USDM 形式の要求記述の曖昧さを軽減する記述構文パターンの提案とその評価

南 伸二[†1]　宮田 朋人[†1]　山本 稔郎[†2]　中島 毅[†2]

概要：システム開発における要求仕様の記述は自然言語が用いられることが多いが，自然言語の自由度の高さゆえに曖昧さを導入しやすく，また機械的な確認がしにくいため，要求仕様に含まれる抜け・誤り・曖昧さも見逃されがちである．本研究では，要求仕様記述形式の一つである USDM 形式で記述した要求仕様を対象に，曖昧さを軽減するための記述構文パターンを提案し，そのパターンへの適合を確認できるツール機能を開発した．記述実験によりその効果を評価し，記述構文パターンに沿った要求仕様記述を行うことで，要求記述の曖昧さが軽減され，要求の記述品質が改善することを確認した．

キーワード：要求仕様，品質評価，記述ルール

A proposal and evaluation of syntax patterns for disambiguation of requirements description in USDM format

SHINJI MINAMI[†1]　TOMOHITO MIYADA[†1]　TOSHIRO YAMAMOTO[†2]
TSUYOSHI NAKAJIMA[†2]

Abstract: Using a natural language for the description of requirement specifications in system development easily introduces ambiguity because of the high degree of freedom in its description, and leaks and mistakes of requirements also tend to be overlooked because of difficulty to confirm mechanically. In this paper, we proposed and evaluated the description syntax patterns for the requirement specification description in the USDM format. Describing requirement specifications along with the description syntax patterns proved to reduce their ambiguity, and as a result, to improve the quality of requirements.

Keywords: requirement, quality evaluation, description rules

1. はじめに

要求分析は，システム開発の初期段階において，このシステムで何を実現すべきかを明らかにした上で，要求仕様をまとめる作業である[1][2]. 要求分析で作成する要求仕様は，多くの場合自然言語を用いて記述されている．自然言語は，誰でも記述しやすく理解しやすい反面，その自由度の高さゆえに，要求仕様の曖昧さを導入しやすく，また機械的な確認が難しいことが多い．要求仕様の標準的な構成は定められている[3]ものの，要求仕様の抜け・漏れや誤りも見逃されがちである．要求仕様の曖昧さ，抜け，誤りなどの欠陥は，開発の後工程での実装誤り・抜けや試験漏れなどの原因となる．

要求仕様の欠陥を削減し，品質を向上する手法として，要求仕様を自然言語以外で記述する手法や，自然言語での記述様式を限定する手法が提案されている．要求仕様を自然言語以外で記述する手法として，SysML や UML などの構文と意味が定義された図的言語を使って定義しようとする試みがあり，要求仕様の欠陥の予防や検出に大きな効果を出している[4]．しかし図的言語だけではこれまで自然言語で記述していた要求仕様範囲を全てカバーするには至っていない．

自然言語を使った要求仕様記述手法の一つが，USDM（Universal Specification Describing Manner）である．USDM とは，顧客要求からシステム仕様までを同一の階層的な表の様式を用いて記述する手法であり，仕様の抜け・漏れを防止するのに一定の効果がある[5]．しかし，USDM は，様式の中に記述内容を書くルールを提供していないため，記述内容によって曖昧さの問題を引き起こしやすい[6].

本研究では，USDM 形式の記述の曖昧さの課題を解決するために，USDM 形式での要求仕様記述における記述ルール，特に記述構文パターンを定義し，その記述構文パターンにもとづいて要求仕様を記載することで，自然言語での要求仕様記述においても，要求仕様の曖昧さが軽減され，記述品質が改善されることを示す．

2 節では，この研究に先行する関連研究を示し，自然言語で記述された要求仕様の記述の曖昧さを改善する取り組みについて示す．3 節では，USDM 形式の曖昧さを軽減する記述構文パターンを提案し，4 節で，提案した記述構文パターンを評価するために使用したツールについて簡単に述べる．5 節で評価実験の方法および結果について述べ，

†1 SOLIZE Engineering 株式会社
SOLIZE Engineering Corp.
†2 芝浦工業大学
Shibaura Institute of Technology

