ソフトウェア工学 32－2
（１９８３．９．２８）

バグ重大度の定量化の試み

福田　昌弘
（富士通株式会社）

1. はじめに

ソフトウェア製品の品質を向上させるためには，品質を定量的に分析，把握し，それに基づいて適切な対策をとることが必要である。中でも品質の重要な要因である信頼性を対象とする場合には，バグを分析することが不可欠である。バグの分析にあたっては，件数がまず基本となる。しかしオペレーティングシステム（以下ＯＳと略す）のバグを分析する場合，システムダウンを引き起すような重大なバグと，メッセージの綴りミスのような軽微なバグとを同列に扱うことは，利用者の立場から見れば実感に合わない。この解決策として，バグの件数だけでなく，その内容も考慮してバグをとらえ，プログラムの信頼性を評価するために，利用者に与える影響を中心とした「バグ重大度」という指標を導入することが一つの方法であると考える。

バグ重大度については，例えばシステムダウンを引き起すバグを重大なものとして，その他のバグとは別に取り扱うといった形で，これまでにも定性的には用いられてきている。しかし次に挙げるような問題があるため，特に大規模なソフトウェア製品においては，定量的に用いられることは殆どなかった。

(1) 重大度を構成すると考えられる要因が数多く存在し，しかもその中には利用者の感覚に依存するものがあるため，特定の理論から重大度の算出式を演繹的に導き出すことは困難である。

(2) ＯＳのように時には数メガステップに及ぶような大規模な製品においては，対象となるバグ情報が多量かつ多岐にわたるため，これらを確実にしかも効率よく分析し蓄積する手段を持たないと，重大度の算出式の導き出し及び検証が行えない。

(3) よく研究対象として取り上げられる数キロステップから数十キロステップの規模のプログラムでは，少数の調査者で全てのバグ情報をとらえることができる。しかし大規模な製品を対象とする場合には，多数の分析者による分析作業の分担が不可欠であり，このため分析結果に担当者間のバラツキが生じやすい。

しかしながら，以上の問題を解決し定量的に取り扱う手段を講じないと，大規模なソフトウェア製品のバグの重大度を的確に把握することは不可能である。そこで次章に述べるような方法で，これらの問題を解決しながらバグ重大度の定量化を試みた。

2. バグ重大度の定量化

次のようなアプローチでバグ重大度の定量化を行った。

(1) バグ重大度を構成する要因の洗い出し
(2) (1)で選ばれた要因の観測方法の確立
(3) バグ重大度指数算出式の導出と検証

2.1 バグ重大度を構成する要因の洗い出し

バグ重大度というのは，従来定性的で漠然としたものであったので，これを定量化するために，まず重大度を構成する要因と考えられるものを洗い出し整理した。洗い出しにあたっては，利用者の観点から見た重大度ということに的を絞り，ＯＳのプログラム作成者でなく，実際にＯＳを利用している人間が中心となっておこなった。

このような作業の結果，要因を次の５つに集約した。

a　バグが利用者群に与える影響の度合
b　バグが顕在化した時点で発生する現象
c　バグの顕在化を抑止するための代行手段，回避手段の有無
d　バグが顕在化する以前の状態に戻す作業の困難さの度合
e　バグが顕在化するための条件

例えば要因ｃやｄは，プログラム作成者による従来のバグ分析では余り重視されていない要因であるが，利用者にとっては重要な要因であると判断した。

2.2　要因の観測方法の確立

ＯＳに代表される大規模製品においては，バグ情報が多量かつ多岐にわたる。これらの情報を確実に管理するために，我々はこれまで個々のバグについて，トラブルシューティングに関する情報や修正処置に関する情報をデータベース化して一元管理している。バグ重大度を構成する要因の観測を，バグの検出過程に合わせて定常的にかつ出来るだけ少ないコストで行えるようにするために，この既存のバグデータベースを観測の基点とした。すなわち図－１に示すように，「バグ分析票」と呼ばれる定型の帳票を用意し，バグデータベースに新しいバグが記録されるたびに，バグ重大度の構成要因の採点結果を蓄積するという方式をとることにした。

