Technical White Paper

Developing compliant critical software systems with multicore processors

マルチコアプロセッサ搭載システムのスタンダード準拠と WCET について

www.ldra.com

内容

はじめに

マルチコアプロセッサと、それが日常生活にもたらした利点は周知されています。これらは 2000 年代初頭からパーソナルコンピュータで採用され[1]、NVIDIA は 2010 年頃からモバイルデバイスでの利点への理解を促進していました[2]。マルチコア設計は、プロセッサのクロック速度や精度を維持するための冷却といった、物理的な制限に対処します。単一のプロセッサチップ上の追加コアにより、デバイスメーカーは中央処理装置(CPU)で処理できるデータ量を効果的に増やして、クロック速度の問題を回避しました。

これらの原則が確立されると、当初は 2 個のコア設計であったものが、4 個、6 個、さらには 10 以上のコアによって補完されるまで、そう長くはかかりませんでした。初期の設計では、すべてのコアは常に互いに同一であり、つまり、均質(または対称)のマルチコアプロセッサ(MCP)でした。

これらのプロセッサは、SMP(対称型マルチプロセッシング)オペレーティングシステム用に構築されています。これらのオペレーティングシステムは、負荷を分散するためにコア間でプロセスをスケジュールします(これは、Windows、Linux、iOS、および Android が MCP の機能を活用するために使用するアプローチです)。

タスクは、実行される処理の流れの中の連続したコードシーケンスとして定義できます。ノート PC やモバイルデバイスら産業用制御システム、自動車システム、航空アプリケーションに至るまで、コンピューティングシステムは、1 つ以上のタスクを同時に実行するように設計されています。

ここでの重要な差別化要因は、さまざまな運用環境で許容される刺激に対する応答時間です。

ノート PC やモバイルデバイスは、制御プロセスが期限なしで特定の速度で実行される標準的な制御処理を展開します。これらのシステムのパフォーマンスを向上させることは、単にプロセッサの速度を上げることなのです。

リアルタイム制御システムは閉ループであり、データの収集、そのデータの処理、システムの更新には、厳しい時間枠が課されます。ハードウェアとソフトウエアのアーキテクチャも進化しており、これらの時間枠の計算と管理に影響を与える可能性があります。アーキテクチャが何であれ、時間枠を逃すとシステムの安定性が低下し、セーフティクリティカルなアプリケーションに壊滅的な影響を与える可能性があります。

産業ピラミッド

実際には、この時間枠の重要性は、閉ループ制御システムと開ループ制御システムの間の単純な二項対立ではありません。たとえば、図 1 の産業ピラミッドは、製造業の環境に典型的な階層構造の制御アーキテクチャを表しています。

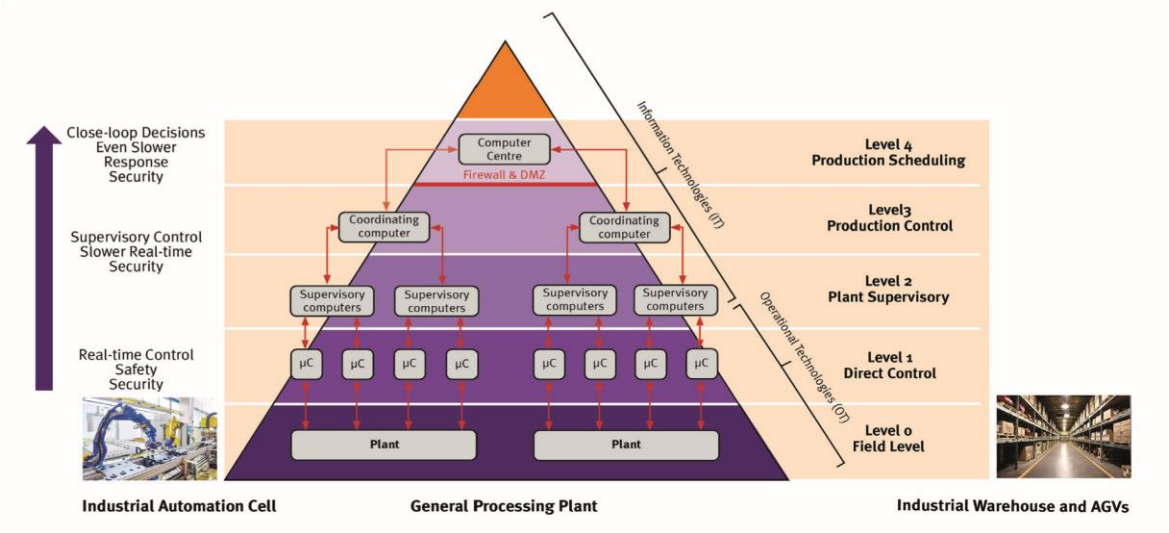
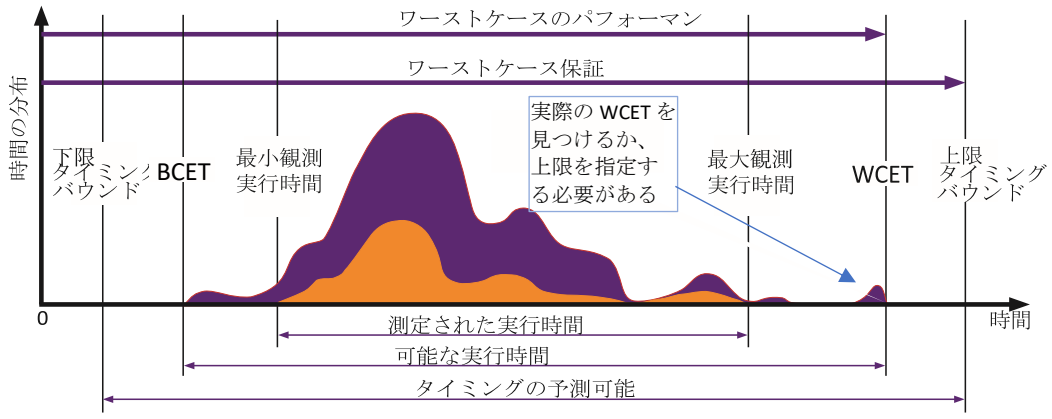


図 1: 産業（またはオートメーション）ピラミッド

これは、産業用自動生産プラントの様々な情報技術(IT：Information Technology)と運用技術(OT：Operational Technology)のレイヤー(「レベル」と呼ばれます)を分類します。すべてのレベルには、独自のタスクとインフラストラクチャがあります。図 1 に示す産業ピラミッドでは、レベル 0(Field Level)はデバイスとセンサーに最も近く、レベル 4(Production Scheduling)は製造現場から最も遠くにあります。デバイスやシステムの特性はそれに応じて異なります。レベル 0 のシステムでは、安全性、セキュリティ、および厳密なリアルタイム制御ループ(マイクロ秒からミリ秒)が最重要です。レベル 2 の監視制御システムでは、低速ではあるが決定論的なループ(数ミリ秒から秒)が必要です。レベル 3 および 4 のシステムには、開ループシステムで十分である可能性が高く、リアルタイム要件はそれほど厳しくありません。

