ZIPC TERASの導入と活用事例の紹介

NECスペーステクノロジー株式会社
技術本部 ソフトウェア技術部
小林　友樹

1. 概要

宇宙機へのミッション要求の多様化、高精度化に伴い、ソフトウェアも複雑および大規模となってきている。そのため、より一層、ソフトウェア開発の上流から要求管理を確実に実施する必要性が高まっている。その一方で、作成する文書や品質記録も膨大になることで、文書作成そのものや品質記録の確認の作業効率が課題となっている。弊社では要求管理を確実に実施するため、トレーサビリティ確認作業の効率化を目的として、トレーサビリティ管理ツールであるZIPC TERASを導入した。本論文では、その活用事例を紹介する。

2. 宇宙事業におけるソフトウェア開発モデル

宇宙事業におけるソフトウェア開発はV字型モデルを基本としている。各ソフトウェア開発プロセスは、ISO/IEC 12207「Information technology - Software life cycle process」[1]に基づいて行われている。例えば宇宙航空研究開発機構殿との契約時には「ソフトウェア開発標準（JERG-0-049）[2]が適用される。「ソフトウェア開発標準」とは、宇宙におけるソフトウェア開発の標準プロセスを定めたものであり、ISO/IEC 12207に基づいている。また、各プロセス中で行われるべき活動も定義されており、それらを実施することで、確実な品質の作りこみ、検証および妥当性確認が行われている。ISO/IEC12207の各プロセスを工程別にマッピングした、V字型のソフトウェア開発モデルを図2-1に示す。

宇宙開発において、トレーサビリティ確認は重要な品質保証活動の1つである。従来からトレーサビリティ管理の仕組みは確立され、実施されてきた。V字モデルの工程毎に作成された成果物に対してトレーサビリティ表が作成され、上位の仕様が確実に落とし込まれていることをレビューや審査会の中で確認する。トレーサビリティ確認が十分であることは工程移行の判定の重要な基準の1つとなっている。

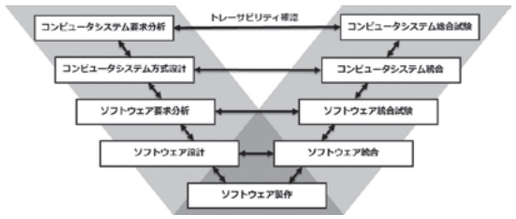

図2-1　宇宙事業におけるソフトウェア開発モデル

3. ZIPC TERASの導入

本章ではZIPC TERAS導入の目的および期待する効果を説明する。

3.1 なぜZIPC TERASの導入が必要か

従来からトレーサビリティ管理は確立されている宇宙事業だが課題もある。ソフトウェアの複雑化および大規模化に伴い、作成する文書や品質記録が膨大になることで、文書作成そのものや品質記録の確認の作業効率の課題が増えている。トレーサビリティ管理に関して主に、以下の各作業効率における課題がある。

① トレーサビリティ表作成作業の効率
② トレース漏れの確認、逆トレーサビリティ確認作業の効率
③ 複数文書にまたがる変更時の確認作業の効率

これらの課題を改善するためZIPC TERASの導入を行った。次項からこれらの課題に対して、ZIPC TERAS導入に期待する効果を述べる。

3.2 トレーサビリティ表作成作業の効率化

まず、課題の1つ目「トレーサビリティ表作成作業の効率」についてZIPC TERAS導入に期待する効果を述べる。

現状トレーサビリティ表は、図3-1に示すように仕様書の目次や機能・性能名等を基にコピー&ペーストまたは直接タイピングにより作成している。

ソフトウェアの複雑化および大規模化に伴い、当然トレース対象の量も多くなる。またトレーサビリティ結果は1：1や1：複数とは限らず複数：複数となる場合もある。そのため、トレーサビリティ表を作成する作業の効率に課題があるのが現状である。また、量が多い分、人為的ミスも誘発しやすい作業となっている。加えて、仕様書を変更した際、トレーサビリティ表の更新も当然必要であり、修正から確認までの再実施が必要となる。この時、文書とトレーサビリティ表が整合しなくなる問題が起こり得る。仕様書とトレーサビリティ表間が不整合の状態のままだと、例えば次の変更の時に影響解析が正しく行われず、不具合を引き起こす要因にもなるため回避しなければならない。

