Vol.2013-DD-88 No.8
2013/1/18

統合ソフトウェアドキュメンテーション環境としての DITA ベース CMS の開発

坂井麻里恵[†1]　大場みち子[†2]　伊藤恵[†2]　奥野拓[†2]

ソフトウェアの開発プロセスにおいて，多くのドキュメントが作成される．しかし，ワードプロセッサ等のアプリケーションで作成された一般的なドキュメントでは，ドキュメント数の増加に伴ってトレーサビリティの低下，メンテナンスコストの増大という問題が発生する．技術文書向けの XML 標準である DITA に基づいてドキュメントを記述し，コンポーネント単位で再利用することで，それらの問題を解決する．ソフトウェアドキュメントを DITA のトピックとして記述するために，ドキュメントを構造化し，トピック粒度の検討を行った．また，DITA に基づいたソフトウェアドキュメンテーション環境としての CMS の構築を目指し，必要な機能の検討を行った．

Building DITA-based CMS as Integrated Documentation Environment

MARIE SAKAI[†1]　MICHIKO OBA[†2]
KEI ITO[†2]　TAKU OKUNO[†2]

General software documents have some problems with traceability and maintenance. This study aims to solve those problems by using DITA, an XML-based architecture for authoring technical documents. In DITA architecture, documents consist of topics and maps. Topics have small amount of information which can reuse, and maps specify contents of each output by organizing topics. It is difficult to break down general documents into reusable topics. In this paper, granularities of topics are defined. As a result of analysis of software documents, it was found that there are three types of structures in software documents, and those need to be broken down into different-sized topics. Besides, a CMS as a software documentation environment is needed for software documentation using DITA. Functions required for the CMS are considered.

1. 序言

ソフトウェアの開発プロセスにおいて，提案書，要件定義書，設計書といった多くのドキュメントが作成されている．これらのドキュメントは，一般的に大きく三つの問題を抱えている．まず，ドキュメントの増加に伴うトレーサビリティの低下, メンテナンスコストの増大の二つである．これら二つの問題は，ドキュメント各々が独立し，各ドキュメント単位で内容が完結していることに起因する．複数ドキュメントを横断しての情報閲覧が困難であるために，ドキュメント数が増加するほど，全体を通してのトレーサビリティが低下していく．また，内容の一部に変更が生じた際も，影響が波及する部分をドキュメント毎に修正する必要があり，メンテナンスコストが大きい．これらは，ドキュメントの内容に不整合が生じる原因にもなっている．ドキュメントの内容の不備は，それらを参照して作成したソフトウェアにも引き継がれるため，結果的にソフトウェア自体の品質の低下を招く恐れがある．三つ目の問題は，多くのドキュメントが汎用のワードプロセッサやスプレッドシートを用いて作成されるために，執筆と同時に書式設定を行う必要が生じることである．これは非本質的な作業であるため，生産性の低下に繋がっている．

一方，近年，マニュアルやヘルプファイルなどの技術文書において, XML ドキュメントのアーキテクチャ標準である DITA（Darwin Information Typing Architecture）が用いられている[1]．DITA に基づくドキュメントは，再利用可能な情報の単位であるトピックと，トピックの組み合わせからなる文書構造を定義するマップで構成される．この構造をソフトウェアドキュメントに適用すると，関連する情報を一括して参照することや，同じ記述を複数ドキュメントで再利用することが容易になる．しかし，従来のソフトウェアドキュメントはドキュメント単位での管理を想定して記述されているため，容易にトピック単位に分割することができない．

本研究は，DITA を適用することでソフトウェアドキュメントにおける問題を解決し，ドキュメントの作成・管理を効率化することを目的とする．ドキュメント間のトレーサビリティを確保することで，内容に変更が生じた際の修正漏れを防ぐことができ，結果として不整合が生じる可能性が低下する．ドキュメントの整合性は，それらを参照して作成されるソフトウェアの品質にも影響するため，不整合の解消はソフトウェアの品質の向上に繋がる．

ソフトウェアドキュメントに DITA を適用するためには，従来のドキュメントを分析し，適切な粒度でトピックに分割する必要がある．そこで，ソフトウェアドキュメントの

†1 公立はこだて未来大学大学院
Graduate School of Future University Hakodate
†2 公立はこだて未来大学
Future University Hakodate

構造化を行い，適切なトピックの粒度について考察する．

また，DITA のコンテンツは一般的にコンテンツマネジメントシステム（CMS）で管理されるため，ソフトウェアドキュメンテーション環境としての CMS の構築を行う．

