ソフトウェアのFMEAのレビューからは、このプロセスが発展し、MIL-1629A規格のすべてのコンポーネントがもはや保持されていないことがわかる。表VIにソフトウェアのFMEAにおいて一般的にみられる要素を示す。表VIIには、ソフトウェアの詳細リスク分析の作業産物についてのプロセスのパターンを概説する。この種の分析における主要な要素には以下が挙げられる：

- 特定のソフトウェアの要素（例えば、機能、または高位ではモジュール）
- 特定コンポーネントの不具合の挙動（特定の障害とともに発生するエラーの種類を含む）
- 特定の患者に対するハザードに至るシステム挙動の説明。患者に対するハザードはIRAに引用されたものにすべきである。新しいハザードは、IRAの改訂が必要な可能性を示唆している。これは、IRAが初期段階を超えて妥当である場合に特に必要である。
- ソフトウェア試験チームが強制的に特定のソフトウェアエラーを発生させたときに、軽減メカニズムを実証できる軽減策の詳細。タグ付けによって、軽減策の追跡が保証される。

セイフティクリティカルソフトウェア

ソフトウェアのリスクマネジメントでは、セイフティクリティカルソフトウェアの特別な処理を必要とすることがある。このソフトウェアは前述のルールを適用して特定することができる。ソフトウェアの設計者はリストに詳細なインプットを提供することができるが、通常、詳細リスク分析によって確定される(表VIII)。

種類	時期	形式	対象者	レビュー	詳細	トレース	影響
一連の正式に承認されたルールに従って決定されたセイフティクリティカルであることが明確なソフトウェアを認識する	設計段階の範囲内で実施する	ルールを添付した一覧表	実行および試験チーム コンフィギュレーション管理はプロジェクトレベルである	詳細リスク分析と一致していることを保証するためのソフトウェアの品質チームによるチェック	詳細	品質計画に従ったソフトウェアの広範囲の品質保証活動を受けると記載されたすべてのモジュールを保証しなければならない	確実なソフトウェア配布のために必要とされる通常のプロセスに加えて、安全性に特異的な検証タスクを定義する傾向がある

表VIII.　セイフティクリティカルソフトウェアのプロセスのパターン

ソフトウェアの品質計画は、セイフティクリティカルソフトウェアに適用される追加レベルの検証活動を定義すべきである。これらの活動は一般的に以下の項目が含まれる：

- 正式な従来のFagan式の設計およびコード検査
- 独立したブラックボックステスト
- 非常に詳細な独立した構造テスト（full-statementおよびdecision-predicateテストを含む）
- 発生した措置とすべての欠陥が処理されたことを保証するための進捗監査および最終監査

フォールトインサーション（不具合挿入）テスト

AMDリスクマネジメントモデルの成功は、軽減メカニズムによってリスクが軽減することを実証する最終ステップにかかっている。このステップは、文書化されたプロトコールに基づく正式に承認された試験、結果の収集、発見された問題の処理、監査完了からなる。この試験は、少なくとも2回実施されるべきである。1回目は、すべての検証活動を受けたソフトウェアに関するエンジニアリングプロトタイプについて試験を実施すべきである。次に、試作品、すなわち準備完了ソフトウェアを内蔵したGMP製造医療用具について試験を実施すべきである。

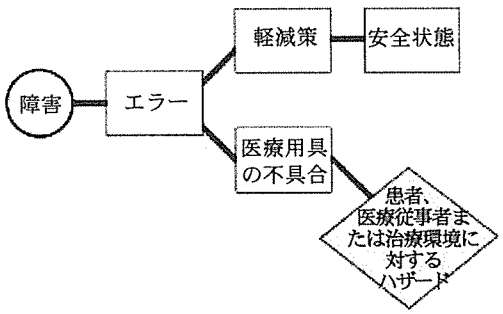
種類	時期	形式	対象者	レビュー	詳細	トレース	影響
以下について実証する：軽減メカニズムがリスク軽減に有効であることを実証するよう努める	プロトタイプおよび試作段階で実施	試験グループが定めたテンプレートに従う	開発；要約が記述され、終了時の監査が実施されるような形式で結果を収集しなければならない	正式である可能性があるが、必ずしも実施されるとは限らない	非常に詳細	IRAおよびDHA作業産物タグへの遡及を支援する	不具合の安全性を決定付ける重要ステップ；修正および再試験サイクルに至ることがある

