メントプロセスの後の段階において）特に以下の重要な成果を得られる可能性がある。

- ソフトウェアによる危害の発生を防止するハードウェアリスクコントロール手段
- 危害を及ぼす可能性のあるソフトウェア機能の仕様からの除去
- 危害の防止にソフトウェアを使用するリスクコントロール手段（IEC 62304:2006の細分箇条5.2.3参照）
- 低い欠陥密度で実装しなければならないソフトウェアの部分、及び特別試験の対象としなければならないソフトウェア仕様の部分の特定（IEC 62304:2006の細分箇条4.3）
- 予期せぬ悪影響から生じる危害を防止するために（より低いソフトウェア安全クラスの）ソフトウェアアイテムから分離しなければならないより高い安全クラスのソフトウェアアイテムの特定（IEC 62304:2006 の細分箇条4.3及び5.3.5参照）。この点に関する詳細な検討については、第6.2.2.5項を参照すること。

ハザードを十分に明らかにするには、医療機器の臨床用途をよく理解しなければならない。また、ソフトウェアには複雑なユーザーインターフェースの可能性などの複雑性という特別な難しさがある。したがって、ソフトウェアハザードの特定だけを単独で行うことはできない。ソフトウェアハザードの特定は、臨床専門家（臨床支援及び技術サービスの専門家）、ソフトウェアエンジニア、システム設計者、及びユーザビリティ／人間工学の専門家などからなる分野横断的なチームによって、システムレベルで実施することが望ましい。

ハザードの特定では、医療機器の性質から起こりうる危害（例えば患者の切り傷、被ばく、あるいは感電）、さらにはソフトウェアの使用に関連して発生する新たなハザードを考慮するのが望ましい。後者の例として、以下が挙げられる。

- 臨床家又は患者に対する誤った情報の提供
- 患者の取り違い（医療機器に患者の詳細又は処方の情報が保存されている場合）
- ソフトウェアの異常による治療の遅れ又は拒否

注記：多くの医療機器の場合、治療の遅れや拒否は患者に危害を及ぼすとはみなされない。

特定されるハザードには、仕様に従って動作しているソフトウェアに関連するハザードと、ソフトウェア異常に関するハザードの両方が含まれることが望ましい（第6.1項も参照）。

多くの場合、医療機器のユーザーインターフェースはソフトウェアによってさらに複雑になっている。特に、ソフトウェアを組み込んだ医療機器は情報を扱うことが多い。これは患者利益の点から正当化されるだろうが、例えば次のような、誤った情報又は誤って使用された情報に関する新たなハザードを考慮することが望ましい。

- 誤ったデータ入力
- 使用者による表示の誤読
- 使用者による警報の誤解又は無視
- 過剰なデータ又は過剰な警報数による使用者への過負荷（IEC 62366[5]参照）

4.4 各ハザード状態に関するリスクの推定

4.4.1 一般

ISO 14971:2007から抜粋

4.4 各ハザード状態に関するリスクの推定

ハザード状態を招くおそれのある合理的に予見可能なイベントシーケンス又はイベントの組み合わせについて検討し、結果として生じたハザード状態を記録すること。

注記 1：過去に認識されていないハザード状態を明らかにするには、特定の状況を網羅する体系的手法を使用できる（附属書 G 参照）。

注記 2：ハザード状態の例は、H.2.4.5 及び E.4 で示している。

注記 3：ハザード状態はスリップ（動作のミス）、ラプス（記憶のミス）、及びミステイク（確認のミス）から生じうる。

特定した各ハザード状態について、利用可能な情報又はデータを使用してその関連リスクを推定する。危害の発生確率が推定不能なハザード状態については、リスク評価及びリスクコントロールで使用するために、考えられる結果を列挙する。これらの活動の結果は、リスクマネジメントファイルに記録する。

危害の発生確率又は危害の重大性を定性的又は定量的に分類するために使用するシステムすべてを、リスクマネジメントファイルに記録する。

注記 4：リスクの推定には、発生確率及び結果の分析が盛り込まれている。用途によっては、リスク推定プロセスの一定要素だけが検討対象となるかもしれない。例えば、初期ハザードと結果の分析だけで十分な場合もある。D.3 も参照。

注記 5：リスクの推定は、定量的又は定性的に行うことができる。システム上の欠陥から生じるリスクの推定も含めたリスク推定手法は、附属書 D で示している。附属書 H では、生体外診断医療機器に関するリスクの推定に役立つ情報を提供している。

注記 6：リスクの推定に用いる情報又はデータは、例えば次から取得できる：

a) 公表規格
b) 科学技術データ
c) 公に報告されている事故を含む、既に使用中の類似医療機器の実地データ
d) 典型的な使用者を使ったユーザビリティ試験
e) 臨床上の証拠（エビデンス）
f) 適切な調査の結果
g) 専門家の意見
h) 外部品質評価スキーム

適合性は、リスクマネジメントファイルの検査によって確認する。

ソフトウェア関連のリスクを推定するためには、まずソフトウェアを含むハザード状態を特定することが必要である。ソフトウェアは、ハザード状態の原因となるイベントシーケンスの根本的原因になる場合もあれば、ハードウェア故障の検出を意図したソフトウェアのように、シーケンスの他の原因になる場合もある。ソフトウェアには、SOUPコンポーネントや過去に開発したコンポーネントの再利用品が含まれる可能性もある。

リスク推定は、特定した各ハザード状態から生じる危害の確率及び危害の重大性に基づいて行われる。ソフトウェア異常から生ずる危害の確率を推定するのは非常に困難なため（箇条4.3.3参照）、危害に至るイベントシーケンスにおけるソフトウェア異常などのハザード状態のリスクを推定する際にソフトウェア異常の発生確率を使用するときには、注意が必要である。

4.4.2 特定方法

ハザード状態におけるソフトウェアの潜在的役割の特定には、様々な手法が使用できる。これらの手法はアプローチが異なり、ソフトウェア開発の様々な段階で役立つ可能性がある。唯一の正しい手法というものは存在しない。ISO 14971:2007の附属書Gにリスク分析のいくつかの手法についての情報が記載されている。

フォルトツリー解析（FTA）は、全体としての医療機器から始める、伝統的なトップダウン手法である（IEC 61025 [3] 参照）。この手法は、主として危害の原因分析に使用される。FTAは危害が発生することを前提とし、その危害について存在するはずのイベント又は条件を、ブール論理を使用して特定する。そのイベント又は条件を段階的にさらに詳細に解析していき、危害を予防すると期待されるリスクコントロール手段がひとつ以上特定できるまで解析を続ける。FTAは、ハザード状態を引き起こすイベントシーケンスに関与するソフトウェアアイテムの特定に使用できる。

故障モード影響解析（FMEA）は、コンポーネントやサブシステム（ソフトウェアの場合は、IEC 62304 におけるソフトウェアアイテム）から始め、「この要素が故障したとするとどのような結果になるか？」という問い掛けを行う、ボトムアップアプローチである（IEC 60812 [2] 参照）。

各ソフトウェアアイテムにどのソフトウェア欠陥があるかを予想するのは困難なことから、FMEAではまず各ソフトウェアの安全に関わる要求事項をリストアップし、この要求事項が満たされない場合にどのような結果になるかを考えることから始める。

そうすることにより、どのソフトウェアアイテムの故障が危害に結びつくかを特定でき、どのような種類の故障を防ぐ必要があるかも特定できる。

ハザード状態を引き起こす可能性のあるイベントのシーケンス又は組み合わせを特定するとき、医療機器の主要性能に直接関わるソフトウェア（血糖値を計算するアルゴリズムなど）及び関連するハザードに固有の原因に焦点を絞るのが最も容易である。また、軽微な故障モードを招くおそれがありしたがってひとつ以上の医療機器ハザードを発生させる可能性のあるソフトウェアの原因を考慮することも重要である。ソフトウェア原因の例については、附属書Bを参照すること。

注記：固有の原因とは、その機能性が機器の臨床機能に明らかに関わっており、機器ハザードの原因のひとつになっているソフトウェアの欠陥である。例として、試験結果を算出するアルゴリズムの欠陥が挙げられる。

ソフトウェアアイテム内で何が故障するかを厳密に予測することは困難であるが、欠陥のカテゴリを特定することは可能であり、そのカテゴリのそれぞれには周知のリスクコントロール手段がある。例えば、データ破損は、チェックサム手順を使用することで検出及び予防可能な故障に分類される。ソフトウェア原因の例とその取扱い方法については、附属書Bを参照のこと。製造業者は、自社製品に関するソフトウェア欠陥のカテゴリの独自のリストを維持管理することが望ましい。

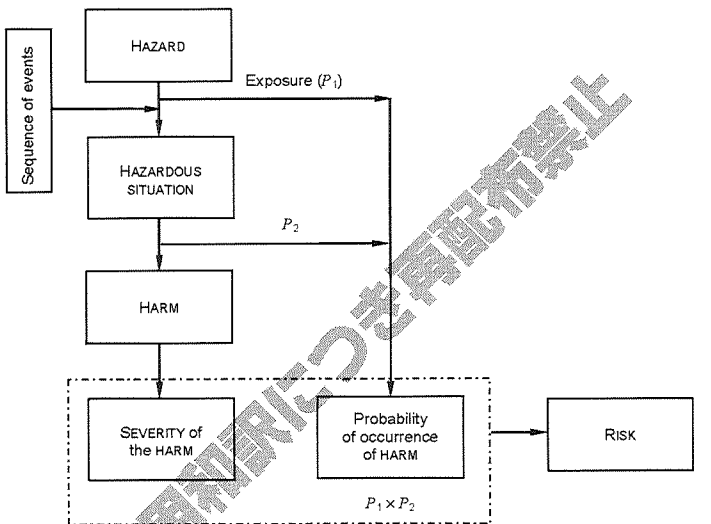
4.4.3 確率

ソフトウェアのある特定のバージョンにおけるソフトウェア異常は、そのソフトウェアのすべてのコピーにも存在する。しかし、個々のコピーへの入力がランダムであるため、そのソフトウェア異常がソフトウェア故障につながる確率を推定するのは非常に困難である。

ソフトウェア故障発生確率の推定手法について、見解の一致は存在しない。ハザード状態の原因となるイベントシーケンス内にソフトウェアが存在する場合、ハザード状態のリスクを推定するときにソフトウェア故障発

生確率を考慮に入れることはできない。このような場合、最悪のケースの確率を考えるのが適切であり、ソフトウェア故障発生確率は 1 とすることが望ましい。シーケンス内の残りのイベントの確率が推定可能な場合（そのイベントがソフトウェアでない場合）は、その確率をハザード状態の発生確率（図1のP_1）として使用してもよい。それができない場合は、ハザード状態の発生確率を 1 に設定すべきである。

ハザード状態が危害に至る確率（図1のP_2）を推定するには、通常、臨床で危害の発生を防げる見込みのあるハザード状態と、危害が発生する可能性が高いハザード状態とを識別するための臨床知識が必要とされる。



NOTE P_1 is the probability of a HAZARDOUS SITUATION occurring.
P_2 is the probability of a HAZARDOUS SITUATION leading to a HARM.

