改善活動

車載ソフトウェアにおけるソフトウェア仕様書の改善

姫路事業所　技術第1部　技術第1課
岩元 慶介

1．まえがき

当所は電動パワーステアリング(EPS)制御用ソフトウェアの開発を担当している。従来のEPSのソフトウェア開発は小規模開発であった。しかし、近年は機能安全規格(注1)として制定されたISO26262(注2)への対応や自動運転などの付加価値機能の導入により、ソフトウェア規模が年々増加している。

当所はソフトウェア規模の増加に対応するため、ソフトウェア製作を協力会社に委託している。委託先へはソフトウェア仕様書を提示し、Q&Aにより内容を合意している。しかし、近年に変更したソフトウェア構造に関わる部分は、内容の合意までに時間を要した。その結果、開発計画に遅れが生じ、対策が必要となった。

本稿では、ソフトウェア構造の明確化を目的として実施したソフトウェア仕様書の改善を紹介する。

（注1）システムの機能不全により引き起こされるリスクを、許容できる状態に移行するための手法や管理方法を定めたもの
（注2）自動車分野における機能安全の国際規格として、2011年11月に発行。

2．ソフトウェア開発プロセスと課題

2.1　ソフトウェア開発プロセス

EPS制御用ソフトウェアの開発プロセスをV字プロセスで表すと図1となる。

(1)当所が対応するプロセス

当所は、V字プロセスの左側では顧客からの要求仕様に基づき要求分析を実施し、分析結果を踏まえてソフトウェア仕様書とソフトウェアテスト仕様書を作成し、作成したソフトウェア仕様書を委託先へ提示する。V字プロセスの右側では委託先から納入されたソースコードに対して、ソフトウェアテスト仕様書に基づくソフトウェアテストを実施する。

(2)委託先が対応するプロセス

委託先は、提示されたソフトウェア仕様書を確認し、不明点がある場合は、Q&Aを繰り返し、仕様を明確にした後に、詳細設計書とソフトウェア統合テスト仕様書を作成する。次に、詳細設計書に基づきコーディングと単体テストを実施し、ソフトウェア統合テスト仕様書に基づきソフトウェア統合テストを実施する。委託先は詳細設計書、ソースコード及びソフトウェア統合テスト仕様書・成績書を当所へ納入する。

