ZIPCのカーステレオ開発への導入

日本ビクター株式会社
カーエレクトロニクス事業部　技術部　コア開発グループ　主席技師
栗原　功

１．はじめに

これまで、当社のカーステレオのソフトウェア開発は社内で要求仕様を作成して実際のソフトウェア設計、作成は外部ソフトハウスが行うというスタイルが大半で社内設計は一部だけでした。また、対象となる市場はアフターマーケットが中心でしたが、急速に縮小しており、当社としても車メーカーへのセット納入に対応出来る開発体制に転換していく必要に迫られております。

今回開発した商品は米国のアフターマーケット向けのHD radioレシーバですが、車メーカーからの要求に対し迅速に対応する為に、ソフトウェアの社内開発化を図りました。これまでセット全体のソフトウェア開発を行ったことのないメンバーで開発を行わなければならなかった為、ソフトウェア全体の内容を把握し易いという点からZIPCを導入することにいたしました。

また、車メーカーから求められる信頼性の高いソフトウェア設計を目指すには、設計段階での抜け、漏れの作り込みを防ぎ、検証段階で不具合が発覚した場合でも解析を行い易く、修正に伴って派生する不具合を抑えることが出来る開発ツールとしてZIPCが適していると判断して採用いたしました。

実は前年度もZIPCを使用して同様のモデル開発を行ったのですが、諸般の事情により途中段階で開発を中断いたしました。今回の開発では、開始段階からその反省点を踏まえた上で進めることにいたしました。

以降、その中でも主な重要反省点「開発プロセスにおける手順の踏襲」を中心に御紹介いたします。

２．開発プロセスにおける手順の踏襲

これまでの当部門のソフト開発は、基本設計の古い既存ソースコードに対して、新商品としての要求仕様の変更、追加を行ってソフトウェアを成立させていました。しかし、標準プロセスとして設計段階で行うべき作業、手順、内容の定義、実施が徹底されていなかった為、設計書無しでソースコードを直接編集することもありました。前回の開発でも、単にZIPCを使用するだけでソフトの内容が理解し易くなり信頼性も向上するという思い込みもあって、これまでと同様に十分な設計検討無しでいきなり仕様の変更、追加部分をSTM（状態遷移表）で記述していました。

CATS殿には前機種の開発当初より開発に密着したサポートをいただいておりましたが、開発中断後にまとめていただいた指摘項目として開発プロセスの理解を深めることを第一に挙げられました。

一方で、車メーカーからもCMM、CMMI、Automotive SPICE等に基づいて、設計プロセスを重視してソフトウェアの設計品質を改善することを強く求められています。今回のソフトウェア開発ではこの開発プロセスの改善を重要視して進めることにいたしました。

２．１　ZIPCに適した開発プロセス

ZIPCを使用した開発プロセスの中で（図１）、設計段階ではSTMを段階的に作成し、各段階でシミュレーションを繰り返すことにより下位への不具合の持ち込みを防ぐ仕組みとなっています。このプロセスで最終段階に作成したSTMを使用して自動コード生成を実施します。今回の開発では　仕様STM → 基本STM → 詳細STMの三段階としました。設計に適用する段階の数や各段階での内容についてはCATS殿のアドバイスを受けて設定を行いました。

２．２　日程計画と実績

今回の開発は着手からROMリリースまでの期間が約8ヶ月間であり、初めて量産セットの開発にZIPCを全面的に導入するには短すぎると判断し、ZIPCでの設計の適用範囲を導入効果が最も高いと思われるキー入力等の各イベント発生か

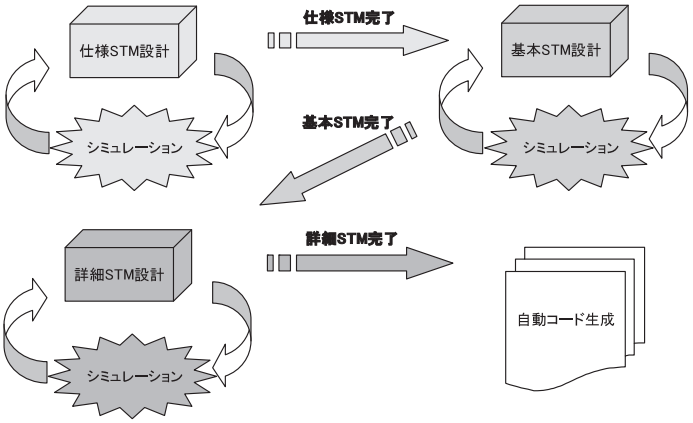

図１　ZIPCを使用した開発プロセス

ら各種SourceでのMode分岐までに止めることにしました。また、前回の反省点の１つとして挙げられた開発ステップ初期のソフトウェア構造の構想段階（基本STM）に十分な時間をかけることにしました。表１が今回の日程計画とその実績ですが、基本STMの検討については3段階に分けてその内容の吟味を行っています。