図1 – ハザード、イベントシーケンス、ハザード状態、及び危害の関係図 – ISO 14971:2007 附属書Eより

多くの場合、危害の発生確率を推定することは不可能かもしれず、リスクを危害の重大性だけに基づいて評価することが望ましい。そのような場合のリスクの推定は、ハザード状態から生じる危害の重大性に焦点を合わせるのがよい。
ソフトウェア故障の発生確率を推定することは不可能であるが、多くのリスクコントロール手段が、ソフトウェア故障がハザード状態に至る可能性を低減しているのは明らかである。ソフトウェア異常によるメモリの破損を例にとってみると、メモリのチェックサムを実施することにより、故障を検出してハザード状態の確率を低減できる可能性がある。ただし、チェックサムですべての損傷を検出できる保証はない。正確に言うと、そのような破損の大多数を検出することで、そのリスクを許容できる水準まで軽減するものである。チェックサムを実施する前又は後にハザード状態の確率を推定することはできないが、チェックサム実施後のハザード状態の確率が実施前よりも低くなることは断言できる。リスクコントロール手段がリスクマネジメント計画で特定された残留リスクの許容基準を満たすのに有効であると実証する責任は製造業者にある。

つまり、ソフトウェアのリスク推定では、起こりうるソフトウェア故障の確率を推定しようと試みるので

はなく、万一故障が起きた場合の危害の重大性及びその相対的確率に主眼を置くことが望ましい。

注記：これは、同じ重大性を持つ複数のハザードを見分けるのに役立ち、それによって実際の危害の確率がより高いハザードにもっと注力できるようになる。

4.4.4 重大性

ソフトウェアが原因となるリスクの重大性の推定は、使用すべきソフトウェア開発プロセスに影響を及ぼす。IEC 62304 では、プロセスの厳密さはソフトウェアが原因となる危害の重要性に依存する、とされている。

ISO 14971 では、リスク評価の目的で重大性のレベルをどのように定義するかを製造業者に委ねているが、IEC 62304 のソフトウェア安全性分類と関連付けて重大性のレベルを定義しておくのがよい。さもないと、重大性の分類を二回（リスクマネジメントプロセス全体で必要なリスク評価時、及び IEC 62304 に従ってソフトウェアの安全性クラスを決定するとき）実施する必要があるかもしれない。

5 リスク評価

ISO 14971:2007 から抜粋

5 リスク評価

特定した各ハザード状態について、製造業者はリスクマネジメント計画で定める基準を使用してリスク軽減が必要か否かを判断する。リスク軽減が必要でない場合、6.2 から 6.6 までに示す要求事項はこのハザード状態に適用しない（つまり 6.7 まで進む）。このリスク評価の結果は、リスクマネジメントファイルに記録する。

注記 1：リスク許容性の決定に関する指針は、D.4 で示す。

注記 2：医療機器設計基準の一部として関係する規格を適用すると、リスクコントロール活動を構成し、6.3 から 6.6 に示す要求事項を満たす場合がある。

適合性は、リスクマネジメントファイルの検査によって確認する。

4.4.3 に記載したように、ソフトウェア故障の確率を推定するのは難しい。そのために危害の確率が推定できない場合には、危害の重大性のみに基づいてリスクを評価することになる。

6 リスクコントロール

6.1 リスク軽減

ISO 14971:2007 から抜粋
6 リスクコントロール

6.1 リスク軽減

リスク軽減が必要な場合、6.2 から 6.7 までに記載するリスクコントロール活動を実施する。

リスク軽減のソフトウェア面を6.2から6.7で検討する。

6.2 リスクコントロールオプション分析

ISO 14971:2007 から抜粋

6.2 リスクコントロールオプション分析

製造業者は、リスクを許容可能な水準まで軽減するのに適切なリスクコントロール手段を特定する。

製造業者は、次のリスクコントロールオプションの一つ以上を、次に掲げる優先順位で使用する：

a) 本質安全設計

b) 医療機器自体又は製造工程の保護手段

c) 安全性情報

注記 1：オプション b) 又は c) の実施について、製造業者はリスクが許容できるか否かを判断する前に、合理的に実施可能なリスクコントロール手段が考慮され適切なリスクの軽減を提供するオプションが選択されているプロセスに従うことができる。

注記 2：リスクコントロール手段は、危害の重大性を下げること、危害の発生確率を下げること、又はその両方が可能である。

注記 3：多くの規格が、医療機器の本質安全設計、保護手段及び安全性情報について定めている。また、他の多くの医療機器規格が、リスクマネジメントプロセスを統合した要素を備えている（例：電磁両立性、ユーザビリティ、生体適合性）。関連規格は、リスクコントロールオプション分析の一部として適用することが望ましい。

注記 4：危害の発生確率を推定できないリスクについては、D.3.2.3 を参照。

注記 5：附属書 J で提供する安全性情報に関する指針

選択したリスクコントロール手段は、リスクマネジメントファイルに記録する。

リスクコントロールオプション分析中に、要求されるリスク軽減の実施が不可能と製造業者が判断した場合、製造業者は残留リスクのリスク／効用分析を行う（6.5 へ進む）。

適合性は、リスクマネジメントファイルの検査によって確認する。

6.2.1 複雑なシステムのためのリスクコントロールオプションの選択

6.2.1.1 一般

複雑なシステムでは、ハザード状態につながりえるイベントシーケンスが多数あることが考えられる。そうしたシーケンスの個々のイベントすべてに対してリスクコントロール手段を適用するのは不可能かもしれないし、適用する必要もないかもしれない。選択した一部のイベントにリスクコントロール手段を適用し、危害の全体的な確率を許容レベルまで軽減すれば十分である。

以下の3つの細分箇条では、三種類のリスクコントロール手段をどのようにソフトウェアに実装するかについて、概要を示す。さらに、どのイベントにリスクコントロール手段が必要かについても論じている（6.2.1.5を参照のこと）。

6.2.1.2 本質安全設計

本質安全設計は通常、医療機器の不安全な機能を取り除くことによって、あるいはその機能がより安全な方法（すなわちハザード状態を回避する又は最小にする方法）で実装されるように設計を変更することによって達成される。これは、多くの場合、設計を簡素化し、実装しやすく、かつ使用者にとっては操作しやすくなる、という効果がある。

このことは、ソフトウェアに実装される機能について特に当てはまる。ソフトウェアシステムには顧客の要望を可能な限り無差別にすべて盛り込みたい、という誘惑がある。しかし、これはソフトウェアコンポーネント同士が互いに影響しあう可能性を大幅に増やす結果となり、予期しないハザード状態を生じさせる。医療機器及びそのソフトウェアの開発の初期の段階でリスクマネジメントを適用することにより、これを回避しながら顧客の大部分を満足させることが可能になる。

ほとんどの場合、ソフトウェアに適用する本質安全設計には、以下が含まれる。

- 不必要な機能を削除する
- ハザード状態につながるイベントシーケンスを回避するためにソフトウェアアーキテクチャを変更する
- 使用時のヒューマンエラーの可能性を低減するためにユーザーインターフェースを簡素化する
- ソフトウェア異常を回避するためにソフトウェア設計ルールを明示する
  この例として以下がある。
- 動的メモリ割り当てに関わるソフトウェア異常を回避するため、静的メモリ割り当てのみ使用する
- プログラミングエラーにつながりやすい構造を回避するため、プログラミング言語の限定的なバージョンを使用する

6.2.1.3 保護手段

ソフトウェアを使用する医療機器のための保護手段は、ハードウェア又はソフトウェアのどちらに実装してもよい。保護手段の設計は、その保護手段がそれを適用する機能から独立していることを示すべきである。ソフトウェア保護手段をハードウェアに適用する場合、又はその逆の場合には、これは比較的容易である。

ソフトウェアに実装してソフトウェアに適用する保護手段を選択する際には、ひとつの原因から複数の故障が生じる可能性を回避することが重要となる。保護手段によってハザード状態を検出及び／又は予防する場合、製造業者は、その保護手段と主要性能を実現するソフトウェア機能とをはっきりと分離する必要がある。

例えば、患者に治療を提供するソフトウェアをひとつのプロセッサで走らせ、ソフトウェア保護手段を実現するソフトウェアは別のプロセッサで走らせる、といった方法がある。

6.2.1.4 安全性情報

医療機器にソフトウェアを使用すると、使用者から見た挙動がより複雑になりがちである。そのため、安全のための情報への依存度が高くなりやすい。その情報は簡単な画面上の警告から複雑なユーザマニュアルや限定研修コースまで様々である。このような資料のボリュームと複雑さは、優れたユーザーインターフェースの設計に注意を払うことで、減らすことができる（IEC 62366 [5] を参照のこと）。

6.2.1.5 どのようなイベントにリスクコントロール手段が必要か

ハザード状態を引き起こす可能性のあるイベントシーケンスは数多くある。そうしたシーケンスの個々のイベントすべてに対してリスクコントロール手段を適用するのは不可能かもしれないし、適用する必要もないかもしれない。選択した一部のイベントにリスクコントロール手段を適用し、危害の全体的な確率を許容レベルまで軽減すれば十分である。

どのイベントを検出、防止、又は発生確率を低くするかを決定する際には、当然のことながら、ハザード状態につながる可能性のある一連のイベントを詳細に列挙するとよい。フォルトツリー解析（4.2.2参照）は、イベントシーケンスは提示しないが、シーケンスの特定には使用できる。リスクコントロール手段の正しい動作は、ANDゲートにFALSE入力として現われ、ANDゲートへの他の入力に関わりなく、危害の防止につながる。図2にFTAの一部を示す。このFTAでは、誤ったソフトウェア出力（それ自体があるイベントシーケンスの出力）が、不安全な状況を検出してその出力の有害な影響を防止する（動作を停止させるなどして）ように設計されたリスクコントロール手段によって、患者に傷害を及ぼすのが防止されている。誤ったソフトウェア出力が発生しても、患者に傷害が及ぶのはリスクコントロール手段がうまく機能しなかったときのみとなる。ただし、誤ったソフトウェア出力が患者への傷害につながらないことを確実にするために、それにつながるイベントシーケンスを詳細に探索する必要はない。

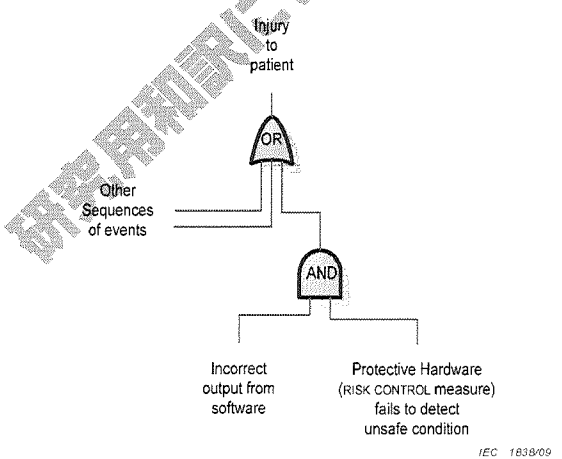


図2 – 誤ったソフトウェア出力が危害を発生させるのを防止するリスクコントロール手段を示したFTA

リスクコントロール手段を適用するポイントの例として、以下がある。

- 全体としてのソフトウェアシステムへの入力
- 全体としてのソフトウェアシステムからの出力

– ソフトウェアモジュール間の内部インターフェース

ソフトウェアへの入力の範囲を制限するリスクコントロール手段を適用することにより、不安全な出力を防止できる。また、ソフトウェアの異常による危害につながる入力の発生確率を低減することができる。これは、ソフトウェアが試験されていない、予期しない動作をする確率を低減するからである（4.4.3参照）。

ソフトウェアへの入力の範囲の制限は、ソフトウェア又はハードウェアのリスクコントロール手段を使用することによって行える。以下に例を挙げる。
– ソフトウェアのリスクコントロール手段で入力をチェックし、不安全な、又は矛盾した値を拒否できる。
– ハードウェアのリスクコントロール手段では、権限のない人間がデータを入力できないように部屋をロックすることなどが考えられる。

リスクコントロール手段を医療機器又はそのソフトウェアの出力に配置すると、ソフトウェアの出力値が安全範囲内であり内部的に矛盾がないかチェックし、そうでない場合には危害を防止することができる可能性がある。これを実現する手段の例としては、以下がある。
– 出力値をチェックしてそれが安全範囲から逸脱するのを防ぐ、ソフトウェアのリスクコントロール手段
– 患者に印加されるエネルギーを制限する、ハードウェアのリスクコントロール手段
– 警告ラベルと固定配線による停止スイッチの組み合わせからなるリスクコントロール手段。このリスクコントロール手段は、コンポーネントの操作者がハザード状態を検出できることが前提となる。

医療機器又はそのソフトウェアの出力に配置されたリスクコントロール手段に加えて、ソフトウェアコンポーネントの入力及び出力にもリスクコントロール手段を適用できる。これにより、ソフトウェアのより小さい部分の入力及び出力をチェックし、危害を防止することが可能になる。

ひとつのパラメータについて、医療機器が安全に動作する範囲をひとつだけ指定するのは不可能な場合もある。しかし、「安全動作限界」、すなわち、医療機器が安全に動作する限界を形成する複数のパラメータの組み合わせを指定することは可能である。医療機器が安全動作限界内で動作しているかどうかの評価には、ソフトウェアを使用できる。例えば、ソフトウェアは装着部の測定温度と曝露時間を組み合わせて患者の火傷の可能性を検出することができると考えられる。

臨床家はソフトウェアの入力及び出力の安全範囲を知っているが、医療機器の設計ではその安全範囲が予想できないケースもある。そのようなケースでは、その医療機器を臨床家の指定どおりに動作させることを目的として、リスクコントロール手段を使用することができる。ソフトウェアの入力と出力との間の矛盾の検出には、ハードウェア又はソフトウェアのリスクコントロール手段を使用する。

例えば、臨床家が医療機器を使用して治療を処方するが、処方は患者によって大きく異なる場合がある。この場合、入力値又は出力値を分析するだけではハザード状態は検出できない。それでも、ソフトウェアのリスクコントロール手段を適用することによって、医療機器の出力が入力（意図する処方）に正確に一致するようにすることができる。