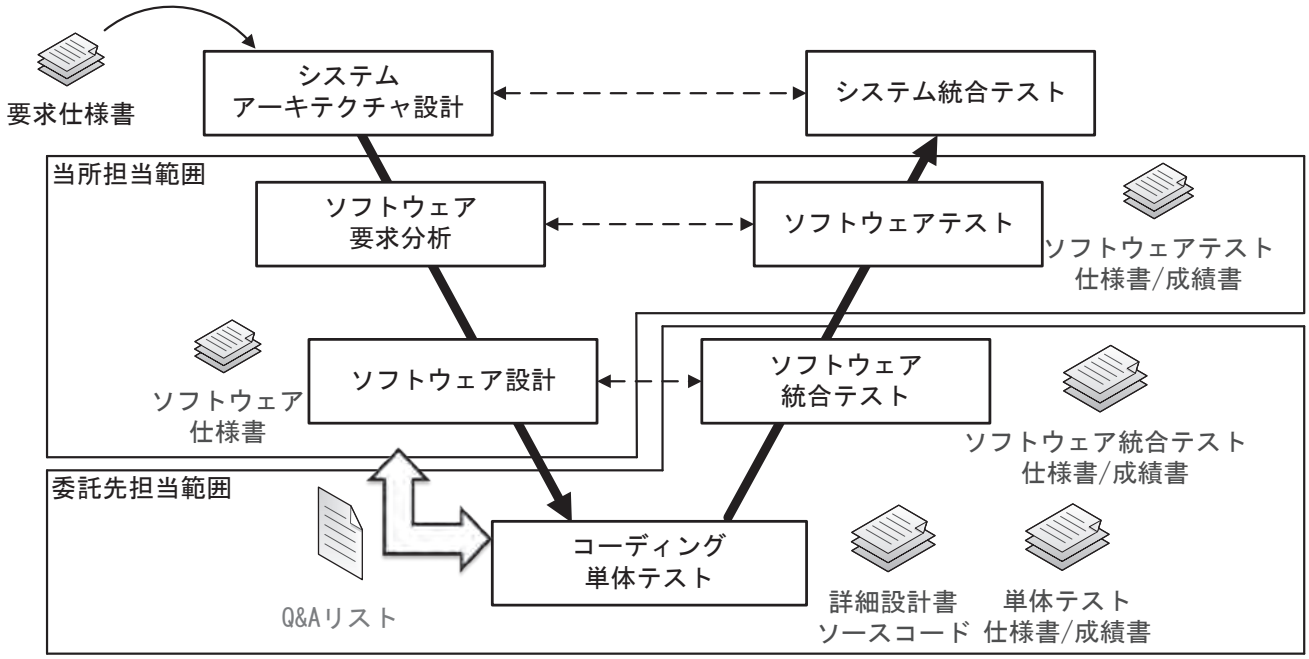

図1．EPS制御用ソフトウェアの開発プロセス

2.2　ソフトウェア開発の課題

開発計画を遅延させる大きな要因として、Q&Aの多発がある。この発生原因を調査すべく、「Q&Aリスト」の分析を行った。結果を図2に示す。大別すると、原因はソフトウェア仕様書の記載不足であり、ソフトウェア仕様書の記載不足箇所は、ソフトウェア構造に関わる部分が多くを占めた。また、ソフトウェア構造に関わる部分の中でも近年に変更した部分が多いことがわかった。

上記のことから、近年に変更したソフトウェア構造に着目して仕様書を見直し、Q&A件数を30%削減することを目標とした。

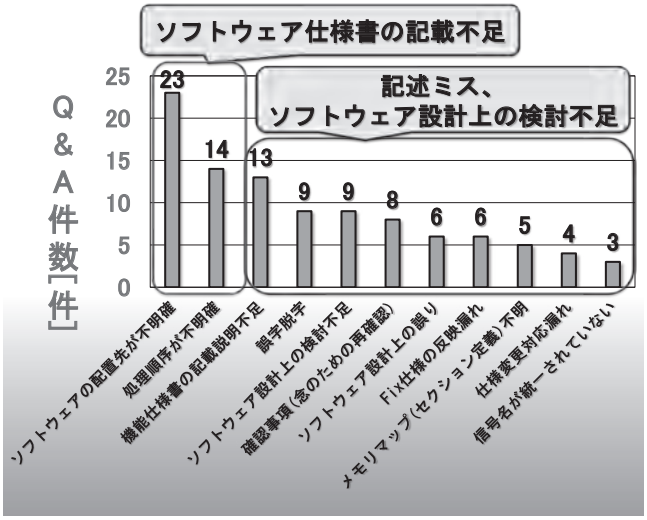

図2．Q&Aリストの分析結果

3．ソフトウェア仕様書の改善

近年、EPS制御用ソフトウェアは、安全規格対応のためにAUTOSAR(Automotive Open System Architecture)のアーキテクチャが導入されている。

AUTOSARのアーキテクチャでは、ASIL(Automotive Safety Integrity Level)と制御周期の依存関係を理解し、ソフトウェアを変更する必要があるが、ソフトウェア仕様書に明記できていなかったことで、委託先からの不明点に関する問合せ、確認が多く発生していた。そこで、ソフトウェアの静的な視点から配置図を、動的な視点からシーケンス図の導入を検討した。

3.1　ソフトウェア構造の変更点

(1)機能安全のASIL割り当てと干渉回避

機能安全とは、ISO26262 Part1において、“電気電子(E/E)システムの機能不全のふるまいにより引き起こされるハザードが原因となる、不合理なリスクの不在"と定義され、システムが故障した際に“リスク"を“許容できる状態"に移行することを求めている。

リスクの決定は、表1に示すとおり危害度S、発生頻度E、制御難易度Cの3つの指標がある。危害度Sは低い方からS1・S2・S3の3段階で表し、同様に発生頻度Eは4段階、制御難易度Cは3段階で表す。危害度S、発生頻度E、制御難易度Cの組み合わせからQM(Quality Management)とASILに大別され、ASILは低レベルからASIL-A・ASIL-B・ASIL-C・ASIL-Dの4段階に分類される。

EPSは、ハンドル操作に係るシステムのため、故障が発生した際、人命に関わる恐れがあり危害度は最も重いS3。自動車を運転中は常に使用されるので発生頻度が高くなりE4。故障が発生すると危害を回避することが困難なので制御難易度は回避困難なC3となり、最も厳しいASIL-Dとなる。

しかし、故障診断ツールとの通信機能などハンドル操作に関わらない付属処理は、危害が小さくS1、使用頻度が低くE1、危害の回避が容易C1であることから、最も低いQMになる。

このように複数種類のリスクが共存したシステムで機能安全を満足するために、ASIL-D機能が他のASIL機能から影響を受けないように、ASIL間の干渉を回避する必要がある。

