

HF120 ターボファンエンジン 高信頼制御ソフトウェアの開発

Development of High Reliability Control Software for HF120 Turbofan Engine

手塚 誠*
Makoto TEZUKA
神野 英章*
Hideaki JINNO

杉本 昇平*
Shohei SUGIMOTO
赤井 勇太*
Yuta AKAI

河井 圭助*
Keisuke KAWAI

要 旨

航空機エンジン電子制御システムのアプリケーションソフトウェア開発において、一般的にその開発プロセスは、米国航空無線技術協会から発行されたガイドラインであるDO-178Bに適合する必要がある。また、その中でソフトウェア開発時に実施すべきことは、開発組織が独自に定義する必要があり、弊社は、その独自プロセスによりソフトウェア開発を進めた。

アプリケーションソフトウェア開発は、要件定義、設計、実装の設計フェーズと、レビュー、テスト、解析の検証フェーズに分けて実施された。また、日本、アメリカ、メキシコという時差のある地域間での共同開発においては、本プロジェクトのために構築された開発システムを使用して、信頼性を維持しながら文書や設計データの管理および共有を実現した。

弊社は、本ソフトウェア開発を通じて、DO-178Bに適合した高信頼ソフトウェア開発プロセスを確立し、制御システムソフトウェアを含めたHF120ターボファンエンジンの米国連邦航空局による型式認定をGeneral Electric社と共同で取得することができた。

ABSTRACT

When developing application software for an aircraft engine electronic control system, the development process is required to comply with DO-178B, one of the guidelines issued by the Radio Technical Commission for Aeronautics. All activities to be conducted during software development should be defined by each development organization, and Honda developed the software based on the process defined by Honda.

The application software was developed separately during the design phase consisting of requirement definition, software design and software implementation. The verification phase consisted of review, testing and analysis. In addition, Honda established the development system for the project tailored for the joint development between Japan, USA and Mexico with international time differences. Documents and design data could be shared and managed while maintaining the integrity within the development system for this project.

As a result, Honda established a highly reliable software development process for the HF120 turbofan engine control system in compliance with DO-178B. Jointly with General Electric Company, Honda obtained Type Certificate issued by the Federal Aviation Administration for the HF120 turbofan engine, including the engine control system software.

1. ま え が き

航空機用エンジンは一般的に生涯運用期間が長く、その制御装置にも耐久性が求められる。また、航空機用エンジンは、センサ等の故障が発生したとしても、その状況に応じて適切な制御をおこない、エンジンを作動させ続ける必要がある。弊社は、HF120ターボファンエンジン(HF120)

の制御システムとして、Full Authority Digital Engine Control(FADEC)を採用し、General Electric社(GE)と共同で開発をおこなった。また、HF120制御システムは、運用期間中にも機体メーカーや米国連邦航空局(Federal Aviation Regulations Administration:FAA)からの変更要求に柔軟に対応できるよう、エンジン制御の主要特性をソフトウェアで実装している。

*航空機エンジン R&D センター

また、Fuel Pump Metering Unit(FPMU)や Engine Control Unit(ECU)等のハードウェア、ECUに搭載するオペレーティングシステム(OS)やアプリケーションソフトウェア(AS)が、制御システムの構成部品であり、弊社はASの開発を担当した。さらに、システムインテグレーション、検証およびエンジン適合性の確認もおこなった。

制御ソフトウェア開発は、耐空性を確保するための開発プロセスのデファクトスタンダードであるDO-178B⁽¹⁾を満たすように実施される必要がある。DO-178Bはソフトウェア開発時に実施すべきことを規定するものであるが、実施プロセス自体は各開発組織が独自に定義し、そのプロセスによって作り出されたソフトウェア開発データを客観的証拠として、ソフトウェア開発の適合性が証明される。

本稿は、弊社の構築した航空機用エンジン制御ソフトウェアの高信頼性開発プロセスを紹介するものである。

2. 制御システム概要

2.1. 航空機用エンジン制御

航空機用ターボファンエンジンには精度の高い推力制御が要求される。また、エンジンの加減速時には、圧縮機のサージ、ストール、さらに失火を回避しながら、推力応答性を確保する必要がある。

