F|FI)P(FI|SL = L) \\ &= P(FD = F|FI = T)P(FI = T|SL = L) \\ &+ P(FD = F|FI = F)P(FI = F|SL = L) \\ &= 0.3 \times 0.68 + 0.9 \times 0.32 \\ &= 0.492 \end{aligned}$$

となる．よって，求めたい分布は,

$$\begin{aligned} P(FI = T|SL = L, FD = F) &= \frac{0.68 \times 0.3}{0.492} = 0.415 \\ P(FI = F|SL = L, FD = F) &= \frac{0.32 \times 0.9}{0.492} = 0.585 \end{aligned}$$

となる．同様の計算を行うことによって，図 2 に示すようにノード Specification(変数 SP) に対する事後確率分布も求めることができる．

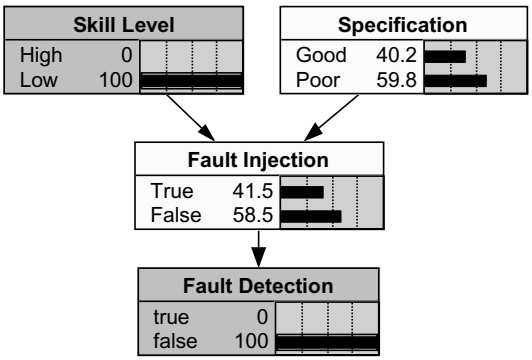

図 2. 求められた確率分布

3 BBN に基づく混乱予測モデル

3.1 混乱プロジェクトの定義

開発現場の状況が制御不可能になってしまうプロジェクトを混乱プロジェクトと呼ぶ [13]．しかし，これでは漠然としているので，ここでは混乱プロジェクトを次のように定義する．

基本的には開発コストと開発期間の 2 つについて開発計画作成時の推定値と実績値のずれを見て，いずれかが基準値を超えていれば，混乱プロジェクトであると判断する．しかし，これでも必ずしも十分ではないので，プロジェクト管理者にインタビューを行って同意が得られたものだけを最終的に混乱プロジェクトと定めることにする．

3.2 問題分析アンケート

ここでは，これまでの研究 [9, 14, 15] でも利用してきた問題分析アンケート (表 5) について説明する．

問題分析アンケートは，プロジェクトマネージャが無意識につかんでいると思われる混乱要因を引き出す目的で作成された．問題分析アンケートの設計にあたっては，リスク管理に関する専門書や論文 [3, 5, 7]，また協力企業における経験に基づくチェックリストを調査した．その結果，混乱プロジェクトを引き起こす問題要因を次の 5 つに整理した．

(1) 要求仕様に関する問題点

(2) 見積りに関する問題点

(3) 工程計画作成に関する問題点

(4) 組織体制に関する問題点

(5) プロジェクト管理に関する問題点

これらはそれぞれより詳細なレベルの項目に展開されている．例えば要求仕様に関する問題点は，(R-1) 要件が不明確なままでの要求，(R-2) 要件の説明力不足，(R-3) 要件の理解不足，(R-4) 要件に対する顧客側，実現側相互の合意不足，(R-5) 要件，仕様変更管理不足と詳細化されている．

評価についてはプロジェクト管理者が “ある”，“どちらでもない”，“ない”，“分からない” のいずれかを記入する．なお，本研究では問題分析アンケートへの回答を分析するための基礎データとするために，“ある”，“どちらでもない”，“ない” という回答に評価値 3，2，0 を対応づける．また “分からない” という回答については，問題要因はプロジェクト中で見受けられなかったが “ない” とは言い切れない，と解釈して評価値 1 を対応づける．

表 5. 問題分析アンケート

R. 要求仕様	
R-1	要件が不明確なままでの要求
R-2	要件の説明力不足
R-3	要件の理解不足
R-4	要件に対する顧客側，実現側相互の合意不足
R-5	要件，仕様変更管理不足
E. 見積り	
E-1	見積りの重要さの認識不足
E-2	見積りの根拠不足
E-3	見積りの項目不足
E-4	技術的課題に対する楽観的過ぎる見積り
E-5	非技術的圧力に妥協
P. 計画作成	
P-1	計画に対する上級マネージャのレビュー不足
P-2	作業に対する責任分担不明確
P-3	作業成果物定義不十分
P-4	マイルストーンやレビュー時期の設定不足
P-5	進捗管理方法不明確
P-6	計画に対する関係者全員のコミットメントなし
O. プロジェクト体制	
O-1	要員のスキル不足
O-2	プロジェクト体制整備の必要性認識不足
O-3	モラール不足
M. プロジェクト管理	
M-1	工数不足
M-2	進捗状況報告不足
M-3	進捗管理データ不足