2. ソフトウェアドキュメンテーションの現状

まず，一般的なソフトウェアドキュメンテーションの現状と，その問題点について述べる．

2.1 ソフトウェアドキュメントの性質

一般的なウォーターフォール型開発では，要件定義，設計，実装，テスト，運用といった一連の作業工程に従って開発を行う．各工程の成果は主にドキュメントとしてアウトプットされ，その内容は開発が進行する毎に段階的に詳細化されていく．そのため，作成された一連のドキュメントには，類似した内容や互いに依存関係のある内容が多く含まれている．図 1 は，依存関係にあるドキュメントの例である．「業務一覧」のドキュメントにおいて，見積書を作成する業務について触れられており，「機能一覧」のドキュメントではそれを受け，見積書作成機能の説明が記述されている．さらに，「入出力一覧」のドキュメントでは，見積書作成機能における入出力項目が記述されている．

このように，多くのソフトウェアドキュメントは互いに関連する内容を含んでいる．そのため，ドキュメントを参照する際や内容の一部を変更する際，該当箇所のみでなく，関連する他のドキュメントにまで遡る必要がある．しかし，ワードプロセッサやスプレッドシートで作成された従来のドキュメントは，それぞれが独立して管理されており，複数ドキュメントを横断しての閲覧には適していない．こうした必要性から，ソフトウェアドキュメントにおけるトレーサビリティを確保するために様々な手法が研究されてきた．

2.2 XML の利用

ドキュメントのトレーサビリティを確保する手法の一つに，XML を用いたドキュメンテーション手法がある．松島らは，ソフトウェアドキュメントとそれらの間のリンクを XML で記述することにより，ドキュメントを閲覧する際，関連のあるドキュメント要素も同時に一覧できるシステムを提案している[2]．関連するドキュメントを提示することによって，ドキュメントの保守や閲覧を効率良く行うことができる．このような手法は数多く提案されているが，いずれの手法も XML 文書の互換性において共通の問題を抱えている．ソフトウェアドキュメントを XML で記述する場合，文書内で利用する要素名やその出現回数などは，ドキュメントの構造に応じて独自に定義できる．しかし，作成したドキュメントは独自の型に依存するため，外部組織との間で読み書きするための互換性が失われてしまう．

一方で，技術文書向けの標準となる型を定義したものも

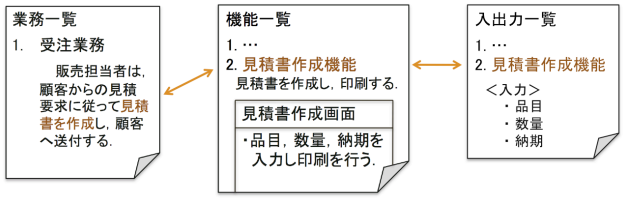

図 1　ドキュメント間の依存関係

Figure 1　Dependency Between Software Documents

存在する．DocBook がその一例である[3]．このような標準を利用すれば互換性は保たれるが，利用できる要素や構造が予め用意されたものに限られるため，ドキュメントの記述の自由度は制限されてしまう．そこで，本研究では，双方のメリットを併せ持つ DITA を用いる．

3. ソフトウェアドキュメントへの DITA の適用

次に，DITA におけるドキュメントの構造と，一般的な XML ドキュメントとの違いについて説明する．そして，ソフトウェアドキュメントへ適用した際の利点と，適用に際しての課題を述べる．

3.1 DITA に基づくドキュメントの構造

DITA(Darwin Information Typing Architecture)とは，XML で記述する技術文書のための標準仕様である[1]. 2005 年に OASIS によって標準化され，主に欧米で導入が進んでいる．

DITA に基づくドキュメントは，トピックとマップと呼ばれる要素で構成され，共に XML で記述される．トピックは，異なるコンテクスト間で再利用可能な単位の情報である．マップでは，それらのトピックの組み合わせとなる文書の構造を定義する．この 2 つの要素により，情報をトピック単位でリポジトリに保存し，必要に応じてマップ上でトピックを組み替えることで，同じトピック群から様々な形のドキュメントを出力することができる構造になっている．

この構造の利点は，トピックが文脈に依存せず，複数のドキュメントで再利用できることにある．従って，トピックは可能な限り小さい粒度に分割されていることが望ましい．トピックの粒度が大きいと，再利用できる範囲が狭まってしまい，トピック単位で管理するメリットを十分に得られない．

3.2 トピックの型

DITA では，トピックの基本の型として Concept，Task，Reference，Glossary の 4 種類の型を提供しており，それぞれ異なる用途に沿った要素が定義されている．例えば，Task トピックは特定の目標を達成するための手続きを記述するのに適した型であり，作業手順をステップ毎に記述するための<steps>という要素が用意されている（図 2）．これらの型は，技術文書のために定義された汎用的な構造であるた