先に述べたように，バグ重大度を構成する要因を５つに集約したが，いずれも観測値が定量的な形で直接得られるものではないため，次に観測値を定量的に表現するための評価尺度を作成した。すなわち各要因について，これまでのバグ事例からバグの分類を行い，個々の要因における重大度に合わせて数値コードを割り当てた。その詳細を表－１～表－５に示す。

続いて，この評価尺度を用いて，実際に検出されたＯＳのバグの中から選んだサンプルを対象に，バグ重大度の要因についての分析及び採点作業を試行した。なお分析及び採点作業を複数の人間が分担して行う場合に問題となる担当者間の判断のバラツキを最小限にするために，予め「分析手続マニュアル」を準備し，各担当者はこれに従って分析を行う方式をとった。試行にあたっては，デルファイ法を参考にして次のような方法でリファインを繰り返した。

(1)　実際のＯＳのバグの中から一部をランダムに抽出し，複数の分析者が「分析手続マニュアル」に従って，各々独立に分析及び採点を行う。
(2)　(1)の採点結果を集計し，各分析者間のバラツキを検討してその原因を探る。
(3)　(2)で明らかとなったバラツキの原因を取り除くために，「分析手続マニュアル」の説明を修正，追加したり，事例を補足する。さらに評価尺度についてもコードの統合，分割を行う。
(4)　(2)の集計結果を各分析者にフィードバックするとともに，(3)で改良した分析方法を用いて，新たなサンプルについて，(1)の作業に戻る。

以上の試行を通じて，バグ重大度を構成する各要因の観測値を，分析者間のバラツキの少ない形で得る方法が出来上がった。

2.3　バグ重大度指数算出式の導出と検証

前節で述べた観測方法により，実際に検出されたＯＳのバグについての分析及び採点結果を蓄積すると同時に，ここで得られたバグ重大度の要因の観測値を用いて，バグ重大度指数算出式の導出及び検証を行った。

各要因についての観測値は，前節で述べたように個々の要因における重大度に対応したものとなっているので，これらを組み合せることによりバグ重大度指数が求められる

と考えた。そこでこれらの各要因の観測値を和又は積の形で結合した複数の算出式モデルを，バグ重大度に関するこれまでの定性的な経験則を満たす形で導き出した。例えば代行・回避手段のないバグの方が，それらがあるバグより利用者にとって重大であるという関係を満足するようにし，いずれも利用者にとり重大なバグの方が，重大度指数の値が大きくなるようにした。

次にこれらの算出式モデルの間で，最もよくバグの重大度を表しているものを選択するために，以下のようにして評価検証をおこなった。

(1) 分析及び採点作業の終わったバグの中から，2件ずつのバグの組み合わせをランダムに抽出する。
(2) (1)で抽出された2件ずつのバグの組み合わせの各々について，いずれが重大であるかを，複数の評価者により判定する。
(3) (1)で抽出されたバグの各々について，先に導き出した複数の算出式モデルに各要因の採点結果を代入し，重大度指数を算出する。そして各々のバグの組み合わせについて，重大度指数の大小関係を比較する。
(4) (2)で得られた人間による判定結果と，(3)で得られた算出式モデルによる判定結果を比較する。ここで値の大小関係のみに着目して比較を行っているのは，一方のバグが他方のバグの何倍重大であるかということまで人間に判断させるのは，個人差が大きくなりすぎるためである。

以上の評価検証作業の結果，次に示す算出式(1)が，人間の判断に最もよく一致した。

$$X = \ln\{(\Sigma a + 1) * (b + 1) * (\Sigma c + 1) * (\Sigma d + 1) * \Sigma e\} \quad \cdots\cdots (1)$$