対して、ZIPC TERASを用いた場合、仕様書をZIPC TERASにインポートし、解析されたモデルを基にトレーサビリティ表が自動生成される。トレース対象間の紐付けは対応する項目同士をモデル上で繋げるだけであり、トレーサビリティ表を作成する作業に掛ける時間を大幅に削減することが可能である。また、紐付けが直観的でわかり易いため人為的ミスの軽減も期待できる。さらに、仕様書変更時には、変更した仕様書をZIPC TERASへ再インポートする事で、自動的に仕様書とトレーサビリティ表の整合性を保つ事ができる。従来実施していたトレーサビリティ表の手動更新が不要となり、ZIPC TERASへ再インポート後のトレーサビリティの変更差分を確認するだけで済むようになる。

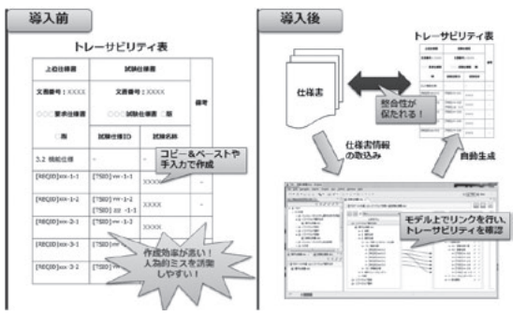

図3-1　トレーサビリティ表作成作業の効率化

3.3 トレース漏れ確認、逆トレーサビリティ確認作業の効率化

次に、課題の2つ目「トレース漏れ確認、逆トレーサビリティ確認作業の効率」について、ZIPC TERAS導入に期待する効果を述べる。

従来、トレース漏れ確認は図3-2に示すようにトレーサビリティ表を用いて行っていた。しかし、トレース対象が膨大でトレーサビリティ表が複数ページにまたがる場合は、トレース漏れに気づけない可能性がある。

また、トレース漏れを見つけるために逆トレースの観点で確認することも多い。その場合、既にあるトレーサビリティ表とは別に逆トレーサビリティ表を新規に作成していた。つまり、単純に2倍の作業量がかかっていたことになる。

対してZIPC TERASでは、トレース漏れの項目が図3-2に示すように視認性高く表示される。また、フィルタリングを行うことでトレース漏れの項目だけを自動でピックアップできる機能もあり、トレース漏れ確認の効率化を図ることが可能である。

逆トレーサビリティ確認作業についても、新たに逆トレーサビリティ表を作成しなくてもZIPC TERAS上で確認が可能である。また、ZIPC TERASのモデル情報を基に逆トレーサビリティ表も自動生成できるため、逆トレーサビリティ表が確認の確証として求められる場合にも作成作業の効率化が期待できる。

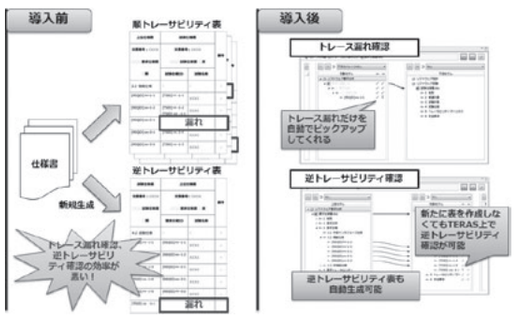

図3-2　トレース漏れ確認、逆トレーサビリティ確認作業の効率化

3.4 複数文書にまたがる変更時の確認作業の効率化

最後に、課題の3つ目「複数文書にまたがる変更時の確認作業の効率」について、ZIPC TERAS導入に期待する効果を述べる。

トレーサビリティ表は各工程にて確認対象文書毎に作成している（例えば、ソフトウェア要求仕様書、ソフトウェア設計書、ソフトウェア試験仕様書毎に作成している）。

不具合や要求変更時に影響範囲の評価を行う際は、それぞれのトレーサビリティ表を確認しなければならないのが現状である。

対して ZIPC TERAS では、全工程のトレース結果が 1 つのリポジトリで管理されるため、一気通貫で影響範囲を確認することが可能である。例えば、図 3-3 に示すように変更箇所と紐付いている影響範囲を自動で抽出することが可能である。また、ZIPC TERAS V2.0 から追加された Link Viewer を用いれば、複数の文書を串刺しした全体像を確認することができる。これによって変更時の影響範囲確認の効率化が期待できる。

