1 開発段階 － 定義		
活動	**定義**	**価値**
プロセス要件の定義	ソフトウェアによる部分的又は全面的な自動化について検討しながらプロセスを定義する。この場合、プロセスとしては、製造プロセス又は品質システムプロセスが考えられる。プロセス要件の定義では、プロセス又はソフトウェアのリスク分析を行う際に検討されるプロセス内における検証又は予防策についても説明する。 この活動からのアウトプットは以下のいずれか一つ又は両方で文書化される。 • プロセスフロー図 • ビジネス、製造、又は品質システムのプロセス内で行われた活動を定義する要件書	ライフサイクルの後の方で決定される事項の基礎を確立し、ソフトウェアの使用に関するコンテクストを定義する。プロセス内におけるソフトウェアと上流及び下流の活動との境界を明確にする上で役立つ。 ビジネスプロセスの要件を“ユースケース”に変換し、それを利用してソフトウェアの詳細な要件及び受入試験のための試験例を作成できる。 プロセス要件は、ソフトウェア販売業者を選定する際、業者から提供されたテストスクリプトのベースとして利用できる。
プロセス障害リスク分析	プロセス障害が機器の安全性/効率性、製造担当者、環境、又は品質システムに及ぼす影響を判定する。 プロセス障害リスクは、FMEAなどの公式な方法、又はプロセスに障害がある場合に発生する可能性がある危害のタイプを簡単なリストにまとめるなどの非公式な方法で判定できる。	プロセス障害の影響及び重度を確定し、それによりソフトウェア開発及び検証努力のレベルに関する決定を促す。
意図する使用	単純なソフトウェアの意図する使用の定義は、少ないセンテンス又はパラグラフで構成される。一方、大型で複雑なソフトウェアの意図する使用には、数ページにわたる広範な文書、及びソフトウェア要件の詳細が含まれる。リスクは、意図する使用の定義の深さを決定する上で重要な要素でもある。 意図する使用の要素は以下の通り。	意図する使用の定義は、ソフトウェアの適用範囲及び要件を理解し、そのソフトウェアの使用におけるリスクのレベルを評価するための基礎である。 意図する使用を定義することは、規制対象の活動を実行するソフトウェアを製造業者がどの程度信頼しているかを理解するための実践的な方法である。 ソフトウェアの目的及び意図は、システムの使用及びコンテクストに関する高度な視野と関連リスクに

1 開発段階 ― 定義		
活動	**定義**	**価値**
意図する使用（続き）	• ソフトウェアの目的及び意図 ○ この要素は以下のものによって定義されるソフトウェアの使用を説明する。 • ソフトウェアの使用：ソフトウェアに関する主な質問への回答を含む（セクション 4.3.1.4 参照） • 規則に準じた使用：ソフトウェアの使用によってコンプライアンスがサポートされる場合に規則への具体的なリファレンスにより、品質システムとの関連で、ソフトウェアがどのように使われるかの決定を含む（セクション 4.3.1.4 参照） • 品質システムプロセス内におけるソフトウェアと他のソフトウェア又はユーザーとの境界 ○ 既製ソフトウェアの場合、その目的及び意図する使用は、既製ソフトウェア開発業者の観点からみた一般的な使用ではなく、機器製造業者の観点からみた既製コンポーネントの具体的な使用を説明するものである。 ○ ソフトウェアの意図及び目的は、プロセスに対する高度な視点を必要とし、ソフトウェアを品質システムプロセスの要素又はコンポーネントの一つとみなす。 ○ ソフトウェアの意図及び目的は、ソフトウェアが何をするのかではなく、ソフトウェアのコンテクストを説明する。 • ソフトウェアの使用に関する要件 ○ この要素は、ユースケース、ユーザー要件、又はその他のシステム要件の形でソフトウェアの使用を説明し、ソフトウェアの目的及び意図を満たすためにユーザーがソフトウェアに求める機能を定義する。	関する見識をもたらしてくれる。これにより、検証アプローチの審査担当者のための基本的な情報がもたらされる。 ソフトウェアの使用に関する要件は、ユーザーがそのソフトウェアを使って何をするのかを読む人に正確に理解してもらうために必要なユーザーベースの視野をもたらしてくれる。 ソフトウェアに関する要件は、受入試験のための基本的な枠組みをもたらしてくれる。また、ソフトウェアに期待される機能を説明するコミュニケーションツールとしても有用である。 ソフトウェアに関する要件は、ソフトウェアの開発者が何を開発すべきかを正しく理解し、ソフトウェアの試験担当者が何をテストすべきかを理解するための詳細な透視図をもたらしてくれる。また、ソフトウェアアーキテクチャ、ソフトウェア設計、ソフトウェアコード、及びソフトウェア試験を開発するための確実な基礎をもたらしてくれる。