6.2.2 リスクコントロール手段

6.2.2.1 概要

ソフトウェアに対して適切なリスクコントロール手段を効率よく実装するには、製品開発及びソフトウェアライフサイクルに関する理解について、慎重に検討するのが望ましい。リスクコントロール手段の種類によっては、設計の初期段階であれば非常に簡単に実装できるが、開発の後の段階になると実装が不可能になるものもある。製品開発プロセスの早期段階でソフトウェアをリスクマネジメントの観点から慎重に検討しないと、医療機器の安全性について、適切なソフトウェア操作への依存が期せずして過度になってしまうようなハードウェアの決定がなされる可能性がある。

極めて重大なソフトウェアアイテム（例：欠陥があれば死に至るおそれのあるアイテム）と安全性に影響を及ぼすおそれがないソフトウェアアイテムを区別するために、ソフトウェアアイテムを分離してソフトウェア安全クラスを割り当てることは、有効な手法となりうる。ソフトウェア安全クラス分類に関しては、

IEC 62304:2006の細分箇条4.3を参照すること。

ソフトウェア安全クラスの割り当ては、より重大なソフトウェアアイテムの検証及びコンフィギュレーション管理活動で、厳格度をより上げ焦点をさらに絞るための基礎として役立つ。これを行った場合は、悪影響を注意深く検討するのが望ましく、重大性の低いソフトウェアアイテムでも、それがより重大性の高いソフトウェアアイテムに影響を及ぼすおそれがある場合は、それと同じ格付けにするのがよい。また、IEC 62304 ではひとつの活動又は作業内で異なる複数の手法の使用を許可していることにも留意するべきである（IEC 62304:2006 の細分箇条5.1.4では、その手法を定義することを求めている）。製造業者は、ソフトウェアの安全分類でクラスCに分類されるソフトウェアアイテムを識別するための枠組みを作成してもよい。例えば、高度に複雑なソフトウェアアイテムにはより正式な検証方法（すなわちコードレビューではなくコードインスペクション）を使用することなどが考えられる。

なお、ソフトウェアアイテムをまず安全性に関係するアイテムとして分類し、その後一定のリスクコントロール手段や設計選択によってその重大性を低くして処理することもできる。リスクマネジメントを正しく実施すれば、分離と本質安全設計により安全性に関係するソフトウェアを最小限のサブセットに減らすことができる。

ソフトウェアの安全性を確保するには、製品開発ライフサイクルを通して様々な活動が必要になる。故障分析の正式手法などの信頼性技術には、完全なリスクマネジメント手法は含まれていない。また、信頼性と安全性は関連していることが多いものの、全く同じ属性ではないことを理解しておくことも重要である。信頼性に重点を置いたライフサイクルプロセスでは、安全性を十分に達成できない場合がある。

一部の特定のリスクコントロール手段についての詳細と、リスクの特定の原因にどのように対処すべきかの手引きを 6.2.2.2、6.2.2.3、6.2.2.4、6.2.2.5、及び 6.2.2.6 に記載する。

6.2.2 リスクコントロール手段及びソフトウェアアーキテクチャ設計

6.2.2.1 概要

ソフトウェアアーキテクチャは、リスクを軽減する保護手段のためのソフトウェアメカニズムだけでなく、本質安全設計によってリスクをコントロールするのに使用されるソフトウェアの機能も記述することが望ましい。

6.2.2.2 アーキテクチャの機能による本質安全設計

ソフトウェア制御機能に関わハザードは、例えばその機能の実装にハードウェアを使用するなどの手段によって回避できるかもしれない（同様に、ハードウェア機能に関わハザード（消耗、疲弊）は、ソフトウェアの使用によって回避できるかもしれない）。

時には、上位レベルでの設計に関する決定によって、ハザードを完全に回避できる場合もある。例えば、ハードウェアの観点からは、AC電源の代わりにバッテリーを使用することによって感電リスクを排除できる。同様に、ハザードを招くおそれのあるプログラミングエラーのクラス全体を、上位レベルでの設計によって排除可能な場合もある。例えばメモリリークは、静的データ構造だけを使用することによって回避できる。

ソフトウェアを使用したシステムで特に問題になるのは、ソフトウェアが共有できる物理的インフラには限度がないという認識である。この認識は誤りである。

システムには要求されたときに必要なタスクすべてを実行できるだけの十分なリソースが確保されるべきである、というのがシステム設計の原則である。この原則を、ハードウェアだけでなくソフトウェアにも適用するのが望ましい。あるソフトウェアアイテムが安全性に関与している場合には、リスクアセスメントで以下の問題に取り組まなければならない。

– 安全性に関係するそのソフトウェアアイテムは必要時にプロセッサへのアクセスを確保できるか？

– 安全性に関係するそのソフトウェアアイテムは、不安全状態が事故に発展する前にタスクを完了するのに十分なプロセッサ時間を確保できるか？

– 他のソフトウェアアイテムがその安全に関わるソフトウェアアイテムを破損させたり、その動作を妨げないことを実証できるか？

これらの問題は、安全性に関与するソフトウェアがひとつのプロセッサを安全に関与しないソフトウェアと共有しなければならない場合に、特に重要である。なぜならば、安全性に関与する機能と安全性に関与しない機能との間でリソースの獲得競争が発生するからである（分離については6.2.2.2.4を参照）。

上記の問題のすべてが設計者に見えるようにする開発手法を選択するのが望ましい。例えば、すべてがうまくいってオペレーティングシステムに余裕があるときであれば作動するようなプロセスとして、安全性に関係するソフトウェアアイテムを設計するのでは十分ではない。開発手法は、スケジューリング、優先順位、及びタイミングを周到に設計できるようなものであるべきである。

6.2.2.3 フォールトトレラント（耐故障）アーキテクチャ

医療機器では、患者又は使用者の安全を確保するための機能が数多く要求される。そのような機能には、中断や遅延の許されない臨床機能、及び保護的リスクコントロール手段を実現する機能がある。

フォールトトレラント設計は、医療機器の信頼性を高めるアプローチとして非常に一般的である（ソフトウェアエンジニアリング実施担当者のための参考資料としては、Pullum[7] やBanatre[8]などがある）。フォールトトレラント設計の狙いは、コンポーネントに欠陥がある状況（ソフトウェア異常を含む）においても、安全関連機能が確実に動作し続けるようにすることである。

通常、フォールトトレラント設計は冗長性を利用する。これはひとつのコンポーネントが正しく機能しないときに動作を継続させるために不可欠なコンポーネントを単純に複製することや、あるいは故障を検出して動作を別の動作モード（おそらくは機能を制限したモード）に切り替えるコンポーネントを追加することが考えられる。

ソフトウェア故障が発生しても重要な機能が継続できるように、フォールトトレランスを利用することができる。この場合、同じソフトウェアのコピーを複数使用する単純な冗長性では十分ではない。はぜならば、そのソフトウェアのコピーのそれぞれに同じ欠陥が存在するからである。

そのような場合には多様性が必要となる。例えば、ソフトウェアエラーを検出して回復プログラムを実行するために追加的ソフトウェアを使用することが考えられる。その追加的ソフトウェアはそれが監視するソフトウェアのいずれの機能も共有しないようにし、そうすることによってひとつのソフトウェア欠陥が両方のソフトウェアの故障につながらないようにするべきである。

より重大なケースでは、同じ機能をふたつ以上のソフトウェアアイテムで実行するが、設計と実装は、共通の仕様から始まって、各ソフトウェアアイテム単独で行う。これが「多様性プログラミング」である。ただし、複数の異なる開発エンジニアが同じ間違いを犯す傾向があり、それが多様性を無効にしてしまうおそれがあることに留意する必要がある。また、共通仕様に要求事項が含まれている可能性もある。さらに、誤作動するソフトウェアが影響しないようにするために、ボーティングのような何らかの手段を使用する必要がある。ひとつのボーティング手法を実現するには、少なくとも3の異なるソフトウェアアイテムが必要であろう。

フォールトトレランスを提供するために冗長性を使用する際は、多様性の有無に関わらず、故障が存在していることを使用者に知らせることが重要である。さもないと、フォールトトレラントな医療機器が、実際には安全性が低下した状態で動作しているにもかかわらず、安全に動作しているように見えてしまうことがある。

6.2.2.2.4 ソフトウェア原因によるリスクを軽減するための分離

ソフトウェア欠陥が同じハードウェア上で実行されている無関係なソフトウェアのエラーを引き起こす可能性もある。製造業者は、安全性に関係するソフトウェアアイテムを安全性に関わらないソフトウェアアイテムから分離し、安全性に関わらないソフトウェアアイテムが安全性に関係するソフトウェアアイテムの動作に干渉できないようにする方法を選択するべきであり（IEC 62304:2006の細分箇条5.3.5参照）、また、その分離が効果的であることを実証すべきである。これには、ソフトウェアアイテム間の意図しない競合を回避するためにソフトウェアアイテムがリソースを適切に使用することの実証も含まれる。

ソフトウェアアイテム間の効果的な分離は、ソフトウェアアイテムが予期しない相互干渉にさらされるような以下の状況に対処しなければならない。

複数のソフトウェアアイテムが共有ハードウェア（例えばプロセッサ、記憶装置、及びその他の入力／出力装置）上の時間を奪り合うときに、ソフトウェアアイテム間に意図しない相互干渉が発生することがある。これにより、ひとつのソフトウェアアイテムが意図した時間に実行されなくなるおそれがある。十分なハードウェアを提供することはアーキテクチャ上の機能であり、その機能はすべてのソフトウェアアイテムを要求されたときに実行できるだけの十分な時間を確保するための適切な仕様と計画の対象とすべきである。

同じメモリ内に複数のソフトウェアアイテムが共存することがある。この場合、ひとつのソフトウェアアイテムが予期せずに他のソフトウェアアイテムに属するデータを書き換えてしまうおそれがある。極端なケースでは、ひとつのソフトウェアアイテムが別のソフトウェアアイテムのコーディングを偶発的に書き換えてしまうこともある。多くのプロセッサ及びオペレーティングシステムでは、ハードウェアの補助でメモリ使用量を分離する手段を提供している。そのような手段がある場合は、必ず使用するべきである。そうした手段のほとんどは、ソフトウェアアイテムのひとつに欠陥があったとしても、意図しない相互干渉を防いでくれる。

複数のソフトウェアアイテムが広域変数、環境変数、及びオペレーティングシステムのパラメータを含む変数を共有している場合にも、意図しない相互干渉が発生することがある。これにより、ソフトウェアアイテムのひとつに欠陥がある場合に、ソフトウェアアイテム間で意図しない通信が発生する可能性がある。ソフトウェアアイテム間での変数の共有は最小限にとどめることが望ましい。必要であれば、共有変数は少数の特定のソフトウェアアイテムによってのみ書き換え可能で、その他のすべてのソフトウェアアイテムは共有変数を読み込むだけで書き換えられないようにするために、エンジニア全員に対してルールを通知するべきである。

最も強力な分離の手段は、相互干渉してほしくない複数のソフトウェアアイテムを別々のプロセッサで実行することである。しかし、上記で推奨するアーキテクチャ設計を実現すれば、ひとつのプロセッサ上でも妥当なレベルの分離が得られる。

ラボ環境でのシステム試験で所与のテストケースに対して十分な物理的リソースと時間リソースがあることが示されても、現場での使用においては、適用負荷又は実行環境（同じボックス上で別のプロセスが実行されている）により、危害を生じるようなソフトウェア故障が発生する。

一方、ラボにおけるテストケースで、性能が低くソフトウェアのスピードアップを性急に図るために不当な手段が取られているということが示された場合、その手段が設計を台無しにし目に見えない悪影響による他の新たなリスクを生む可能性がある。

分離を効果的なものにするには、通常動作で次のことを実証することが望ましい。

a) データフローの破損が防止されている：安全性に関わらないソフトウェアアイテムは、安全性に関連するデータを変更できない。

b) コントロールフローの破損が防止されている。

- 安全性に関係する機能は、安全性に関わらないソフトウェアアイテムの動作に影響されることなく、常に適時に実行可能である。
- 安全性に関わらないソフトウェアアイテムは、安全性に関係するソフトウェアアイテムを変更できない。

c) 実行環境の破損が防止されている：安全性に関係するソフトウェアアイテム及び安全性に関わらないソフトウェアアイテムの両方が使用するソフトウェアシステムの部分（例：プロセッサレジスタ、デバイスレジスタ、メモリアクセス特権）の破損は起こりえない。

上記のいずれかを侵害するイベント（例：ハードウェア故障）が検出され、安全性の継続の確保に必要な措置がシステムによって実施されるようにすることが望ましい。

6.2.2.3 保護手段の詳細

多くの場合、本質安全設計ですべてのハザードを回避すること及びすべての潜在的故障に対してフォールトトレラント設計を実装することは現実的ではない。それらのケースでは、保護手段が潜在的ハザードに対処するための次善のアプローチとなる。一般に、保護手段は潜在的なハザード状態を検出することによって作動し、結果を軽減するために自動介入するか、又は警報を発生させて使用者に介入を促す。
例えば治療用X線システムは、施設のドアが開けられた場合にX線発生装置をシャットダウンするソフトウェアロジック又はハードウェアを使用したインターロックシステムを備えている場合がある。そのインターロック機能は、治療上の役割を一切担っていない。その唯一の目的は、意図しない放射線被ばくの危害を軽減することにある。