め，一般的な技術文書の大半をこれらの型に従って記述する事ができる．

一方で，DITA の特徴に特殊化という仕組みがある．特殊化とは，既存の型から派生する形で，新たに独自の型を定義することをいう．特殊化した型は元となる型やその要素を継承するため，特殊化後の型の定義情報を持たなくとも，元となる要素に置き換えて解釈することが可能である．つまり，特殊化の仕組みに則っている限り，独自の型をどのように定義しても，汎用的な要素に置き換えることでドキュメントを読み書きすることができる．この仕組みにより，DITA は，他組織との間でのドキュメントの互換性と，独自に型を定義できる柔軟性を共に兼ね備えている．

3.3 DITA 適用の利点

DITA の仕組みに基づいてドキュメントを作成することで，ソフトウェアドキュメントにおける三つの問題の改善が期待できる．

まず，複数のドキュメントに散在する情報から，関係する部分だけを一括して参照することが容易になる．図 3 のように，従来は「業務一覧」「要件定義書」「画面設計書」の三つのドキュメントに分かれていた情報を，トピック単位に切り分けてリポジトリに保存する．すると，改めて文章を加筆することなく，「機能 A」という一つの機能についてのみ記述されたドキュメントを出力することができる．このように，目的に応じてマップを作成することで求める情報を取り出すことができるため，従来のドキュメントに比べトレーサビリティが向上すると考えられる．

また，トピックは複数のマップから参照する形で，ドキュメント間で共有することができる．従って，トピックの内容を変更する際，同じトピックを含む全てのドキュメントをそれぞれ修正する必要は無い．元トピックの変更が全ての共有先に反映されるため，修正回数や範囲が小さくなり，メンテナンスコストの削減が見込める．

さらに，XML で記述することで書式情報と内容が分離されるため，非本質的な書式設定の作業を執筆と同時に行う必要が無くなり，生産性の向上が期待できる．これらの利点から，本研究では DITA に基づいたソフトウェアドキュメンテーション手法の構築を目指す．

3.4 DITA 適用における課題

DITA を適用することでソフトウェアドキュメントにおける問題を解決することができるが，ソフトウェアドキュメントを DITA のトピック単位に分割するために，適切な粒度の検討が必要である．また，ドキュメンテーションに伴う作業も従来とは異なるため，DITA ドキュメンテーションに適した環境の構築が必要である．

DITA ではトピック単位でドキュメントを管理することから，1 項目毎に内容が独立し，トピック単位に分割しやすいマニュアルやヘルプファイルといったドキュメントに主に用いられてきた．また，本研究の対象であるソフトウェアドキュメントに適用した事例として，Diaz らの研究がある[4]．Diaz らは，ソフトウェアプロダクトライン開発という手法において，ドキュメンテーションに DITA を用いることを提案している．ソフトウェアプロダクトライン開発は，ドメイン毎にソフトウェアを開発し，それらを組み合わせて製品化する開発手法である．この例も，DITA のアーキテクチャとソフトウェアの開発アプローチが類似しているため，適用が有効であると考えられている．

一方，一般的なウォーターフォール型開発におけるドキュメントはそれぞれ文脈を持っており，容易にトピックへ分割することはできない．分割するためには，一連のドキュメントに含まれる情報とそれらの関係を分析し，適切なトピックの粒度を決定する必要がある．そこで本論文では，ソフトウェアドキュメントの構造化を行い，適切なトピックの粒度について考察する．

また，DITA に基づくドキュメンテーションでは，特殊化やトピック・マップの執筆など，従来のソフトウェアドキュメンテーションとは異なる作業が発生する．そのため，それらの作業を補助する統合ソフトウェアドキュメンテーション環境を構築することが望ましい．そこで，統合ドキュメンテーション環境に必要な機能の検討も加えて行う．

Task型

```
<task>
    <title>タイトル</title>
    <taskbody>
        <prereq>前提条件</prereq>
        <context>背景情報</context>
        <steps>
            <step>ステップ1</step>
            <step>ステップ2</step>
            ...
        </steps>
        <result>期待される成果</result>
    </taskbody>
</task>
```