表1．リスクの決定

低 → 高

危害度	発生頻度	制御難易度 C1 容易に回避可能	C2 通常は回避可能	C3 回避困難
S1 軽度	E1 極低	QM	QM	QM
	E2 低	QM	QM	QM
	E3 中	QM	QM	ASIL-A
	E4 高	QM	ASIL-A	ASIL-B
S2 重度	E1 極低	QM	QM	QM
	E2 低	QM	QM	ASIL-A
	E3 中	QM	ASIL-A	ASIL-B
	E4 高	ASIL-A	ASIL-B	ASIL-C
S3 生命を脅かす	E1 極低	QM	QM	ASIL-A
	E2 低	QM	ASIL-A	ASIL-B
	E3 中	ASIL-A	ASIL-B	ASIL-C
	E4 高	ASIL-B	ASIL-C	ASIL-D

低 ↓ 高

QM：Quality Management
ASIL：Automotive Safety Integrity Level

(2)AUTOSARの導入によるソフトウェア構造の変更

AUTOSARとはソフトウェアモジュールの共通化を図り、再利用性の向上を目的とした規格である。ASIL間の干渉回避には、MPU(注3)を用いた不正アクセス防止機能が有効であるため、AUTOSARが提供するMPUの設定モジュールが使用されている。

ソフトウェア構造の変更として、従来は、ROM、RAM及びStackを分割せず共通で使用していたが、AUTOSAR導入後は、ASIL間の干渉を回避するために、ROMとRAMはASILに応じて、StackはASILと制御周期に応じて分割した。これにより、ASIL間の干渉回避を実現した。図3にイメージ図を示す。

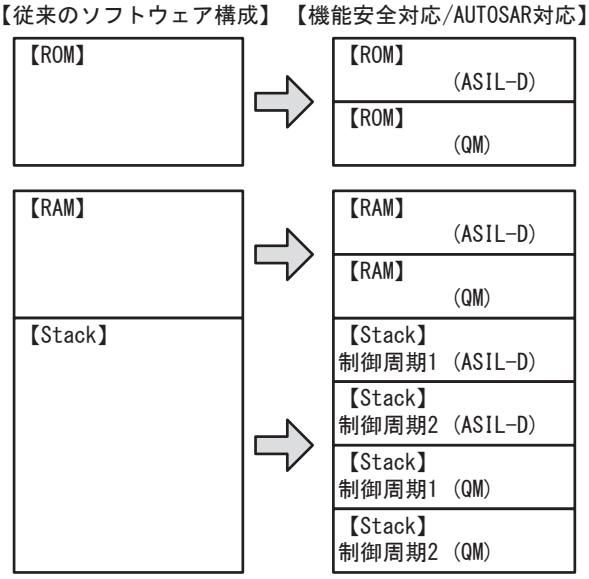

図3．ソフトウェアの構造変更内容

3.2 ソフトウェア仕様書の改善

(1)機能安全に適応した配置図の導入

従来は図4に示すDFD(DataFlowDiagram)を使用し、左から右へ処理が流れるように機能の構成要素を配置し、データの流れを矢印で示すことで各構成要素間の依存関係を記載していた。導入した配置図(図5参照)はASILと制御周期の関係を意識して、大項目をASIL、中項目を制御周期、小項目として各構成要素を機種間の共通・非共通で区分配置することにより、ASILと制御周期の依存関係を整理し、ソフトウェアの構成要素間を矢印で結ぶことで依存関係を明確化した。