3.3 分析対象データ

あるソフトウェア開発企業の協力により，1996 年から 1998 年までに開発された 40 プロジェクト (PR1～PR40) に対して表 5 に示すアンケートを実施し，その結果の提供を受けた．なお，これらのプロジェクトは全て開発が終了しており，混乱したかどうかの判定もすでに SEPG によって行われている．アンケートの結果は表 6 に示す形式で得られている．

3.4 ベイジアンネットに基づくモデル化

上記のデータを用いてプロジェクトの混乱を予測するために，まず，ベイジアンネットによるプロジェクトの混乱の因果関係モデルを作成する．ベイジアンネットの作成に当たっては，それぞれの問題要因間の関係なども考慮に入れたモデルを作成することもできる．今回は簡単化のため，プロジェクトの混乱という事象には「要求仕様」「見積り」「計画作成」「プロジェクト体制」「プロジェクト管理」の 5 要因が直接に影響を及ぼすものと考えた．そして，この 5 要因にはそれぞれ分類に属するアンケート項目が示す要因が直接に関連していると考えた．

表 6. アンケート結果

プロジェクト	R-1	R-2	R-3	···	M-3
PR1	0	0	0	···	0
PR2	0	0	0	···	0
PR3	0	0	0	···	0
PR4	3	3	2	···	0
PR5	0	0	0	···	0
PR6	0	3	2	···	2
PR7	0	0	2	···	0
PR8	0	2	3	···	0
PR9	0	2	0	···	2
PR10	2	2	0	···	3
:	:	:	:		:
PR40	2	2	2	···	2

以上の考えに基づいて作成したモデルを図 3 に示す．各要因は，アンケートでの回答に応じて 4 段階の離散値を持つ変数とする．また，最終的な目的変数「TROUBLED」は「混乱」「成功」の 2 値をとる離散変数とする．

図 3 のモデルに対して，先に述べたプロジェクトのうち 1996 年と 1997 年に行われた 32 プロジェクト (PR1～PR32) のデータを与えて確率分布表を作成する．なお，一連の BBN モデルの構築作業には Norsys Corporation の Netica [10] を使用した．

3.5 モデルの適合度の確認

作成された混乱予測モデルのデータへの適合度を確認するために，モデルを作成する時に用いた 1996 年と 1997 年に実施されたプロジェクトのアンケートデータをモデルへ適用して，各プロジェクトに対して混乱する確率の計算を計算を行った．ここで，モデルにデータを適用するとは，BBN の各変数に対してある値になる確率が 1 になると定めることである．

その結果，計算された変数「TROUBLED」の確率が 0.5 以上であれば「混乱」と予測し，そうでなければ,「成功」と予測した．実際の混乱プロジェクト数とモデルが予測した混乱プロジェクト数を比較した結果を表 7 に示す．

表 7 中の列は混乱予測モデルが予測したプロジェクトの数を表す．つまり，列が「混乱」ならモデルはプロジェクトが混乱すると予測,「成功」ならプロジェクトが混乱しないと予測する．一方，表 7 中の行は実際にプロジェクトが混乱したかどうかを表す．つまり，行が「混乱」ならプロジェクトが実際に混乱した,「成功」ならプロジェクトが実際は混乱しなかった，という事象を表す．