２．２．１　1stROMリリースの遅れ

日程表の中で1stROMリリースから実績結果に大きな遅れが生じています。1stROMはハード確認が目的であり、下位層の機能が実装されていれば良いのですが、ZIPCの開発プロセスでは上位層から作り込まなければならない為、全体の設計が終了するまでの時間をハードの開発日程に合わせる事が出来ませんでした。

今後の開発では、ソフトウェア設計の着手をハードウェア設計に対して十分先行させて行う必要があります。また、ZIPCのVIP機能を使ってハードの開発日程とは切り離した形でソフトの作り込みを進めることが出来る体制を構築していきます。

開発目標	日程計画	実績結果	日程差分
ZIPC 仕様 STM 開始	4 月 3 日	4 月 3 日	
ZIPC 基本 STM 1st 開始	4 月 17 日	4 月 17 日	
ZIPC 基本 STM 2nd 開始	5 月 11 日	5 月 11 日	
ZIPC 基本 STM 3rd 開始	5 月 17 日	5 月 17 日	
ZIPC 詳細 STM 開始	5 月 24 日	5 月 24 日	
1st ROM リリース	7 月 13 日	**8月9日**	27 日
ES 1st リリース	7 月 30 日	**8月20日**	21 日
ES 2nd リリース	8 月 20 日	**9月14日**	26 日
全機能組込み完了	9 月 6 日	**11月2日**	56 日
PP ROM リリース	9 月 12 日	**10月15日**	33 日
最終 ROM リリース	11 月 30 日	**12月5日**	5 日

表１　開発の計画日程と実績

２．２．２　全機能組込み完了の遅れ

また、全機能組込み完了の日程でも遅れが大きくなっていますが、これは特定のメンバーに業務が集中してしまい設計業務が遅延したにも関わらずリカバリー、メンバーの再配置の判断が遅れた為です。しかし、設計／組込みが完了したタスクから順次検証作業を行っていく事で、最終的には元々の日程計画からの遅れを問題ない程度に抑えることが出来ました。図２の不具合曲線からも評価期間中、不具合の検出と解決がコンスタントに行われているのが判ります。

現在では、開発計画立案時に業務分担を適切に行うと共に進捗遅れに対する是正タイミングと方法についてルールを作成して管理することにより対処することにしています。

２．３．　不具合発生の要因

表２が検出された不具合の要因一覧になります。表のタスク欄の中でDP：表示関係の問題が一番多くなっていますが、これは前述の特定メンバーへの業務の集中がこのDPタスクの不具合増大として現れたものです。しかし、このDPタスクの不具合数を除外しても、今回の開発ではソフトウェア構造の構想段階（基本STM設計）に時間を掛けたにも拘わらず、不具合の要因から見ると設計の誤り、漏れが最も多くなっています。

これは弊社のSTMによる設計プロセスが未だ不完全であったことに原因があります。以下にその要因と対応策を挙げます。

２．３．１．仕様未確定

要因：ハードの日程に合わせる為に、仕様の全てが明確になっていない状態でSTM設計に移行していました。また、我々にとっては今回STM設計に費やした時間では、未だ足りなかったといえます。評価段階で発見された仕様の抜け漏れの影響が広範囲の場合、複数のSTMを修正しなければならない為、多くの時間をロスすることになりました。