1 開発段階 － 定義		
活動	定義	価値
意図する使用（続き）	○ 使用に関する要件は、詳細なソフトウェア要件の視点ではなく、プロセス及びユーザーワークフローの視点をもたらしてくれる。 ○ ソフトウェア使用に関する要件は、詳細な文書に記録された追跡可能なものである。 • ソフトウェアに関する要件 ○ この要素は、目的/問題が主にソフトウェアの意図及び目的並びに使用によって定義される場合、問題を解決したり、目標を達成するために、ソフトウェアが満たさなければならない条件又は性能を説明する。 要件要件は、正確、完全、明白、及び測定可能又は客観的に検証可能なものでなければならない。 ○ この活動のアウトプットは、特注のソフトウェア開発をソフトウェアの設計に利用可能な場合、及び既製のソフトウェアを調達の基準として利用可能な場合、ソフトウェア要件仕様書（IEEE）であると考えられる。	

1 開発段階 － 定義		
活動	定義	価値
検証プランニング	検証プランニングの定義は以下の通り。 • （GCSSDT）“プロジェクトのために採用されたアプローチを説明する管理文書。この計画では、通常、行うべき仕事、必要な資源、採用すべき方法、遵守すべき構成管理及び品質保証の手順、プロダクトの組織などについて説明する。プロジェクトの中には、統合計画、セキュリティ計画、試験計画、品質保証計画などが必要なものもある。” • （NIST）“開発段階の後で実施される検証活動に関するプランニング。リスク及び/又はハザード分析活動の後に行われる。ソフトウェアのリスクが低い場合、‘最低限の計画’で必要な活動が指示される場合もある。” 検証プランニングは2段階で実施される。 • 最初に、開発段階から定義段階にかけて、検証の文書化で期待される詳細及び努力のレベルを定義し、精査（経営陣の関心、部門横断的な参加、及び独立審査）のレベルを定義し、定義段階に含める活動を選択する。 • インプリメンテーション段階では、定義段階及び関連するリスク分析活動での決定に基づき、適切な検証活動を選択する。 検証プランニングからのアウトプットは、ソフトウェアが一貫して意図する使用の要件を満たしているという信頼を確立するために行われる活動を説明する計画である。	検証プランニングでは、各種の提出物に求められる厳密さ及び精査のレベルに関する決定事項、それらの提出物に求められる内容の範囲、及び使用するツール及び方法の選択を文書化する。これにより、検証プロセス全般で応用される批判的思考の証拠がもたらされる。

1 開発段階 － 定義		
活動	定義	価値
ソフトウェア要件の公式審査	（IEEE）“ソフトウェア又はハードウェア品目に関する要件がプロジェクト担当者、管理職、ユーザー、顧客、又はその他の関係者に提示され、コメント又は承認が得られるまでの間のプロセス又はミーティング。例として、ソフトウェア要件の審査が挙げられる。 要件審査は、意図する使用の定義のいずれか又はすべてのレベル（ソフトウェアの目的及び意図する使用、ソフトウェアの使用に関する要件、又はソフトウェアに関する要件）で個別又は複数同時に行われる。	ソフトウェア要件仕様書は開発及びインプリメンテーションプロセスの主要なインプットであることから、ソフトウェア要件仕様書が正確、完全、明白、及び測定可能又は客観的に検証可能なものであることを保証するために公式審査が行われることが多い。
ソフトウェア開発ライフサイクルモデルの選択	ソフトウェアライフサイクル全体の開発に関する部分で使用するライフサイクルの方法及びコントロールを定義する。 IEC 62304は一部のソフトウェアのプロセス標準として最適なものと考えられる。	これまでの結果、適切なライフサイクル開発モデルを選択することで、ソフトウェアのクオリティ及び信頼性が向上している。

1 開発段階 － 定義		
活動	**定義**	**価値**
リスクマネジメントプランニング	リスクマネジメントのプランニングには、以下のサブ活動のいずれか又はすべての準備が含まれる。 • 高リスクを伴う機能の特定 • 規制又はリスクの影響から見たソフトウェア機能の特定 • 機能分割及びリスクのトレーサビリティ • ソフトウェアが危害につながる確率を確定する目的で“分割度”（直接または何らかのプロセス方法による）を利用した潜在的な危害に関するソフトウェアの機会/近似性の特定 • リスク評価、リスク審査、リスク分類の活動 • ライフサイクルプロセスを通じたリスク分析の反復 リスクマネジメントプランニングのアウトプットとして、リスクに関係するこのソフトウェアのさまざまな問題エリアを分析するためのアプローチ、及び故障モード影響解析（FMEA）や故障樹解析（FTA）などのリスク分析法の選択が挙げられる。	リスクマネジメントのプランニングは、ソフトウェアの潜在的な危害に関する見識をもたらし、検証努力におけるツール及び努力レベルの選択を促してくれる。
製造/業務プロセス内におけるリスク予防策の特定	リスク/ハザードの予防策（手順管理など）を特定するメカニズム。予防策が用意され、適切に機能していることを確認するための継続的なモニタリングを含む。	リスク要因がプロセスに入り込むのを予防する。