図 2　Task トピックの例
Figure 2　Example of Task Topic

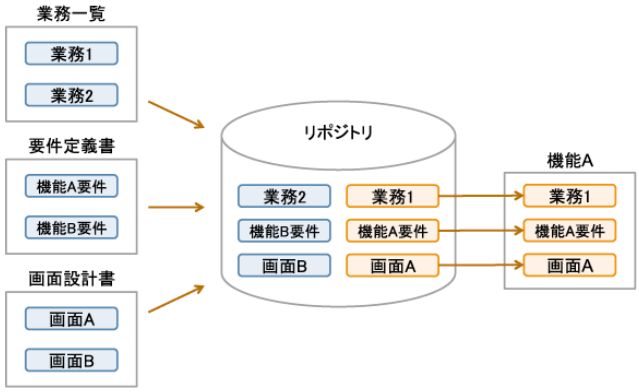

図 3　DITA 適用イメージ
Figure 3　Software Documents Written in DITA

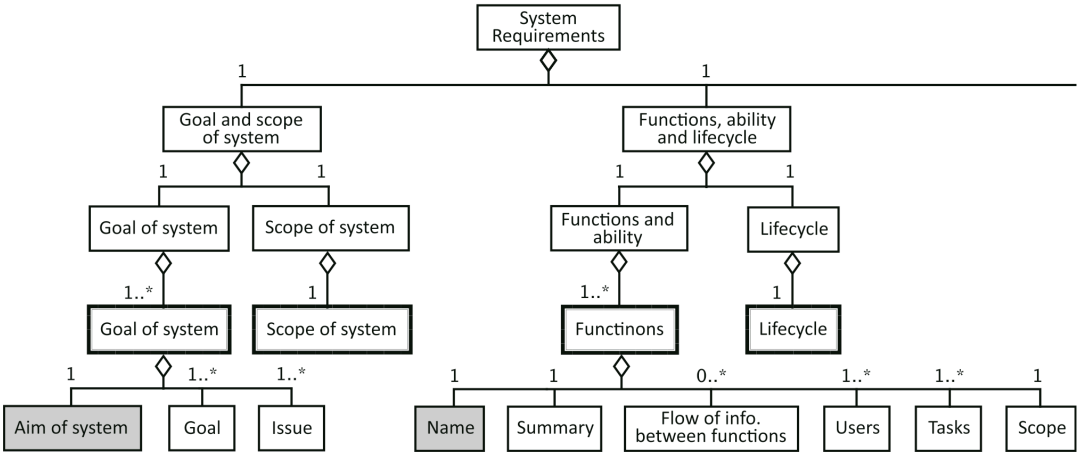

図 4　「システム要件」のドキュメント構造化モデルの一部

Figure 4　A Part of the Structured Document Model about “System Requirements”

4. ソフトウェアドキュメントの構造化

一連のソフトウェアドキュメントの構造化を行い，各ドキュメントに記述される項目とそれらの関係を構造化モデルとして可視化する．その上で，どのような粒度でトピックに分割するべきかを考察する．

4.1 トピックの粒度による影響

DITA では，トピック単位でドキュメントを保存し，それらを目的に応じて組み合わせる．つまり，トピックはある特定の文書内で一度だけ用いられるのではなく，組み合わせを変えて再利用されることを想定している.そのため，トピックは単体で意味を持ち得る最小の単位で作成することが理想とされている.トピックの粒度が細かいほど,様々な文脈上で引用することができ，再利用性が増す.

一方，トピックの粒度を細かくすると，粒度が荒い場合に比べトピック数が増加する．それに伴い，トピック毎の変更履歴やトピック同士の関係性などの管理が煩雑になる可能性がある．従って，全てのトピックを最小単位で作成するのではなく，再利用されない，もしくは荒い粒度でも再利用が可能な項目についてはある程度まとめて扱う必要がある.

そこで，ドキュメントの構造化にあたっては，まずトピックとなり得る最小単位の項目を洗い出す．その後，最小単位でトピックとするべきか，まとめて扱うべきかを再利用性の観点から考察する.

4.2 ドキュメント構造化モデルの事例

ドキュメント構造化モデルを作成するため，まず，既存の構造化モデルの事例の検討を行う．次に，本研究で対象とするドキュメントを定義し，それらのドキュメントについて構造化モデルを作成する.

ドキュメントに含まれる情報とその関係性を可視化するために,構造化を行っている研究事例がある.元山らは，モデルに基づいたソフトウェアレビューでドキュメントの不整合を検出する目的で，ドキュメントの関連をメタモデルで定義している[5]．元山らの定義したメタモデルは設計仕様書を対象としており，業務処理，制約，業務プロセス等,仕様書を構成する要素とそれらの関係を図示している．しかし，元山らのように詳細にメタモデルを定義するためには，ドメインを限定しなければならない．モデルがドメインに依存してしまうと，他のドメインに適用する際にモデルの再検討が必要となる．本研究では，ドメインに依存せず，汎用的に適用できるモデルの作成を目指す.