時には（つまり、医療機器の機能性の喪失がハザードを招かない場合）、役割を犠牲にして安全性が達成される場合がある。例えばラボ用血液分析装置の場合、結果を出力できなくてもハザードは招かないが、誤った結果を出力すればハザードにつながりかねない。この場合、防御的プログラミングチェックで予期せぬ不具合が示されたときには、分析装置の動作を続けるのではなくシャットダウンすることでリスクを軽減する。フェイルセーフアーキテクチャでは、システム又はコンポーネントの不具合若しくはその他のハザード状態は、機能の喪失につながる可能性はあるが、操作者及び患者の安全は守られる。一方、フェイルオペレーショナルシステムでは、システムは安全に動作を継続できるが性能は低下する（例：キャパシティが下がる、応答時間が遅くなるなど）。

6.2.2.4 ハザード状態の迅速な防止と通知

リスクコントロール手段は、ハザード状態防止の確率を向上させるものが重要となる。

危害の防止に加え、検出された状態を使用者に通知することも考慮に入れるべきである。これがなければ、リスクコントロール手段が正しく機能しなかった場合に、危害が発生するおそれがある。

また、ソフトウェアのリスクコントロール手段を実行する頻度も考慮に入れることが望ましい。ソフトウェアのリスクコントロール手段は、危害が発生する前に危害につながる状態を検出できるように、十分に高い頻度で実行されるべきである。

6.2.2.5 ソフトウェア異常に対するリスクコントロール手段

ソフトウェアには特有の難しさがある。つまり、ハザード状態につながるイベントシーケンス内の一部のイベントが発見されていないソフトウェア異常に起因している可能性があり、その異常がどこで発生するか、それがどのような影響を及ぼすかを予見することが困難だ、という点である。

リスクコントロール手段はソフトウェア異常に起因する危害の確率を低減できる。ソフトウェアアーキテクチャ内には、通常、リスクコントロール手段でそれまでのイベントの性質に関わりなく危害の確率を低減できる箇所があるものである。これを慎重に行えば、ソフトウェア異常が危害を引き起こすのを防止する目的でそのソフトウェア異常の性質を厳密に予想する必要がなくなる。

このアプローチが実際的でないケース、例えば予防手段がソフトウェアに実装されている場合では、そのソフトウェアの保全性を確保するための手段を講じることが望ましい（6.2.2.6参照）。

6.2.2.6 リスクコントロール手段としてのプロセス

ソフトウェア異常がハザード状態につながるイベントシーケンスの一因となる可能性がある場合、リスクコントロール手段を講じることで危害の発生を防止することはできないかもしれない。この場合の最善のソリューションは、ソフトウェア異常が発生してもハザード状態につながらないような本質安全設計を行うことである。

これが不可能な場合、効果的なソフトウェア開発プロセスを使用することによって、ソフトウェア異常の発生確率を低減できる可能性がある。プロセスのリスクコントロール手段は、異なるタイプのリスクコントロール手段と組み合わせかつプロセスが詳細に定義されているのであれば有益である、というのが共通した認識である。

ソフトウェアが意図する機能を確実に実施できるという信頼度が非常に高いこと、またそのソフトウェアに欠陥がないことの信頼度が非常に高いことを立証できるのであれば、そのソフトウェアを保全性の高いコンポーネントと見なすことができる。この高レベルの信頼を達成するには、製造業者はソフトウェア開発プロセスが、信頼性が高く欠陥のないソフトウェアを作り出せること実証しなければならない。そうすることにより、そのようなプロセスを使用しているのでソフトウェア異常の発生確率を抑えられると断言できる。

ソフトウェア開発プロセスの厳格度を上げることによりソフトウェア異常の数を減らせる、というのは広く認められている。ただし、医療機器ソフトウェアを試験することでソフトウェア異常の数を低減できるとはいえ、ソフトウェアが計画されたすべての試験に合格したからといってソフトウェア異常が残っていないと決めてかかってはならない。なぜならば、臨床使用では、計画された試験に含まれていなかったシーケンスがソフトウェアへの入力に含まれるからである。医療機器ソフトウェアは非常に複雑で網羅的に試験することができないため、厳格な試験もハザード状態の確率を低くする手法として見なすことができるに過ぎない。また、試験だけでは、ソフトウェアが保全性の高いコンポーネントであると見なせるだけの信頼度を立証するには不十分である。

厳格なソフトウェア開発プロセスを定める上での出発点は、IEC 62304で定める活動と作業をソフトウェア開発プロセスに含めることだと言えるだろう。厳格なソフトウェア開発プロセスを開発する際に考慮すべきその他の事項としては、次が挙げられる：

- スタッフの能力 ･･･ 技能、資格、経験、及び訓練（誰がソフトウェアを開発するのか？）
- 手法 ･･･ 仕様、設計、コーディング、及び試験方法の適切性（どのような開発プロセスか？）
- 厳格度、形式性、及びレビューと検査の適用範囲（静的分析をどの程度実行するか？）
- ツール ･･･ コンパイラ、要求事項トレーサビリティ、コンフィギュレーション管理ツールなどのツールの質（ソフトウェア開発中にどのようなツールを使用するか？）

保全性の高いソフトウェアが開発できるかどうかの見込みは、堅実に実施できる反復可能なプロセスが存在するか否かに依存する。

ソフトウェア異常から生じるハザード状態のリスクを軽減するために厳格な開発プロセスを実施する場合、ソフトウェア異常から生じる故障の頻度を示すデータを収集し分析することによってリスクコントロール手段の有効性を実証することが望ましい。プロセスが保全性の高いソフトウェアを作り出しているとの主張を裏付けるためには、ソフトウェア故障が全くない、又は非常にまれであることを示す証拠を提供するべきである。

6.2.3 出所が不明なソフトウェア（SOUP）に関する考慮事項

システム設計中にSOUPの使用が決定されることがよくある。医療機器の潜在リスクが高いほど、SOUPの潜在的故障モードをより入念に分析し、リスクコントロール手段を特定するべきである。通常、SOUPを修正してそのSOUPが故障した場合にハザード又はハザード状態につながらないようにそのSOUPを監視又は分離するのに必要な新しいリスクコントロール手段を組み込むことはできない。また、SOUPに起因するすべての潜在的ハザードを特定するのに十分な内部設計情報を入手することもできない。そのため、SOUPが故障したときにそれがハザードを引き起こさないようにそのSOUPを監視又は分離するのに必要なリスクコントロール手段を提供できるように、システム及びソフトウェアアーキテクチャを設計するのが望ましい。

医療機器にSOUPが含まれる場合、医療機器の安全性が損なわれないように注意を払わなければならない。そのような状況では、「ラッパー」、つまりミドルウェアアーキテクチャが必要となる場合がある。そのミドルウェアは以下のいずれかの機能を有するのがよい。

a) SOUPの使用したくない機能の使用を防ぐ。

b) SOUPと医療機器ソフトウェアとの間で正しい情報が転送されるように論理チェックを実施する。

c) 医療機器が必要とする追加情報を提供する。

SOUPに関するその他の重要問題として、市販のオペレーティングシステムや通信システムの使用がある。徹底したリスクアセスメントに基づき、安全性を損なうことなくソフトウェアプラットフォームの変更（例：安定性、セキュリティ）を許可するアーキテクチャを構築することが望ましい。このようなリスクアセスメントには、ネットワークセキュリティパッチのインストールなど、医療機器の安全性の保全性を確保するために必要な変更頻度の分析を含めるのがよい。

6.3 リスクコントロール手段の実装

ISO 14971:2007から抜粋

6.3 リスクコントロール手段の実施

製造業者は6.2で選択したリスクコントロール手段を実装すること。各リスクコントロール手段の実装を検証すること。検証結果をリスクマネジメントファイルに記録すること。

リスクコントロール手段の有効性を検証し、その結果をリスクマネジメントファイルに記録すること。

注記　：　有効性の検証には、妥当性確認活動も含めることができる。

適合性は、リスクマネジメントファイルの検査によって確認する。

リスクコントロール手段を特定したら、それを実装し、その有効性を検証する必要がある。

リスクコントロール手段が適切に実装されていてリスクをコントロールする上で有効であることを検証することは、ソフトウェアにとって不可欠である。分析と試験の両方が必要となる可能性が高い。主要検討事項には以下が含まれる。

a) 安全性に関係するすべてのソフトウェアアイテムが特定され、そのソフトウェアのあらゆる関連バージョン及び変異形（例：異なるプラットフォーム、異なる言語、異なる医療機器モデル）において、安全性に関係するすべての機能性が特定され、実装され、試験されるようにするためのトレーサビリティ

b) リスクコントロール手段を試験する際の、広範な異常状態及びストレス状態での試験を含む、厳格度及び適用範囲の拡大

c) リスクコントロール手段及び安全性に関係する機能性に変更を加えた場合（その変更が安全性に影響を及ぼすことを意図していないとしても）、そのリスクコントロール手段及び安全性に関係する機能性に対する重点的な回帰試験

保全性の高いソフトウェアを作り出すのに十分に厳格と考えられるプロセスを使用する場合でも、その結果を検証しなければならない。

根本原因分析が行われ、安全性に関係する異常の重大性のレーティング（IEC 62304:2006参照）が追跡・評価されている場合には、検証活動及び妥当性確認活動中に収集した異常情報が役立つことが多い。安全性に影響を及ぼした異常を評価して、異常がリスク分析で特定されたかどうか、及び特定したリスクコントロール手段が十分だったかどうかを判断することができる。ソフトウェア異常に関するデータは、ソフトウェア開発プロセスの有効性の実証、又はリスクの軽減を主張するために改善を必要とするソフトウェア開発プロセスの状況の特定に使用されることがある。

ソフトウェアのリスクコントロール手段の妥当性は、ハードウェアのリスクコントロール手段の妥当性ほど明らかではないかもしれない。このため、ソフトウェアを扱うときには、リスクアセスメントの文書に従来要求されていたものとは違う書式が必要かどうかを検討することが望ましい。そのために役立ちうる方法のひとつが、「セーフティケース」を書くことである（附属書E参照）。

6.4 残留リスクの評価

ISO 14971:2007から抜粋

6.4 残留リスクの評価

リスクコントロール手段を適用した後、リスクマネジメント計画で定める基準を使用して残留リスクを評価すること。この評価の結果は、リスクマネジメントファイルに記録する。

この基準を使用して残留リスクが許容可能と判定されなかった場合は、さらにリスクコントロール手段を適用すること（6.2参照）。

許容可能と判定された残留リスクについて、製造業者は開示すべき残留リスク及びその開示のために附属文書に含める必要がある情報について決定する。

注記　：残留リスクの開示方法に関する指針は、附属書Jで提供している。

適合性は、リスクマネジメントファイル及び附属文書の検査によって確認する。

ソフトウェアに起因する残留リスクは、システムレベルでの医療機器に関わ残留リスクに含めなければならない。ソフトウェア異常の確率を推定するのは困難であることから、残留リスク評価では通常、許容できないリスクにつながるすべてのイベントシーケンスについて、その発生確率を低減する、又は危害の程度をリスクマネジメント計画（3.4.1参照）で定義したレベルに抑えるためのリスクコントロール手段を講じるかどうかを決定する必要がある。

是正されていないソフトウェア異常が（検証活動及び妥当性確認活動で）特定された場合は、それらを分析して安全性に関係するソフトウェアに影響を及ぼすかを判断することが望ましい（IEC 62304:2006の細分箇条5.8.3参照）。影響を及ぼすと判断された場合には、それらの異常から生じるリスクを評価し、それ

らが影響を与える可能性のあるすべてのハザード状態の残留リスクの評価に使用する必要がある。

6.5 リスク／効用分析

ISO 14971:2007 から抜粋

6.5 リスク／効用分析

残留リスクがリスクマネジメント計画で定める基準を使用して許容可能であると判定されず、更なるリスクコントロール手段が実施不可能な場合、製造業者は意図する使用による医学的利益が残留リスクを上回るかどうかを判断するため、データや文献を収集しレビューを行うことができる。この証拠が、医学的利益が残留リスクを上回るという結論を立証しなければ、そのリスクは許容できないままとなる。医学的利益が残留リスクを上回る場合は、6.6 へ進む。