2 開発段階 － インプリメンテーション		
活動	**定義**	**価値**
ソフトウェア障害の分析（リスク分析）	自動化するプロセス及びプロセス障害の分析で特定された問題エリアに関係するソフトウェア障害の影響を判定する。安全なソフトウェア慣行を確立するにはAAMI TIR32の採用が最適と考えられる。リスク評価及びリスク評価法の詳細については付録Bを併せて参照すること。 詳細なソフトウェア障害分析の具体的なテクニック： • ソフトウェア故障樹解析：ソフトウェアが原因でそれ自体が危険な状態に陥る可能性がないことを証明したり、ソフトウェアが危険な状態に陥る可能性のある環境条件を明らかにするために使用する。 • 故障モード影響解析（FMEA）：特定の故障モードがソフトウェアの挙動に及ぼす影響を判定するために使用する。	この活動により、ソフトウェア内にある高リスクエリアの試験に重点を置き、設計に採用すべきリスク予防策の特定を推進することができる。ソフトウェアの要件の脆弱な部分又は欠落した部分を明らかにし、ハザード又はリスクの潜在的な可能性が高いエリアでのロバスト性を向上させることができる。
ソフトウェアアーキテクチャの文書化及び審査	ソフトウェアアーキテクチャは、ソフトウェア要因の上部構造及びそれらの要因の関係定義する。 この活動では、アーキテクチャを文書化し、正確性、完全性及びソフトウェアの性能について審査（検証）を行う。	コミュニケーションを改善し、要件をサポートする上でのアーキテクチャの妥当性及び適正を検証する。この段階でソフトウェアのリスクを正しく評価することで、予防策を単純化することができる。
設計仕様書	ソフトウェアの要件がどのようにして履行されるかを正確に記載した文書。通常含まれるものとして、ソフトウェア又はコンポーネントの構成、アルゴリズム、制御論理、データ構造、データセットの使用に関する情報、インプット/アウトプットのフォーマット、インターフェースの説明などが挙げられる。 ソフトウェアの設計は、サイジングなど、さまざまな形式の分析によって推進される。初期段階で必要な機能の特定を容易にするために、プロトタイプを設計してもよい。	ソフトウェアがどのように機能するかを理解するために、アーキテクチャとコードの間にある抽象的な層を明らかにしてくれる。具体的なインプリメンテーションの要件を追跡する上で必要な構造を明らかにしてくれる。

2 開発段階 － インプリメンテーション		
活動	**定義**	**価値**
開発/設計審査	審査を実施し、一つまたは複数の構成アイテム用に選択された設計アプローチの進行状況、技術的な妥当性、及びリスク解決策を評価する。この審査には以下の目的がある。 • 構成アイテムに関する要件の適合性を設計ごとに判定する。 • 定義のレベルを評価し、選択されたインプリメンテーションの方法及びプロセスに関連する技術的なリスクを測定する。 • 構成アイテムと機器を構成する他のアイテム、施設、ソフトウェアと人の間にある物理的及び機能的なインターフェースの存在及び互換性を確定する。さらに、場合に応じて、 • 予備的な業務文書及び補足文書を評価する。	設計をコードに変換できるように、ソフトウェアの要件及び設計の仕様書が正しく作成されていることを確定する。 ソフトウェアが複雑で高リスクな場合、又は重要な環境で動作する場合は特に有用である。
ソフトウェア設計におけるリスク予防策の特定	リスク評価で特定されたリスク/ハザードの予防策を特定する。継続的なモニタリングを実施して予防策が用意されて正しく機能していることを確認できる反復的なプロセスでなければならない（手順管理、ハードウェア冗長性など）。	ソフトウェア使用の残留リスクを制限し、それを意図する使用に準じた許容可能なものにする。
コードレビュー/コード検証	欠陥の発見・除去及びコード全体のクオリティ向上を目的としたソフトウェアソースコードのピアレビュー。コードレビューには以下の種類がある。 • 単一ピアによる非公式なレビュー • 非公式なグループレビュー • 公式ミーティングでのウォークスルー • 役割及び責任が割り当てられた公式な検査 コードレビュー及びコード全体のクオリティは、共通コード化基準を確定し、それを遵守することで向上する。	ソフトウェア設計のコードへのインプリメンテーションが適切なことを確認する。プロセスの初期にエラーを発見・除去する最後の機会。