表 1　対象ドキュメント一覧

Table 1　The List of Target Documents

ドキュメント名	作成プロセス
提案依頼書	取得
プロジェクト管理計画書	供給
利害関係者の要件	要件定義
システム要件	開発
システム方式設計	開発
ソフトウェア要件	開発
ソフトウェア方式設計	開発
テスト手順，テストデータ	開発
利用者文書	開発，運用

4.3 対象ドキュメントの定義

ドメインに依存しない標準的なドキュメントとして，共通フレーム 2007 の主ライフサイクルプロセスにおいて作成が推奨されているドキュメント群から，各プロセスの代表的なドキュメントを抽出した[6]．表 1 に，抽出したドキュメントと，それぞれの作成されるプロセスを示す．これらのドキュメントについて構造化モデルを定義することで,一般的なソフトウェア開発において作成されるドキュメントの大部分の内容をモデル上で可視化することができる．よって，本研究では，これらを対象ドキュメントとする.

図 5　レコードパターン
Figure 5　Record Pattern

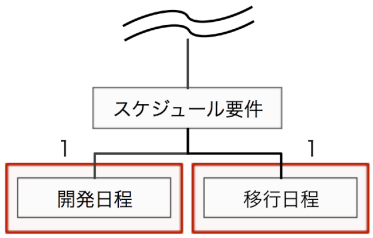

図 6　集合パターン
Figure 6　Aggregation Pattern

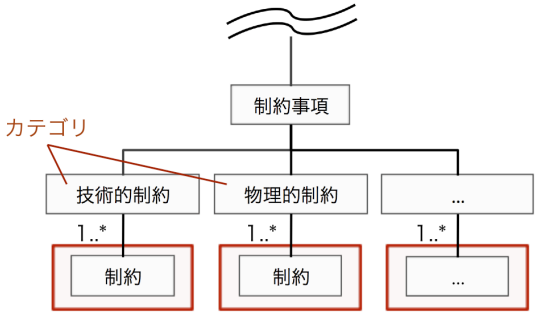

図 7　サブセットパターン
Figure 7　Subset Pattern

4.4 ドキュメント構造化モデルの作成

トピックとなり得る最小単位の項目を洗い出すため，各ドキュメントの構成と記述される具体的な項目をモデルとして可視化する．共通フレーム 2007 では，各ドキュメントに記述すべき項目も定義されている．それらの各項目をノードとし，ドキュメント毎にツリー状の構造化モデルを作成した．また，ノード毎に親ノードに対して一つしか存在しないもの，複数存在するものを区別するため，多重度を記載した．これは，UML のクラス図における多重度と同様の記法を用いている[7].

ただし，共通フレーム 2007 上で記述すべき項目として明記されている項目は，粒度が一定ではない．例えば，「利害関係者の要件」のドキュメントにおける「業務内容」という項目については「手順，入出力情報，組織，責任，権限など」と具体例が記載されているが，「システム要件」のドキュメントにおいては「業務，組織及び利用者の要件」という大きな項目名しか記載されていない．これらの粒度を一定にし，最小単位の項目を明確にするため，さらにノードの細分化を行った．「システム要件」のドキュメントについて，作成した構造化モデルの一部を図 4 に示す．

4.5 ドキュメント構造の分析

ドキュメント構造化モデルにおいて，ソフトウェアドキュメントの文書構造には特徴が見られた．構造化モデルにおける葉ノード同士の関係性から，それらを次の 3 種類に分類した．

- レコードパターン

  レコードパターンは，データベースにおけるレコードのような構造である．兄弟ノードのうちの 1 つをキーとし，その他の兄弟ノードは全てそのキーについての情報を持つ構造となっている．図 5 に例を示す．図のように，「機能名」のノードがキーとなり，「概要」「作業」「実現範囲」などはその機能についての情報となる．

- 集合パターン

  集合パターンは，ある分類に適合する項目の集合となっている構造である．親ノードが集合名，子ノードがそれに含まれる項目名となる．また，レコードパターンとは異なり，兄弟ノード同士は直接関係しない．図 6 に例を示す．図のように，「スケジュール要件」のノードが集合名であり，その集合に含まれる項目として「開発日程」「移行日程」のノードがそれぞれ子ノードとなっている．

- サブセットパターン

  サブセットパターンは集合パターンの特殊な例であり，集合の中にさらにカテゴリ毎の部分集合が存在するものを指す．図 7 に例を示す．図のように，「制約事項」という集合に含まれる各「制約」が，さらに「技術的制約」「物理的制約」といったカテゴリ毎に分類される構造となっている．