最悪実行時間（worst-case execution times）の重要度

この例で示されているように、ハードリアルタイムシステムは、制御するシステムに由来する厳しいタイミング制約を満たす必要があります。一般に、これらの制約を満たすには、実行時間の上限が必要です。最終的に問題となるのは、システムが何か他のことをする必要が生じる前に、必要なことを完了するのに十分な時間があるかどうかです。境界を指定して検証することで、これを統計的に検証できます。リアルタイムシステムが、シングルコアプロセッサで実行され、並行に実行される複数のタスクで構成されているとします。図 2 は、リアルタイムタスク[3] のいくつかの関連する特性を示しています。



WCET: Worst-Case Execution Time, BCET: Best-Case Execution Time, ACET: Average-Case Execution Time

図 2: WCET と BCET は、それぞれプログラムで可能な最長と最短の実行時間です

通常、タスクは、入力データまたは環境のさまざまな動作に応じて、実行時間に一定の変動を示します。すべての実行時間の集合は、上側の曲線で示されています。最短の実行時間はベストケース実行時間(BCET：Best-Case Execution Time) と呼ばれ、最長の実行時間は最悪実行時間(WCET：Worst-Case Execution Time) と呼ばれます。ほとんどの場合、状態空間が大きすぎて、考えられるすべての実行を網羅的に調べて正確な WCET と BCET を決定することはできませんが、これらの値を有用な範囲で推定できる近似値があります。

WCET とマルチコアプロセッサ

シングルコアプロセッサは複数のプロセスを並列に実行することはできず、代わりに高速なスケジューリングを使用して、あたかも並列に実行しているように見せかけます。1973 年に Liu と Layland によって証明されたように[4]、そのようなアプローチを取るための非常にしっかりとした根拠があります。シングルコアプロセッサの場合、マルチタスクのリアルタイムシステムは、十分な CPU ヘッドルーム(容量)が許されている限り、期限に間に合うことが保証されます。

マルチコアプロセッサ(MCP)は、複数のプロセスを純粋に並列に実行するという点で、余分なレベルの複雑さをもたらします。シングルプロセッサのアプリケーションとは異なり、Y 個のプロセッサコアで X 個のタスクを実行して、すべてのタスクが期限に間に合うようなスケジュールを見つける効率的なアルゴリズムがありません。

ノート PC や携帯電話で使用されているオペレーティングシステムは出来の良いものですが、タスクの期限に間に合うことを保証することはできません。

マルチコアプロセッサでは、プロセス間でハードウェアが共有されている場所であればどこでもハードウェア干渉が発生する可能性があることが、この問題を悪化させます(図 3)。たとえば、多くの場合、階層メモリ全体が共有されるため、多くの場所で干渉が発生する可能性があります。これらの干渉チャネルにより実行時間の分布が広がります。実行時間の分布は、幅の狭いピークがあるものではなくロングテールで広くなります[5]。

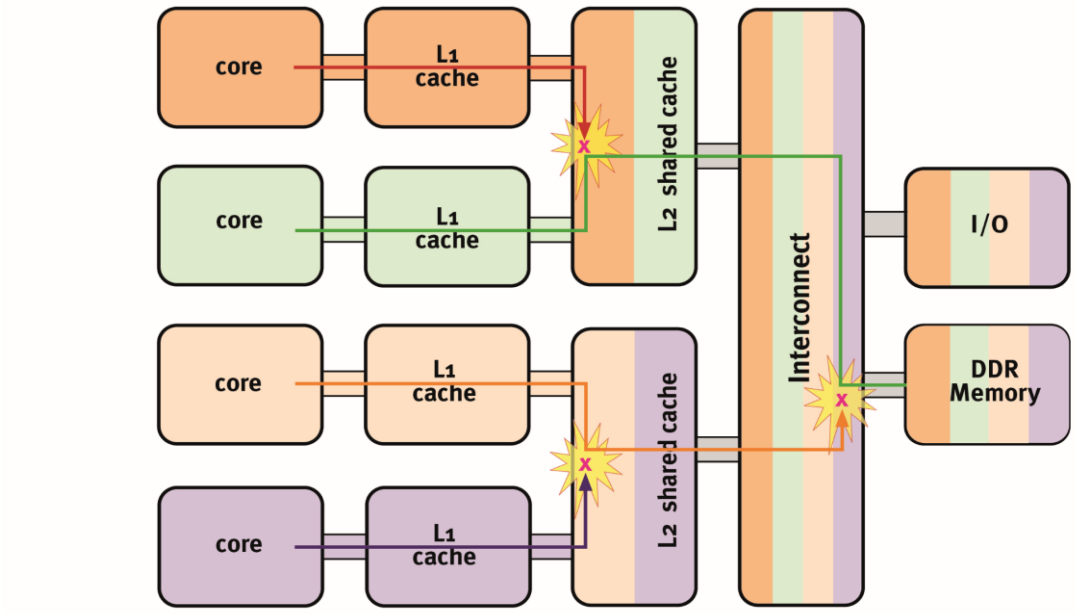


図 3: ハードウェア干渉の発生場所 [5]

セーフティクリティカルなアプリケーションの分野以外では、これらの問題はほとんど重要ではありません。しかし、機能安全が最優先される分野では、特に重要です。

機能安全規格と WCET

ノート PC や携帯電話ではまったく合理的である自由放任主義のアプローチは、ここでは通用しません。分野に跨った安全性を保証する規格では、この問題が適切に処理されることが要求されます。

民間航空機: DO-178C、CAST-32A、AMC 20-193 および AC 20-193

長年にわたり、民間航空機でのマルチコアプロセッサの使用は、マルチコアプロセッサのいくつかの特性に対する懸念から、シングルコアの使用のみに制限されてきました。CAST-32A [6]は、それに対処するために書かれたアドバイザリーポジションペーパーでした。これは、認証局がすべての商用ソフトウエアベースの航空宇宙システムを承認するための主要な文書である DO-178C[7]を補足するものでした。

RTCA の「DOcument」シリーズ（DO-178、DO-330、DO-278 など）は厳密にはガイドラインの集まりであり、規格ではありません。ただし、それらは非公式に規格として知られており、便宜上ここでは規格として参照されています。

同様に、DO-178C は、一般には機能安全規格として分類されていませんが、機能安全の課題に対処しています。

DO-178C は WCET 解析の必要性を確立し、§6.3(ソフトウエアのレビューと解析)、§6.3.4(ソースコードのレビューと解析)、および§11.20(ソフトウエア成果の要約)でそれを強調しました。

CAST-32A とその後継文書である AMC 20-193 [8] および AC 20-193 は、これに基づいて構築されており、DO-178C 準拠のアプリケーションにマルチコアプロセッサを用いる場合に満たすべき基準に関する詳細なガイダンスが含まれています。