2 開発段階 － インプリメンテーション		
活動	**定義**	**価値**
トレーサビリティ分析	設計、コード、試験、リスク/ハザード分析及びリスク予防策に関する要件のトレーサビリティ。プロセス要件へのトレーサビリティを含めることもできる。	トレーサビリティ分析は、未定義の要件を特定し、すべての要件が検証済みであることを確認するためのカバレージ解析を可能にする。 トレーサビリティは、回帰試験及び保守活動でも有用である。設計又はコードに変更が生じた場合、それを要件に準じてトレースバックすることができる。

2 開発段階 － インプリメンテーション		
活動	**定義**	**価値**
販売業者の監査	ソフトウェア販売業者のシステムについて、その業者が安全かつ有用なソフトウェアを供給する上で十分な能力を備えていることを購買者に保証するために必要なレベルに達しているかどうかを評価する。販売業者の監査には、以下をはじめとするさまざまな方法を利用できる。 • 開発及びサポートの慣行に関する質問を記載した簡単なアンケートを業者に送付する。 • 業者が供給するソフトウェアに対するユーザーの要求を調査する。 • 業者を選定し、試験又は欠陥追跡などの問題エリアについて現場監査を行う。 • ソフトウェアの仕様、開発、試験、導入及びサポートに関連する全業者のシステムについて、品質システムの徹底的な現場監査を行う。販売業者監査を実施することで、同等又はより重要な意図する使用について業者システムの使用状況を確認できる。 チェックすべき具体的な要素として、以下の例が挙げられる。 • 業者が作成した"既知の問題リスト"のチェック • 業者が作成した基本システム検証資料のチェック • "Out-of-the-box"ソフトウェア作業フロープロセス図のチェック • "Out-of-the-box"標準レポートライブラリーのチェック • 標準作業フロー及び業務規則に加えられた構成変更のギャップ分析 • 業者から供給された導入検査及び認定用自動化試験ツールの説明及び転帰	販売業者のプロセスを知れば、ソフトウェア製品に対する自信を強めたり、弱いエリアを認識できるほか、ユーザーが必要に応じて受入試験をカスタマイズすることができる。問題の報告手順及び問題解決プロセスをきちんと定義していれば、問題が発生した場合にそれを解決する上で役に立つ。

3 開発段階 ー 試験		
活動	定義	価値
試験プランニング	試験プランニングでは、ソフトウェアがその意図する使用に準じていることを裏付ける信頼醸成をサポートする試験活動の全体的なアプローチを定義しなければならない。しかし、ソフトウェア試験だけでは、ソフトウェアがその意図する使用に準じていることを裏付ける信頼醸成には不十分である。試験と他の検証テクニックを併用して、包括的な検証アプローチを実現する必要がある。 リスク推進要因などの要素を踏まえて試験のレベルを決定し、開発者試験やユニット試験、統合試験、ユーザー試験、負荷試験、操作試験などの適切な試験法に基づき、ソフトウェアが要件及び設計の仕様に準じていることを裏付ける上で適切なレベルの信頼を獲得しなければならない。 試験で確認された不具合を記録し、それに対処するためのプロセスを定義する。 （IEEE）意図する試験活動の適用範囲、アプローチ、リソース、及びスケジュールを明記した文書。それにより、試験項目、試験すべき機能、試験作業、責任、必要なリソース、及び不測事態対応計画を特定する。	価値の低い活動に費やされる時間を短縮して検証努力を節約する。 試験活動の役割及び責任を明らかにする。 これにより、試験に採用するアプローチ及び方法の根拠がもたらされる。監査に利用可能な以下の証拠を得ることができる。 • 全体的な検証努力に必要な適切なボリュームの試験が実施されている。 • すべての不具合への対処が適切に行われている。