5. トピックの粒度についての検討

文書構造の違いによって，情報の再利用性や，分割すべきトピックの粒度にも違いが生じると考えられる．そのため，構造毎に情報の再利用性を検討し，適切なトピックの粒度について考察した．

5.1 トピックの粒度の定義

まず，ソフトウェアドキュメントにおける 3 種類の構造について，それぞれに該当するドキュメントの実例を元に，そのドキュメント内で情報の再利用が発生するかどうか検討を行った．ドキュメントの実例としては，情報処理推進機構ソフトウェア・エンジニアリング・センターが公開している「超上流から攻める IT 化の事例集」を用いた[8].

再利用性の検討の結果から，それぞれの構造毎に適切なトピックの粒度について考察する．本論文で定義する 2 種類のトピックの粒度を図 8 に示す．大トピックは，構造化

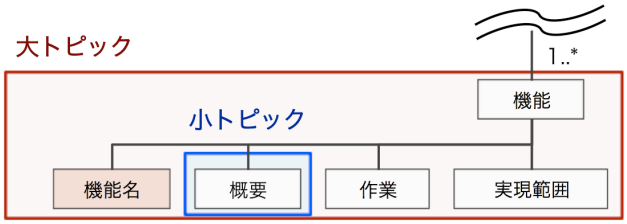

図 8　トピックの粒度
Figure 8　Topic Granularity

モデルにおける複数のノードをまとめて 1 つのトピックとする粒度であり，小トピックはノード単体を 1 つのトピックとする粒度である．

レコードパターンにおいては，大トピックが 1 レコード分の情報にあたり，小トピックが各フィールドの情報にあたる．レコードパターンでは，兄弟ノードを合わせて一件分の情報となるため，基本的にはそれらのノードを全てまとめてトピックとするのが適切であると考えられる．しかし，再利用性を検討した結果，フィールド単位の情報が個別に再利用される場合があることがわかった．それらの項目は，単体でトピックとして独立させておくことが望ましい．従って，レコードパターンでは，大トピックを基本とし，そのうち再利用される可能性のある項目を小トピックとするネスト構造が適していると考えられる．

集合パターンでは，一つのドキュメント内で情報の再利用は発生しないことがわかった．従って，大トピックの粒度が適していると考えられる．ただし，今回は一つのドキュメント内でのみ再利用性の検討を行っているため，さらに他のドキュメントの記述内容と比較することで再利用が発見できる可能性もある．そのため，引き続き検討が必要である．

サブセットパターンでは，各ノード単位で情報の再利用が発生することがわかった．従って，小トピックの粒度が適していると考えられる．

5.2 対象ドキュメントへの適用と評価

次に，「利害関係者の要件」「システム要件」の 2 つのドキュメントに，定義した粒度を適用した．その上で，トピック数や DITA で記述した場合の XML タグ数を試算し，作成した構造化モデルの葉ノード，すなわち最小単位をトピックとした場合との比較を行った．表 2 にトピック数を，表 3 に XML タグ数をそれぞれ示す．表から，「利害関係者の要件」のトピック数は最小粒度の場合と比較して 41%，XML タグ数は 25%減少した．同様に，「システム要件」のトピック数は 80%，XML タグ数は 81%減少した．この結果は，ドキュメント内での再利用性を損なわない範囲でトピックの粒度を大きくすることが可能であり，結果として管理コストを抑制することが可能であることを示している．

ドキュメント別に見た場合，トピック数と XML タグ数ともに，「利害関係者の要件」と比較して「システム要件」の減少率が大きくなっている．これは，パターンの内訳からシステム要件のレコードパターン数が多いためであることがわかる．システム要件ドキュメントでは，その前段階の利害関係者の要件と比較して，記述が整理され，よりデータベースの関係表に近い形式による記述の割合が増加していることを反映している．

表 2　最小粒度と提案粒度におけるトピック数
Table 2　The Number of Topics of Smallest-granularity and Proposed Structurization

ドキュメント	パターン	最小粒度	提案粒度	削減率
利害関係者の要件	レコード	225	130	42%
	集合	62	7	89%
	サブセット	75	75	0%
	合計	362	212	41%
システム要件	レコード	850	160	81%
	集合	78	11	86%
	サブセット	20	20	0%
	合計	948	191	80%

表 3　最小粒度と提案粒度における XML のタグ数
Table 3　The Number of XML Tags of Smallest-granularity and Proposed Structurization