エンジン制御システムは、機体からの要求推力にエンジン状態や外気条件、機体運転条件等を考慮して、必要な燃料流量を決定し、燃料流量は、Fuel Metering Valve(FMV)の位置により決まる燃料流路面積を変化させることで制御される。なお、航空機用ターボファンエンジンは、空気密度が低下する高高度で使用されるため、高高度における低流量域と地上における高流量域の両方で十分な燃料制御精度が必要である。

2.2. 制御システム構成

HF120 制御システム構成を Fig. 1 に示す。システムの主な構成部品は、ECU、エンジンセンサ、FPMU、それらと航空機をつなぐ機体インタフェース部である。

ECU やセンサ類の電装部品は、高信頼性を実現するために二重系の冗長設計としている。ECU は Cross Channel Data Link(CCDL)によって、両電気系統の健全性を比較し、より健全なシステムを選択してエンジンを制御する。つまり、一系統の電装部品に故障が起きたとしても、即座に故障系統の使用を停止し、残りの系統に切り替えることで、適切なエンジン制御を継続することが可能な構成とした。

FPMU は制御機能を実現するための重要な構成機器であり、主な機能は、燃料を昇圧し、エンジン燃焼器に供給することである。燃料流量制御に使われる FMV やサージとストール回避のための Bleed-off Valve(BOV) Actuator、緊急時燃料遮断のための Shut-off Valve(SOV)は、FPMU に内蔵される。さらに FADEC の作動は、基本的に FPMU に内蔵される Permanent Magnetic Alternator(PMA)による FADEC 専用電源を使用しており、機体電源を必要としない。

推力レバー位置やコックピットスイッチ信号等の推力制御に必要な機体情報は、機体インタフェースを通して ECU に伝えられる。また、エンジン運転状態を示す信号は、機体インタフェースを通して機体に送られる。

3. 高信頼性ソフトウェアの開発概要

3.1. 制御システム要件

航空機用エンジンの制御システムは、米国連邦航空規則を満足する必要がある。ここには、エンジン制御に対する耐空性基準の規定があり、エンジン推力の基本制御性や耐

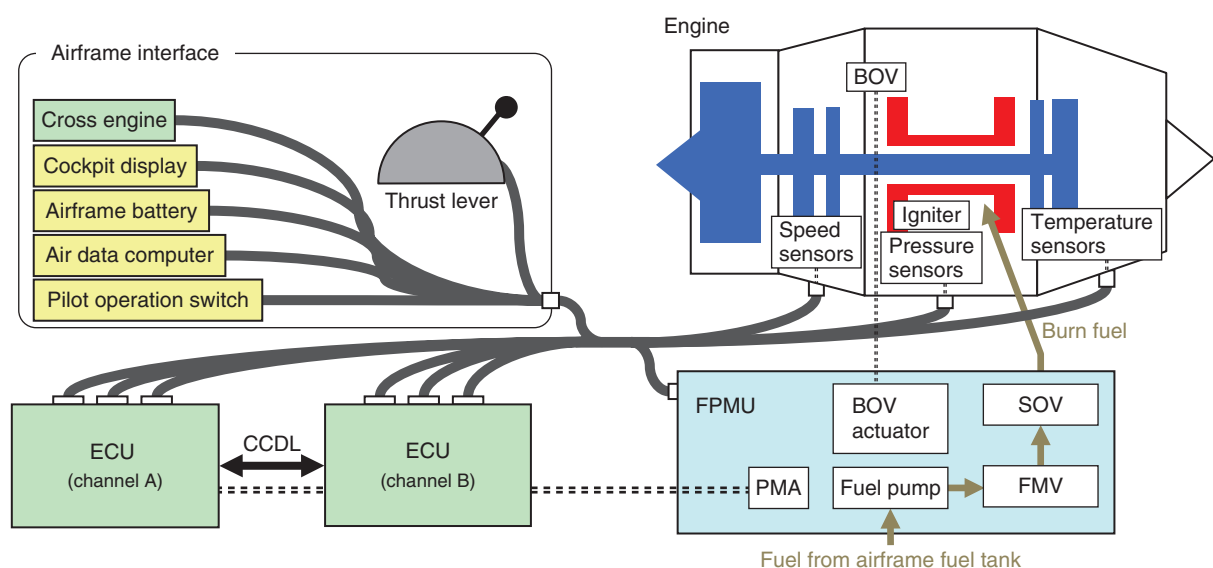


Fig. 1 FADEC system

電磁干渉性、信頼性など、エンジンを安全に運行するための規定が定められている。

機体メーカからは、エンジンシステム要求仕様書や機体エンジンインタフェース仕様書が発行され、これらの上位要件に基づき、エンジンメーカとしてのエンジン仕様書が作成される。これらエンジン仕様書に規定される要件は、制御システム仕様書へフローダウンされ、さらに、ソフトウェア仕様書や詳細設計仕様書へフローダウンされる。