3 開発段階 ― 試験		
活動	**定義**	**価値**
ユニット試験	(1) (NIST) 誤字、統語誤用、及び論理エラー用モジュール、ユニット設計の正しいインプリメンテーションのためのモジュール、及びユニット要件適合のためのモジュールの試験。 (2) (IEEE) 単一のソフトウェア要素 (ユニット又はモジュール) 又はソフトウェア要素の集合のための設計のインプリメンテーションを確認する目的で実施する試験。	オーナーの管理下でコーディングが行われる場合にのみ該当 (改良、インターフェースなど)。 ユニットごとに検証のための試験を行う代わりに、開発業者のユニット試験を監査/モニタリングすることができる。 システムのレベルからは通常到達できないレベルでのソフトウェア試験に価値をもたせる。その好例として、管理された試験環境で"絶対に起きてはならない状況"及びなかなか起こり得ない状況について、エラー状態及びエラー回復の試験を行う場合が挙げられる。
データ検証	データの正確性を確認するために遂行される活動。データの移送、変換、又は試験の一環として、若しくは独立して行われ、状況に応じて統計学的サンプリングを含めることもできる。	データの正確性における信頼を醸成する。
統合試験	(IEEE) ソフトウェア要素、ハードウェア要素、又はその両方を組み合わせて試験を行い、ソフトウェア全体が統合されるまで、それらの連携を評価する通常の試験進行。	この活動の価値は、データ構造を経由して通信しなければならない主要なソフトウェアコンポーネントの試験、又は複数のハードウェア要素をカバーする機能を備えたソフトウェア、及び通信手段が電子的で通信プロトコルによって制御されるソフトウェアの試験を行うところにある。ユニット試験と同様、このタイプの試験は、タイミングやエラー回復、デッドロックなど、システム又はユニットのレベルでシミュレーションが難しい連携を調査する上で必要となる。

3 開発段階 ― 試験		
活動	定義	価値
ユースケース試験	ユースケース試験は、システム又はコンポーネントの内部機構又は構造を無視した機能試験の一形態であり、選択されたインプット及び実施条件に反応して生じたアウトプットに重点を置く。ユースケースごとに関連のあるインプットパラメータを用意し、実際の使用条件をシミュレーションすることが特定された数値のセットをそれぞれのパラメータに割り当てることができる。目標を達成の手順を示す所定のフローを利用して、一連のユースケースを関連づけることができる。	システム全体が正しく機能し、そのシステムが現実的な使用シナリオ及び作業量を処理し、実際の生産に近い環境においてユーザーに受入可能な方法でデータを効率的に処理できることを保証する。
インターフェース試験	アウトプットからインプットに至るまでの全体的なデータ移送経路を考慮しながら、ソフトウェアアプリケーション間のインターフェースを確認する。これは直接的なインターフェース試験又は100%データ検証により遂行可能である。試験活動には、インターフェースが正常と異常の両ケースで要件に準じて仕様限界又は境界条件で機能することを確認するための戦略が含まれなければならない。	データフロー検証は、高リスクアプリケーションの既知のリスクを低減するための試験法の一つである。 この活動の価値は、異種のソフトウェアコンポーネント又はアプリケーションをカバーする機能を備えたソフトウェアの試験を行うところにある。経験上、コード又はデータをわずかに変更しただけでも、境界条件で適切な試験を実施しなければ、ソフトウェアインターフェースは故障点になりやすいことが判明している。
回帰試験	（NIST）ソフトウェアの開発及び保守段階で行われた変更又は修正に起因するエラーを発見するために、それ以前にプログラムが正しく機能していたときの試験例を再実施すること。 ソフトウェアがその意図する使用を満たすために実行しなければならない基本試験例を定義・記録する必要がある。回帰試験活動を正しく実行するために、保存可能な構成パラメータセット又はデータセットの定義又はベースライン化が必要となる場合もある。	修正後も引き続きソフトウェアが当初の要件仕様に適合していることを確認する。 他のモジュールやアプリケーション、オペレーションシステム、データベース構造に小規模又は大規模な変更が加えられた後も、ソフトウェア又はソフトウェアモジュールがその意図する使用に適合していることを証明する際に使える非常に有用なツールである。