干渉チャネルに関して、これらは具体的に次のことを強調しています。「*MCP プラットフォームによってホストされているソフトウエアアプリケーション間で干渉を引き起こす可能性のある干渉チャネルを特定し、それらの干渉チャネルの影響を緩和し、選択した緩和手段を検証することが重要です*」

安全関連システム: IEC 61508

IEC 61508 [9]「電気・電子・プログラマブル電子安全関連系の機能安全」は、参照規格として広く受け入れられています。多くの場合、セーフティクリティカルなシステムの開発に直接適用されますが、その汎用性により、業界および分野に固有の規格を導出するための理想的な「空白のキャンバス」にもなります。

IEC 61508 パート 3 §7.9 は、ソフトウエアの検証に関係しています。§7.9.2.14 では、タイミング性能の検証についてより具体的に述べており、「*タイミング性能の検証：時間領域での動作の予測可能性を検証する必要がある*」ことを要求しています。さらに、「*タイミング動作には、最悪実行時間が含まれる場合がある*」と注記しています。

車載アプリケーション: ISO 26262

一般的な IEC 61508 規格の派生物である ISO 26262 [10]では、ソフトウエアの安全要件にタイミング制約を含めることを要求しています。コードレベルでの最悪実行時間とシステムレベルでの応答時間の両方を考慮する必要があり、「*適切なスケジューリング特性*」への参照があります。

時間的制約、特に最悪実行時間、もソフトウエアアーキテクチャ設計(パート 6 §7.4.5) の一部です。

鉄道および GTS システム: EN 5012x シリーズ

IEC 61508 の他の派生物と同様に、EN 5012x [11] シリーズの規格は、応答タイミングとメモリの制約に関する詳細を示しています。EN 50128 §D.45 は、「*平均および最悪の条件下での需要分布を特定する分析を実行する*」ことを要求しています。

医療機器: IEC 62304

IEC 62304 [12] も IEC 61508 派生物であり、米国 FDA 規制 [13] の影響も大きく受けています。IEC 62304 §5.2.2 は、機能および能力要件の定義に「*タイミング要件*」を含める必要があることを示唆しています。

影響とベストプラクティス

WCET の解析

シングルコアプロセッサの場合でも、第一原理に基づく WCET の計算は簡単ではありません。実行時間のエンドツーエンド測定や、各関数、ループなどでそれぞれのアセンブラ命令をカウントし合計するといった手動の静的解析手法など、いくつかの方法が存在します。

この目的のための静的解析ツールは存在しますが、数学的な解析による WCET の確定値の計算は、一般的なケースでは解決できません。それゆえ、Reinhard Wilhelm らによると、このようなアプローチでは近似値を適用する必要があり、これは正確である必要がありますが、完全である必要はありません[3]。その結果、そのようなツールは必然的に「安全側」に倒すことになります。これは、何もないよりはましですが、精度がすべてである環境では、MCP によって導入される追加の予測できない変化がないとしても、理想的とは言えません。

ただし、特定のプラットフォームでの実行に依存しないソフトウエアコードの特性を測定するために利用できる、長年にわたって実証済みのメカニズムがあります。

たとえば、Halstead のメトリクス[14] は、ソフトウエアモジュールのサイズ、ソフトウエアの複雑さ、データフロー情報を評価するための、さまざまな言語でのアルゴリズムの実装や表現を反映したもので、これらはソースコードの静的解析から正確に計算できます(図 4)。このようなアプローチでは、コードのどのセクションが最も処理時間を要求するかを特定できますが、最大経過時間の絶対値を提供することはできません。

Halsteads (*Cashregister.c*)

File	Total Operators	Total Operands	Unique Operators	Unique Operands	Vocabulary	Length	Volume
Total for Cashregister.c	149	230	16	56	72	379	2338

図 4: LDRA ツールスイートの TBvision を使用して計算された Halstead のメトリクス [15]

同じ静的解析から関数コールグラフも生成され、この解析によって明らかになった最も要求の厳しい関数がコードベース全体のコンテキストで実行される場所を視覚化する手段を示します (図 5)。

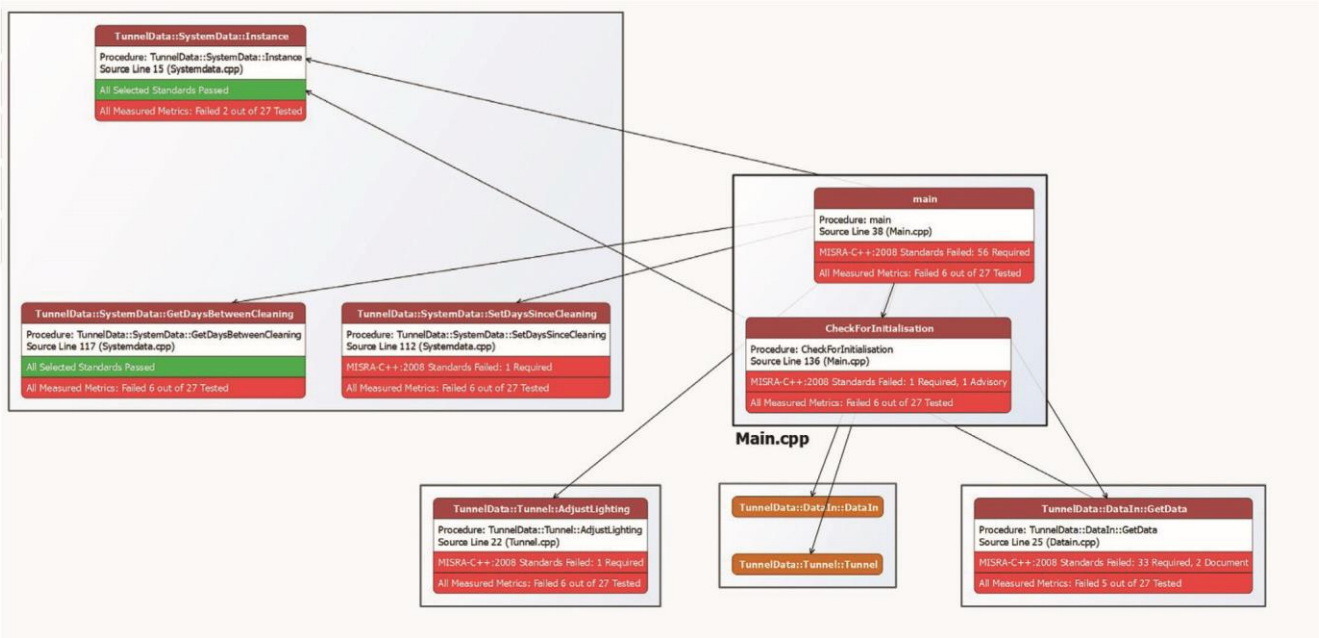


図 5: LDRA ツールスイートの TBvision によって生成された関数コールグラフは、最も要求の厳しい関数が呼び出される場所を視覚化する手段を提供します

その情報が特定の関数またはコールツリーの経過時間にどのように変換されるかを確認するには、それを実行する環境で測定することが最良の方法です。これは、補助的な TBwcet モジュールを備えた LDRA ツールスイートの単体テストツールである TBrun を導入することで動的に測定できます (図 6)。