ドキュメント	パターン	最小粒度	提案粒度	削減率
利害関係者の要件	レコード	1490	1125	24%
	集合	383	196	49%
	サブセット	300	300	0%
	合計	2173	1621	25%
システム要件	レコード	5530	900	84%
	集合	428	155	64%
	サブセット	80	80	0%
	合計	6038	1135	81%

6. 統合ドキュメンテーション環境の構築

DITA に基づいたドキュメンテーションでは，特殊化やトピック・マップの執筆など，従来のソフトウェアドキュメンテーションとは異なる作業が発生する．そのため，効率的にドキュメントの作成・管理を行うためには．適切なドキュメンテーション環境を用意することが望ましい．

本研究では，ソフトウェアドキュメントの作成・管理のためのプラットフォームである統合 DITA ドキュメンテーション環境の構築を目指す．開発者による設計書の作成・編集，管理者によるドキュメント全体の構造や執筆状況の管理，その他のステークホルダによるドキュメントの閲覧などを 1 つのプラットフォーム上で行えるようにすることを想定し，必要な機能の検討と，実装方法の検討を行う．

表 4　各フェーズにおいて必要となる機能
Table 4　Function Needed in the Three Domains

フェーズ	機能	概要
編集	特殊化	特殊化したトピック型を作成する.
	XML コンテンツ執筆	DITA のトピックとマップを執筆する.
	レイアウトの編集	出力するドキュメントのレイアウトなどの編集を行う.
管理	関連の設定	コンテンツ同士の関連付けを行う.
	コンテンツの閲覧	画像や外部ファイルを含めた各コンテンツを閲覧する.
	検索	キーワードや関連からコンテンツを検索する.
利用	出力	DITA コンテンツを PDF や XHTML などの形式に変換し，出力する.
	ドキュメント閲覧	出力したドキュメントを閲覧する.

6.1 統合ドキュメンテーション環境としての CMS

多くの技術文書は，コンテンツマネジメントシステム（CMS）によって管理されている．CMS を用いることで，次のようなことが可能となる．

- 複数の開発者が，ドキュメントを介して協調作業を行うことができる．
- 執筆者が分散環境にあっても，ドキュメントの共同編集を行うことができる．
- 多量のコンテンツやそれらの関係を，1 つのリポジトリ内で管理することができる．

CMS の中でも，DITA のトピックのようなコンポーネント単位でのコンテンツ管理を想定しているものを特に CCMS（Component Content Management System） という．DITA に対応した CCMS 製品も存在するが，それらは汎用的なコンテンツ管理を目的としており，トレーサビリティ管理などのソフトウェアドキュメンテーションに必要な機能が整備されていない．そのため本研究では，統合ドキュメンテーション環境としての DITA ベース CCMS の構築を目指す．

6.2　システム要求

DITA に基づいたソフトウェアドキュメンテーションを行うために必要な機能を検討する．樋川は，既存の DITA ツールを，編集，管理，利用という 3 種類のフェーズに分類している[9]．編集フェーズでは，トピックやマップなどの XML コンテンツを作成する．管理フェーズでは，コンテンツ自体やそれらの間の関連を管理する．利用フェーズでは，XML コンテンツからドキュメントを出力し，閲覧する．これらの各フェーズに必要な機能を，既存のツールの持つ機能に基づいて定義した．結果を表 4 に示す．

また，ソフトウェアドキュメントは開発プロセスを通して複数の開発者によって作成・編集されるため，管理者はそれらのドキュメントの状態を CMS 上で確認できる必要がある．従来，このような作業はドキュメンテーションと

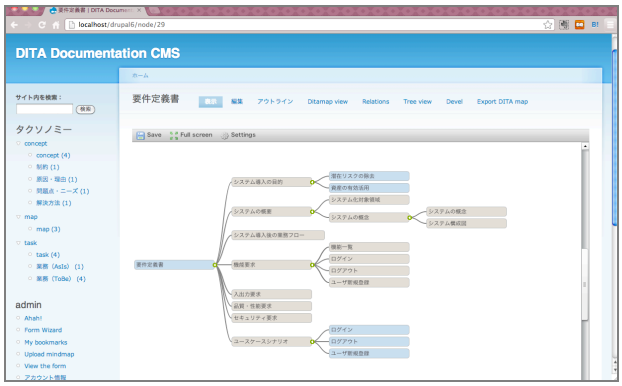

図 9　“DITA integration for Drupal” 画面イメージ
Figure 9　The Module “DITA integration for Drupal”