3 開発段階 － 試験		
活動	定義	価値
業者供給の試験スイート	これらのスイートは、ソフトウェアソリューションの最大能力をテストし、エンドユース環境でのソフトウェアの性能に対する大きな信頼を獲得することができる。しかし、講じられているリスク予防策の試験を含め、定義された意図する使用及び試験の完全性に対する妥当性についても評価が行われなければならない。ソフトウェアの使用期間中、販売業者に試験スイートの保守を要請する契約が必要になる場合もある。	“通常使用”又は基本構成におけるソフトウェアの精度に対する信頼を確立する上で有用なツール。ビジネスでの“ユースケース”を補強し、エンドユース環境での正しい機能を保証しなければならない。
ソフトウェアシステム試験	(IEEE) 統合されたハードウェア及びソフトウェアの試験を実施し、ソフトウェアが所定の要件に適合していることを確認するプロセス。このような試験は、開発環境とターゲット環境の両方で実施される。この試験には、サブセットとして、ソフトウェアのインストール及び構成検証活動が含まれなければならない。 ソフトウェア検証は、ソフトウェアがその意図する環境で意図するユーザーに使われる場合の適合性を確認するものであり、ソフトウェアシステムの試験とは異なる。ソフトウェアシステムの試験では、ソフトウェアに関する要件が適切に履行されていることだけを確認する。 自動化製造システムの場合、プロセス検証試験で、これらの試験の一部又は全部をカバーすることができる。品質システムアプリケーションの場合、ソフトウェアの作業指示書に規定された手順をすべて実施すれば、ソフトウェア試験の要件をカバーすることができる。	この試験は、ユーザー試験に最も類似している。このような試験の中には、手順が細かく定められていて再現性が高く、専門知識を有する担当者によって実施されるものもあれば、意図する使用環境で意図するユーザーによって行われるものもある。使用エラーで発見不能なソフトウェア設計及びインプリメンテーションのエラーを発見できるところに価値がある。 この試験では、適切なバージョンのソフトウェア及び関連ハードウェアとソフトウェアの構成が、ターゲット環境で仕様書又は設計に準じて正しく設置されていることを確認する。
ユースケース試験	ユースケースに基づいて実施される試験。それらのユースケースで定義されている代替フロー及びエラー状態を含む。	ユースケース試験は、ソフトウェアと対話するユーザーの観点から行われる。ソフトウェアとの通常及び異常な対話の際にユーザーが遭遇するであろう欠陥を発見できる。

3 開発段階 ― 試験		
活動	定義	価値
ノーマルケース試験	（GPSV）通常のインプットによる試験。 （IEEE）（1）結果を観察又は記録し、ソフトウェア又はコンポーネントの何らかの側面を評価しながら、所定の条件下でソフトウェアまたはコンポーネントを操作するプロセス。（2）既存の状態と要求された状態の差（バグなど）を発見し、ソフトウェアの機能を評価する目的でソフトウェアを分析するプロセス。	要件の妥当性を検証する。 備考：期待される有効なインプットだけでソフトウェア製品の試験を実施しても、そのソフトウェア製品を徹底的にテストしたことにはならない。従って、ノーマルケース試験は、それ自体ではソフトウェア製品の信頼性に対する十分な信用を獲得することはできない。
ロバスト性試験（ストレス試験）	（GPSV）予想外の無効なインプットが与えられた場合、ソフトウェア製品が適切に動作することを実証するソフトウェア試験。 そのようなテストケースを特定する方法として、同値類分割、境界値分析、及び特殊例特定（エラー推測）などが挙げられる。 （IEEE）所定要件の限界又はそれを超えた状態でシステム又はコンポーネントを評価するための試験。 このタイプの試験では、ソフトウェアが障害を起こす負荷及びその障害発生の仕組みを判定できる。性能試験と同じプロセスを採用しながら、非常に高いレベルの負荷をシミュレーションした状態でストレス試験が実施される場合が多い。	ソフトウェア内の弱点を特定し、ソフトウェアが通常の生産作業負荷を受けた状態で機能することを証明できる。 リソースの問題を特定し、システムアーキテクチャの弱点を判定するのに役立つ。 負荷を受けた状態のグレースフルデグラデーションが結果として破局的な障害に至らないことが望ましい。
アウトプット強制試験	（GPSV）所定（又はすべて）のアウトプットがシステムによって適切に生成されていることを確認するためのテストインプットを選択すること。 アウトプット強制では、システムから特定のアウトプットを生成する目的でいくつかのテストケースを設計する。ここでは、システムの反応を誘発するインプットではなく、希望するアウトプットを生成することに重点が置かれる。	要件及びシステムアウトプットの妥当性を確認する。