表IX.　フォールトインサーションテストのプロセスパターン

いわゆるフォールトインサーションテスト（*fault-insertion testing*）、*このテストは軽減ソフトウェアおよびハードウェアを強制的に機能させるもので、その結果は通常、後に*

医療用具が安全状態で作動するというものである。オペレーティングユニットが意図的に組み込んだ障害を発見するまでにはかなりの時間を要する可能性があるため、大部分のテストは意図的に誘発したエラーのパスに従う(表IX)。例えば、試験者はスタックポインターを破壊するために組み込んだ障害の発生を待つのではなく、システムを一時停止させ、スタックポインターを破壊し、停止した実行状態からシステムをリスタートさせることができる。このようなテストは、医療用具の構造についての詳細な知識を有する状態で実施しなければならないため難しい。通常、改変された医療用具は外部の支援ハードウェアを使用することが必要とされる。ハードウェアおよびソフトウェア環境を変えると、通常は臨床使用に適さないユニットが得られる。現在、フォールトインサーションテストは、工学雑誌においてかなりの注目を集めている[8,9]。

障害とエラー



ソフトウェアの障害は開発プロセスの様々なポイントで生じる可能性がある。ソフトウェアの障害は、特別な瞬間に現れるいくつかの状況が組み合わさることによって誘発されるまで、休止していることがある。障害はエラー、またはコンピュータ環境の不正確な状態を招く。ハザードを予防する状態にマシンを移行させるハードウェアまたはソフトウェアの軽減メカニズムによって、このような状態になることもある。リスクマネジメントプロセスに欠陥がある場合、エラーが拡大し、結果的に医療用具の不具合および患者に対する潜在的に有害な作用をもたらす可能性がある。

特別な問題

COTS用途　医療用具へのCOTSソフトウェア内蔵が一般的になりつつある。これは、FDAの「Guidance for Off-the-Shelf Software Use in Medical Devices」という表題の文書案によって認められた[10]。迅速に市販を実現したいという要望に刺激され、将来的に製造業者による医療用具へのCOTSソフトウェアの組み込みは継続すると思われる。しかし、COTSソフトウェアは医療用具の安全性を脅かす可能性があるため、ソフトウェアのリスクマネジメントプロセスに従って検討されなければならない。

AMDアプローチをCOTSソフトウェアに応用することができる。最初に、開発チームはシステムにおけるソフトウェアの役割を認識し、文書化しなければならない。COTSソフトウェアの不具合はIRAに含まれる可能性がある。その特別な性質のため、開発チームがCOTSソフトウェアについて個別のIRAを作成することが妥当と考えられる。この分析では、使用されるソフトウェアパッケージの種類にかかわらず認められる従来の故障モードについて検討すべきである。例えば、ファイルシステムは、破壊されたレコードの復帰に失敗することがある；分析によって、破壊されたレコードに基づいて作動するときのシステム安全性に対する影響が示されなければならない。これが、safety-wrapper philosophyの開発につながることもある。この場合、COTSソフトウェアの不具合を発見し、システムをCOTSソフトウェアと切り離すための特別なソフトウェアが開発される。これには、保存前のレコードの内容に関するエラー検出コードの作成と、その後のこのコードと保存レコードに対して作成されたコードとの照合が含まれると思われる[11]。

COTSソフトウェアの広範囲にわたる検証は、開発チームがソースコードを知ることができないため困難であることがある。しかし、医療用具に特異的なコードとCOTSソフトウェアとのインターフェースにエラーを導入することによって、COTSソフトウェアの不具合に対するシステム安全性の感受性をテストから特定することができる。

ツールとリスクマネジメント　ソフトウェア開発は、ビジュアルデザインツールから欠陥の処理を追跡するデータベースのコンパイラーに至るまでの多数のソフトウェアツールに依存している。これらのツールに組み込まれた障害が、製造コードを破壊する可能性のあるエラーの原因となる可能性がある。これらの障害は革新的な組織が現在検討しているソ