医学的利益より小さいと実証されたリスクについて、製造業者はその残留リスクの開示に安全性情報が必要か否かを判断する。

この評価の結果は、リスクマネジメントファイルに記録する。

注記 ：D.6 も参照。

適合性は、リスクマネジメントファイルの検査によって確認する。

ソフトウェアに関する追加の手引きはない。

6.6 リスクコントロール手段から発生するリスク

ISO 14971:2007 から抜粋

6.6 リスクコントロール手段から発生するリスク

リスクコントロール手段の影響を、次の事項についてレビューする：

a) 新しいハザード又はハザード状態を招く；
b) 既に特定したハザード状態の推定リスクが、そのリスクコントロール手段の採用で影響を受けたか否か。

新しい又は増加したリスクには、4.4 から 6.5 までに従って対処する。

このレビューの結果は、リスクマネジメントファイルに記録する。

適合性は、リスクマネジメントファイルの検査によって確認する。

ソフトウェアのリスクコントロール手段の導入によりその医療機器の他の部分にどのような影響が及ぶかを入念に調査できるようにするためには、厳格な変更管理（IEC 62304:2006の細分箇条8.2参照）を含め、厳格なソフトウェアコンフィギュレーション管理プロセスが不可欠となる（IEC 62304:2006の細分箇条7.4も併せて参照のこと）。

ISO 14971は、設計／開発プロセスについて規定していない。この影響のひとつとして、リスクコントロール手段が実装されている場合、ISO 14971では単にその手段が更なるハザードを招いていないことを保

証するためにその手段をレビューすることを求めているに過ぎない、ということがある。この要求を、実装が完了した後にのみこの疑問を調査せよとの指示として解釈するべきではない。

ソフトウェアのリスクコントロール手段では、ソフトウェアが実装されるまでこのレビューを保留しないことが特に重要である。ソフトウェアのリスクコントロール手段が指定されたら、これを直ちにコンフィギュレーション管理下に置き、新しいハザード又はハザード状態を誤って発生させてしまうなどの悪影響を見つけるためにレビューすることが望ましい。

ソフトウェア設計をさらに著しく複雑にするリスクコントロール手段の実装は、潜在的なソフトウェア異常をさらに増加させたり新しいハザード状態を招く可能性がある。リスクコントロール手段はできる限りシンプルにし、常に新しいリスクアセスメントの対象とすることが望ましい。

このレビューは、（少なくとも）ソフトウェアの設計後及びソフトウェアシステムの試験後に繰り返すことが望ましい。

6.7 リスクコントロールの完全性

ISO 14971:2007から抜粋

6.7 リスクコントロールの完全性

製造業者は、特定したすべてのハザード状態から生じるリスクが確実に考慮されるようにする。この活動の結果は、リスクマネジメントファイルに記録する。

適合性は、リスクマネジメントファイルの検証によって確認する。

ソフトウェアがハザード状態の要因となっている場合には、リスクコントロールの完全性にIEC 62304:2006の細分箇条7.3.3を含めることを検討するべきである。

7 残留リスク全体の許容性の評価

ISO 14971:2007から抜粋

7 残留リスク全体の許容性の評価

すべてのリスクコントロール手段を実装及び検証した後、製造業者はリスクマネジメント計画で定める基準を使用して医療機器が呈する残留リスク全体が許容できるか否かを判断する。

注記 1：残留リスク全体の評価に関する手引きについては、D.7を参照。

残留リスク全体がリスクマネジメント計画で定める基準を使用して許容可能と判定されない場合、製造業者は意図する使用による医学的利益が残留リスク全体を上回るかどうかを判断するため、データや文献を収集しレビューを行うことができる。この証拠が、医学的利益が残留リスク全体を上回るという結論を裏付ければ、その残留リスク全体は許容可能と判断することができる。

そうでなければ、その残留リスク全体は許容できないままとなる。

許容可能と判断した残留リスク全体について、製造業者は、その残留リスク全体を開示するためにどの情報を附属文書に含める必要があるかについて決定する。

注記 2：残留リスクの開示方法に関する手引きは、附属書 J で提供している。

残留リスク全体の評価の結果は、リスクマネジメントファイルに記録する。

適合性は、リスクマネジメントファイル及び附属文書の検査によって確認する。

残留リスク全体を評価するためには、すべてのリスクコントロール手段を実装することが必要である。これには、ソフトウェアをそれが使用される各システムコンフィギュレーションとの関連において評価することも含まれる。

システム試験活動（すべてのソフトウェア機能性とハードウェアのリスクコントロールに関する）の結果は、許容基準と組み合わせて評価するべきである。また、残留するソフトウェア異常はすべてリスクマネジメントファイルに記録し、それらが許容できないリスクの一因とならないことを確認するために、評価するべきである（IEC 62304:2006の細分箇条5.8.2及び5.8.3を参照）。この評価は、必要に応じて臨床／アプリケーションの専門家による独立した学際的なレビューで受け入れられることが望ましい。また、附属文書に情報を含めることが必要になる場合もある。

8 リスクマネジメント報告書

ISO 14971:2007から抜粋

8 リスクマネジメント報告書

医療機器を商品流通に向けてリリースする前に、製造業者はリスクマネジメントプロセスのレビューを実行する。このレビューでは、少なくとも次の事項を保証すること：

- リスクマネジメント計画が適切に実施された；
- 残留リスク全体が許容できる；
- 関係する生産／生産後情報の取得のための適切な手法が実施されている。

このレビューの結果をリスクマネジメント報告書に記録し、リスクマネジメントファイルに含める。

レビューの責任は、リスクマネジメント計画において妥当な権限を持つ者に割り当てることが望ましい（3.4 b 参照）。

適合性は、リスクマネジメントファイルの検査によって確認する。

リスクマネジメントプロセスのレビューの一環としてIEC 62304:2006の箇条6及び細分箇条7.3.3を含めることを検討するべきである。

9 生産／生産後情報

ISO 14971:2007から抜粋
9 生産／生産後情報

製造業者は、生産段階及び生産後段階における医療機器又は類似機器に関する情報を収集及びレビューするためのシステムを確立、文書化し、保守する。

医療機器に関する情報の収集及びレビューのためのシステムを構築するときには、製造業者は特に次を考慮することが望ましい；

a) 操作者、使用者、又は医療機器の設置、使用、及びメンテナンスの責任者が作成する情報を収集及び処理するメカニズム；
又は

b) 新規格又は改訂規格

また、当該システムは、市場の類似医療機器に関する一般に入手可能な情報も収集及びレビューすることが望ましい。

この情報を、考えられる安全性との関連について評価するのが望ましい。特に次のことについて評価する：
- それまで認識されなかったハザード又はハザード状態が存在しないか
- ハザード状態から発生する推定リスクがもはや許容できないものになっていないか

上記のいずれかの状況が発生した場合：

1) 以前に実施したリスクマネジメント活動への影響を評価し、リスクマネジメントプロセスへの入力としてフィードバックする。
2) 当該医療機器のリスクマネジメントファイルをレビューする。残留リスク又はその許容性が変わった可能性がある場合は、先に実装されているリスクコントロール手段への影響を評価する。

この評価の結果は、リスクマネジメントファイルに記録する。

注記 1：生産後監視の側面には、国家の規制の対象となる部分もある。そのような場合、追加の手段が必要となることもある（例：予想生産後評価）。

注記 2：ISO 13485:2003 の 8.2[6]も参照。

適合性は、リスクマネジメントファイル及び他の妥当な文書の検査によって確認する。

ソフトウェアのリスクマネジメントは、ソフトウェアメンテナンスプロセス（IEC 62304:2006の箇条6参照）及びソフトウェア問題解決プロセス（IEC 62304:2006の箇条9参照）を含め、ソフトウェアライフサイクルの初めから終わりまで継続する。

IEC 62304:2006の箇条6は、医療機器ソフトウェアのリリース後にフィードバックを受領、文書化、評価、及び追跡するための手順書の使用を定めたソフトウェアメンテナンス計画を確立するよう製造業者に求めている。メンテナンス計画はまた、ソフトウェアリスクマネジメントプロセスの使用及び医療機器ソフトウェアのリリース後に生じた問題を分析し解決するためのソフトウェア問題解決プロセスの使用についても定めている。

ソフトウェア問題解決プロセス（IEC 62304:2006の箇条9参照）の使用によって、リスクマネジメント活動はソフトウェア問題の調査とその問題の安全性との関連に関する評価に統合される。臨床専門家、ソフトウェアエンジニア、システム設計者、ユーザビリティ／人間工学の専門家を含む学際的なチームをこの調査及び問題の評価に関与させることが重要である（3.3参照）。

SOUPも、ソフトウェアメンテナンス計画及び生産後リスクマネジメント活動の重要な側面である。一部のSOUPは、その特性（ウイルス防護ソフトウェアなど）から、頻繁に更新されるため、製造業者はそのことをソフトウェア保守計画で考慮に入れなければならない。

SOUPの故障又は予期しない結果、及びSOUPの陳腐化（サポート終了）は、医療機器の残留リスク全体が許容できるかどうかの判定に影響することがある。そのため、ソフトウェアシステムの開発及び保守にSOUPの監視及び評価活動を組み入れる必要がある。これらの活動でSOUPの更新、アップグレード、バグ修正、パッチ適用、及び陳腐化に対応する。公に入手可能な異常リスト及びSOUPの現地性能を積極的に監視し評価することにより、製造業者は既知の異常がハザード状態を引き起こす可能性のあるイベントシーケンスにつながるかどうかを事前に判断することができる（IEC 62304:2006の細分箇条6.1 f）、7.1.2 c）、7.1.3、及び7.4.2を参照のこと）。

製造業者からリリースされるSOUPパッチ又は更新には、医療機器の安全性及び有効性に必須ではない追加的機能が含まれていることがある。そのようなSOUP更新については、リリースする医療ソフトウェアから削除可能な過剰コンポーネントがないか分析し、ハザード状態を招くおそれのある予期せぬ変更を回避することが望ましい。

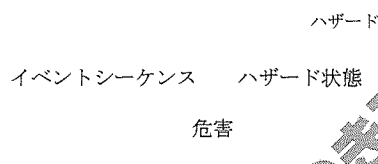
ソフトウェアアイテムの変更の場合と同様に、製造業者はSOUPの更新によって影響を受けるソフトウェアアイテムを知っておくこと及び回帰試験を実施することが望ましい（IEC 62304:2006の細分箇条7.4、8.2、及び9.7を参照のこと）。

附属書 A
（参考情報）

定義に関する審議

ISO 14971でキーとなる用語として「ハザード」と「危害」がある。このふたつの用語を理解することが、規格を理解する上で重要である。危害は身体的傷害若しくは人間の健康に対する害、又は財産若しくは環境に対する害を言う。ハザードは「危害の潜在的な源」と定義され、ハザードには（多くの）原因がある。

ISO 14971:2007では、ハザードは、一連の事象（イベントシーケンス）又はその他の状況が「ハザード状態」に至るまでは、危害を生じさせない、と定義されている（図A.1参照）。このイベントシーケンには、単独のイベントと複数イベントの組み合わせの両方が含まれる。ISO 14971:2007の附属書D.2では、ハザード、予見可能なイベントシーケンス、及びハザード状態の例について手引きを提供している。



図A.1 – イベントシーケンス、危害、及びハザードの関係図

ひとつのハザード又はハザード状態の原因は、ハザード状態を生じさせることが合理的に予測可能なイベントの組み合わせである。ひとつのハザードには考えられる原因（イベントシーケンス）がひとつ、いくつか、あるいは数多くありえる。

熱や電気エネルギー、懸垂部分とは違って、ソフトウェア自体はハザード（危害の潜在的な源）ではない。例えばソフトウェアとの接触によって負傷するおそれはない。しかし、ソフトウェアによって人がハザードにさらされる可能性はある。言い換えれば、ソフトウェアがハザード状態の原因になる可能性がある。ソフトウェア故障は（いかなる性質のものであれ）ハザードからハザード状態への変化を助長することが多い。

ソフトウェアは、主にハザードを顕在化させてハザード状態を作り出すイベントシーケンスの一因となることで、ハザード状態に寄与する。

表A.1 – ハザード、予見可能なイベントシーケンス、ハザード状態、及びそれによって生じる可能性のある危害の関係

ハザード	ソフトウェアが関与する予見可能なイベントシーケンス	原因	ハザード状態	危害
電気エネルギー	眼球インプラントによって印加される電流を制御するソフトウェア出力が高すぎる	(1) ソフトウェアのアルゴリズムに限界がある (2) ソフトウェアは正しく規定されているが、異常がある	インプラントによって患者の目に過剰な電流が印加される	重度の火傷 失明
臨床機能の喪失	生命維持を目的として設計されているソフトウェアが生命維持機能を提供できない	(1) ソフトウェアが例外的な入力データを処理できない (2) ソフトウェアが誤った機器設定を検知できない (3) ハードウェアがソフトウェアのタイムリーな動作を支援するのに十分なリソースを備えていない	(1) 必要なときに機器が治療をっ提供できない (2) 正しくセットアップされていない状態で機器が動作する (3) 機器が生命に関わる状態を警告しない	患者の状態の悪化 死亡
臨床スタッフによる患者軽視	ソフトウェアの入力及び出力が使用者を混乱させる、又は使用者の判断を誤らせる	(1) 人間/ソフトウェアインタフェースが使用者にとって分かりづらい (2) ソフトウェア出力が使用者の反応能力を上回っている (3) 使用者がソフトウェアの限界を理解していない	(1) 患者の治療ミス (2) 緊急事態にタイムリーに対処できない (3) ソフトウェアに頼りすぎて人間が自分で決断しなくなる	患者の状態の悪化 死亡

