4.2 バグ区分

本項では、バグ管理項目の中の“バグ区分”について説明します。

バグ区分はバグの内容を分類した情報項目で、開発プロジェクトや組織内でのバグ全体を対象に、品質評価（バグの傾向分析）を行ったり改善ポイントを検討したりする際にこの情報を用いることができます。

バグ発生工程別に分類したバグ区分を表 4.3、バグ管理プロセス運用上必要な区分を表 4.4 に示します。

表 4.3、表 4.4 の項目については以下の通りです。

「工程」：バグが発生した工程。ESPR におけるソフトウェア・エンジニアリング・プロセスに基づいて「４つの工程」と「工程共通」に分けています。
「種別」：バグの種類を示しています。
「説明」：バグの「種別」について説明しています。
「例」　：バグの「種別」について例を示しています。

一つのバグについてそのバグ区分を検討した結果、該当する種別が複数考えられる場合は、最も重要と考える種別を選択します。また、いったんは「その他」の種別に分類した後、バグ分析の過程で適切な種別に分類し直すこともあります。

バグ区分は、バグ分析を行う開発プロジェクトや組織の特性を考慮してそれぞれで適切なバグ分析を行うために、必要に応じて選択または変更します。

表 4.3　バグ発生工程別に分類したバグ区分

工程	種別	説明	例
要求定義（外部）	記述誤り（外部）	外部の要求仕様書における記述の間違い、不明瞭、漏れなどによるもの。 ※外部＝顧客、システム、ハードウェアなど	・異常時の動作仕様に間違いがあり、システムとして安全状態に移行しないケースがあった。 ・制御に使用する外部入力信号の定義がもともと 1 本漏れていた。 ・マイコンのポートに割り付けられた入（出）力信号の情報がハードウェア要求で間違って指示されていた。
要求定義	記述誤り	ソフトウェア要求仕様書などにおける記述の間違い、不明瞭、漏れなどによるもの。	・状態遷移表（図）の遷移条件が間違っていた、または抜けていた。 ・故障検知のためのフェイルセーフ仕様を間違っていた。 ・フローチャートの矢印表記が無かった。
	機能の欠如	ソフトウェア要求仕様書などにおける記述で、要求されている機能全体の抜けによるもの。	要求定義（外部）には定義されていた機能が、ソフトウェア要求仕様書でその機能自体の記述が無かった。
	機能の定義誤り	ソフトウェア要求仕様書などにおける要求の定義が誤っているもの。要求されていない機能が追加されているものも含む。	・外部の要求仕様で未定義の内容を勝手な解釈でソフトウェア要求仕様に盛り込み、結果、顧客から間違った仕様であると指摘された。 ・既存のソフトウェア要求仕様書を流用したが、開発中の製品では不要な機能の記述が含まれていた。 ・開発途中の仕様変更で削除された機能がソフトウェア要求仕様書から削除されていなかった。
設計	データの誤り	データの取り扱いに関する誤りによるもの。	・取り得るパラメータの範囲が間違っていた。 ・条件判定で使用するデータの種類が不足していた。 ・条件判定で使用するデータの種類が間違っていた。 ・データの初期値が間違っていた。 ・使用するデータ変数名が間違っていた。

工程	種別	説明	例
設計	アルゴリズム／制御の誤り	計算手順や演算方法に関する誤りによるもの。	・データの計算式が間違っていた。 ・計算の順番が間違っていた。 ・意図しない演算のオーバーフローが発生していた。
	インターフェースの誤り	インターフェース仕様（設計）関係の誤りによるもの。 ・システム間のデータ形式（構造、量）の誤り。 ・プログラム、タスク間のデータ形式の誤り。　など その他、無用な依存関係を持ったインターフェースもここに分類する。	・引数の値が間違っていた。 ・引数の型が間違っていた。 ・戻り値の使い方が間違っていた。
	タイミングの誤り	タスク間のタイミング関係の誤り、設計不十分によるもの。 ・タスク間の実行条件（処理順序や割り込み処理の優先順位）の誤り。	・データの初期化・更新タイミングが間違っていた。 ・多重割り込み処理において、割り込み処理の優先順位が間違っていた。 ・割り込み許可（禁止）タイミングが間違っていた。 ・関数の呼び出しタイミングが間違っていた。
	リソースの誤り	・リソースの確保・解放忘れによるもの。 ・処理負荷の見積もりに関するもの。	・マイコンリソース（例えばレジスタ）の使用方法が間違っていた。 ・mallocのサイズが少なかった。 ・ファイルのフリーをしていなかった。 ・二重解放していた。 ・決められた処理時間内に処理が終わらなかった。 ・ROM、RAMが不足していた。
	エラーチェックの誤り	エラーチェックの抜けによるもの。 ・関数、メソッド呼び出しの戻り値の扱いの誤り（エラーチェック抜けなど）。 ・入力データのチェックの誤りなど。	・偶数パリティとするところを奇数パリティにしていた。 ・NULLを引数にとる可能性のある関数の設計仕様で、NULLチェックを記述していなかった。
	記述誤り	設計書における上記種別以外の記述の間違い、不明瞭、漏れなどによるもの。	・フローチャートの矢印表記が無かった。 ・設計書バージョンの記述が間違っていた。
	機能の欠如	設計書における記述で、要求されている機能全体の抜けによるもの。	既存の設計書を流用したが、開発中の製品に必要な機能が抜けていた。
	機能の設計誤り	設計書における機能の設計全体が誤っているもの。要求されていない機能が追加されているものも含む。	・既存の設計書を流用したが、開発中の製品では不要な機能の記述が含まれていた。 ・開発途中の仕様変更で削除された機能が、設計書から削除されていなかった。