フトウェアのリスクに相当する。組み込まれた障害に対処するために以下のステップを実施することができる：

- 各ツールからの主たるリスクの脅威を認識する。少なくともすべてのツールおよびそのリスク原因への寄与を考慮する初期リスク分析を実施する。
- ツールからのリスクの削減におけるすべての検証プロセスの有効性を考慮する。典型的なアプローチは、決定操作者（decision operator）にとって欠陥のあるコンパイラーが生成する欠陥のあるマシンコードが構造試験によって発見できるかどうかを考慮することであるかもしれない。ツールがセイフティクリティカルソフトウェアに及ぼす影響に特に配慮しなければならない。
- 各ツールとともに用いられる限定的な一連の機能を定義する。ツールのすべての機能が使用されることはまれである。このステップの実施は不可欠である。
- おそらくは典型的な作業で構成される単純な試験を用いて、新たにリリースされた各ツールについて受入れ試験を実施する。欠陥が発見された場合、開発チームに連絡する。
- ソフトウェアのベンダーと直接連絡をとり、すべての既知の欠陥について理解する。一部のベンダーはこの作業をサポートするウェブサイトを開設している。確実に開発チームの全員が欠陥について認識しているようにする。既知の欠陥を誘発する可能性がある使用を発見するためのコード検査チェックリストを調整する。

結論

ソフトウェアのリスクマネジメントは、ソフトウェアを用いる医療用具数が増加するにつれて、ますます重要になりつつある。言い換えれば、このことによって、医療用具の安全性におけるソフトウェアの役割を特定する特異的活動が必要となる。この役割を認識するには、原因を、患者、操作者、臨床環境に対するリスクを軽減する軽減メカニズムと組み合わせる、確実なエンジニアリングが必要である。さらに、プロセスステップをパターンにあてはめることができ、反復した原因と軽減策のパターンの適用によって再利用と有効性が達成される。最後に、ソフトウェアのリスク分析を拡大して、COTSソフトウェアおよび開発に使用される他のツールの役割を明確にし、分離することが重要である。

引用文献

1. AM Davis, ed., *IEEE Software*14, no. 3 (1997).

2. BJ Wood and JW Ermes, "Applying Hazard Analysis to Medical Devices," *Medical Device & Diagnostic Industry* 15, no. 1 (1993): 79–83.

3. BJ Wood and JW Ermes, "Applying Hazard Analysis to Medical Devices," *Medical Device & Diagnostic Industry* 15, no. 3, (1993): 58–64.

4. WW Gibbs, "Software's Chronic Crisis," *Scientific American* 271, no. 3 (1994): 86–95.

5. NG Leveson, "Software Safety: Why, What, How," *ACM Computing Surveys* 18, no. 2 (1986): 25–69.

6. *Reviewer Guidance for a Pre-Market Notification Submission for Blood Establishment Computer Software* (Rockville, MD: Center for Biologics Evaluation and Research, Office of Blood Research and Review, Division of Blood Applications, January 1997).

7. James M. Utterback, *Mastering the Dynamics of Innovation* (Boston: HBS Press, 1994).

8. JA Clark and DK Pradhan, "Fault Injection—A Method for Validating Computer-System Dependability," *Computer* 28, no. 6 (1995): 47–56.

9. J Voas, "Fault Injection for the Masses," *Computer* 30, no. 12 (1997): 129–130.

10. "Guidance for Off-the-Shelf Software Use in Medical Devices," FDA, CDRH, June 4, 1997, draft document.

11. Richard W Hamming, *Coding and Information Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1980), 21–34.

参考文献

Hohmann, L, "Getting Started with Patterns," *Software Development* 2, no. 2 (1998): 55–61.

Leveson, Nancy G. *Software: System Safety and Computers.* New York:

Addison-Wesley, 1995.

Parrish, Edward A, ed., *Computer* 31, no. 6 (1998). This issue contains several articles on the risks of COTS.

Storey, Neil. *Safety-Critical Computer Systems.* Essex, England: Addison Wesley Longman, 1996.

Voas, Jeffrey M and Gary McGraw. *Software Fault Injection*, New York: John Wiley & Sons Inc., 1998.

Principals for Computers in Safety-Related Systems—DIN V VDE 0801. Publication 1990-01. Munich, Germany: TÜV Product Service.

Functional Safety—Safety Related Systems—IEC 1508 (draft). Geneva, Switzerland: International Electrotechnical Commission.

Bill J. Wood is vice-president of research and development for RELA (Boulder, CO).