なお、ソフトウェア開発を含めてすべてのコンポーネント開発は、これらの要件に基づきおこなわれる。

3.2. ソフトウェア開発手法

Fig. 2 の V 字開発フローに従い、ソフトウェアの開発を実施した。左側は設計フロー、右側は検証フローを示す。

この V 字プロセスで示される個々の作業結果は、構成管理データベースで管理されている。構成管理はソフトウェアのリリース管理、ソフトウェア構成要素のバージョン管理、構成要素の変更に関する変更要求管理から成り立っており、すべての作業に対して、いつだれが何の目的で何をどう変更したのか、といった作業トレーサビリティを記録することを目的としている。

3.2.1. ソフトウェア設計手法

FADEC 開発は定義されたエンジン仕様を受けて、システム要件を定義するところから始まる。FADEC に求められる機能要求および非機能要求を明確にし、ハードウェアおよびソフトウェアの要件が定義できるよう各種インタフェース仕様書を定義する。

HF120 開発において、顧客の要求はシステム要件の一部として定義され、次にこれらシステム要件からソフトウェア要件を定義した。DO-178B に準拠した検証作業では、このソフトウェア要件に対する 100% のテストカバレッジを示すことが求められる。このため、ソフトウェア要求仕様書作成に対して一般的に求められる作法に忠実に従いソフトウェア要件を作成し、すべての要件が検証可能となるようにした。

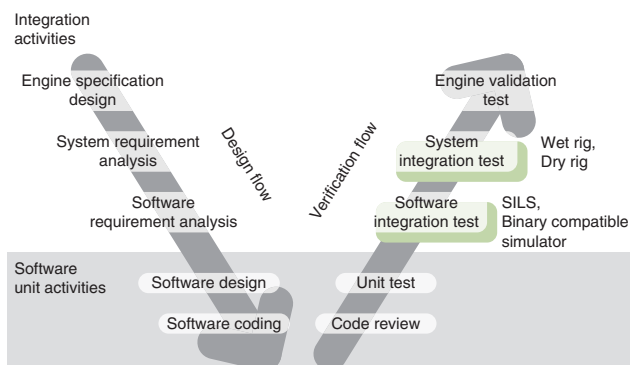


Fig. 2 Software development flow

その後、ソフトウェア要件に基づいて設計、ソフトウェア実装をおこなう。HF120 開発において、ソフトウェアの振る舞い記述に MATLAB/Simulink および Unified Modeling Language を用い、それらのモデルを C 言語で実装した。Simulink で記述されたモデルは、コード自動生成ツールを用いて実装した。

3.2.2. ソフトウェア検証手法

検証作業はソフトウェア開発の大部分を占め、そのやり方自体が制御ソフトウェアの信頼性に影響を与える。検証作業はレビュー、テスト、解析で構成される。Fig. 2 に示したコードレビュー、ユニットテスト、ソフトウェア統合テスト、システム統合テスト、エンジン適合テストが主要な検証作業であり、それ以外に各種解析によって検証作業全体の完全性を確保している。

4. ソフトウェア検証手法および環境

HF120 開発においては、モデル化範囲の異なる四つの統合テスト環境を準備し、各検証作業に最適なテスト環境を選択した。

統合テスト環境は、モデル化範囲によりソフトウェア統合テスト環境とシステム統合テスト環境に分類される。各統合テスト環境のモデル化範囲を Fig. 3 に示す。Software In the Loop Simulation (SILS) とバイナリ互換シミュレータはソフトウェア統合テスト環境に分類でき、ドライリグとウェットリグは、システム統合テスト環境に分類できる。