X：バグ重大度指数
a：バグが利用者群に与える影響の度合
b：バグが顕在化した時点で発生する現象
c：バグの顕在化を抑止するための代行手段，回避手段の有無
d：バグが顕在化する以前の状態に戻す作業の困難さの度合
e：バグが顕在化するための条件

〔注〕Xがとりうる値の範囲は，0.69～9.47である。

検証のためのデータが充分に揃っていないため，現時点で算出式(1)がバグ重大度指数を算出するための最適解であるかどうかは決定できないが，人間の判断との相違が5%以下であったので，実用に耐えうると考え，(1)式を採用した。

3. 大規模製品への適用結果

前章で述べたバグ重大度の構成要因の観測方法及びバグ重大度指数算出式(1)が，ソフトウェア製品の信頼性を向上させるためのバグ分析作業に有効であるかどうかを確かめるために，これらを用いて実際の大規模ソフトウェア製品のバグの重大度についての調査を行った。その結果について以下に述べる。なおここで対象とした製品は，大型の汎用ＯＳであり，プログラムの規模は数メガステップである。

3.1 バグの重大度指数別分布

対象ＯＳの全検出バグについて，算出式(1)により求めた重大度指数Xに従って，その分布を示したものが図－2である。個々のバグの重大度指数が，全体として正規分布に近い分布をしていることがわかる。この結果は，従来の定性的にとらえた感覚とも矛盾せず，算出式(1)の妥当性の一例証といえる。又全検出バグのうち，システムダウンを引き起こしたバグについて見ると，重大度の値の大きい方に偏って分布しており，システムダウンを引き起すバグは重大であるという従来の定性的にとらえた感覚をよく表して

いる。さらに細かく見るとシステムダウンを引き起さないバグにも重大度の値の大きいものがあるなど，単にシステムダウンを引き起すか否かだけによるバグの重大度のとらえ方では不充分だった点についての，今回の試みによる改善の効果が表れていることがわかる。

3.2 バグ重大度指数の階層によるコンポーネント別バグ割合の変化

対象ＯＳを構成する各コンポーネント別のバグ件数の割合が，算出式(1)により求めた重大度指数Ｘによる階層ごとにどう変化するかを示したものが図－３である。重大度指数Ｘの大きい階層ほどコンポーネントＢのバグの占める割合が高いことがわかる。このことは，コンポーネントＢのバグを減らすことが，対象ＯＳにおいてバグによる利用者への悪影響を減らすためのキーポイントとなることを示唆しているといえる。すなわち対象製品の品質向上対策に際して，改善のキーポイントの発見にこの方法が有効な手段であるといえる。

3.3 バグ重大度指数による重み付けをしたコンポーネント別バグ件数割合

対象ＯＳを構成する各コンポーネント別のバグ件数の割合について，算出式(1)により求めた重大度指数Ｘによる重み付けが，どのような効果を表すかを示したものが図－４である。ここで重み付けに用いた重大度指数Ｘの最大値と最小値の差が小さいため，重み付けをした場合としない場合との差は僅かである。ただし重み付けをしたことによる増減は，定性的にとらえられていた各コンポーネントに対する判断とよく一致している。

4. まとめ

本稿では，ＯＳに代表される大規模なソフトウェア製品における信頼性把握の一手法として，利用者から見たバグ重大度という概念を導入するとともに，バグ重大度を定量的に取り扱う手法を提案した。さらにこの手法を実際のＯＳに適用し，その有効性について検討した結果，信頼性向上のためのバグ分析に有効であることがわかった。