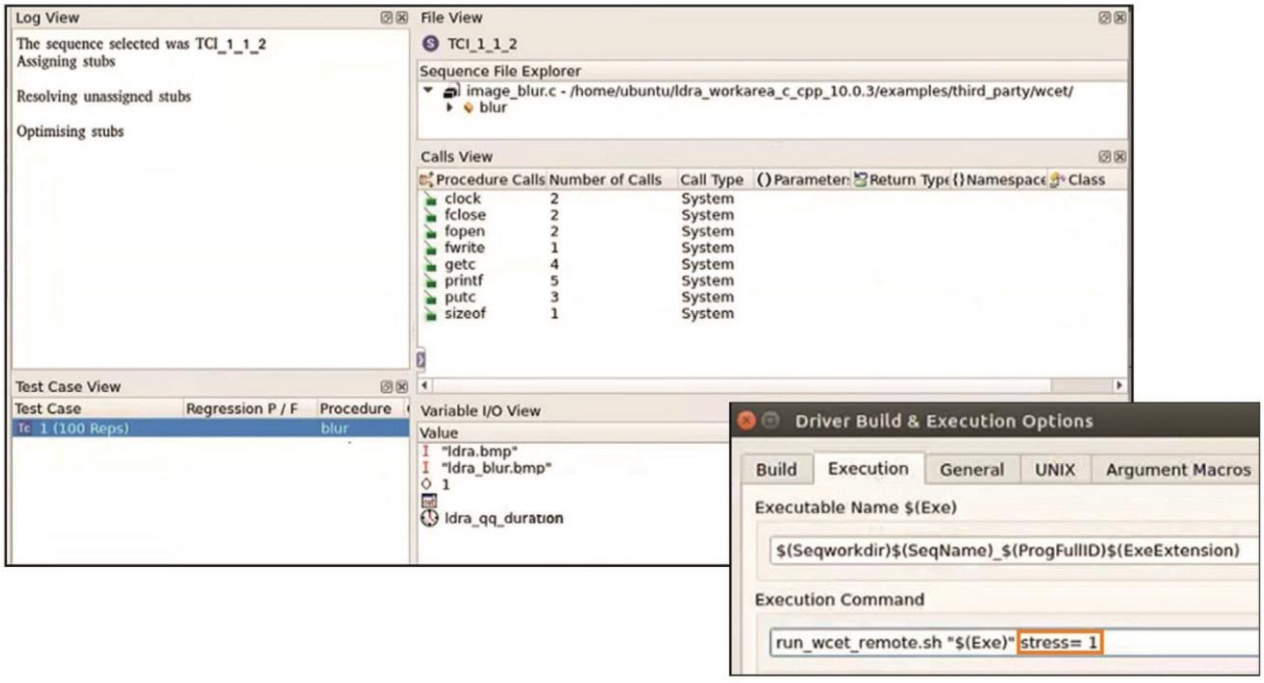


図 6: LDRA ツールスイートの単体テストツールである TBrun [16] と TBwcet [17] を使用して実行時間を測定する

このようなツールは測定を自動化し、指定されたテストを繰り返し実行して、実行時間の変動をグラフィカルに表示します (図 7)。これにより、ユーザーは好みのハードウェアストレスユーティリティ (stress-ng [18] など) を選択して、ターゲットプロセッサ上の種々の共有リソースにさまざまな程度で負荷をかけることができます。

この例で、干渉が実行時間に悪影響を及ぼしていることがわかります。観測された WCET は 4.50 億 CPU ティック強（約 3.08 億から増加）で、平均実行時間は 2.29 億 CPU ティック弱（約 1.16 億から増加）です。カバレッジの目標が満たされ、観測された WCET が範囲内にある場合、干渉の緩和は適切です。そうでない場合、結果のデータはシステムをさらに最適化するために必要な情報を提供します。