は分離されており，確認プロセスにおいて誤りや漏れを引き起こす原因となっている．そのため，本研究では，表 4 に加えて次のような機能を実装する．

- 各ドキュメントのトピック執筆の進捗状況を確認する．
- 記述されたトピックやドキュメントのレビュー，承認を行う．

これらの機能により，ドキュメンテーションの進捗状況を定量的に把握する事ができるため，ソフトウェアの生産性と品質向上に繋がることが期待できる．

6.3 DITA ベース CMS の実装

本研究では，オープンソース CMS である Drupal を拡張することで，統合ドキュメンテーション環境としての DITA ベース CMS の構築を行う[10]. Drupal は PHP で記述されたオープンソースの CMS であり，モジュールを作成することで必要な機能を自由に追加することができる．Drupal を用いる理由は次の 4 点である．

- Web CMS である．
- オープンソースで，カスタマイズ可能である．
- Apache と PHP が動作する全てのプラットフォーム上で稼働する．
- DITA コンテンツを作成･管理するためのモジュール “DITA integration for Drupal”が作成されている．

既存モジュール“DITA integration for Drupal”の画面イメージを図 9 に示す．このモジュールは複数のサブモジュールを含んでおり，次のような機能を持っている．

- DITA コンテンツをアップロードする．
- Task トピック，Concept トピックを執筆する．
- マインドマップのインタフェースを用いて, DITA マップの編集を行う．
- DITA コンテンツを PDF2 や XHTML に変換し，ダウンロードする．

これらの機能を持った既存モジュールに基づき, 統合ドキュメンテーション環境としての DITA ベース CMS の構築を行う．現在，プロトタイプの作成を進めており，トピック

同士の関連づけの設定，各マップから参照されているトピックの一括閲覧など，トレーサビリティ管理に必要となる機能の開発を優先的に行っている．

7. 結言

ソフトウェアドキュメントにおけるトレーサビリティの低下，メンテナンスコストの増大，生産性の低下の三つの問題を，DITA を用いて解決する手法を提案した．DITA の適用に向けて, 共通フレーム 2007 から抽出したドキュメントの構造化モデルを作成し，トピックの適切な粒度について検討を行った．また，DITA に基づいたソフトウェアドキュメンテーションを補助する統合ドキュメンテーション環境を構築するため，必要な機能と実装方法の検討を行った．

トピックの粒度の検討の結果，ソフトウェアドキュメントは 3 種類の特徴的な文書構造を持っており，その構造毎にトピックの適切な粒度も異なることがわかった．ドキュメントをそれぞれの構造に適した粒度に分割することで，トピックの管理にかかるコストを削減できることが伺えた．今後は，より広い範囲での再利用性の検討及びトピックの粒度の調整を行っていく．また，必要に応じて，ソフトウェアドキュメントに適したトピックの特殊化を行う．

統合ドキュメンテーション環境については， Drupal の拡張モジュールとして，定義した機能の実装を行う．その後，実装した機能を用いて，ソフトウェアドキュメンテーションにおける DITA 適用の有効性の評価を行う．

謝辞 本研究は科研費(23500127) の助成を受けたものである．

参考文献

1) Comtech Services, Inc. DITA コンソーシアムジャパン訳：DITA 概説書，エスアイビーアクセス(2010).
2) 松島弘幸，小田利彦 : XML 技術を利用したソフトウェア文書理解支援システム CrystalScope の開発：情報処理学会研究報告．DD，98(107)，pp. 1-8(1998).
3) OASIS DocBook Technical Committee: DocBook.org, http://www.docbook.org/.
4) Oscar Diaz, Felipe I. Anfurrutia, Jon Kortabitarte: Using DITA for Documenting Software Product Lines, DocEng'09, pp. 231-240.
5) 元山厚，中谷多哉子：ソフトウェアレビューのための設計仕様メタモデルの提案, 電子情報通信学会技術研究報告, KBSE, 知能ソフトウェア工学 110(305)，pp. 1-6.
6) 情報処理推進機構ソフトウェア・エンジニアリング・センター編：共通フレーム 2007 第 2 版 ～経営者，業務部門が参画するシステム開発および取引のために～，オーム社(2009).
7) 児玉公信：UML モデリング入門 本質をとらえるシステム思考とモデリング心理学，日経 BP 社（2008）.
8) 情報処理推進機構ソフトウェア・エンジニアリング・センター：超上流から攻める IT 化の事例集, http://sec.ipa.go.jp/tool/ep/.
9) 樋川恭平 : XML ベースのコンテンツ管理システムにおける現状と課題，情報処理学会研究報告，Vol. 2010-DD-78(3), pp. 1-6.
10) Official website of Drupal, [Online], Available: http://drupal.org/.