バグについて考察する場合，なぜバグが作り込まれたかといったプログラム作成者の立場から見た分析が必要なことは言うまでもない。しかしそれだけでなく，ここで提案したバグ重大度のような利用者の立場から見た分析も，ソフトウェアを製品として取り扱う上で，不可欠であるといえる。又バグ重大度のように人間の感覚に関連の深いことがらを定量的に取り扱うためには，様々な方法が考えられるが，どのような手法を用いるにせよ，本稿で述べたようなバグ情報の定常的な蓄積方法と，分析者による個人差を防止するための分析手順の標準化が，データ考察の前提条件として不可欠である。

今回使用した算出式(1)については，バグ重大度指数の値の大小関係に着目して検証を行ったので，重大度指数の絶対値，特に最大値と最小値の倍率の妥当性について，今後さらに広くデータを収集し，検証を進めていく必要があると考えている。

又バグ重大度の概念を導入することにより，ソフトウェア製品の信頼性を向上させるという命題を，残存バグの個々の重大度の累積値を減少させるという命題に置換することが可能であることが，製品への適用を通じてわかってきた。そこで後者の命題にさらにバグの検出及び修正のためのコストの概念を導入すれば，制約された工数の中で最も効果的な品質向上対策を行うための作業指針を，バグ重大度と検出・修正コストとの比の最適化問題として解くことが出来ると予想されるので，今後の研究課題としたい。

最後に，本研究を進めるにあたり，試行作業への協力及び貴重な助言をいただいた富士通㈱第一ソフトウェア事業部検査部の関係者各位に感謝します。

〔参考文献〕

(1)菅野：「ソフトウェア工学における品質管理（QC）と品質保証（QA）」，情報処理 Vol.21 No10,1980

(2)菅野：「ソフトウェアデザインレビュー」，日科技連

(3)武田他：「ソフトウェア・エラーの分類と実例」，情処学会25回全国大会 1E-8,1982

表－1　バグが利用者群に与える影響度

a	a_1	a_2	a_3	a_4
ユーザ種別	オペレータ	システム管理者	端末利用者	センタ利用者
主な業務	システムの運転・操作	システムの作成・保守共通資源の維持等	ＴＳＳ，ＤＣ等の端末操作	バッチジョブの依頼者等（左記以外）
0	支障なし	支障なし	支障なし	支障なし
1	——	——	特定少数の端末利用者のみ支障あり	特定少数のセンタ利用者のみ支障あり
2	支障あり	支障あり	不特定多数の端末利用者に支障あり	不特定多数のセンタ利用者に支障あり

表－2　バグが顕在化した時点で発生する現象

b	内　　容
7	システム全体の処理が不可能
6	サブシステムの処理が不可能
5	システム，サブシステムの処理が異常
4	システム，サブシステムのコマンド処理が異常
3	ジョブの処理が異常
2	性能が異常
1	メッセージが異常
0	その他