この機能は、干渉の調査を実施し、結果として干渉を緩和することで、最終的に WCET の検証要件が満たされることを実証するのに役立ちます。

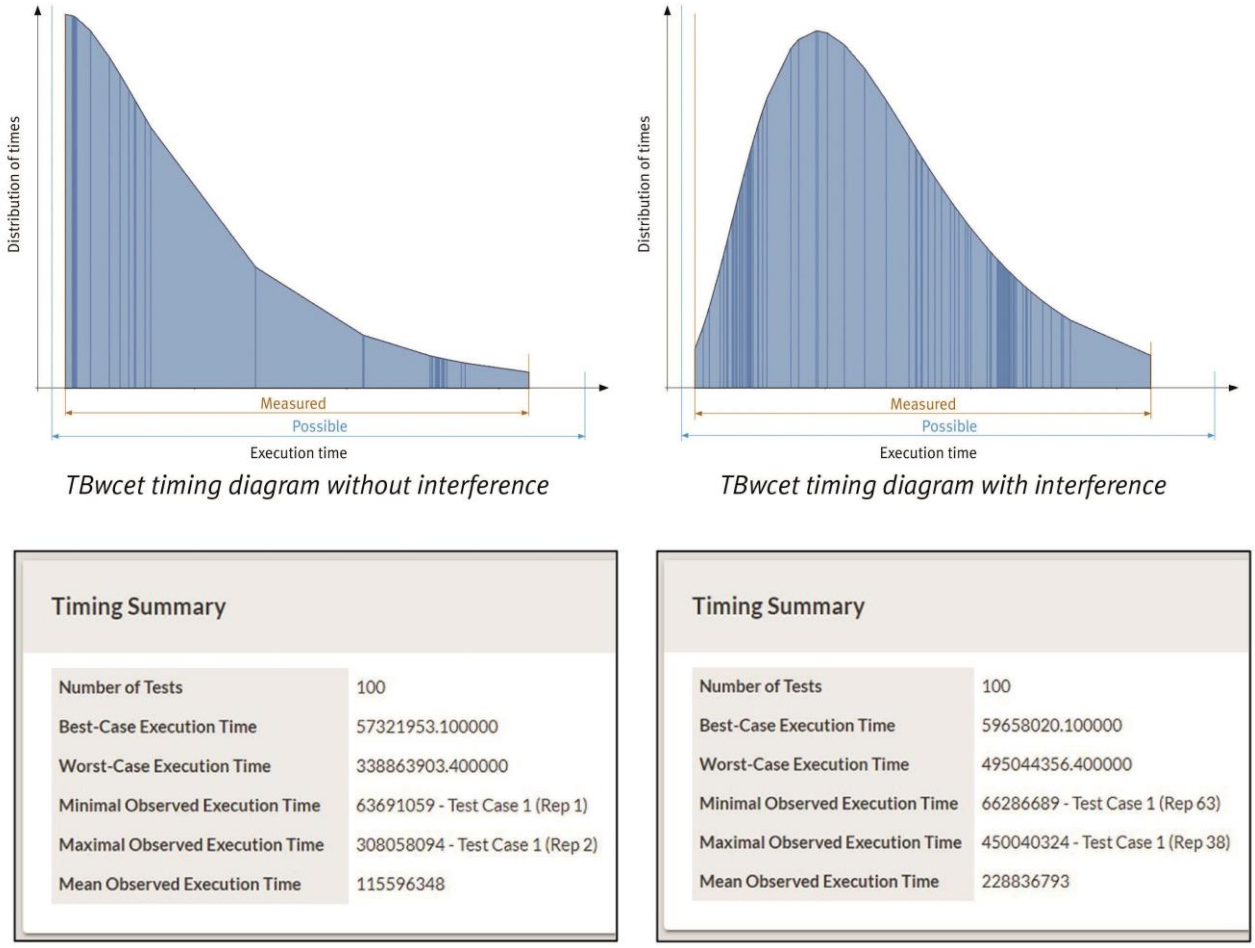


図 7: LDRA ツールスイートによって表示される実行時間の分布を示すヒストグラム。

干渉調査と反復開発

このしっかりとした基盤にもかかわらず、MCP 環境ではシングルコアと比較して、検証と妥当性確認が実証に基づいたものになるため、プロジェクトマネージャーは変化する要件や構成に迅速に適応できる能力を持つことが不可欠です。Dan Iorga 他 [19] は、干渉の測定とチューニングにゆっくりと着実にアプローチすることを推奨しています。このような干渉調査がシステムやソフトウエア要件の変更につながる可能性が高く、逆に、システムの機能要件の変更が、新しい干渉チャネルを推進したり、既存の干渉チャネルに影響を与えたりする可能性が高くなります。 (図 8)。

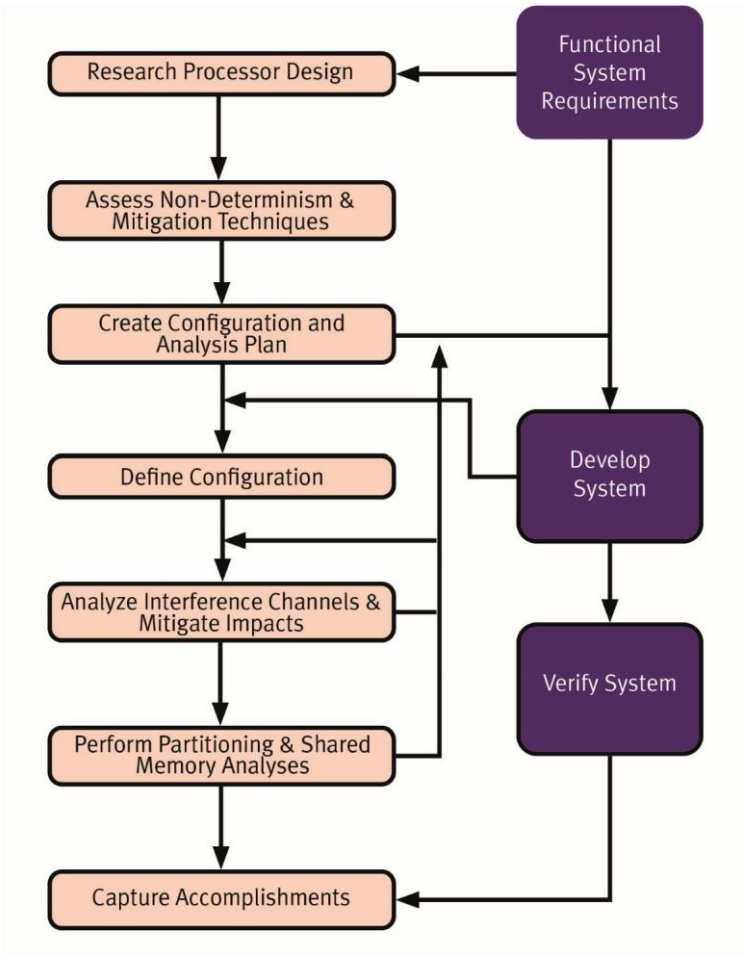


図 8: 推奨されるマルチコア認定アプローチ [19]

このような状況では、検証と妥当性確認を改めて実施するために何を再検討する必要があるかを追跡し、プロジェクトをスケジュールどおり、予算内に収めるために、自動化されたメカニズムが不可欠です(図 9)。

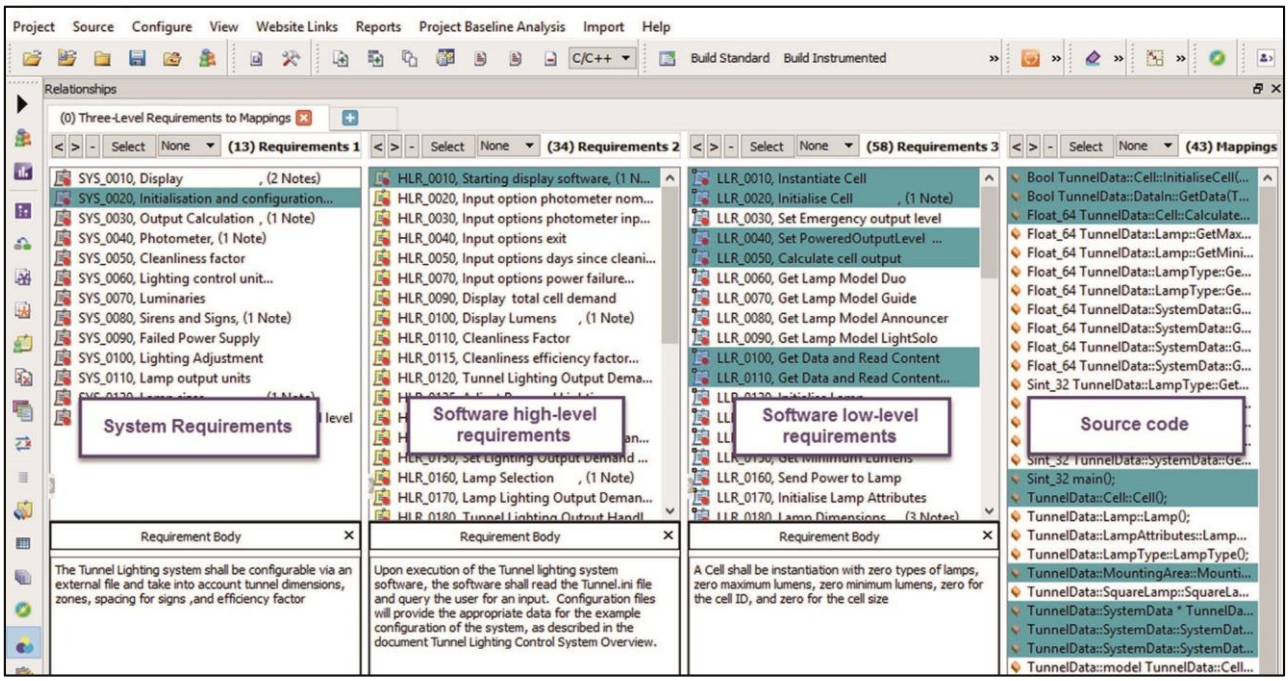


図 9: LDRA ツールスイートの TBmanager [20] を使用した要件と規制目標のトレースの自動化

堅牢な分割

この反復的なアプローチは、第一原理から干渉に対処するメカニズムを提供しますが、堅牢なリソースと時間の分割を提供する RTOS またはハイパーバイザーの使用が役立つ可能性があります。民間航空部門では、CAST-32A/A(M)C 20-193 が次のようにそれを具体的に考慮しています。