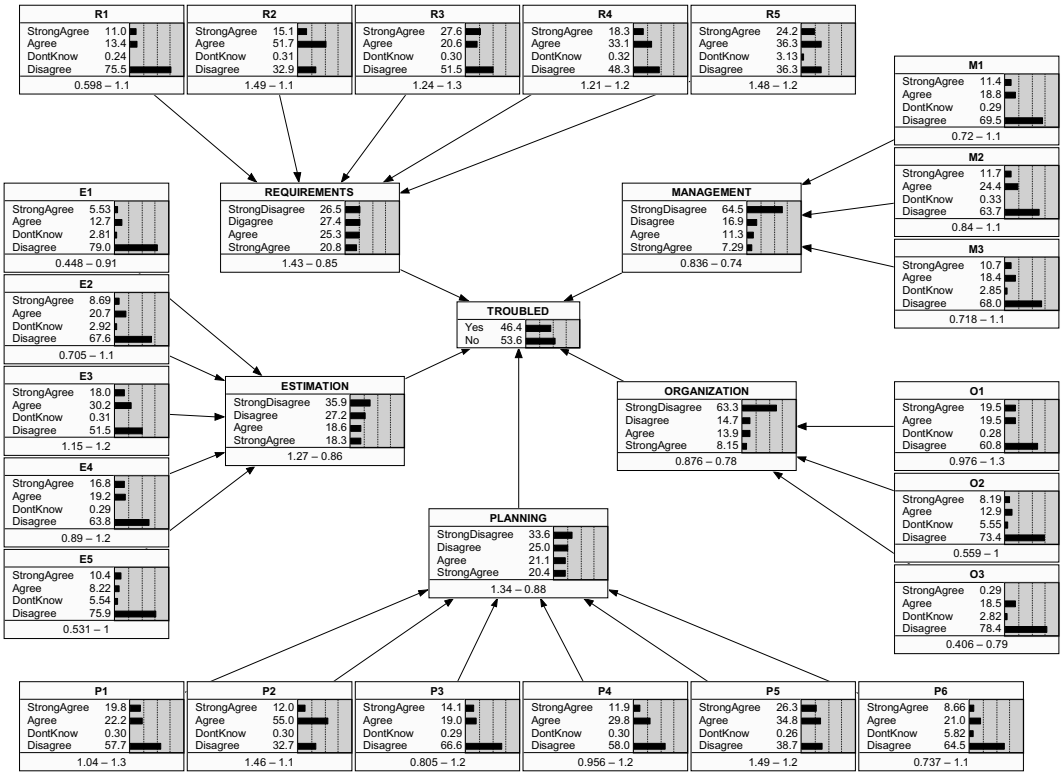

図 3. BBN に基づく混乱予測モデル

表 7. データの適用結果

実測	予測	
	混乱	成功
混乱	9	1
成功	0	22

この結果，32 プロジェクトに対するデータの当てはめはおよそ 97 %の精度でモデルの結果と一致している．このことから，モデルのデータへの適合度は非常に高いことが分かる．

4 適用実験

ここでは，モデルを作成するのに用いたものとは別のデータセットを用いた 2 つの適用実験を行い，提案する手法の有効性を確認する．どちらの実験でも利用するのは 1998 年に実施されたプロジェクト (PR33〜PR40) から得られたアンケート結果である．

実験 1 すべての問題要因 (アンケート項目) を予測モデルへの入力として与えて，予測結果を調べる．

実験 2 一部の問題要因のみを予測モデルへの入力として与え，予測結果を調べる．

なお，BBN の説明でも述べたが，実験 2 のように一部の値のみを与えても意味のある結果を得ることができるのが，本手法の特長の 1 つである．

4.1 適用実験 1: すべての問題要因を利用した予測

まず，8 件のプロジェクト (PR33〜PR40) から得られた問題要因をモデルに入力し，それぞれのプロジェクトに対して混乱する確率を求めた．その確率が 0.5 以上となるものを「混乱」と予測し，それ以外のものを「成功」と予測した．その結果をまとめて，実際の混乱プロジェクト数とモデルが予測した混乱プロジェクト数を比較した．表 8 にその結果を示す．

この結果，8 件のプロジェクトに対して BBN を用いた混乱予測モデルは 75 %の精度でプロジェクトが混乱するかしないかを正しく予測している．

表 8. データの適用結果

実測	予測	
	混乱	成功
混乱	3	0
成功	2	3

4.2 適用実験 2: 早期に判明する問題要因を利用した予測

BBN を用いた混乱予測モデルでは，一部のパラメータの値が未設定の場合でも事前に与えられた確率分布に基づいて混乱の確率を算出できるようになる．

そのためプロジェクトの問題要因の中で，早期に判明すると思われる要因のみを抽出して図 3 の混乱予測モデルに適用した．以下に取り出した要因を示す．

R. 要求内容不明確・不確定・把握不十分

R.1 ソフトによる実現を要求する側が、何を要求したいか分かっていなかった。

E. 過小見積

E.2 過去の成功パターンで安易に見積った (見積もり根拠が不明確であった)

E.5 政治的圧力に妥協してしまった．

P. 工程計画不十分

P.1 マネージャによる実現性の検証不足．

P.2 作業分割構造（ＷＢＳ）およびプロジェクト組織構造の明確化不十分．

P.3 各作業分担の工程毎成果物の定義が不十分．

P.4 マイルストーン、チェックポイント、レビュー時期の設定がなかった．

P.5 進捗管理方法不明確．

P.6 計画に対する設計者全員のコミットメントがなかった。（担当者は実現不可能と思っていた）．

M. プロジェクト管理

M.3 管理のための最小限のデータ (進捗，工数など) 収集を行っていなかった．

8 件のプロジェクト (PR33～PR40) から収集されたデータから，これら 10 個の問題要因に関するデータだけを取り出したものを実験 1 と同様の要領で予測モデルに適用した．このとき，上に挙げたもの以外の要因については，モデルの構築時に作られた確率分布を利用する．