附属書 B
（参考情報）

ソフトウェア原因の例

表B.1に、ハザードに関わることの多いソフトウェアの機能分野をリストアップし、そのハザードの潜在的原因である状況の例を示す。また、リスクコントロール強化に役立つと思われる、ソフトウェア開発時に確認すべき事項の例も挙げている。この表に示されている情報の中には、すべての医療機器ソフトウェアに適用できるわけではないものも含まれる。特定の医療機器にこれらの例が当てはまるかどうかは、その医療機器の意図する使用、医療機器のシステムレベルでの設計、医療機器におけるそのソフトウェアの役割、及びその他の要因によって左右される。この表は出発点としてのみ使用していただきたい。

この表はすべてを網羅しているものではなく、安全で効果的なソフトウェアを開発するための思考過程の手助けをするものである。

表B.1 – ソフトウェア機能領域による原因の例

ソフトウェア機能領域	ハザード原因の例	確認事項
警報及び警告：		
優先度	優先度の低い警報が優先度の高い警報の表示又は警報音を隠してしまう 重大な警報が抑制されない	複数の警報状況にシステムがどう対応すべきかを仕様書に明記しているか？ 複数レベルの警報があるのか？ レベルの高い警報はレベルの低い警報の音声に優先して出力されるか？ 使用者が警報に気付いたことを知らせるまで持続させるべき警報はあるか？
保護措置	警報の状況又は警報の種類ごとの保護措置が明確でない 警報がクリアされた後に保護措置をどのように解除するかが規定されていない、又は明確でない	保護措置がユーザビリティ問題を発生させないか、つまり、使用者が保護措置から安全に移行できるか？
シャットダウン／セーフモード／リカバリー	セーフモード措置が十分でない セーフモード措置が新たなハザードを生み出している	セーフモード措置は意図する使用に適したものか？ 臨床スタッフはそのセーフモードシナリオをレビューしたか？ セーフモード状態は使用者にはっきりと分かるか？
ユーザーインターフェース	密集している又は貧弱なユーザーインターフェースが「真の」警報状況を隠している 警報に対して取るべき措置が明確でない	臨床スタッフはユーザビリティの保護手段をレビューしたか？
ログ	永続的なエラーが未解決の故障を示している ログに記録された警報が誤った患者と対応付けられている	検出したエラーはログに記録されているか？ ログは十分に大きいか？ ログの記憶装置は信頼できるものか？ ログはどのように解除されるか？ 使用者はログ解除のタイミングを承知し

		ているか？
可聴性	バックグランドノイズが警報音を消してしまう 警報音量が非常に大きいので操作者が警報を無効にするための代替的手段を探す 音声システムが故障して使用者がそれに気付かない。	可聴警報の設計で意図する使用の環境を考慮したか？ 使用者はユーザーインターフェース設計の要求事項の開発に関与したか？ 音声システムは、電源投入又は患者に関してどのように検証されているか？

表B.1 – ソフトウェア機能領域による原因の例 (続き)

ソフトウェア機能領域	ハザード原因の例	確認事項
重要な電源サイクル状態		
データの完全性	シャットダウン時に不揮発性の書き込みが進行中である 電源OFF/ONで処置を再開するための重要パラメータが維持されない 潜在的なソフトウェア異常により動作中に重要パラメータが誤って上書きされる	電源が落ちたときに進行中だったメモリ書き込みはどうなるか? ソフトウェアはもうすぐ電源が切れることを知らされるか? 電源投入時に不揮発性記憶装置は検証されるか? 重要パラメータは使用前にチェックされるか?
リセット	予期せぬリセットの後にコンポーネントとの同期がとれない 差し迫った故障の前兆かもしれないリセットが繰り返されるが、検知されない	リセットがリスクコントロール手段として使用されているか? リセットサイクル中に入出力制御が損なわれるか? 使用者にリセットが知らされるか?
リカバリー	電源投入時の初期化中に、意図する使用に医療機器が利用できない 電源投入時に不揮発性故障－対処方法が分からない 重要設定が工場出荷時の設定に戻ったことに使用者が気付かない	復帰時間は安全性の問題か? 医療機器の可用性は安全性の問題か? フェールセーフ保護手段によって、不揮発性記憶装置はどのような影響を受けるか?
電源モード	低電源状態で可聴機能又はその他の重要なユーザーインターフェース機能が利用できない 低電源状態に移行すると、気付かないうちに重要な割り込み機能が無効になる	低電力モードで支障をきたすリスクコントロール手段がないか? 低電源モードからのソフトウェアの復帰は、可能なスタートアップ状態として検証・妥当性確認活動で検討済みか?
重要なユーザーコントロール機能／ユーザビリティ		
調整／ナビゲーション／選択	ユーザーインターフェースソフトウェアプロセスは数値変更に対応するがコントロールプロセスが新しい数値を取得しない	新しい数値に調整が加えられたときに選択も確認もされない場合、使用者には通知されるか? パラメータ調整時、変更に二段階の操作を必要とすべきか?
データ入力	使用者が範囲外の数値を入力する 使用者が、範囲内だが意図しない数値を入力する	使用者に確認を促すべきか? ソフトウェアは妥当性確認のためにデータ入力をチェックしているか? 非常に重要な入力又はエラー取消しの確認に、監督者権限のログインを必要とするか?
アクセス性	遮断用制御ボタンが隠れている。 外科用手袋をして操作するとタッチスクリーン上の制御ボタン類が機能しない	使用者は、安全関連機能にアクセスするために何層のレイヤーをナビゲートしなければならないか?
画面切換	警報によってディスプレイの画面が「自動的に」切り替わる	すべての自動画面切換えを評価したか?
ユーザーインターフェース設計	一般的なタイプの色覚異常を考慮せずに色を使用する どの条件で警告が発生するか使用者が判断できない	色覚障害のある操作者はエラーメッセージをどのように解釈するか? ユーザーインターフェース設計の要求事項の開発に使用者が関与したか?

表B.1 – ソフトウェア機能領域による原因の例 (続き)

ソフトウェア機能領域	ハザード原因の例	確認事項
表示		
診断用画像	向きが逆 画像と患者との対応付けの誤り	正しい画像の向きを確保するために使用されている手法はあるか? 画像はどのように患者に対応付けられているか?
診断波形	不適切な表示フィルター エイリアシング、歪み、スケーリングエラー、時間軸誤差、非可逆圧縮	表示にはどのような周波数内容が要求されるか? 臨床スタッフがその要求事項をレビューしたか? 表示フィルターの特性は完全に明らかにされたか、つまり全入力範囲で合格・不合格だったものは何か?
ハードウェア制御		
アルゴリズム モーター制御	積分飽和、エイリアシング、タイミング、オーバーフロー、ポートエラー（シリアルポート／パラレルポート）	サンプリングレートはいくつか? PID制御の場合、積分器のゲインは制限されているか? アルゴリズムは製造されたハードウェアの全バリエーションにわたって特性が明らかにされたか? フィードバック制御の場合、フィードバック信号の妥当性確認のためにどのようなチェックが行われているか? 使用されているマイクロプロセッサ及びコンパイラについて、すべてのデータタイプが評価されたか?
エネルギー印加	治療の初め及び治療中継続的にすべてのエネルギー「ゲート」がチェックされない 安全性システムが故障したが使用者がそれに気付かない	すべてのアサーションを定期的に継続的に検証しているか? 治療制御ソフトウェアと安全監視ソフトウェアに「共通モード」エラーは存在しないか? 安全性モニタは電源投入ごと又は患者ごとに検証されているか?
ディスクリート	ビット固着 ポーリング間隔のせいでビットの変更が検出されない	ソフトウェアはビット固着 (固定されて変化しない) を検出するか? ポーリングレートについてシステムエンジニア又はハードウェアエンジニアと検討したか?
キャリブレーション／セルフテスト 範囲チェック アッセイキャリブレーション（ソフトウェア特有のキャリブレーション）	使用説明が不十分なため使用者が医療機器のキャリブレーションを正しく行えず、誤ったキャリブレーション定数になる ゼロでない信号で自動ゼロ化動作が実施される、すなわち予期せぬ圧力がカフ又はラインに掛かる、若しくはトランスデューサーに力が加わる	ソフトウェアはキャリブレーション値 (つまり傾き又はオフセット) の合理性チェック又は妥当性チェックを実施しているか? 使用者が自動キャリブレーション又は自動ゼロ化に気付くか?
ハードウェア故障検出	ハードウェア故障は検出可能だが使用者に報告されない この状態で医療機器の使用が継続される 電源投入後にハードウェア故障が発生する 電源投入時にソフトウェアはハードウェア故	すべてのハザード結果が使用者に報告されているか? ハードウェア故障は、電源投入時、各処置又はセッションの前、若しくは秒当たり一回などの継続的なベースでチェックするのがよいか?

	障しかチェックしない	
自動洗浄	使用者が洗浄又は消毒のプロセスを途中で止める	ソフトウェアは強制的にサイクルを最後まで実施するか？ ソフトウェアによる不完全な洗浄／消毒サイクルの検出を無効化できるか？

表B.1 – ソフトウェア機能領域による原因の例（続き）

ソフトウェア機能領域	ハザード原因の例	確認事項
流体送出	不適切なキャリブレーションの検出 すべての液体「ゲート」のチェックを最初に行わず、また治療中に継続してチェックすることもしない。	すべてのアサーションを定期的に継続的に検証しているか？ 安全性システムを無効にできるか、つまりチューブを安全クランプに固定していない状態でポンプを動作させることができるか？
生命維持	安全状態が定義されていない 多くのシャットダウンパスのうちのひとつが割り込みを無効にしない 生命維持機能のバックアップがない	処置や安全なシャットダウンシーケンスの遅延の影響を含め、対象母集団（例：成人、新生児）の範囲で安全状態の定義及び分析を徹底して行ったか？ ソフトウェアは「機能停止」モードをサポートし使用者に状況を通知することができるか？
モニタリング		
判断	監視用ソフトウェアの共通モードエラー 競合状態が誤った判断結果を招く	治療制御と治療監視用ソフトウェアは個別に開発されたか？ この判断ポイントに関して、ソフトウェア設計は競合状態が発生する可能性を排除又は最小化しているか？
非アクティブ化	監視システムがサブシステムをシャットダウンしたことに制御システムが気付かない 医療機器側で非アクティブ化されたパラメータをネットワーク接続されたシステムがロギングする	制御サブシステムは監視サブシステムのアクションを認識しているか？ 非アクティブ化されたパラメータは使用者又はネットワーク接続されたシステムにどのように伝達されるか？
表示	表示された数値が更新されていないが使用者はそれに気付かない 表示書き込みが複数の優先レベルで実行される	「固まった」表示をどうやって使用者に気付かせるのか？ 映像の「コンテキスト」はプリエンプションの前に保存されるか？
計測	データ取得タイミング又はサンプリングレートの誤り	信号の周波数内容に対してサンプリングレートは適切か？ 計測値がソフトウェアレイヤーを通して一貫した単位で格納されているか？
インターフェース		
引数の受渡し不良	機能はマイクロリットル単位で数値を受け渡すが、ドライバはミリリットル単位での数値を求めている 不正なポインタが受け渡される 揮発性メモリへのポインタが受け渡され、処理前に数値が失われる	各ソフトウェア機能は受け渡された引数を検証しているか？ ソフトウェア言語は、より堅牢なタイプのチェックをサポートしているか？ ソフトウェアは、数値に関してソフトウェアパッケージを通して一貫した単位で設計されているか？ 引数は優先度の高い処理レイヤーで修正されるか？

ネットワーク	ソフトウェアがホストコンピュータの応答を待つ無限ループに入る ネットワーク上の複数の医療機器に同一の「名前」が与えられ、データ損失につながる ネットワーキングデータの処理がCPUサイクルを独占し、安全性又は意図する使用の機能のためのリソースが不足する	ソフトウェアはどのような物理ネットワーク接続状態にも耐えられるように設計されているか? リモート接続は、コマンド又は偽データを繰返し送信することでシステム性能を下げることができるか? 医療機器はネットワーク名が既に使用されていないかどうかチェックするか?