SILS は、FADEC システムの構成要素すべてをモデル化したテスト環境であり、エンジンプラントモデルとのク

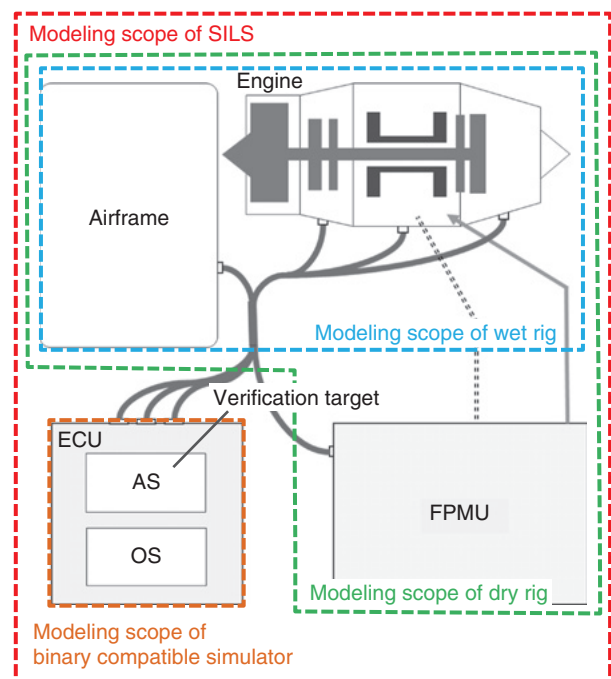


Fig. 3 Modeling scope

ローズドループシミュレーションが可能である。一方、バイナリ互換シミュレータは、エンジンプラントモデルを組み込まず、ECU ハードウェアと OS のみがモデル化されたオープンループのテスト環境として構築されている。

ドライリグは、エンジンプラントモデルの演算結果に基づいて ECU を電氣的に駆動できるシミュレータと ECU を組み合わせたテスト環境で、実 ECU を用いたローズドループシミュレーションが可能である。ウェットリグは、FPMU に実燃料を流してテストできる環境であり、実 ECU と実 FPMU を組み合わせたローズドループでシミュレーションが可能である。

4.1. ソフトウェア統合テスト

4.1.1. SILS テスト

ソフトウェアの評価を正確におこなうために、シミュレータ設備や ECU のハードウェア特性、ノイズ特性を排除した理想状態のエンジン挙動とソフトウェア挙動のシミュレーション環境が必要となる。また、ノイズ注入テストでは、ノイズ特性を精度よくソフトウェアに注入できるシミュレーション環境が必要である。このようなシミュレーション環境に対する要求を満たすために、SILS を利用する検証手法が採用される。

実際の検証過程においては、推力制御、耐ノイズ性に関する検証において SILS が活用された。推力制御に関する検証では、ソフトウェアのコーディングの完了と同時に SILS を構築し、この SILS 上で高度やキャビン抽気条件などのエンジンの使用条件を網羅し、得られたすべての制御応答が設計要件を満たすことを確認した。次に、実 ECU を含めたドライリグ上で同じ条件のテストを実施し、SILS から得られた制御応答との誤差が許容範囲内であることを検証する。このような二段階のステップを採用することで、理想的な環境上でソフトウェアが要件を満足すること、さらに、そのソフトウェアが実 ECU 上でも要件を満足することを実証できた。

耐ノイズ性の検証では、SILS の特徴を活かし、想定されるあらゆるノイズパターンをソフトウェアに対して正確に注入することができ、さらに、テストの再現性も得られるため、ソフトウェアの耐ノイズ特性を正確にかつ効率的に検証することができた。

4.1.2. バイナリ互換シミュレータテスト

ソフトウェアの要件の中には、ハードウェア特性に依存した内容が詳細に記述される場合がある。たとえば通信やストレージなどのアクセスタイミングなどがこれに該当する。このような要件をテストで確認するには、ハードウェアの挙動を時間精度よく制御できるテスト環境が必要となり、AS の検証における下記の目的を満たすためにバイナリ互換シミュレータを準備した。

(1) ECU の内部故障を任意のタイミングで発生させるこ

とができる。

- (2) 任意のタイミングで任意パターンの通信データを ECU に送り込むことができる。
- (3) 任意のタイミングでストレージデバイス内の状況を仮想的に確認する。
- (4) ECU にロードする AS の実行可能ファイルがバイナリレベルの互換性をもって実行でき、AS の処理結果が実 ECU と完全一致する。