3 開発段階 ― 試験		
活動	定義	価値
インプットの組合せ試験	ソフトウェアユニット又はシステムが操作の実行中に遭遇する可能性があるインプットの組合せを利用した試験法。 エラー推測では、システムの特定エリアで発生が予想されるエラーの項目別リストを作成し、それらのエラーをチェックするためのテストケースを設計する。 因果関係のグラフ化は、テストケースに採用するソフトウェア製品へのインプットの組合せを体系的に特定するための機能的ソフトウェア試験法の一つである。	ここで特定される機能的試験法は、個別又は単一の試験インプットに重点を置く。ソフトウェア製品の大半は、それぞれの使用条件下で複数のインプットを使って操作が行われる。 エラー推測を拡充してインプットの組合せを特定できるが、これは特殊なテクニックである。 エラー推測は、科学的な試験法というより技能的な試験法といえるが、システムの履歴に精通した試験官が実施すれば非常に効果的である。
ベータ試験	少数のクライアントを対象に販売業者がライブ環境で実施する試験。 （プレスマン）一ヵ所又はそれ以上のエンドユーザー施設で、開発者による監督が行われない環境において、顧客がソフトウェアのライブアプリケーションで実施する受入試験。	ベータ試験は通常、所定の手順や監督のない状態で行われる。意図する使用の分析や試験に最適なシミュレーションである。一般的にラボで実施される試験ほど厳密ではないが、予測不能な使用状態をテストするのに適した方法といえる。 しかし、ベータ試験はライブ環境で行われるため、検証が完了するまで製品を保留の状態にしたり、独立した代替的な対策を講じて、試験中のソフトウェアに依存することなく、プロセスのアウトプットのクオリティを検証しなければならない。
性能試験	（NIST 500-234）反応時間、CPU使用率、及び操作で数量化された他の機能の要件に準じて、ソフトウェアシステムがその程度作業を実行するかを測定すること。 性能試験では通常、自動化された試験スイートが使われる。それにより、通常、最大、及び例外のさまざまな負荷条件を簡単にシミュレーションすることができる。	システムが生産現場の環境で要求された反応を達成できるかどうかを確認する。 この種の試験は、使用頻度の高いアプリケーションにおける性能上の弱点を特定する上で特に有用である。 性能試験は、購入検討中のサーバやミドルウェアなど、サードパーティ製品の環境でのベンチマーキングにも応用できる。

4 開発段階 － 導入		
活動	定義	価値
ユーザー手順のレビュー（ソフトウェアの使用に関する手順を含む。ソフトウェアの使用が適切に定義されていることを保証する）	ソフトウェアの使用に関連するユーザー手順/指示のレビュー。指示は完全、正確、かつ明確か。ソフトウェアの使用に関連するユーザー文書の正確性を審査/検証する。	ソフトウェアが“正確”でもユーザー指示が不正確だと、ソフトウェアは正常に機能しない。不正確な場合、ソフトウェアの誤用につながる可能性がある。
アプリケーションの社内トレーニング	文書に規定され、ソフトウェアに特化したトレーニング活動。	ソフトウェアの使用又は習得が“困難”な場合に重要性を増す。ソフトウェアのリスクプロフィールにあまり依存しないソフトウェア特性の一つである。
据付時適格性確認	ソフトウェアが、インストール指示書の規定に準じてインストールされ、機能していることを裏付ける信頼を確立する。	ハードウェア及びソフトウェアのバージョン/構成、すべてのコンポーネントの連携機能が正常であることを確認する。
運転時及び性能適格性確認（プロセス検証が行われる場合）	運転時適格性確認 － 製造プロセス及び関連システムが所定の限度及び許容範囲内で正常に動作可能なことを裏付ける信頼を確立する。 性能適格性確認 － プロセスの有効性及び再現性を確立する。	これらの活動は、プロセス検証に必要なものであり、エンドユース環境でソフトウェア開発試験を実施する機会をもたらしてくれる。ソフトウェアが正常に機能することを裏付ける信頼も確立する。
最終受入試験	最終的な導入の直前に行うシステムの試験。通称“稼働試験”と呼ばれる。	ソフトウェアがその意図する使用に適合し、正常に機能していることを裏付ける信頼を確立する。
オペレータ検定試験	トレーニング受講者がトレーニングで示す適格性を確認すること。	製造アセンブリステーションで要求されることが多い。オペレータエラーを減らす手段。

5 保守段階		
活動	**定義**	**価値**
保守プランニング	保守プランニングに関連する方法として、以下の例が挙げられる。 • フォワードプランニング ― この方法は、フォワードプランニング及びソフトウェアへの変更予測をカバーする。定義されたプロセスとソフトウェアインプリメンテーションに特化した活動又はツールの標準セットを利用し、ソフトウェア変更を分類・評価するための全体的な枠組みを提供する。セキュリティのアップグレード又はパッチに関するプランニング及び戦略を含む場合もある。この方法は、保守段階に入る前、最初のソフトウェアインプリメンテーションで利用できるほか、保守段階では随時利用可能である。 • 審議中の変更プランニング ― この方法は、ソフトウェアの変更が審議中に行われるプランニングをカバーする。このプランニングは通常、審議中の変更に特化した活動に重点を置く。このプランニングは、ソフトウェアの*保守段階*に行われる。	このツールは、ソフトウェアの検証された状態を維持する活動に構造をもたらしてくれるほか、初回導入時から保守活動時にかけて重要なソフトウェアに関する重大な情報を収集すると考えられる特定ソフトウェアの変更評価及びインプリメンテーションの統一アプローチを提供する。例えば、ソフトウェアに特注のコンポーネントがある場合、このコンポーネントは、既製の基本ソフトをアップグレードする際、取り扱いに関する特別なニーズを有する。特注コンポーネントは、アップグレードの際に、再構成又はメインアプリケーションからの隔離が必要となる場合もある。
既知の問題分析	購入したソフトを使って規制プロセスを自動化する場合、使用中の検証済みソフトウェアに適用される定期的なアップグレード及び/又はパッチを受け取ることがある。既知の問題分析は、インストールされたソフトウェアの使用及び/又は検証された状態への影響について、販売業者が認知するソフトウェアのあらゆる問題を評価するプロセスである。	この分析を行えば、悪影響の原因となり得る変更が加えられたソフトウェアのインストールを回避することができる。
互換性試験	(GCSSDT)複数のソフトウェアシステムが情報を交換する能力を判定するプロセス。	どの外部ソフトウェアがバージョンの不一致により現行のソフトウェアに影響を及ぼすかを特定する。 アウトプットは、ソフトウェアがリソースをインターフェースで連結したり、共有する他のソフトウェアに有用な場合もある。
インフラ互換性分析	ソフトウェアインフラへの変更がインストール済みソフトウェアにどのような影響を及ぼすかを判定するプロセス。これには、ハードウェア又はシステムのロケーションへの変更が含まれる。	インフラの変更がソフトウェアに関する予測不能な機能の問題の原因になるのを回避する。