実際の混乱プロジェクト数とモデルが予測した混乱プロジェクト数を比較した結果を表 9 に示す．

表 9. 早期に判明するデータの適用結果

実測	予測	
	混乱	成功
混乱	2	1
成功	0	5

この結果，8 件のプロジェクトに対して混乱予測モデルは 87.5%の精度でプロジェクトが混乱するかしないかを正しく予測している．これはモデルの予測精度として十分実用可能な値である．この結果，BBN を用いた混乱予測モデルがプロジェクト早期の段階での混乱予測にも有効に利用できると考えられる．

4.3 考察

ここでは，実験 1 と実験 2 の結果について考察する．実験 1 と実験 2 の予測精度 (75%と 87.5%) を比較すると，多くの情報を与えた実験 1 のほうが実験 2 よりも低い予測精度となっている．この原因として，問題要因として抽出した要因の中に，予測を正しくない方向へと導いてしまう要因が入っている可能性が挙げられる．現時点では，その原因特定にまでは至っていないが，今後の研究の中で，そうした要因の影響を分析していく必要があると考えている．

今回用いた混乱予測モデル (図 3) は，各要因間の依存関係を表したモデルとはなっていないため，すべての問題要因が混乱という事象だけに関与する形となっている．そのため，上で述べたような現象が発生しているとも考えられるので，要因間の依存関係の分析などを通じて，今後より詳細なモデル化を行う必要があると考えている．

また，今回の実験で利用したデータの総数もそれほど多いとは言えない．より多くのデータを対象とした統計的な分析を実施していく必要がある．

5 まとめと今後の課題

本研究では，ソフトウェアプロジェクトの混乱をベイジアンネットを用いて予測する手法を提案した．ベイジアンネットを用いることにより，全ての問題要因をモデルに取り込みつつ，そのうちの一部のパラメータの値だけを指定した場合でも予測が可能になった．すなわち，開発初期の段階で，全ての問題要因が判明していないような場合についても，より正確な予測を提供できると期待される．提案手法の有効性の評価のために，実際のソフトウェア開発現場から収集した問題要因のデータをモデルに適用し，評価実験を行った．実験の結果，プロジェクトの早期段階で収集できるメトリクスのみを用いた場合でも，非常に高い精度での混乱予測が可能となることを確認した．

今後の課題としては，問題要因相互の依存関係をより詳細に分析していく必要がある．また，広範なデータの収集を行い，モデルのパラメータ設定を充実させなければならない．さらに，従来行ってきた研究で得られた知見を利用し，より精度の高いモデルを構築していくことが必要である．

参考文献

[1] T. R. Adler, J. G. Leonard, and R. K. Nordgren. Improving risk management: moving from risk elimination to risk avoidance. *Information and Software Technology*, 41:29–34, 1999.

[2] B. W. Boehm. Industrial software metrics top 10 list. *IEEE Software*, 4(5):84–85, 1987.

[3] E. H. Conrow and P. S. Shishido. Implementing risk management on software intensive projects. *IEEE Software*, 14(3):83–89, 1997.

[4] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic networks and expert systems*. Springer-Verlag, 1999.

[5] R. E. Fairley and P. Rook. Risk management for software development. In *Software Engineering*, pp. 387–400. IEEE CS Press, 1997.

[6] J. Jiang and G. Klein. Software development risks to project effectiveness. *Journal of Systems and Software*, 52:3–10, 2000.

[7] D. W. Karolak. *Software Engineering Risk Management*. IEEE CS Press, CA, 1996.

[8] J. Kasser and V. R. Williams. What do you mean you can't tell me if my project is in trouble? *DoD Software Tech News*, 2(2), 1998.

[9] O. Mizuno, T. Kikuno, Y. Takagi, and K. Sakamoto. Characterization of risky projects based on project managers' evaluation. In *Proc. of 22nd International Conference on Software Engineering*, pp. 387–395, 2000.

[10] Norsys Corporation. Netica. http://www.norsys.com/.

[11] S. Russell, P. Norvig, 古川康一監訳. Artificial Intelligence: A Modern Approach (エージェントアプローチ人工知能). 共立出版, 1997.

[12] R. C. Williams, G. J. Pandelios, and S. G. Behrens. Software risk evaluation (sre) method description (version 2.0). Technical Report CMU/SEI-99-TR-029, Software Engineering Institute, 1999.

[13] E. Yourdon. *Death March : The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects*. Prentice-Hall Computer Books, 1997.

[14] 足立卓也, 水野修, 菊野亨, 高木徳生, 坂本啓司. アンケート調査に基づく開発中のプロジェクトの混乱予測とその予測作業支援システムの開発. ソフトウェアシンポジウム 2000 論文集, pp. 146–153, Jun. 2000.

[15] 濱崎考成, 水野修, 菊野亨, 高木徳生. リスク管理のためのアンケート回答のクラスタ分析と混乱プロジェクト発見への応用. ソフトウェアシンポジウム 2002 論文集, pp. 159–166, July 2002. 松江.