本テスト環境は、ECU の CPU シミュレータ上に、AS とそれ以外のソフトウェアでモデル化されたコンポーネントを統合することで実現され、モデル化対象は OS、ECU ハードウェア、エンジンの一部、およびエンジン診断ツールとした。

この環境において、ECU の内部故障を任意のタイミングで発生させることで、要求に対するテストカバレッジを得ている。代表的な活用事例として CCDL の故障注入テストがある。ECU ハードウェアは、単一のきょう体内部に 2 セットの冗長なボードを持ち、これらのボード間で CCDL 通信を用いて制御データを共有しているが、本テスト環境では、この二重冗長化されたハードウェアと CCDL をモデル化しており、CCDL に対して断線やデータ化けを擬似的に発生させることができる。これによって故障の発生パターンを網羅することができ、システムのロバスト性について確認できた。

上記のような検証時の有用性以外にも、AS と OS の同時開発における本環境の有用性も確認されている。HF120 開発では AS と OS のインタフェースを先に定義して、AS の開発とモデル化された OS の開発を同時に実施し、実 AS と実 OS を統合する前に、モデル化された OS で模擬的な統合テストをおこなった。この事前の統合テストをおこなうことによって、実 OS の受け入れテストの際のインタフェース不整合をスムーズに解決できる。具体的には、実 OS とモデル化された OS を突き合わせることでその振る舞いの違いを洗い出し、それが要件の未達事項なのか、それともインタフェース定義の課題なのかなどを即時に判断することができる。

4.2. システム統合テスト

4.2.1. ドライリグテスト

ドライリグでは、OS と AS を ECU ハードウェアにロードし、ECU に対する入出力信号を制御することでテストをおこなう。ソフトウェア統合テストの優位性は、組み込み対象である電子部品上でのソフトウェアの動作結果が得られる点や、ECU に対する電気インタフェースが任意に設定できる点にある。したがって、ドライリグでの断線や短絡のテストは、ECU を構成するハードウェア、OS、AS 間での複雑な影響伝播の検証を可能としている。

ドライリグの構成を Fig. 4(a) に示す。エンジンモデルの

演算結果に基づきシミュレータから ECU へ電気信号が入力される。ECU からの出力信号は、エンジンモデルへの入力となる。

システム統合テストの信頼性を高めるため、ドライリグテストにおいては約 9 割のテストケースについて自動化をおこなった。さらに、HF120 開発における検証の自動化は、シミュレータの自動操作だけでなく、ECU ハードウェアや OS などのバージョン情報取得や統計的解析をするためのデータストレージ、表やグラフを使ったテスト報告書の作成までを対象とした。このような自動化手法を確立したことで、以下のような観点から、ソフトウェアの高信頼化が実現できた。

- (1) 人の手では困難な回数のテストを繰り返し実行することができる。これによって、ECU ハードウェアが許容誤差を逸脱した挙動を示さないかどうかを検証でき、テスト結果をもって、ECU およびその制約を受けるソフトウェアの確からしさを確認した。
- (2) 人の手では設定が困難な短いタイミングでのテストを実行することができる。これによって、現実には発生しづらい複合的な外部条件におけるシステムの振る舞いをテストによって確認した。
- (3) 完全にテストの再現性が保障され得るという特徴を生かし、開発中のソフトウェア変更に対してその変更との関係性が薄い機能に関するテストも実行した。このテスト結果を変更前のソフトウェアから得られたテスト結果と比較し、その誤差を解析することで変更による影響範囲を特定した。このテスト結果からの影響解析は、OS や ECU の変更があった場合にも適用され、統合システムとしての健全性を監視した。
- (4) 自動化と同時に標準化も実施することで、テストの高品質化を達成した。具体的には、要件ベースカバレッジを満足できる入力条件や許容誤差の決定方法、さらに報告書の構成やグラフによる信号の見せ方について標準化した。さらに標準化された工程はライブラリとして用意することで、多数の検証者が標準化された工程を取り込めるよう工夫した。

4.2.2. ウェットリグテスト

FMV 制御や SOV 制御等の FPMU に関連する検証は、FPMU をモデル化した環境である SILS やドライリグで実施されるが、エンジンテスト前の最終的な検証として、実 FPMU とソフトウェアを組み合わせるためにウェットリグ環境を構築した。