対策：全ての仕様が明確になってから、STM

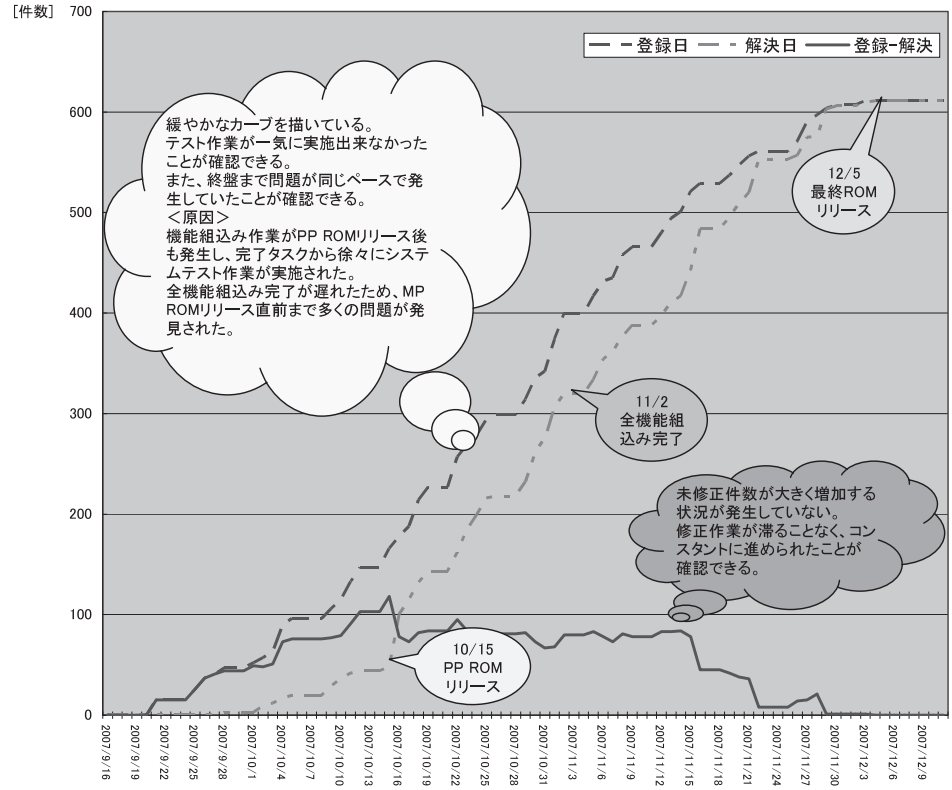

図２　不具合曲線

		不具合要因																													
		計画－企画上の甘さ	計画－計画立案の甘さ	設計－マイコンハード不具合	設計－周辺ハード不具合	設計－設計漏れ	設計－設計誤り	設計－設計の表現不足／冗長	設計－ソフト仕様誤解	設計－マイコンハード仕様誤解	作成－コーディング漏れ	作成－コーディング誤り	作成－未作成	作成－ファイル化誤り	作成－ROM／RAM圧縮ミス	作成－カット&ペースト時のミス	作成－仕様変更・追加	作成－仕様削除	作成－仕様漏れ	作成－仕様の誤り	作成－仕様の表現不備	作成－相互理解不足	作成－処理時間短縮	作成－ROM／RAM圧縮	確認－周辺ソフト不具合	確認－仕様通りの動作	ＡＬＬ
タスク	BIOS	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	BT	0	0	0	0	6	13	0	1	0	1	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	3	30
	CD	0	0	1	7	25	19	0	6	0	3	0	3	0	0	0	1	0	0	0	0	1	0	0	0	0	0	66
	CDCH	0	0	0	0	10	10	0	0	0	11	12	5	0	0	0	0	0	0	0	1	0	0	0	1	0	0	50
	DP	0	0	0	0	31	134	0	0	0	16	12	20	0	0	1	4	0	0	1	1	2	0	0	0	2	224
	etc	0	0	0	0	15	12	0	1	0	1	3	3	0	0	0	3	0	2	2	0	0	0	2	0	0	44
	IPOD	0	0	0	0	5	1	0	0	0	1	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	12
	JBUS	0	0	0	0	4	3	0	0	0	4	0	1	0	0	1	0	0	0	0	0	3	0	0	0	0	16
	KEY	0	0	0	0	2	6	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	11
	POWER	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	PSM	0	0	0	0	1	3	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	7
	SAT	0	0	0	0	5	11	0	0	0	0	1	3	0	0	0	2	0	0	0	0	0	0	0	0	0	22
	SERVICE	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
	SOUND	0	0	0	0	3	9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	14
	TUNER	0	0	0	0	25	10	0	0	0	3	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	44
	ZIPC	0	0	0	0	4	27	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	35
	ALL	0	0	1	7	139	260	1	8	0	41	40	44	0	0	2	13	0	2	3	3	6	0	2	2	6	580
	割　合	0%				72%					22%			5%											1%		100%