「MCP プラットフォームが堅牢なリソースと時間の分割の両方を提供することを確認した申請者は、MCP でアプリケーションを個別に確認し、WCET を個別に決定できます」

とはいうものの、RTOS、分離カーネル、またはハイパーバイザーによって提供される堅牢な分割は、その構成設定に応じた効果が得られるだけです。堅牢な分割により、多くの潜在的な干渉チャネルを排除できますが、干渉の緩和が効果的であることを実証する責任は、開発者に残されています。

クリティカルリアルタイムマルチコアアプリケーション:その他の考慮事項

WCET の他にも、マルチコアプロセッサの使用は、機能安全規格によって推進されている他のいくつかのプラクティスに影響を与えます。

準拠の証明

セーフティクリティカルなシステムの開発では、マルチコアに関する考慮事項は、機能安全関連プロセスの補足事項にすぎないことを覚えておくことが重要です。たとえば、CAST-32A または A(M)C 20-193 に準拠するシステムは、DO-178C の要件も満たしている可能性が高いです。

プロジェクト要件(図 9)と 1 つ以上のプロセス標準およびガイダンス文書の目標の両方に対する双方向のトレーサビリティを自動化することで、プロジェクト管理の主要な問題点に対処するのに役立ちます。

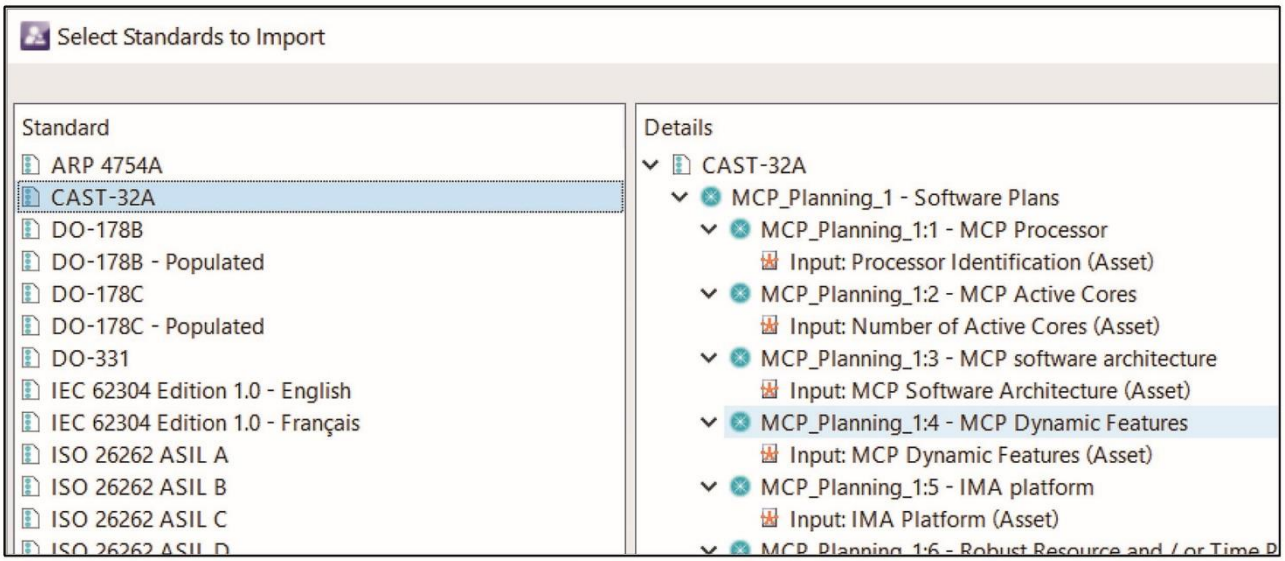


図 10: LDRA のトレーサビリティ機能である TBmanager は、CAST-32A および A(M)C 20-193 の各目標へのトレーサビリティの自動化を支援

コーディング規約への準拠

ほとんどの機能安全規格は、MISRA コンソーシアムによって推進されているようなコーディング規約(または言語サブセット)の使用を奨励しています[21]。これらは、マルチコアプロセッサベースのシステムのコンテキストで重要です。これらの規格によって促進されるルールが、共有リソースを危険にさらす可能性のある実行時の欠陥の挿入を防ぐためです。

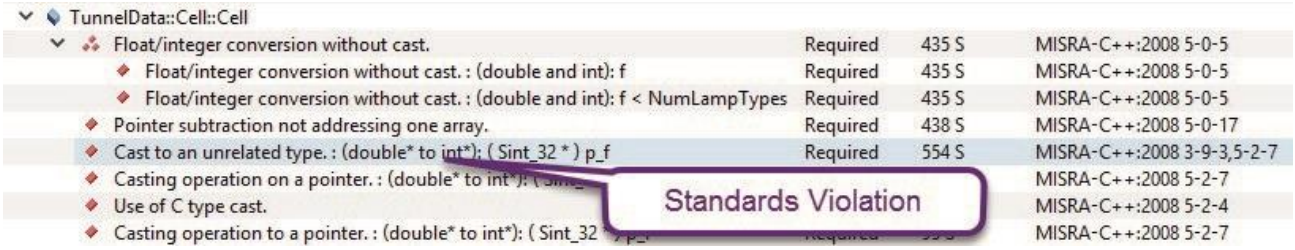


図 11: LDRA の TBvision によるコーディング規約準拠の確認

データ結合解析と制御結合解析

欠陥のある結合などソフトウエアの問題によって実行時間の遅延や変動が引き起され、実行時間の変動が増加する可能性があります。

CAST-32A および A(M)C-193 § MCP_Software_2 は、この点に関して、航空システムに固有のアドバイスで、他のセーフティクリティカルな分野でも同様に有効なものを提供します。それは次のように述べています：

「申請者は、MCP の同一コアまたは異なる複数コアでホストされる個々のソフトウエアコンポーネントすべての間のデータ結合と制御結合が、共有メモリを介したアプリケーション間のインターフェイスと、共有メモリへのアクセスを制御するメカニズムの実行とを含むソフトウエア要件ベースのテスト中に実行され、そのデータ結合と制御結合が正しいことを検証しています」