ウェットリグは、Fig. 4(b) に示すように、実 FPMU と実 ECU、FPMU に実燃料を供給する燃料設備、燃料ノズルを模擬した絞り弁、テストチャンバ、FPMU の駆動に使うモータ、エンジンモデルを模擬するシミュレータから構成される。さらに、燃料設備とテストチャンバは、航空機環境を模擬するために低温から高温まで燃料の温度調整が可能であり、エンジンモデルシミュレータと合わせて、各種の FPMU 運転条件を模擬できる。

エンジンテスト投入前にウェットリグを用いて FPMU 関連制御の機能検証を実施することで、任意の運転条件において、正確かつ効率的に検証作業を完了できた。

4.3. 解析

4.3.1. 要求ベーステストのカバレッジ解析

AS の信頼性を確保するためには、要求のテストカバレッジが 100% であることを証明しなければならない。この完全性を示すため、二段階でカバレッジの解析を実施した。

まず、すべての要件に対してテストが設定されていることを確認し、さらにすべての要件がテスト可能であることも確認した。

次に、個別の要件に対するテストの完全性を確認した。着目した観点は以下の三項目である。

- (1) 実際に起こりうる複数のシナリオがテストされているかという点を確認する。たとえば、エンジン始動条件では、高空始動などを初期状態とする場合も考慮した。
- (2) ある二つの状態を仮定したときに、各々の状態における静的な挙動と、境界をまたいで状態を遷移するときの動的な挙動についてテストされているかという点を確認する。特に飛行状態を示す高度、マッハ数、外気温度などのパラメータについては厳しくチェックした。

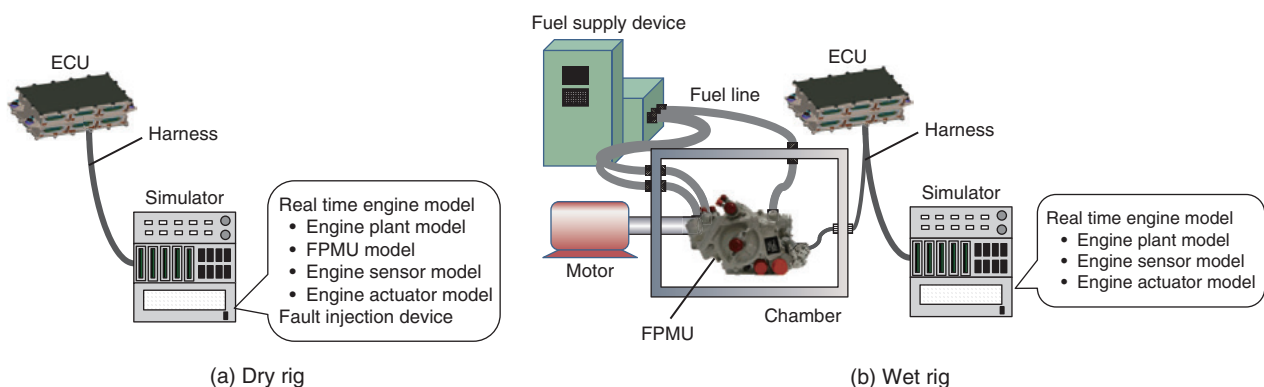


Fig. 4 Dry rig and wet rig

(3) 通常状態に加えて故障状態もテストされているかという点を確認する。これにより、正常なときと同じ動作が得られるか、もしくは故障状態に対応する適切な処理が動作するかなどをチェックした。

解析過程でカバレッジ不足が判明すれば適切にテスト内容を追加する。たとえば、ドライリグテストを採用している要件に対して、ECU ハードウェアの内部故障に対する追加テストが必要と判断された場合、バイナリ互換シミュレータ環境でのテスト項目を追加することで、100% テストカバレッジを確保した。

4.3.2. 実行時間解析

リアルタイム性が要求される制御システムにおいて、タスクが予定時間内に処理を完了できない場合、その制御対象に影響を与える可能性がある。昨今の組み込みシステムに使用される System on Chip (SoC) には、マルチコア、キャッシュ、各種演算コプロセッサ、といった処理高速化機能がハードウェアとして組み込まれており、これらの機能を使うことでシステムの演算性能は飛躍的に向上する。これを踏まえ、タスクが実 ECU 上で取りうる最大実行時間を求めるために下記の解析を実施した。