表２ 修正不具合要因一覧表

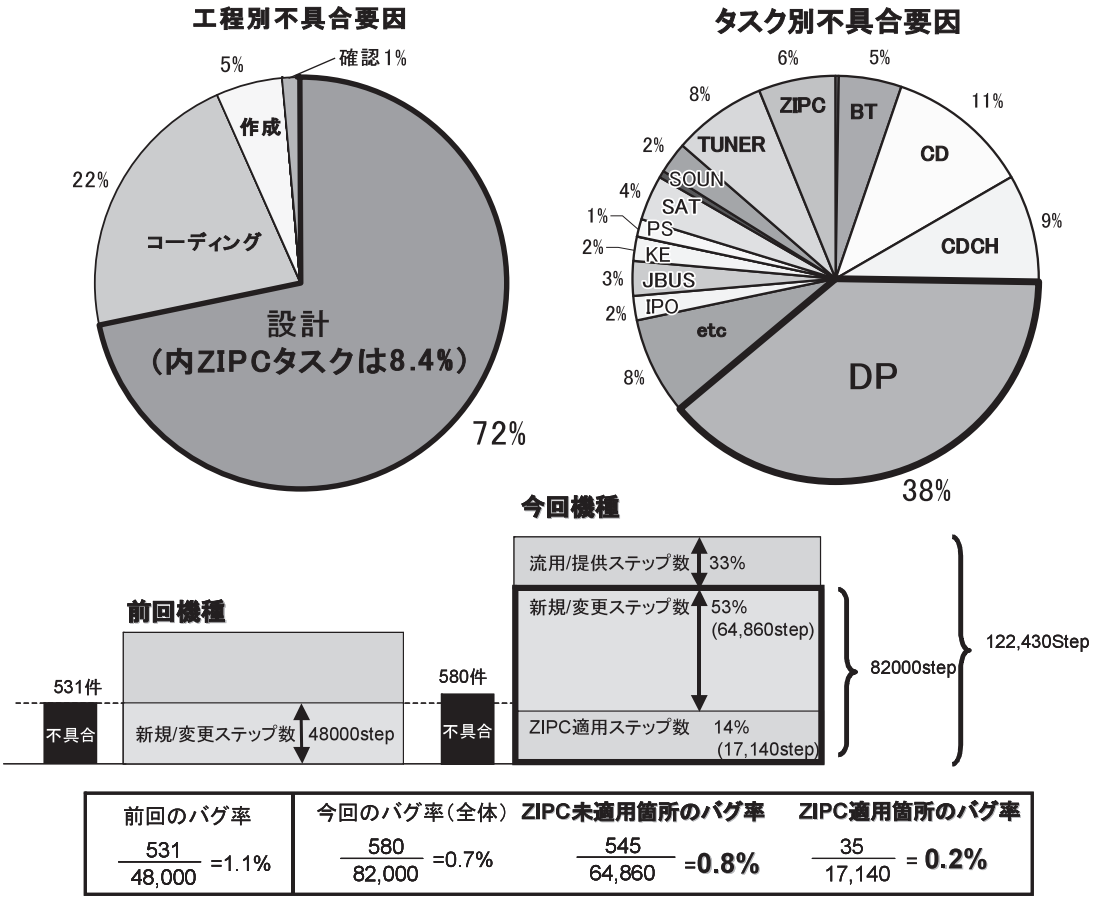

図 3 不具合要因とバグ発生率

設計に移るように開発プロセスにイベントを設ける。また、不明点について管理、フォローする担当を明確にする。

2．3．2．ZIPC開発工程のルール化

要因：図1を見ても判るように、STM上で修正を行う場合、基本STM設計へ戻って設計をやり直さなければならないのですが、詳細STM設計書のみの修正やシミュレーション作業無しでのコード生成などプロセスを逸脱した開発が一部行われました。

対策：計画時に開発ルール、修正ルールを規定して管理、運用する。

2．3．3．ZIPCの使いこなしが未熟

ハードウェアの開発状況によりソフトの開発プロセスが影響されない様にVIP等のシミュレーションをサポートする機能を活用する。

3．結論

以上のように、今回の開発を通じてSTMによる設計ではCATS殿から御指摘いただいたように、ソフトウェアの開発プロセス、ルールが重要視されなければならないことが改めて判りました。これはZIPCに限らずソフトウェア開発ツールを使いこなすにあたって全般的に言える事と思います。

4．今後の展開

現在、弊社ではAutomotive SPICEへの取組を推進しており、車メーカーへの納入品質、信頼性を向上させるためにプロセスの改善をさらに加速させる予定です。

また、ZIPC AUTOSARについても弊社の方向性に合致したツールとして今後の開発での導入に向け検討を行っていく予定です。