表－3　バグの顕在化を抑止するための代行手段，回避手段の有無

c	c_1 (代行手段)	c_2 (回避手段)
2	なし	なし
1	あり	あり
0	不要	不要

表－4　バグが顕在化する以前の状態に戻す作業の困難さの度合

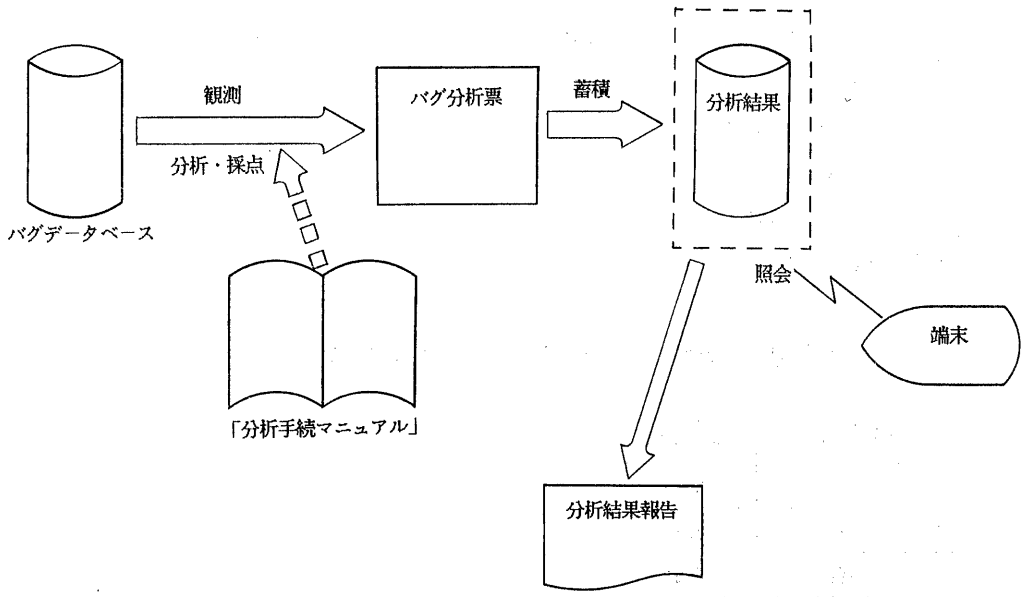
d_1	内　　容
6	復旧にはシステム全体の再起動が必要（再IPL ）
4	復旧には該当サブシステム全体の再起動が必要（再START ）
2	復旧には該当ユーザの再起動などが必要（再LOGON 等）
0	復旧不要

d_2	内　　容
2	d_1 以外の作業として，ユーザ間で共通の復旧作業が必要
1	d_1 以外の作業として，該当ユーザにおける復旧作業が必要
0	d_1 に示す以外の作業は不要