5 保守段階		
活動	**定義**	**価値**
システムモニタリング	以下の方法を利用して、次に紹介するさまざまな観点からシステム全体の"健康"を判定できる。 • 意図する使用が変更されているかどうかの定期的なチェック － ソフトウェアが現行の意図する使用に引き続き対応しているかどうか、意図する使用又はソフトウェア、若しくは両方に変更が求められているかどうかを確定する。 • エンドユーザーによる実際の使用 － ユーザーによるシステムの使用が保守の際にモニタリング可能な情報を生み出す。エラーログ、ヘルプデスクコール、障害レポートを使用できる。 • トレーニング効果の評価 － エラーログ及びヘルプデスクコールは、トレーニングプラン又はプログラムの効果判定のソースになり得る。 • 欠陥分析 － この方法を利用してシステム内のバグの状態をチェックする。バグは社内で報告される場合と販売業者から報告される場合がある。この方法は、社内で開発されたものと既製品、両方システムに応用できる。 • データ監査 － この方法を利用して、データの保守がシステム内で正しく行われていることを確認する。	ソフトウェア又はシステムの継続的な適合性及び応用性に関する情報を提供する。 保守活動（改良、適応又は完全化）のきっかけとなる傾向を特定する。 トレーニングの評価に適したツール。 ソフトウェアが所定の要件を満たしていることを裏付ける信頼をさらに醸成する。 ソフトウェアの意図する使用又はリスク予防策に影響を及ぼす可能性がある既知のバグに対して、問題回避又はシステムの追加変更を行うことができる。 これは、検証プロセスの有効性を評価するツールのほか、新規又は修正要件のソースになり得る。
バックアップ及びリカバリープロセス	バックアップ及びリカバリープロセスには、システムバックアップ、バックアップメディアの保存及び保持、及びバックアップメディアからデータを復元するためのリカバリー手順などが含まれる。	この方法を利用して、データ消失を回復可能なものにする。データ保持の要件を満たすメカニズムを提供できる。

5 保守段階		
活動	定義	価値
運用管理	バックアップとリカバリーのプロセス、モニタリング及びレポート作成のほかにも、ソフトウェアが意図した通りに動作することを保証するための運用管理がある。一般的な方法として、以下の例が挙げられる。 • セキュリティ － この方法は、データの消失、汚染、悪用、及びシステムの意図しない使用を予防するための管理策を利用する。 • アクセス権管理 － この方法は、システムへのアクセスを許可したり、システム内でアクセスを管理することで、ユーザーによるシステムの使用を管理する。 • データベース管理 － データベースを管理し、所定のプロセスを利用して効率的な運用状態でデータベースの保守を行う。 • アーカイブ化 － アーカイブ化の方法を利用して、システムの日常業務に必要とされないデータを管理する。 • 不測事態対応計画 － コンピュータ化されたシステムの障害発生時も運転を継続することが目的である。	意図する環境でシステムを適切に使用することが、検証された状態を維持する上で重要なポイントである。
回帰分析	回帰分析には、トレーサビリティ分析又はインパクト分析など、システムの検証された状態を維持するための活動を決定する作業が含まれる。	この分析を適切に実施すれば、変更案の影響を大きく受けるエリアに検証努力を集中させ、ソフトウェアの変更されない部分が変更の影響を受けないことを確認するための試験など、適切な試験を決定することができる。