工程	種別	説明	例
実装	データの誤り	コードレベルでのデータの取り扱いの誤りによるもの。	・データの初期化をしていなかった。 ・データの初期値が間違っていた。 ・使用するデータ変数名が間違っていた。
	ロジックの誤り	コードレベルでのロジック関係の誤りによるもの。 ・ブランチ：飛び先誤り。条件判定誤り。判断の抜け。判断内容誤り。 ・ループ：終了条件の誤り、ループ回数誤り。初期設定誤り。 ・四則演算処理誤り。 ・不要ロジックあり。ロジック抜け。ロジック位置不適当など。	以下のような間違いをしていた。 ・判定条件として、 (誤)「if (x != 10) {…」 ⇒(正)「if (x == 10) {…」 (誤)「if (x + y) {…」 ⇒(正)「if (x - y) {…」 ・ループ条件として、 (誤)「while (x <= 5)」 ⇒(正)「while(x < 5)」 ・switch文におけるdefault内の処理が無かった、または処理が間違っていた。 ・ポインタ使用時の参照先演算が間違っていた。 ・代入符号「=」と判定符号「==」が間違っていた。
	インターフェースの誤り	コードレベルでのインターフェース関係の誤りによるもの。 ・関数・メソッド呼び出しの引数の誤り。 ・他社製ソフトウェア(購入品)の設定や呼び出し誤り。	・NULLを引数にとらない仕様の関数にNULLを渡していた。 ・呼び出す関数が間違っていた(関数func1を呼び出すところを、func11を呼び出していた)。
	タイミングの誤り	コードレベルでのタスク間のタイミング関係の誤りによるもの。 ・タスク間の実行条件(処理順序や割り込み処理の優先順位)の誤り。	・データの初期化や更新場所が間違っていた。 ・割り込み許可(禁止)場所が間違っていた。 ・関数の呼び出し場所が間違っていた。
	リソースの誤り	コードレベルでのリソースの確保・解放に関する誤りによるもの。	・mallocのサイズ数値が間違っていた。 ・二重解放をしていた。 ・あるリソースの確保には特定レジスタの値を設定する必要があったが、設定が間違っていた。
	エラーチェックの誤り	コードレベルでのエラーチェックの抜けによるもの。	・異常判定処理が無かった。 ・CRC生成多項式の値が間違っていた。

工程	種別	説明	例
実装	機能の欠如	コードの記述で、要求されている機能全体の抜けによるもの。	・設計書に記載されている機能がコード上で存在しなかった。 ・既存のライブラリを流用したが、開発中の製品に必要なコードがライブラリ内に存在しなかった。
	機能の実装誤り	上記以外で機能の実装が正しくないもの。要求されていない機能に対するコードが追加されているものも含む。	・既存のライブラリを流用したが、ライブラリ内に今回の設計に不要なコードが存在していた。 ・開発途中の仕様変更で削除された機能のコードが削除されていなかった。
工程共通	統合の誤り	モジュール統合時の間違いによるもの。	・統合すべきモジュールを選択しなかった。 ・統合不要なモジュールを選択した。
	データベースの誤り	データベース利用方法の誤りによるもの。	テスト実施時に、 ・参照すべきデータベースの版より古い版のデータベースを利用していた。 ・該当機種と異なる機種のデータベースを利用していた。
	OS/パッケージソフトウェアの誤り	他社製ソフトウェア（購入品）に関する問題（ソフトウェアの動作不良など）によるもの。なお、他社製ソフトウェア組込み時の他社製ソフトウェアの設定や呼び出し誤りは、「コード」の「インターフェースの誤り」に分類する。	―
	その他	上記以外をこの種別で扱う。しかし、バグ分析を行う開発プロジェクトや組織の特性から、さらに種別の項目を追加し分類することもある。	―