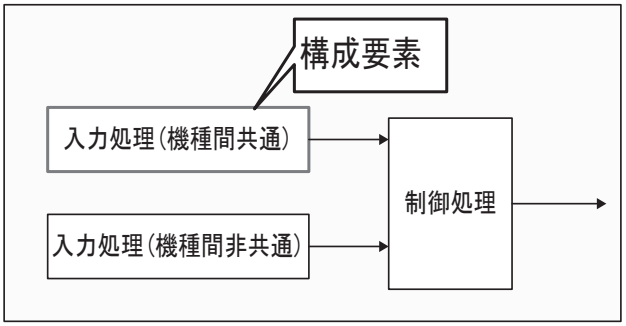

図4．DFD

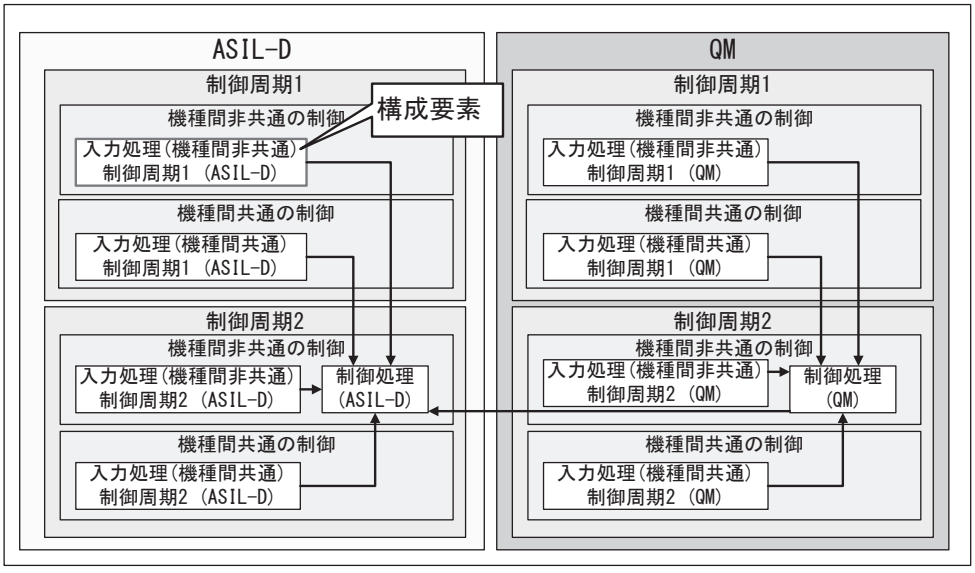

図5．配置図

(注3) マイコンが提供するメモリプロテクション機能。
指定したメモリ領域に対して、不正なアクセスを保護する。

(2) シーケンス図の導入

従来は図6に示すコールツリーを使用しており、最上位を各制御周期のメイン関数とし、上から下に処理が流れるようにサブ関数を記載して、関数単位のコール順序が把握できるようにしていた。そのため、複数の制御周期で実現される機能の場合は、複数のコールツリーから処理順序を確認する必要があり、機能単位の動作を把握することが難しかった。

そこで動的な視点として、機能の実現に必要な各構成要素の相互作用を時系列で表現できるシーケンス図を追加した。作成したシーケンス図を図7に示す。実現する機能をアクターとして設定し、各構成要素を横方向にならべ、縦方向に上から下へ実行過程を示すことで、機能単位での処理順序を明確化することにした。