表B.1 – ソフトウェア機能領域による原因の例 (続き)

ソフトウェア機能領域	ハザード原因の例	確認事項
データ		
臨床情報	システムが誤った患者の記録にアクセスしたのに、ユーザーインターフェース表示でそれがはっきり分からない システムが患者データを誤ったアーカイブに格納する	取違えを検出するループに使用者を置くために、複数の独立した識別子の表示を行うことはできるか? クロスチェックとして、実際のデータと一緒に重要な識別子を埋め込むことはできるか?
レポート	レポートが誤ったデータを提供する、若しくはデータを誤ったシーケンスの中、又は単位なしで特定する	臨床目的のためにどのようなレポートを使用するか? データが誤っている場合の危害の重大性はどの程度か? 臨床家が問題に気付く可能性はどれほど高いか?
データベース	システムレベルの故障又はSOUPの悪影響によるデータ破損	データを使用する前に破損をどのように検出するか? これを起動時だけ行うのではなく使用の都度行うことはできるか?
診断		
意思決定	アーチファクト検出の表示により、ディスプレイ上の不全収縮表示が阻止される	警報表示の階層構造を徹底的にレビューし、また臨床スタッフと共にレビューを行ったか?
データ変換	計算精度エラーによって無効な結果となる アルゴリズムが誤った単位を使用又は表示する	どの程度の計算精度が要求されるか? 十分な精度を確保するために数式をどのようにコード化すべきか?
自動化された予防保全	アプリケーションコードが実際の使用のためにデータをリトリーブしている最中に、バックグラウンド診断でそのデータが一時的に変更される。 バックグラウンド診断が適切なタイミングに干渉する	診断中にアプリケーションプロセスが適切なタイミングでロックアウトされるか? タイミングが重要なサイクル中に診断がロックアウトされるか?
セキュリティ		
コンフィギュレーションオプション	重要なコンフィギュレーションパラメータ又はデータへのアクセスの保護がない、又は不十分	使用者によって変更されるべきでない重要なデータはどれか、あるいは変更に監督者の承認を必要とすべきデータはどれか? 監査証跡は必要か?
機能アクセス	治療又は機器操作のコントロールへのアクセスの保護がない、又は不十分	操作者に対して操作の前にログインすることを要求すべきか? 患者が誤って医療機器を操作するおそれはないか?
インターフェースアクセス	通信インターフェース又はネットワークを通じて渡されるデータ及びコマンドからの保護がない、又は不十分	リモートで許可すべきものは何か? リモートシステムの仮想コントロールに頼ることが望ましいか、そうだとすればその理由は?

表B.1 – ソフトウェア機能領域による原因の例 (続き)

ソフトウェア機能領域	ハザード原因の例	確認事項
性能		
キャパシティ／負荷／応答時間	ピーク負荷中に重要タイミングが影響を受ける。 ピーク負荷下でトランザクション／インプット／アウトプットのシーケンスが影響を受ける、又はデータが失われる。 ピークシステム負荷下でモーター制御が影響を受ける。	ピーク負荷時又はキャパシティの限界に到達したとき、検出不能な方法でデータ又はタイミングが失われるか又は影響を受けるか？ ピーク負荷下でインプット及びアウトプットが正確な決定論的シーケンスのキューに入れられるか？ 重要な機能性及びリスクコントロール手段をこれらのストレス条件下で試験したか？ リスクコントロール手段を実行して限界を検出したか？ 割り込みを設定して重要な時間制約を受ける機能性と他の機能性と分けることができるか？

表 B.1 に示した安全性に関係するソフトウェア機能におけるハザードの潜在的なソフトウェア原因に加え、故障が発生したソフトウェアに関係しないソフトウェアに悪影響を及ぼす可能性のあるソフトウェア原因も数種類ある。ある種のソフトウェア欠陥がそのソフトウェアの安全性に関係する部分に対して予測できない影響を及ぼす可能性がある場合、その可能性を特定し、リスクコントロール戦略及び具体的なリスクコントロール手段を策定することが望ましい。

ハザードに対するそうした潜在的なソフトウェア原因の例、及びその例の場合に考慮すべきリスクコントロール手段を、表 B.2 に示す。要求事項に基づいたシステム試験は、これらのタイプのソフトウェア原因を特定したり、それに対応するリスクコントロール手段及びそのリスクコントロール手段の有効性を検証するには有効ではないことが多い。表の右側の欄に、それぞれの例に適切と思われる静的検証法又は動的検証法の種類に関する手引きを示している。表 B.3 には、静的検証法及び動的検証法の例を挙げる。

表 B.2 – 悪影響をもたらす可能性のあるソフトウェア原因の例

	検証タイプ：	分析：静的／動的／タイミング			
		試験（単体、統合）			
		検査			
ソフトウェア原因	**リスクコントロール手段**				
演算					
ゼロで割る	ランタイムエラートラップ、防御的コーディング	✦	✦	D
数値のオーバーフロー／アンダーフロー	範囲チェック、浮動小数点データ表現	✦	✦	D
浮動小数点四捨五入	堅牢なアルゴリズム	✦		
不適切な範囲／境界チェック	防御的コーディング	✦	✦	S
オフバイワン（OBO）	防御的コーディング	✦	✦	

表 B.2 – 悪影響をもたらす可能性のあるソフトウェア原因の例 (続き)

	検証タイプ：	分析：静的／動的／タイミング			
		試験（単体、統合）			
		検査			
ソフトウェア原因	**リスクコントロール手段**				
ハードウェア関係					
EEPROM使用：長いアクセス時間、磨滅	特殊アクセスモード（ページ／バーストモード）の使用、データ変更時は書き込みのみ、キャッシュは電源喪失時のみ書き込み及びEEPROM更新	✦		T
CPU／ハードウェア故障	電源投入時CPUチェック、プログラムイメージCRCチェック、RAMテスト、クロックチェック、ウォッチドッグチェック、不揮発性記憶装置チェック、ハードウェア応答のタイムアウト及び合理性のチェック、センサーの短絡／接地チェック、既知の信号に対するセンサー応答のテスト		✦	
ノイズ	デジタル入力のデバウンス、アナログ入力のフィルタリング、すべての割り込み（使用・不使用）に割り込み処理ルーチン（ISR）を持たせる			
周辺インターフェース異常	ADC/DACの起動に遅延を持たせる、タイミング及びその他インターフェース要求事項が常に満たされているかを検証、合理性チェック	✦		T
タイミング				
競合状態	更新時に共有リソースを指定して保護（ロック）する、共有リソースへのすべてのアクセスを取り扱う単一で再入不可能なプロセッサを実装する、共有リソース分析を実施する			S
時間デッドラインの見逃し	タイミング要求事項を明確にする、適切なリアルタイム設計アルゴリズム、優先度逆転及びデッドラインの問題を設計で排除する、非決定性タイミング構造を回避する、完成したコードにおけるすべてのタイミング条件を検証する 注記 非決定性タイミング構造には、次のものが含まれる：再帰的ルーチン、ハードウェアからの応答待ち、動的メモリ割り当て、仮想メモリ（メモリページをディスクから又はディスクへスワップ）			T
割り込みの見逃し	シンプルな割り込みアーキテクチャ（優先順位レベル数は最小限に）、ISRを短く速く、割り込みの無効化は頻繁に行わず期間も短くする、適切なリアルタイム設計			T
出力における過度のジッター	ISRを短く速く、非決定性タイミング構造を回避する、適切なリアルタイム設計、周期タスクの始め又は高優先度ISRにおいてすべての出力を更新する			T
ウォッチドッグタイムアウト	ウォッチドッグタイマリセット間の最長時間を決めてタイムアウトを適切に設定する、ハザード状態を避けつつもタイムアウトをできる限り長い期間に設定する			T
モーディング				
異常終了	出口トラップ、ランタイムエラートラップ、適切なウォッチドッグタイマ設計、すべてのプログラム入力についての妥当性確認、電源投入時セルフテスト、すべての「デバッグ」コード及びその他の非生産機能をリリース前にプログラムから除去する、誤って「特殊」モード（サービス、製造）に入れないことを確実にする			D
電源喪失／復帰／シーケンシングの問題	電源投入チェック：CPU、プログラムイメージCRC、RAM、クロック、ウォッチドッグ、不揮発性記憶装置、周辺機器など。適切な状態設計、時間平均数値の初期化、周辺機器の再初期化、非不揮発性記憶措置からのシステム状態の格納／リカバー、外部電圧モニタ／リセット回路			
起動／シャットダウンの異常	電源投入チェック（上記参照）、周辺機器及びデータの適切な初期化、適切な不揮発性メモリ使用、適切な状態設計			
低電源モードの入／切	適切な割り込み設計	✦		T

表 B.2 – 悪影響をもたらす可能性のあるソフトウェア原因の例 (続き)

	検証タイプ:	分析：静的／動的／タイミング			
		試験（単体、統合）			
		検査			
ソフトウェア原因	リスクコントロール手段				
データ問題					
データ破損	シャドーRAM、ブロックCRC又はチェックメモリ、機能を通してデータアクセスをカプセル化する、グローバルデータを最小限にする、データ構造をシンプルに保つ、コンパイラによる構造でのデータの整理方法を承知している、キャストを避ける（後述の「誤ったポインタ」及び「中間データ」も参照）				S
リソース競合問題	共有リソース分析（上述の「競合状態」も参照）				
誤ったポインタ	防御的コーディング：参照解除前の妥当性確認試験、強い型付けの言語を使用する、ポインタの使用を最小限にする、ポインタキャストを避ける	✦	✦		S
データ変換エラー：タイプキャスティング、スケーリング	タイプキャストを避ける、浮動小数点数表現を使用する	✦	✦		S
誤った初期化	初期化前時間平均変数、電源投入時にすべてのデータメモリを「0」にクリアする	✦	✦		S
平均データが範囲外	平均を算出する前（特に電源投入時）に十分なサンプルを確実に取る、又は平均値を既知の（厳しく制御の）好ましい数値にあらかじめ初期化する	✦			
ロールオーバー	合理性チェック	✦	✦		
揮発性データ	ハードウェア、ISR又は優先度が異なる複数のタスクによって変更されるすべてのデータに揮発性記憶装置クラスが使用されていることを検証する	✦			S
意図しないエイリアシング	信号の最も周波数の大きいコンポーネントの少なくとも2倍以上速いサンプルデータ（ナイキスト基準）、信号の周波数帯域の制限				
中間データの使用 注記：計算に対して割り込みや優先処理が発生する可能性がある場合、グローバル（あるいは共有）変数について計算シーケンスを実行することは望ましくない。代わりに、一時的数値変数ですべての計算を実行し、単一の割り込み不可能な指示でグローバル変数を更新する	時間の同期が想定されるデータがすべて同時に更新されることを確実にする、共有リソース分析	✦			
インターフェース問題					
ディスプレイが更新されない	イベントドリブンでなく継続的な更新とする				
人的要因：誤使用	シナリオ再構築のためのログ、コンテキストに応じたヘルプ、シンプルなユーザーインターフェース設計				
ネットワーク問題（例：マルチユーザー）	負荷試験				T
ハードウェア／ソフトウェアコンフィギュレーションエラー／誤ったドライバ	ソフトウェア開発プロセス、コンフィギュレーション管理ツール				
パッチ／更新の不良	プログラムイメージCRC及び電源投入時バージョンチェック、チェックプロトコル改訂、有効期限				
SOUP故障モード：ハングする／復帰しない、割り込みを長くロックし過ぎる、など	SOUPエラッタを調査する、ロバスト設計（例：すべてのブロックコールに対するタイムアウト）、頻繁に使用される又はISRが使用するメモリページをロックする、要求されるSOUP機能のみを使用しその他はすべて削除する	✦			T
ウイルス	ウイルスチェッカー				
ブラウザとウェブの不適合	電源投入時バージョンチェックを統合する、適合性試験				

表 B.2 – 悪影響をもたらす可能性のあるソフトウェア原因の例 (続き)