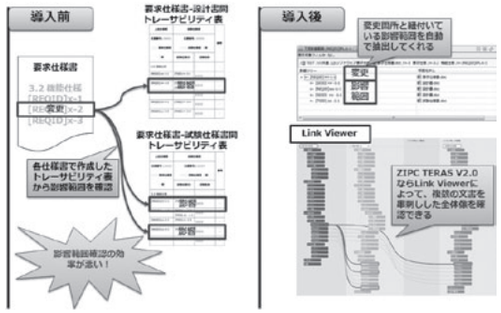

図3-3　複数文書にまたがる変更時の確認作業の効率化

4. ZIPC TERAS の活用事例

本章では、ZIPC TERAS の活用事例を紹介する。事例では主に ZIPC TERAS の機能の 1 つであるタグベースリンク機能に焦点を当てて説明する。

4.1 タグベースリンク機能について

タグベースリンク機能とは、仕様書中にキーワード（トレースタグ、参照タグ）を埋め込むことで、ZIPC TERAS にモデルとしてインポートする際、自動的にトレーサビリティリンクを作成できる機能のことである。

弊社では図 4-1 に示すようにソフトウェア開発において仕様に ID を付与しており、ZIPC TERAS 導入に際して、ID の先頭に ZIPC TERAS 専用のキーワードを追記する方法を採用した。キーワードはプロジェクトや組織で自由に決めることができる。また、ZIPC TERAS では仕様書からキーワードを抽出するためのルールファイルテンプレートが用意されている。このルールファイルは正規表現で記述できるため、これをカスタマイズすることで任意のキーワードを抽出することが可能となる。

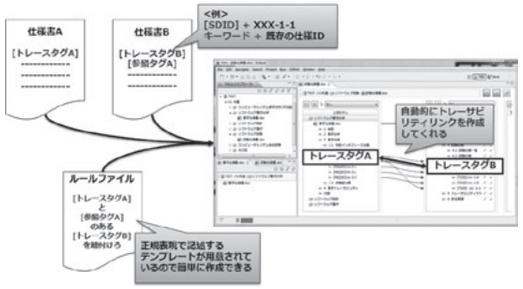

図4-1　タグベースリンク機能

4.2 なぜタグベースリンク機能を採用したか

本項ではなぜタグベースリンク機能を採用したか、その理由を述べる。

ZIPC TERAS におけるトレース手法は大きく 2 つある。1 つは対象同士をマウスでドラッグ＆ドロップして繋げる手動リンクの方法であり、もう 1 つはタグベースリンク機能による自動リンクの方法である。

次にこの 2 つの方法の違いを説明する。まず手動リンクについては図 4-2 の左側に示すように、トレーサビリティ確認を行う際、確認対象の各仕様書と ZIPC TERAS を交互に見ながら ZIPC TERAS 上で紐付けを確認しリンクを繋げる作業となる。仕様書を段階的に作成しながらリンクを繋げる場合、仕様書を更新し、ZIPC TERAS にインポートし、ZIPC TERAS 上でリンクを繋げる作業の繰り返しになる。そのため、作業者としては仕様書と ZIPC TERAS の頻繁な切り替えを回避するため、仕様書が完成してから最後にリンクを繋げる方法を選びがちである。

一方、自動リンクの場合、図 4-2 の右側に示すように、リンク情報を仕様書に埋め込むため、作業者は仕様書と ZIPC TERAS を頻繁に切り替える必要がない。つまり仕様書の作成に集中できるメリットがある。仕様書の作成途中においても、ZIPC TERAS へ適宜インポートすることでトレース漏れの確認を段階的に実施できる。

ここで自動リンクの場合、新たに「キーワードを埋め込む」作業が必要となる。それに伴う作業効率の低下が懸念されるが、作業者の思考過程を考慮した場合、総合的には費用対効果が大きい作業であると考えられる。

例えば設計をする際、担当者は、今から行う設計がまず何の要求に基づくものなのか確認し、作業している。つまり､そのタイミングで上位のキーワード情報である参照タグ等を埋め込むことは、従来頭の中でやっていたことを視える化しているに過ぎない。上位を確認しているタイミングでキーワードを埋め込むため間違いも少なくなり、トレーサビリティ情報自体の品質も向上する。