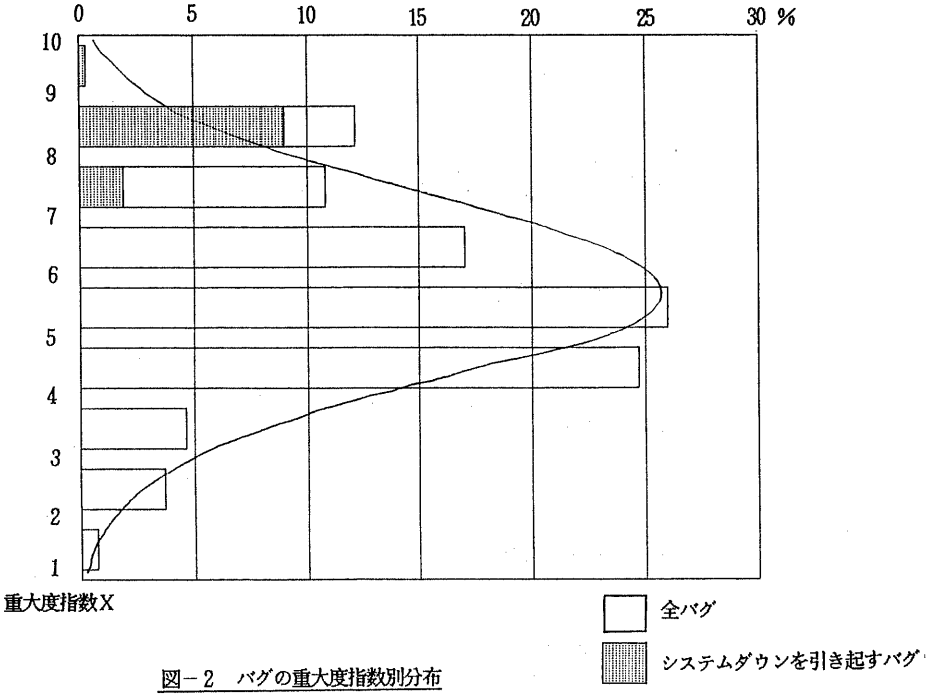
表－5　バグが顕在化するための条件

e_1	内　容
2	正常処理部分で発生
1	異常処理部分で発生

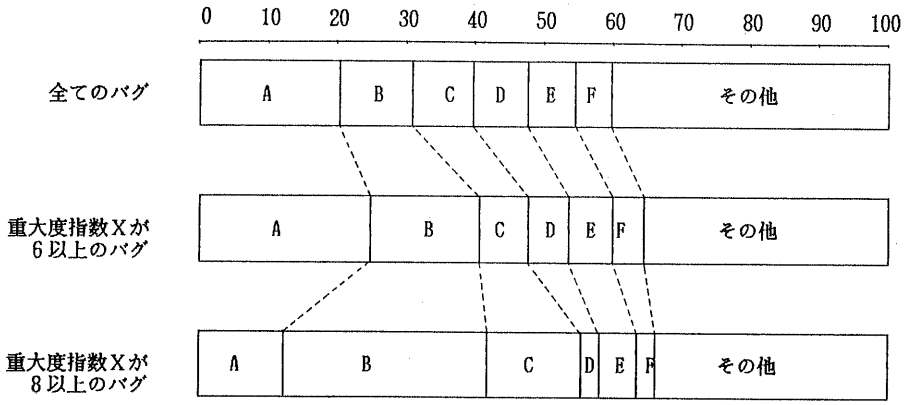
e_2	内　　容
2	外部からみて単純な条件下で常に発生
1	上記以外の場合



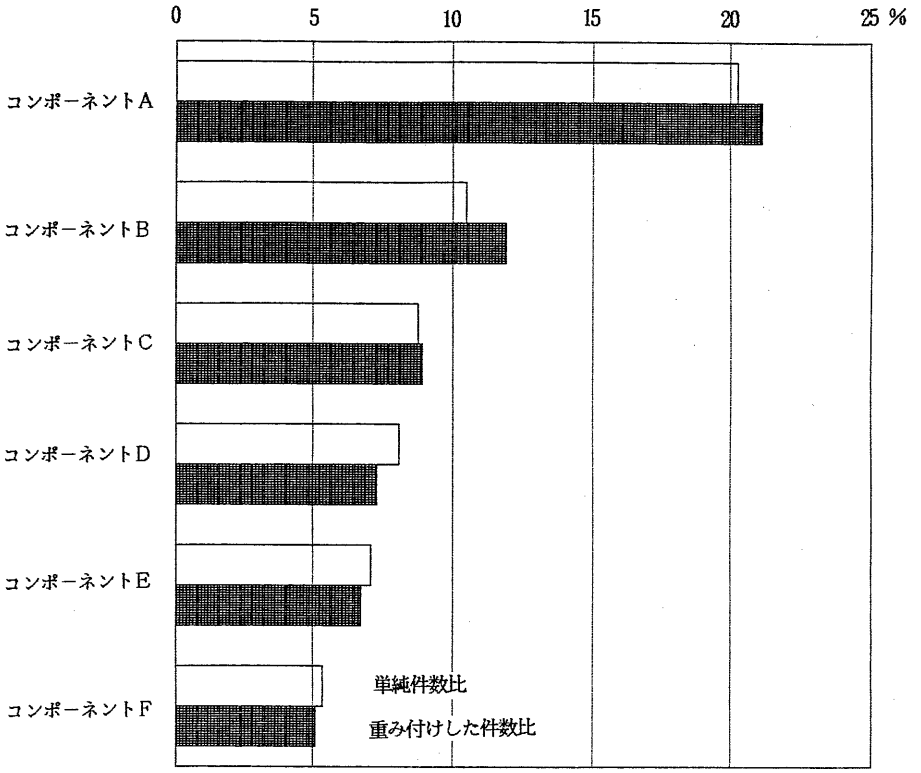
図－1　バグ重大度の構成要因の観測方法



図－2　バグの重大度指数別分布



図－3　バグ重大度指数の階層によるコンポーネント別バグ件数割合の変化



図－4　バグ重大度指数による重み付けをしたコンポーネント別バグ件数割合