派生開発プロセスXDDPへの取り組みによる導入成果に期待

現在の組込みシステム開発の多くは、既存のものをベースにした派生開発が主流となっています。ベースがあり少しの修正だからという理由で短納期が求められる中、膨大なベースソースを理解する十分な時間がないことによる、部分理解に起因するトラブルも増大しています。当社では、解決策として派生開発に最適化したプロセス「XDDP」に取り組んでおり、開発プロセスの改善に活かしていきたいと考えています。

解消されない派生開発のトラブル

組込みソフトウェア開発のほとんどが「派生開発」だとされています。開発者たちにとっても、一から開発を行う新規開発の経験は非常に少ないのではないでしょうか。システムの多機能化に伴い、組込み機器のソフトウェアステップ数は年々増大していますが、開発期間が短くコスト削減も常に求められるため、ベースとなるソースをすべて理解してから開発を行うことは出来にくくなっています。

そのため、開発を行う際には"部分理解"の制約をどうしても受けてしまい、思い込みや勘違いによる不適切な作業も発生しがちです。しかし、こうしたトラブルがなかなか解消されないのは、既存の方法論の多くが新規開発を対象にしたものであり、派生開発特有の問題に対応していないことが大きな原因と考えられます。

これらの解決策として、派生開発に最適化したプロセス「XDDP」をカーナビゲーションシステムの派生開発業務に導入することになり、パイロットプロジェクトで経験を積み、実践での取り組みを進めています。

XDDPの3点セットとは

XDDPとはeXtreme Derivative Development Processの略です。これ以上省けないほどに合理性を追求した派生開発専用のプロセスという意味で、プロセス改善コンサルタントである(株)システムクリエイツの清水吉男氏が提唱している開発プロセスです。

派生開発におけるバグの多くは、ここを変更すればよいと思ったところが最適な箇所でなかったり、関連箇所があることに気づかなかったり、勘違いして不要な修正をしてしまったりしているところから発生しています。そして、従来の開発方法の問題は、

・変更要求をどのように理解したか
・変更要求に対してどこを変更しようとしているか
・どのように変更しようとしているか

を十分に記述していなかったために、担当者の意図を知る方法がなかったことにあります。

XDDPでは、これらを記述することができる、変更要求仕様書(USDM[注1]形式)、トレーサビリティ・マトリクス(TM)、変更設計書の3点セットならびにスペックアウト資料をもとに開発を進めます(図-1)。

まず、変更要求仕様書は、「何を変更するか」というWhatの視点を持って記述するものです。派生開発では、変更要求の意味を的確に理解し、その要求を満たすために変更しなければならない仕様を漏れなく見つけ出す必要があります。この際、「なぜ変更が必要か」の理由を明記することで、当初考えた変更方法だけでは不足していたりすることに気づくことができるようになります。

また、「before」、「after」で記述をするため、現状を記述している「before」でソースコード、各種設計資料の該当箇所を探しやすくなり、さらに、「before」、「after」の「差」から変更の様子(変更行数、難しさ、変更箇所の拡散状況など)を理解することができるようになります。

2つめは、Whereの視点のトレーサビリティ・マトリクスです。変更要求仕様書と同じシート内(図-2)に記述し、左側の仕様書に対してどのソースのどの位置の関数に手を入れるかなどを記入していくものとなります。

成果物	視点	
変更要求仕様書	What	"何を"変更するか? "どういう仕様を"どのような仕様に変更するか? "何を"変更すればよいと認識しているのか? 他に関連して変更しなければならない"仕様"を見つけたか?
トレーサビリティ・マトリクス	Where	その変更が、"どこに"あるのか? 変更する仕様がどこにあると認識しているのか?
変更設計書	How	具体的にどのように変更するのか? どのように変更すれば目的を達成できると認識しているのか?

図-1 XDDP 3点セット

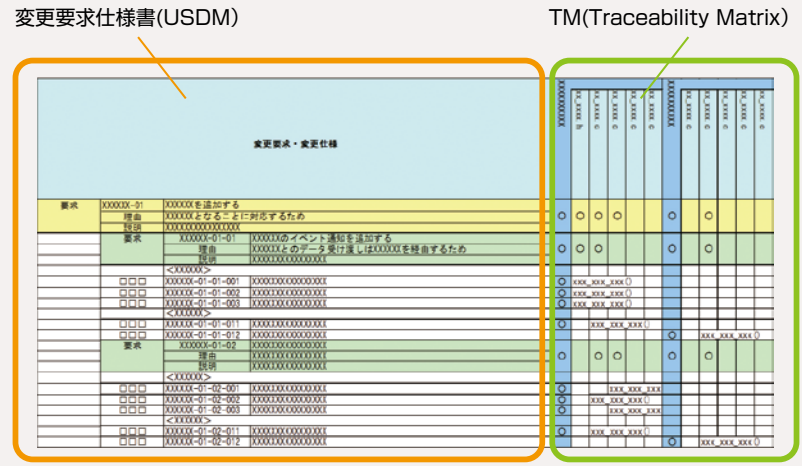

図-2　変更要求仕様書(USDM)とTM (Traceability Matrix)