まず、SoC の内部状態とその命令セットをシミュレートできる環境を準備し、これを用いて AS の実行ファイルを解析した。この解析によって、ソフトウェアに内在する分岐処理において、どちらを通るべきかを机上で判定し、最大の実行時間を要するソフトウェアの実行経路を求めた。また、その実行経路を通るためのタスクに関連するメモリイメージを解析によって導出し、このメモリイメージを実 ECU 上に投入することで、実際のタスク処理時間およびソフトウェア実行経路を取得した。このソフトウェア実行経路が、解析で導いたものと一致することを確認したうえで、計測された時間を最大実行時間として特定した。

HF120 開発では、CPU の負荷マージンが少ない段階でエンジンのメンテナンス機能の実装を開始した。開発中は CPU の負荷マージンを管理するため、本解析を常時実施した。この解析では、後処理によって個別のソフトウェア機能ごとの CPU 占有率が算出しており、必要以上に負荷の高い処理が実装された場合、その情報を可視化しチームとして共有することができた。CPU 負荷低減に関して、チームで対応案を出し合い負荷を下げるための設計努力を積み重ねた。結果、任意のイベントが任意のタイミングで起きたとしても個々のタスクは所定の時間内に完了できるソフトウェアに仕上げることができた。

5. ソフトウェア構成管理

HF120 制御ソフトウェアの開発ライフサイクルは、顧客要求をその代表とするシステム変更要求によって開始される。このシステム変更要求を、要件変更およびソフトウェ

ア設計変更にブレイクダウンし、それぞれの変更の経過を管理する。ソフトウェアのリリースは、システム変更要求の集合としてとらえ、そのリリースに取り込まれた変更点、およびそのリリースを構成するソフトウェアの設計、検証における各種成果物のバージョンを特定できるようにしている。GE および弊社が Configuration Control Board を設置し、一定の基準を設けてシステム変更要求レベルでの実施可否判断をおこなっている。

Fig. 5 にソフトウェア構成管理の仕組みを示し、以下に、これを支える個々の要素について説明する。

5.1. 要件の構成管理

AS の開発に使用する要求仕様書はすべてドキュメント管理ツールを用いて作成した。要求仕様の作成過程において日々変化する仕様書間のリンクを常に最新に保つには、ドキュメント管理ツールの導入が不可欠と考えた。これによってある要求仕様の定義を変更する際に、そのリンクをたどることで他の要求仕様書内の影響範囲を把握できるようになっている。また、要求の文面およびその存在理由の特定以外にも、その要求をどのようにして検証するのかについて要求仕様書のデータベース内に定義した。これによって要求仕様作成段階で設計担当者と検証担当者が、検証可能性の合意を得ることができる。またドキュメント管理ツールは ECU サプライヤでも使用しており、取り交わしたインタフェース仕様を満たすように設計されていることが、客観的に追跡できるようになっている。

5.2. ソフトウェアライフサイクルデータの構成管理

ソフトウェア設計変更要求に基づいて、ソフトウェア設計をおこない、その結果をバージョン管理データベースに登録する。その際、どの変更要求がどのソフトウェアモジュールをどう変更したのかといった変更のトレーサビリティ情報を作成する。これによって、あるリリースを指定すれば、以前のリリースに対して、どのような目的でどのような変更が入ったのかが一目瞭然にわかるようになる。また、構成管理情報にはシステムレベルの変更記述、ソフトウェアの詳細変更記述、ソフトウェアの実装レベルの変化点等が連携して管理されているため、見る人の欲しいレベルの情報を抽出することができる。これによってソフトウェア開発担当者全員が構成管理情報を活用でき関連する設計履歴を把握できるようにしている。

5.3. 構成管理データベースの共有

構成管理情報が最新の開発対象の状態を示すことから、その情報を開発組織間で遅滞なく共有する必要がある。AS の開発では、日本、アメリカ、メキシコにまたがる関係各社の間で、定時データベース同期をおこなうことでこれを実現している。具体的事例として、日本で開発したソフトウェ