静的解析ツールは、ソフトウエアコンポーネントとアプリケーション間の明示的なデータと制御フローに関連する制御結合とデータ結合の解析をサポートします。

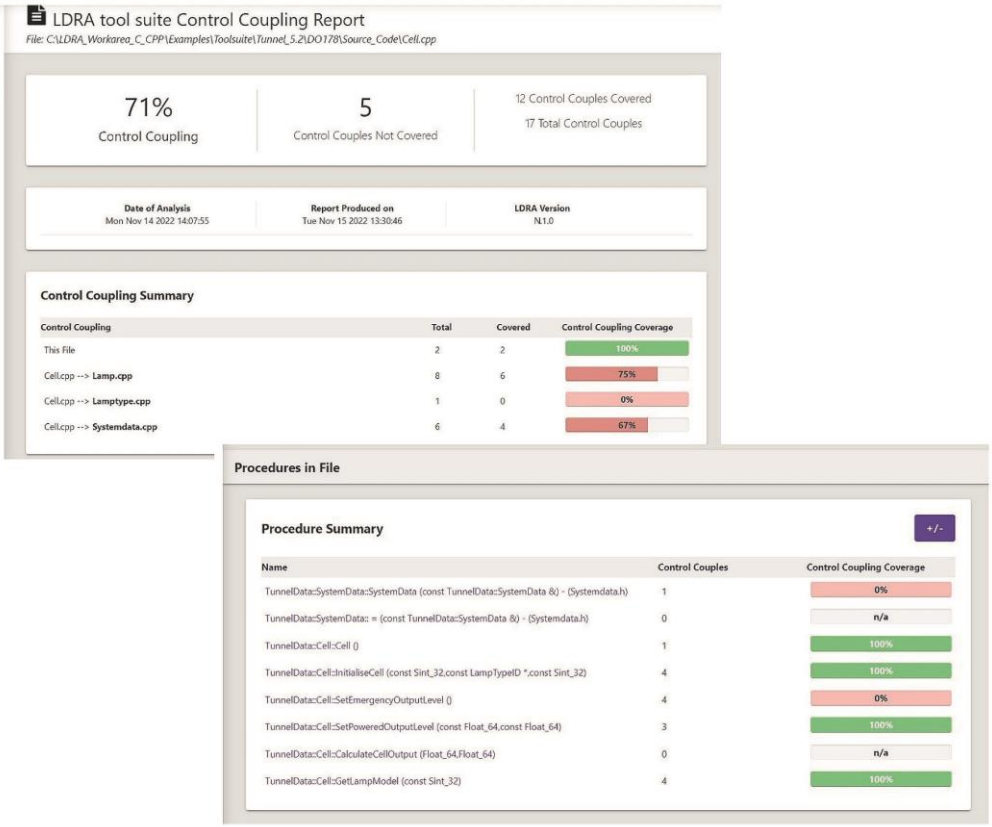


図 12: LDRA ツールを使用したデータ結合と制御結合の解析[22]

ドキュメント内のガイダンスは、プラットフォームレベルでの結合の可能性についても言及しており、前述の WCET 解析手法はそれに関連しています。ほとんどの場合、制御結合はマルチコア環境

では重要な要素ではありません。しかし、データ結合は、コアが通信する必要があるため、重要です。したがって、プロセッサ間でデータが共有される機能を実行するための要件ベースのテストを確立してレビューすることに重点を置く必要があります。

マルチコアプロセッサの構造カバレッジ解析のための効率的なインスツルメンテーション

DO-178 や、IEC 61508、ISO 26262 など機能安全に関する規格では、構造カバレッジ解析が必要です [23]。構造カバレッジデータは、要件ベースのテスト手順で照合されます。ソースコードの「インスツルメントされた」コピーを実行すると、通過したパスが記録され、結果の出力が解析用に提供されます。

ソフトウエアレベルでのカバレッジ解析では、実行を追跡するために、コード内にインスツルメント用のプローブが必要です。これは必然的にパフォーマンスとリソースの両方に影響を与えます。

マルチコアシステムでインスツルメントを有効にするには、その影響を最小限に抑える必要があります。これを達成するための 1 つのアプローチは、各基本ブロックに挿入されたプローブに依存するインスツルメント技法を採用することです (つまり、プローブの数をできるだけ少なくすることを意味します) (図 13)。

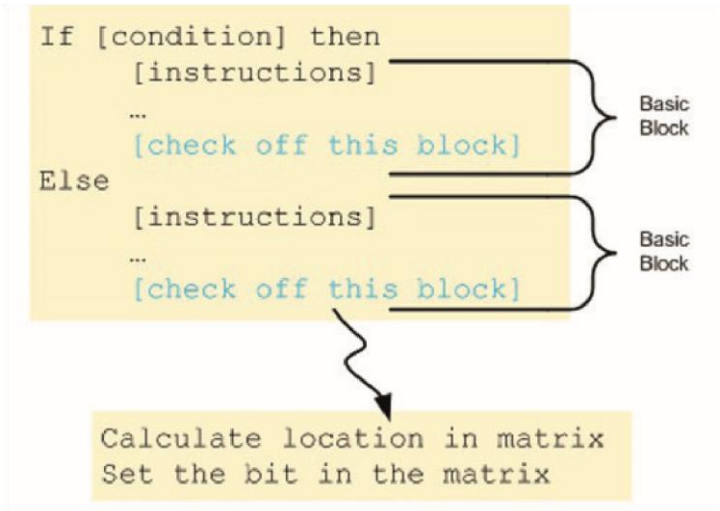


図 13: LDRA のインスツルメントは、基本ブロックにのみ挿入されるプローブを使用します

プローブの数を最小限に抑えることは、カバレッジデータを保存するために必要なメモリにとっても重要です。ビットパックストレージは、プローブ位置に対応するワード内の個々のビットを使用します。その場所は事前に計算されているため、実行時間を節約できます (図 14)。

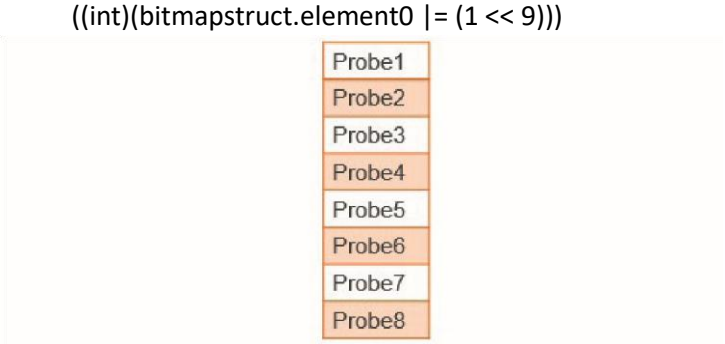


図 14: ビットパックのカバレッジデータストレージ

マルチコアシステムでは、衝突が発生する可能性があります。複数のコアが同じ場所に書き込むと、最初の試行以外はすべて失敗します。ただし、組込みシステムは通常、ループで実行されます。したがって、プローブが繰り返し実行される可能性が高く、衝突のために記録されなかったカバレッジは後続のパスで記録されます。それに失敗しても、報告されるカバレッジは実行されたコードの量を過大評価することは決してないため、フェイルセーフです。