もちろん手動リンクも自動リンクもトレーサビリティを管理する上での品質確保は達成できるものである。しかし、弊社では本項で述べた理由により、自動リンクの方が 3 章で述べた課題に対して有効だと判断したためタグベースリンク機能を採用した。

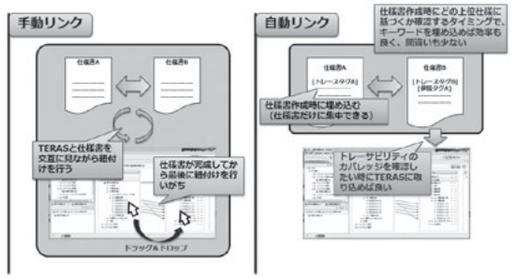

図4-2　手動リンクと自動リンクの比較

4.3 ZIPC TERAS 上でのソフトウェア開発工程の整備

本項から実プロジェクトへ ZIPC TERAS を導入するために行った事例を紹介する。

最初に、ZIPC TERAS 上で弊社のソフトウェア開発工程の定義を取り込む作業を行った。

ZIPC TERAS では TRA 設定ツールと呼ばれる補助ツールが用意されている。このツールでは図 4-3 に示すように、ソフトウェア開発工程を定義し、トレース対象の上流・下流の関係を定義することができる。

TRA 設定ツールでは新規に定義しなくても、デフォルトで一般的なソフトウェア開発工程が用意されている。しかし、社内のソフトウェア開発工程に定義を合わせ、トレースを取るべき上流・下流の関係を整備することで、より効果的に ZIPC TERAS を運用することが可能であると考える。

例えば、TRA 設定ツールにてソフトウェア設計工程の下流トレース対象としてソフトウェア製作工程とソフトウェア統合工程を定義したとする。これによってトレース漏れ確認の際、“ソフトウェア製作工程の成果物とはトレースがとられているが、ソフトウェア統合工程の成果物とはトレースがとられていない” 等のアラームを ZIPC TERAS が挙げることが可能になる。

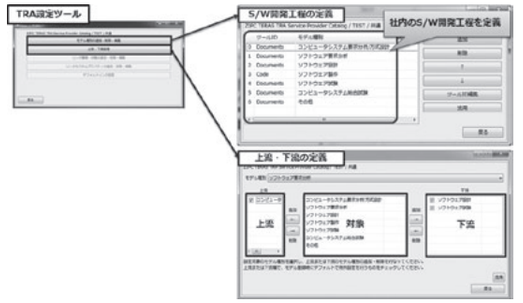

図4-3　ZIPC TERAS上でのソフトウェア開発工程の整備

4.4 仕様書へのキーワード付与のルール化

次に、タグベースリンク機能を有効に使うために、各仕様書へ埋め込むキーワードをプロジェクト内でルール化した。そして実際に各工程で既存の仕様 ID の先頭に ZIPC TERAS 用のキーワードを埋め込んだ。

例えば、要求仕様書で言えば、ID の先頭に [REQID] というキーワードを埋め込むことをルール化した。設計書では、同じように [SDID] を付加し、その下の行には参照元である要求仕様のキーワードと仕様 ID を追記した。ソースコードでは各関数の関数ヘッダ内にコメントとして参照元の設計 ID とキーワードを追記した。そして 4.1 項で紹介したように、各キーワードを抽出するための、ルールファイルも仕様書毎に整備した。

ルールファイルの整備作業は、次プロジェクトでも同じキーワードを採用すれば、ルールファイルは変更なくそのまま使える為、最初に整備することで、費用対効果を高めることができる。

このようにソフトウェア開発の上流から下流全てで一貫してキーワードを付加することで、ZIPC TERAS による自動リンクを有効に活用する下地を構築した。

ZIPC WATCHERS Vol.19

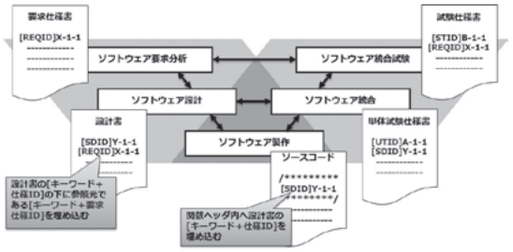

図4-4　仕様書へのキーワード付与のルール化

4.5 確実にトレーサビリティ要素を取込むために（TIPS）

次にこれからタグベースリンク機能を使う人のために TIPS を紹介する。

ZIPC TERAS では仕様書から目次項目等を取り込む際、Word 文書なら Word の機能にある「アウトライン設定」があるものを識別して ZIPC TERASにトレーサビリティ項目を表示している。そのため、弊社の場合では、仕様 ID に対しても確実なアウトライン設定が必要となる。なぜなら仕様 ID への設定漏れがあると、ZIPC TERAS そのものに表示されないため、トレーサビリティ確認が出来なくなるためである。