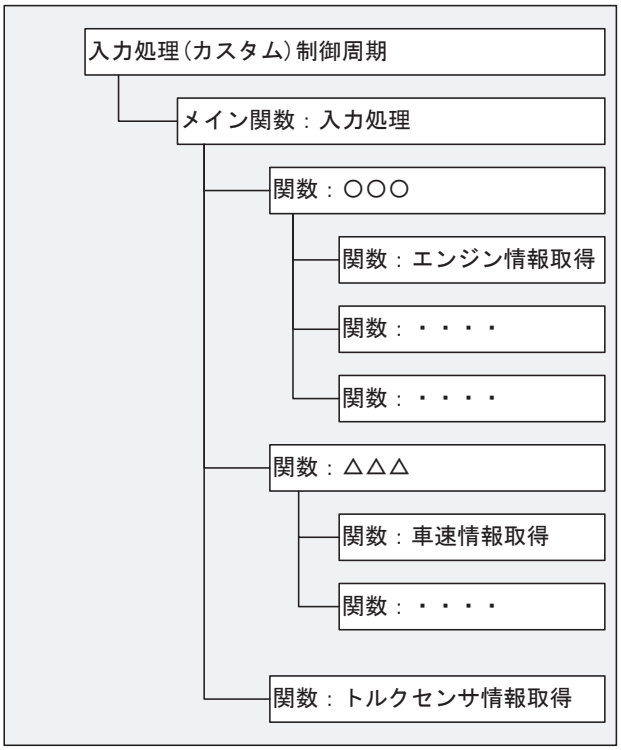

図6．コールツリー

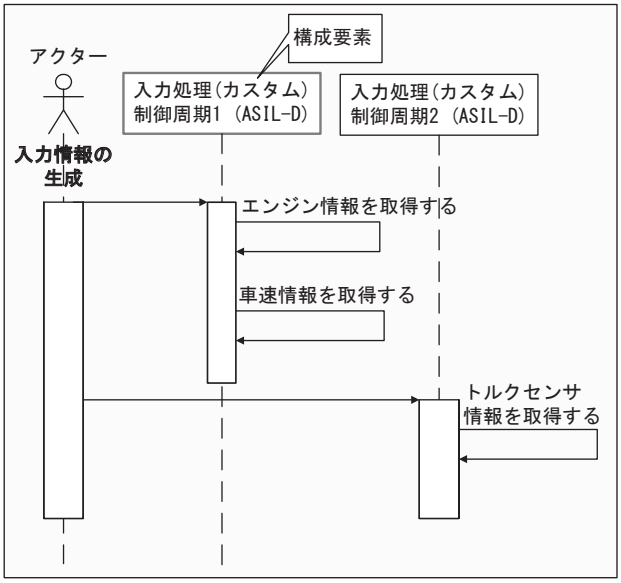

図7．シーケンス図

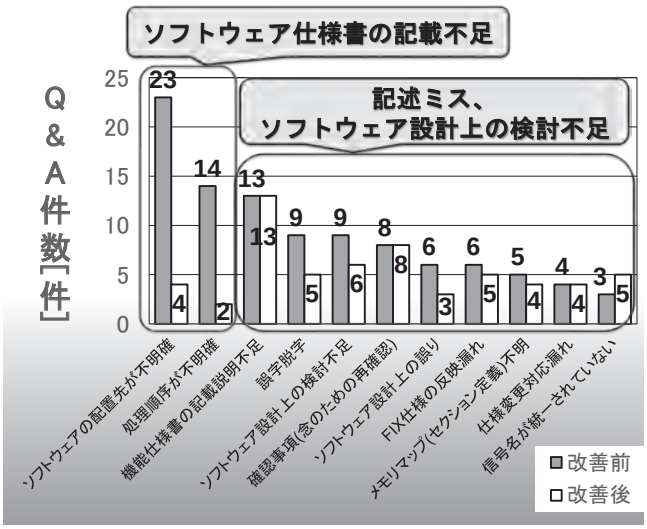

図8．Q&Aリスト分析結果（改善前と改善後の比較）

表2．ソフトウェア変更規模あたりの改善効果

項目	改善前	改善後
ソフトウェア開発ライン数 [Kline]	44	68
Q&A件数 [件]	100	59
Q&A件数 [件/Kline]	2.27	0.87
削減率 [%]	-	61

4．改善効果

4.1 Q&A件数の削減

ソフトウェア仕様書改善による効果を、同程度の規模のソフトウェア開発で比較した結果を図8に示す。ソフトウェア仕様書の記載不足について、改善前の37件(23+14)に対し、改善後は6件(4+2)となり、Q&A件数を約60%削減した。表2に示すとおり、目標の30%を上回る改善となった。

4.2 副次効果:影響分析の精度向上

機能の追加や変更を行う場合、ソフトウェアの影響分析には、従来DFDやコールツリーを用いていたが、機能が複雑な場合は、影響分析に漏れが発生し、その影響で後の工程でも手戻りが発生していた。改善後は作成した配置図とシーケンス図で構成要素の関連と処理順序を明確にしたた

め、影響分析の精度が向上した。

4.3　効果査定と次年度の工数改善見込み

プロジェクト全体での効果査定を行うために、同程度の規模のソフトウェア開発で比較した結果を表3に示す。ソフトウェア設計工数は、ソフトウェア仕様書に配置図とシーケンス図を追加したことで工数が増加するが、委託先とのQ&Aの工数が削減できた。また、ソフトウェアテストでは4.2項で述べた影響分析の精度向上により、作業工数が改善され、ソフトウェアテスト工数削減の効果を得た。

次年度で本改善を適用するプロジェクトの合計生産量が100Kline見込まれるので、次年度では年間327hrの工数削減が可能となる。

表３．ソフトウェア1Klineあたりの改善効果

項目	改善前 [hr]	改善後 [hr]	効果 [hr]
ソフトウェア設計工数	28.68	30.37	+1.69
Q&A工数	4.55	1.74	−2.81
ソフトウェアテスト工数	29.80	27.65	−2.15
合計	63.03	59.76	−3.27

５．むすび

本稿で紹介した配置図及びシーケンス図の有効性を確認できたため、今後はソフトウェア仕様書作成時に配置図とシーケンス図を必ず作成するように、チェックシートへ追加するなど、抜け漏れを防ぎ定着させていく所存である。

最後に、本活動に当たり多くの助力をいただいた関係者各位に深く感謝申し上げる。

執筆者紹介

岩元　慶介　　イワモト　ケイスケ
2007年入社。主に車載システムのソフトウェア開発に従事。現在、姫路事業所技術第1部技術第1課。