セマフォまたはミューテックスを使用する別のアプローチでは、パフォーマンスのオーバーヘッドが大幅に増加するでしょう。しかし、MC/DC 解析に対しては、アトミックロックによって結果の正確さと完全性を保証するため、セマフォやミューテックスは依然として必要です。

結論

セーフティクリティカルなアプリケーションに対するマルチコアプロセッサ (MCP) の適合性に関する懸念は、今に始まったことではありません。しかし、セーフティクリティカルな分野では進展を見ないことが多く、MCP ベースのクリティカルアプリケーションでシングルコアのみを使用するという状況です。他のアプリケーションの世界では MCP をフルに活用し SWaP（サイズ、重量、パワー）の面で非常に大きな利益を得ているのに対して、これは理想的な状況ではありません。

ただし、これらの環境での MCP には課題があります。対称型マルチプロセッシングオペレーティングシステムの使用は実行可能なオプションではなく、ドメイン間の堅牢なパーティション分割が重要です。ただし、RTOS、分離カーネル、またはハイパーバイザーを介してそれが達成されると、ハードウェアの干渉が別の課題になります。

シングルコアの場合でも、静的解析のみによる最悪実行時間 (WCET) の計算は近似値にすぎません。この近似アプローチを、すべてのコアでその近似値が明らかな MCP の追加の要求と組み合わせることは、明らかに次善のアプローチです。最も要求の厳しい実行パスがどこにあるのかを判断し、それらの実行時間を動的に測定する方がはるかに優れています。

1 つの推奨事項は、反復開発モデルを使用してこの課題を克服することです。このような干渉の可能性を最小限に抑えるために、開発時にシステムのテストと適応を繰り返します。高度に複雑なプロジェクトでは、規格の目標とプロジェクト要件の達成を追跡することは、そのようなプロセスを導入しなくても十分に困難なことがあります。これには、統合され、自動化された要件トレーサビリティシステムは、非常に貴重な支援策になります。

要約すると、セーフティクリティカルな分野での MCP の採用には実際の課題がありますが、適切なツールを念入りに使用すれば克服できないものはありません。

Works cited

[1] M. Kyrnin, “Multiple Core Processors: Is More Always Better?,” 28 July 2020. [Online]. Available: https://www.lifewire.com/multiple-core-processors-832453. [Accessed 1 November 2021].

[2] NVIDIA Corporation, “The Benefits of Multiple CPU Cores in Mobile Devices,” 2010. [Online]. Available: https://www.nvidia.co.uk/content/PDF/tegra_white_papers/Benefits-of-Multicore-CPUs-in-Mobile-Devices_Ver1.2.pdf. [Accessed 1 November 2021].

[3] R. W. e. al., “The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools,” April 2008. [Online]. Available: http://www.es.mdh.se/pdf_publications/1258.pdf. [Accessed 1 November 2021].

[4] C. L. LIU and JAMES W. LAYLAND, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” Journal of the Association for Computing Machinery, vol. 20, no. 1, pp. 46-61, January 1973.

[5] T. Loveless, “Challenges building safe multicore systems,” 15 June 2020. [Online]. Available:

https://www.lynx.com/embedded-systems-learning-center/challenges-building-safemulticore-mcp-software-systems. [Accessed 2 November 2021].

[6] Certification Authorities Software Team (CAST), Position Paper CAST-32A - Multicore processors, CAST, 2016.

[7] RTCC, DO-178C “Software Considerations in Airbourne Systems and Equipment Certification”, RTCA, 2011.

[8] EASA, “AMC 20-193 Use of multi-core processors,” 2022. [Online]. Available: https://www.easa.europa.eu/sites/default/files/dfu/annex_i_to_ed_decision_2022-001-r_amc_20-193_use_of_multi-core_processors_mcps.pdf. [Accessed 18th January 2023].

[9] International Electrotechnical Commission (IEC), IEC 61508:2010 “Functional safety of electrical/electronic/programmable electronic safety-related systems” (all parts), IEC, 2010.

[10] International Organization for Standardization, ISO 26262:2018 “Road vehicles - Functional safety”, International Organization for Standardization, 2018.

[11] European Committee for Electrotechnical Standardization (CENELEC), EN 50128 “Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems”, CENELEC, 2020.

[12] International Electrotechnical Commission, IEC 62304:2006+AMD1 Medical device software - Software life cycle processes, IEC, 2015.

[13] U S Food & Drug Administration, “Classify your medical device,” 2 July 2020. [Online]. Available: https://www.fda.gov/medical-devices/overview-device-regulation/classify-yourmedical-device. [Accessed 30 December 2022].

[14] Halstead, Maurice H., Elements of Software Science., Amsterdam: Elsevier North-Holland, 1977.

[15] LDRA, “LDRA Testbed® and TBvision®,” LDRA, [Online]. Available: https://ldra.com/products/ ldra-testbed-tbvision/. [Accessed 23 March 2022].

[16] LDRA, “TBrun®,” LDRA, [Online]. Available: https://ldra.com/products/tbrun/. [Accessed 23 March 2022].

[17] LDRA, “TBwcet®,” [Online]. Available: https://ldra.com/products/tbwcet/. [Accessed 3rd February 2023].

[18] C. King, “Ubuntu wiki: stress-ng,,” 7th October 3030. [Online]. Available: https://wiki.ubuntu.com/Kernel/Reference/stress-ng. [Accessed 3rd February 2023].

[19] Paul J. Parkinson (Wind River) and Harold G. Tiedeman (Rockwell Collins), “Plan With Confidence: Route to a Successful DO-178C Multi-Core Certification,” 28 September 2018. [Online]. Available: https://www.slideshare.net/ICTperspectives/plan-with-confidence-routeto-a-successful-do178c-multicore-certification. [Accessed 5 November 2021].

[20] LDRA, “TBmanager®,” [Online]. Available: https://ldra.com/products/tbmanager/. [Accessed 3rd February 2023].

[21] The MISRA Consortium Limited, “The MISRA website,” The MISRA Consortium Limited, 2021. [Online]. Available: https://www.misra.org.uk/. [Accessed 13 April 2022].

[22] LDRA, “Control coupling analysis and data coupling analysis for embedded software,” [Online]. Available: https://ldra.com/capabilities/data-couplingcontrol-coupling/. [Accessed 3rd February 2023].

[23] LDRA, “Structural (Code) Coverage Analysis in embedded systems,” [Online]. Available: https://ldra.com/capabilities/code-coverage-analysis/. [Accessed 3rd February 2023].