そこで、弊社では VBA マクロを使って、キーワードへ自動でアウトライン設定するように工夫した。これによってアウトライン設定を行う時間の短縮と、先に述べたアウトライン設定漏れを防止することができた。

図4-5　確実にトレーサビリティ要素を取込むために

4.6 ZIPC TERAS 導入の効果

最後に ZIPC TERAS 導入の効果を紹介する。

弊社では、ZIPC TERAS の導入により、トレーサビリティ確認作業において、約 40% の工数削減効果が得られた。

内訳としては、トレーサビリティ確認作業として大きく以下の 3 項目を挙げる。

① トレーサビリティ表を作成する作業
② 仕様書とトレーサビリティ表の整合性確認作業
③ 実際にトレーサビリティを確認する作業

これら 3 つの作業それぞれで工数削減効果があった。

効果が顕著だったのは①のトレーサビリティ表作成作業である。ZIPC TERAS による自動生成によって大幅な効率化が得られたと考える。

次に②の仕様書との整合性確認作業では、もともと作業割合は少なかったものの、ZIPC TERAS によって仕様書とトレーサビリティ表間の整合性が確保されるようになったことで、ほぼゼロにすることができた。

③のトレーサビリティ確認作業においては、逆トレース確認にかかる工数削減効果が大きかった。

また、ソフトウェア設計工程では、設計の一部を関係会社に委託することがあった。そこで関係会社が作成した設計書を弊社が ZIPC TERAS にインポートし、ZIPC TERAS 上でトレーサビリティを確認し、問題点をフィードバックする等のイテレーションを効果的に回すことができた。ZIPC TERAS 導入前に比べ、トレーサビリティ確認のレビューに充てる時間を増やすことにもつながったと考える。

このようにトレーサビリティ確認は、人間が実際に目で見て対応を確認するべきところであり、本来 3 つの作業の中で最も時間を掛けるべきところだと考える。

ZIPC TERAS 導入によって、本当に時間を掛けたい作業に集中できるようになった、という結果が得られたともいえる。

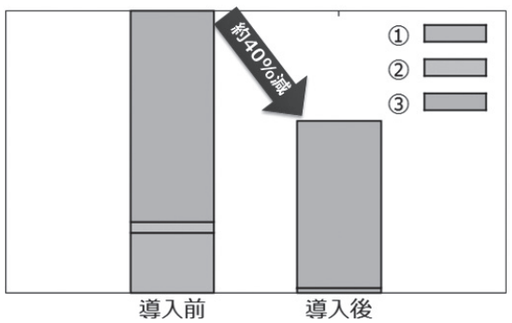

図4-6　ZIPC TERAS導入の効果

5. まとめ

最後に、忘れてはいけないのは、あくまでトレーサビリティを確認するのは人であることである。

ZIPC TERAS は、トレーサビリティ表の自動生成や、トレース漏れを自動検出することにより、人にかかる負担やミスを軽減するために活用するのが有効である。

今回の ZIPC TERAS 導入によってトレーサビリティ確認作業の効率化が可能であることが確認できた。この効率化によって得られた時間で、人にしかできないことに時間を充てることができるようになり、本来ありたい姿に近づけることができる。

今後は、ZIPC TERAS による効率化として、派生開発にも効果が期待できるため、継続して評価していきたいと考えている。例えば、変更箇所の影響解析（開発規模見積りの参考情報等）や、タグベースリンク機能の活用（既存文書へのキーワード埋め込み促進）を検討していきたい。

弊社では 2011 年の当時の一般社団法人 TERAS 設立当初から導入検討を始め、試行を経て、現在の導入に至っている。その中で気づいた改善点や追加して欲しい機能をフィードバックしながら、今後も ZIPC TERAS に期待して活用を継続していきたいと考えている。

参考文献

［1］ Information technology - Software life cycle process ISO/IEC 12207
［2］ 宇宙航空研究開発機構：ソフトウェア開発標準 JERG-0-049