	検証タイプ:	分析：静的／動的／タイミング			
		試験（単体、統合）			
		検査			
ソフトウェア原因	リスクコントロール手段				
その他					
メモリリーク	メモリの動的割り当てを避ける	✦	✦		D
システムデッドロック	シンプルなロックストラテジー（プロセスは一定時間に一つのリソースしかロックできない）、デッドロック分析				S
再入可能性	割り込み（又は様々な優先度の複数のタスク）から呼び出されるサードパーティのライブラリを含むすべての機能が再入可能に設計されていることを確実にする。				D
スタックオーバーフロー	ランタイムスタック保護、最高水位線、スタック分析				S
論理エラー／構文	ソースコード分析ツール（Lintなど）及び／又はコンパイラ警告レベルを使用する 重要な制御ポイントにおける二重ダイバシティ及びクロスチェック				S
無限ループ	ループカウンタ、ループタイムアウト、ウォッチドッグタイマ	✦	✦		
コード破損	電源投入及びランタイムプログラムイメージCRCチェック				
デッドコード	除去されない場合は（カスタム又はソフトウェア内又はパッケージソフトウェアコンポーネントについての）デッドコードの実行開始時に警報を発するか又は安全シャットダウンを実行するエラーチェックを挿入する				D
誤った条件コード	条件的コンパイルが適切かつ必要なときにだけ使用されることを確実にする	✦			
意図しないマクロの悪影響	すべてのマクロパラメータを括弧で囲む				S
リソース枯渇	スタック、ヒープ、及びタイミングの分析				T
誤った警報／警告優先順位付け	ストレス試験				
許可されていない機能（「金メッキ」「バックドア」など）	要求事項及び設計のレビュー、トレースマトリックス				
オペレーション／優先度の誤った順序	「ブレッドクラム」、トラッキングからの呼び出し				
安全状態	独立モニタ				

リスクコントロール手段を確実に意図したとおりに動作させる方法は数多くあり、その一部はより資源集約的である。しかし、そのどれをとっても単独では十分とは言えない。いくつかを表 B.3 に示す。

表 B.3 – リスクコントロール手段を確実に意図したとおりに動作させる方法

静的解析	動的試験	モデリング
ウォークスルー	機能試験	環境モデリング
デザインレビュー	タイミング試験、メモリ試験	タイミングシミュレーション
スニーク回路解析	境界値解析	ケースワークフロー、ユーザワークフローの使用
	性能試験	
	ストレス試験	
	検定	
	エラー推定	
	スレッドに基づく試験	
	使用に基づく試験	
	クラスタ試験	

安全性に関係するソフトウェアが十分な範囲の条件のもとで試験されるようにするには、要求仕様に基づく試験に集中するのではなく、様々な種類の試験 (ストレス試験、境界試験、タイミング試験、電源異常試験、故障試験、SOUP故障試験など) を組み合わせることが望ましい。

附属書C
(参考情報)

潜在的なソフトウェア関連の落とし穴

表 C.1 にリスクマネジメント活動 (ISO 14971:2007 の各箇条による) 時、及びソフトウェアライフサイクル IEC 62304:2006 の各箇条による) 中に回避すべき潜在的なソフトウェア関連の落とし穴を示す。

表 C.1 – 回避すべき潜在的なソフトウェア関連の落とし穴

ISO 14971:2007 箇条 4：リスク分析
－非現実的な低い確率推定値をソフトウェア故障に適用し、非現実的なリスクレーティング、ひいては不適切なリスクコントロール手段を招く。 －（初期開発中又はメンテナンスの一環としてリリース後に）新しいハザード及びハザード状態又は原因が医療機器に追加されたか、あるいは既存のリスクコントロール手段が損なわれたかを判断するためのリスク分析を実行せずにソフトウェア機能を追加する。 －医療機器リスク分析プロセスはシステム及びハードウェアレベルの側面だけを定めるものであり、ソフトウェアの十分なリスク分析との関係を的確に取り扱うものでもなければ、ハザード及びハザード状態の潜在的原因としてのソフトウェア異常に個別の検討を要求するものでもない。 －リスク分析及びソフトウェア開発ライフサイクル手順の厳格度は、医療機器の潜在的危害に比例するものではない。
ISO 14971:2007 細分箇条 4.1 リスク分析プロセス
－リスク分析プロセスは、システムレベル及びハードウェアレベルの側面だけを規定したものである。 －ソフトウェアについては、ハードウェア故障のリスクコントロール手段を実装している場合にだけ対応する。 －リスク分析及びソフトウェア開発ライフサイクル手順の厳格度は、医療機器の潜在的危害に比例するものではない。 －ソフトウェアは、製品開発ライフサイクルの後ろの段階においてのみリスク分析の一環として考慮される。
ISO 14971:2007 細分箇条 4.2 意図する使用の特定 (Subclause 4.2 INTENDED USE identification)
－ユーザー環境／潜在的コンピュータシステムプラットフォームのサブセットのみを考慮する。 －プラットフォーム評価又はセキュリティ若しくはその他のSOUPパッチの必要性を考慮しない。 －潜在的なハザードの原因となる誤使用及びユーザーエラーについて十分に検討せず、したがって対応するリスクコントロール手段が特定されない。
ISO 14971:2007 細分箇条 4.3 ハザードの特定
－十分なリスクマネジメントとして、FMEA又はFTA手法だけで事足りるかのように使用する。 - FMEA又はFTAを、ハードウェア及びソフトウェアから切り離して実行する。 －次のようなハザード及びクラス全体を無視する。 ・予測不能な影響を持つソフトウェアエラー ・ハードウェア故障のリスクコントロール手段として使用するソフトウェア論理のエラー ・医療 機器の意図する臨床目的のためのソフトウェアロジックのエラー（結果計算のためのアルゴリズムなど） ・ソフトウェアプラットフォームの故障－オペレーティングシステム、ライブラリ、SOUP ・コンピュータのコンポーネント及び周辺機器の故障 ・通信インターフェースの故障 ・人的要因

表 C.1 – 回避すべき潜在的なソフトウェア関連の落とし穴 (続き)

次のような仮定に基づいて原因特定に取り組む。 ・ソフトウェアの異常は特定のコンポーネントの機能性に影響を及ぼすだけで、他のコードやデータには悪影響を及ぼさない。 ・ソフトウェアは適切に動作する。 ・潜在的なソフトウェア故障の原因は、その特定、検出、リスクコントロールを行うには数が多過ぎて予測不能である。 ・リスクコントロール手段は、ハザード状態を引き起こすソフトウェアイベントシーケンスの最初と最後にだけ実施すればよい。
ISO 14971:2007 細分箇条 4.4 リスク推定 (Subclause 4.4 Estimation of RISK)
−単一故障状態のコンセプトがソフトウェアの体系的設計の問題及びイベントシーケンスに適用されると決め込む。 −徹底的なものとはなりえない試験によって、特定の故障の確率をゼロにできると決め込む。 −予測不能な悪影響の潜在性を検討せずに、機能性に基づいて、あるソフトウェアアイテムを安全性に関係するものではないと決め込む。 −ハザードのすべての潜在的な使用者及び母集団に対する影響について、十分な臨床上の知識もなく、また十分な知識を持つ専門家を関与させずに (人的要因)、重大性を決める。 −臨床家が故障又は誤った情報に気付くという前提に基づき、低い重大性を割り当てる。 −すべての使用者が医療機器のラベル表示やマニュアルに厳格に従う、又は不用意な誤りを起こすことなく従う、という前提に基づき、低い重大性を割り当てる。 −あるハザードについて計画した一部のリスクコントロール手段を、初期重大性の割り当ての一環として想定する。この想定が誤っている場合、低い初期重大性を割り当てたときに、後でリスクコントロールが不十分であることが明らかになる可能性がある。 −ソフトウェアが使用者に提供する情報の間接的使用、処置の遅れ、並びに医療機器の有効性及び基本性能を考慮せずに、直接患者に及ぶ危害の潜在性だけに基づいて、重大性を特定する。 −臨床家は常にソフトウェアが提供する情報をクロスチェックする、又は誤った情報に気付く、という前提に基づいて低い重大性を割り当て、その他のリスクコントロール手段を実装しない。
ISO 14971:2007 箇条 5 リスク評価 (Subclause 5 RISK EVALUATION)
−主観的なソフトウェア異常確率に基づき、リスクコントロールが不要と判断してしまう。 −ハードウェアの特性を理由にあるハザードをソフトウェアの検討事項から除外したのに、後にそのハードウェアをソフトウェアが寄与因子となるような方法で変更又は除去し、そのソフトウェアに対して追加的リスクコントロール手段を考慮しない。 −ソフトウェアは意図するとおりに動作する、あるいは試験によってすべての異常が見つかる、という前提に基づき、潜在的なソフトウェア異常をハザードの寄与因子として検討しない。
ISO 14971:2007 細分箇条 6.3 リスクコントロール手段の実装 (Subclause 6.3 Implementation of RISK CONTROL measures)
−リスクコントロール手段を、通常状態又は限定状態の下で検証するが、広範な異常状態及びストレス状態の下では検証しない。 −リスクコントロール手段の実装に使用するソフトウェア又はデータがその他のソフトウェアから容易にアクセス可能なコンポーネント又はロケーションにあり、ハザードをもたらす悪影響の潜在性が高くなっている。 −リスクコントロール手段がひとつの動作プラットフォーム又はプログラム形態でのみ検証される。 −一部のリスクコントロール手段が、現象 (例：メモリ故障、競合状態、データ破損、スタックオーバーフロー) を発生させることが困難であるために、実際に実証されない。 −開発時に安全に関わる異常がすべて見つかり、試験を実施することによって現場での適切な動作が保証される、と決め付ける。 −ソフトウェア設計の複雑性を大幅に高めるリスクコントロール手段を実装する。この複雑性により、更なるソフトウェア異常の可能性が高まる、又は新しいハザードを招く。

表 C.1 – 回避すべき潜在的なソフトウェア関連の落とし穴 (続き)

ISO 14971:2007 箇条 9 生産後情報 (Clause 9 POST-PRODUCTION information)
−追加的リスクコントロール手段の導入が考えられる状況で、使用上の誤りによる潜在的に危険な現場でのイベントを無視する。 −確率又は重大性の初期推定値を、現場情報の評価を行うことなく、正確なものと決め込む。 −医療機器が、実装したリスクコントロール手段が不十分なものになりえるような予期せぬ用途に使用されている可能性を見落とす。例えば、HIV検査のためのIVDは個人の使用を意図するものだったが、公衆血液供給のスクリーニングに使用されるようになっている。
IEC 62304:2006 細分箇条 5.1 ソフトウェア開発計画 (Subclause 5.1 Software development planning)
−ソフトウェア計画／ライフサイクルプロセスでリスクマネジメント活動が確立されていない。 −ソフトウェアリスクマネジメント活動が、医療機器リスクマネジメント活動全体に関係付けられていない。 −ソフトウェアリスクアセスメントがライフサイクル内のひとつの段階でのみ行われる。 −ソフトウェアの開発者及び試験者が、リスクマネジメントの教育を受けておらず、経験もない。 −一般的なリスクマネジメント活動でソフトウェアのリスクマネジメントをカバーできると決め込む −ソフトウェアリスクが秩序正しく管理されていない。 −安全性に関する決定のトレーサビリティが確立されていない。
IEC 62304:2006 開発経路が不明のソフトウェア (SOUP) に関する考慮事項 (SOFTWARE of unknown provenance (SOUP) considerations)
−ソフトウェアアーキテクチャを定義する際にリスクアセスメント及びリスクコントロールを考慮せず、本質安全設計によるリスクコントロール手段を実装しない。 −試験をすることによって効果のないアーキテクチャを十分に安全なものにできる、と決め込む。 −アーキテクチャの安全性に関係する部分を特定せず、これらのアーキテクチャ要素が後で変更又は削除されたときに未知の安全性リスクを招く。
IEC 62304:2006 箇条 5.4 ソフトウェアの詳細設計 (Subclause 5.4 Software detailed design)
−正常ケースの取扱いにのみ焦点を当て、インターフェース及びコンポーネント間でやりとりされるパラメータが正しいという前提に基づき、複数レベルのエラーチェックを組み込まない。 −詳細設計ブレーンストーミング及びその後のレビューにおいて、ハザード及びハザード状態を招くおそれのある潜在的ソフトウェア故障及びそれに関連するリスクコントロール手段の特定を検討しない。 −リスクマネジメント活動においてソフトウェア故障の原因 (附属書 B 参照) を無視する。
IEC 62304:2006 細分箇条 5.5 ソフトウェアユニットの実装及び検証 (Subclause 5.5 SOFTWARE UNIT implementation and VERIFICATION)
−低品位の、本質的に不安全な、又は過度に複雑な設計でも、最高のコーディング及び／若しくは試験プロセス、定石、ツール、又は従業員によって補うことができると考える。 −未熟な開発者を重要コードの開発に携わらせる。 −特定の防御的プログラミングの定石を規定して要求することをしない。 −特に重要コンポーネントについて、コード検査又は静的コード分析を行うことなく、もっぱら動的試験に依存する。 −設計要求事項のリスクマネジメントとの関連性を理解せず設計から逸脱する。 −重要コンポーネントに対するユニット試験を、回帰試験の一環として繰り返さず、開発の早期段階で一度しか実行しない。 −試験の焦点をもっぱら動的な、ブラックボックスの、システムレベルの手法に当て、静的及び動的なホワイトボックス検証を実施しない。