変更要求仕様書およびトレーサビリティ・マトリクスの効果としては次のものがあります。

(1) 類似の変更要件と比べることにより「修正箇所の推理」ができる

(2) 変更要求ごとに関連する変更箇所を確認するのが容易になる

(3) 変更要求仕様に対して複数モジュールに関わる変更が一目できる

(4) 複数の担当者が修正に関わっているときの構成管理が容易になる

(5) 異なる要件で同一の関数に修正があったとき、まとめて修正する方法を検討できる

3つめのHowの視点である変更設計書(図-3)には、変更要求仕様で求めていることを実現するために、「どの関数」の「どの部分」を「どのように」変更するかを記述します。変更設計書を書くタイミングは変更要求仕様書とトレーサビリティ・マトリクスのレビューの結果、「最適な修正箇所である」と判断された後になります。「他の箇所で修正すべき」とレビューで判断された場合は、もし先に書いていれば手戻りになってしまいます。

変更設計書には、ソースコードを書くのではなく、文章で記述します。ソースコードを書くと、結果的に修正したのと変わらず、よりよい方法を考えなくなってしまうためです。

以上がXDDP3点セットとなりますが、派生開発ではどうしてもベースとするソースコードを理解する必要があります。概念図的なものを使って機能の全体像やシステムの構成を表したり、データ構造定義やプログラムフローを書いていくことで、理解の手助けとします。これらをスペックアウト資料と呼びます。

社内の教育や開発プロセスの改善へ活用

派生開発は部分理解の制約が少なくありませんが、XDDPでは担当者本人のみならず、レビュアーへも「最適な変更箇所であるか」、「対応箇所に不足がないか」を判断するための材料を与えてくれます。このため、プロセスに起因するバグを大幅に解消でき、トータル工数や手戻り作業も軽減でき、品質向上とコスト削減にも繋がっていきます。過去には、最適な箇所を見つけないまま修正してしまい、その後も繰り返し修正が入ることで新たな不具合を埋め込んでしまったり、デグレード(ソフトウェアのバージョンアップに伴う品質低下)してしまったり、といった経験もしてきており、XDDPの積極的な導入を進めていく考えです。

また、XDDPの考え方はベテラン開発者が無意識のうちに頭の中でやっているかもしれません。若手や新しいメンバーに「どういう考え方でやっているか」を伝えていくことにも繋がり、理由を明記していくことで「なぜ」を考える習慣ができ、仕様書の誤読による不具合も防止できていくと考えています。こうしたことから、XDDPを教育・育成の面でも活用できると期待しています。

今後、社内での活用を広げて、XDDPによる開発プロセスの改善の取り組みに全力を傾注していきたいと考えています。

(エンベデッドアプリケーション事業部
番場正弘)

＜参考文献＞
1.『「派生開発」を成功させるプロセス改善の技術と極意』清水 吉男 著　(技術評論社)
2.「失敗しない派生開発」清水 吉男 著 (技術評論社『Software People vol.8』)

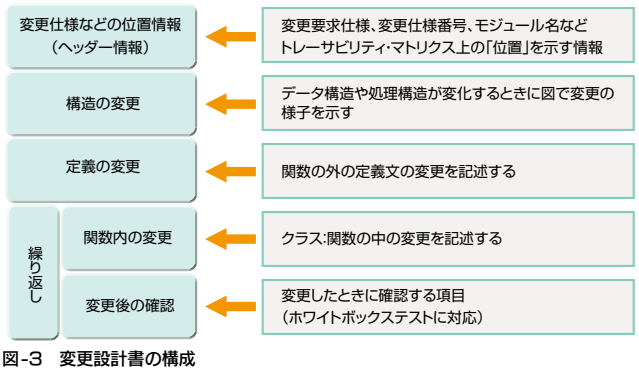

図-3　変更設計書の構成

(注1)Universal Specification Description Manner:XDDPで使用する要求の仕様化方法のドキュメントテンプレート。