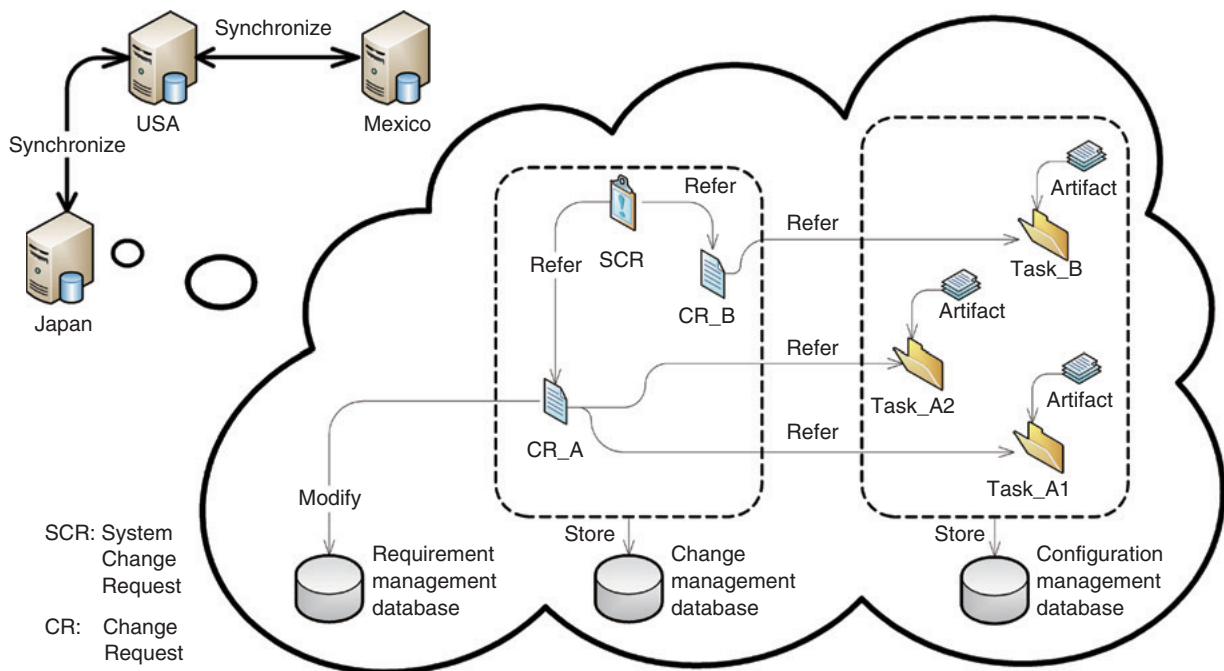


Fig. 5 Software development workflow

アのコードレビューやユニットテストをメキシコで実施したが、検証に伴うソフトウェアの修正サイクルを、この構成管理データベースの共有によって確実なものとした。

この分散型データベースシステムは、あるサイトにおいてデータベースが故障した際にも、他のサイトのデータからデータを復旧できるような構成をとっている。また、各サイトに災害復旧プランの策定を義務付け、その一部としてオフサイトバックアップを要求しており、個々のサイトの可用性およびサイト間の補完的なバックアップを実現することにより総合的な信頼性を確保している。

6. まとめ

本稿にて、HF120 制御システムの中核を成す制御ソフトウェアの信頼性を高めるためのポイントについて説明した。

AS 開発において、抜け漏れなく要件のフローダウンをおこない、設定した要件に対する検証のカバレッジを 100% とすることで、高信頼ソフトウェア開発をおこなった。

ソフトウェアのみに着目したソフトウェア統合テストや、ハードウェアも含めたシステム統合テストを、目的に応じて実施し、さらに、解析によりテストを補完することで、検証のカバレッジおよび信頼性を高めることができた。

また、3 ヶ国にまたがる共同開発の中でも、ソフトウェアの構成を厳密に管理し信頼性を維持できる開発システムを構築した。

HF120 制御ソフトウェアの開発において、以上のように要件設定や検証手法、管理方法に独自のアイデアを加える

ことで、DO-178B に適合した信頼性の高いソフトウェア開発プロセスを確立することができた。

参考文献

- (1) DO-178B, Software Consideration in Airborne Systems and Equipment Certification, RTCA, Incorporated. (1992)

■著 者■



手塚 誠



杉本昇平



河井圭助



神野英章



赤井勇太