6 節では，評価実験の結果について考察する．最後に，7 節ではまとめと今後の課題について述べる．

2. 関連研究

自然言語での要求仕様の記述構文パターンについては，これまでいくつかの提案や研究がなされてきた．

2.1 ISO/IEC/IEEE 29148:2011 [7]

ISO/IEC/IEEE 29148:2011 では,「主体」「動作」「条件」「制約」といった，要求を記述する上での基本的な構文要素の例を提示している．ただし，ここでの記述構文要素は，適切に定義された（well-formed）要求の一例として提示されているものであり，主体，対象，制約条件といった記述要素の記述順を定めることを目的としており，この構文要素を使って記述することによって，その中で利用する用語を適切に規定したり，曖昧さを軽減したりすることを目的としたものではなく，また，どの記述構文パターンをどのような状況で使うべきかについても言及されていない. また，原則として英語の要求文書であることが前提となっている．

2.2 EARS [8]

EARS（Easy Approach to Requirements Syntax）は英語で記述する要求文を遍在型（Ubiquitous），契機型（Event-driven），不測型（Unwanted behaviors），状態型（State-driven），選択型（Optional features）の 5 つの記述パターンと，その組み合わせである複合型（Complex）で，簡潔かつ一意性のある要求を記述するための要求記述言語セットである（表 1）．

EARS は英語に特化した記述構文パターンであり，そのまま日本語に対して適用し，評価できるものではない．また, EARS を日本語訳したとしても, 言語特性の違いから，必ずしも有効性が得られない．たとえば，契機型と不測型を日本語として適切に使い分けることは難しく，結果的に EARS の型が適切に分離できない場合がある．

表 1　EARS の記述構文パターン

パターン名	表現する内容	パターン
遍在型 (Ubiquitous)	常に成立する内容	The <system name> shall <system response>
契機型 (Event-Driven)	トリガーとなる契機に対する反応	WHEN <trigger> <optional precondition>, the <systemname> shall <system response>
不測型 (Unwanted Behavior)	望まない契機に対する反応	IF <unwanted condition or event>, THEN the <systemname> shall <system response>
状態型 (State-Driven)	特定の状態における反応	WHILE <system state>, the <system name> shall <system response>
選択型 (Optional Feature)	特定の機能が備わっている場合の反応	WHERE <feature is included>, the <system name> shall <system response>
複合型 (Complex)	組み合わせ	

2.3 車載ソフトウェア仕様記述言語セットの提案[9]

日本語で記述された車載ソフトウェアの仕様において，EARS を拡張した「記述言語セット」を提案し，提案した記述言語セットによって対象とする仕様書のすべての仕様が記述構文パターンに適合することを示している．

車載ソフトウェアを対象とした仕様に適合するために EARS にはない新たな記述構文パターンが追加されており，USDM の要求仕様記述に適用することは想定されていない．また，評価は適合度のみを対象としており，曖昧さに関する評価などは行われていない．

3. USDM に関する記述構文パターンの提案

3.1 USDM と解析対象

USDM 形式とは，要求と仕様を階層構造で捉えることで要求仕様の抜け・漏れを軽減することを目的とした要求仕様の記述形式であり，要求とともにその要求の根拠である「理由」と補足的な情報である「説明」を記述した上で，要求から導出された仕様を記述することがその特徴である．USDM の様式は, 同じ文書の中に要求文, 理由文, 説明文, 仕様文を構成とする階層構造となっている（表 2）．

表 2　USDM 形式の要求仕様書（様式）

要求		(要求文)
	理由	(理由文)
	説明	(説明文)
□□□	仕様番号	(仕様文)

3.2 記述対象に対するパターン

USDM の 4 種類の記述内容について，提案者の記述解説と，実際の記述例を分析することにより，記述すべき内容を決め，記述例から記述パターンを抽出し定義した．

3.2.1 要求文のパターン

要求文の記述内容は，ユーザの視点からシステムとの界面の仕様あるいはユーザからみたシステムが提供するサービスである．そのため，要求文の主語は，ユーザあるいはシステムであるべきである．しかしながら，主語が明記されていなかったり，サブシステムが主語と想定されるような要求文が存在したりすることで，要求している内容が曖昧になってしまう場合がある．言い換えると，要求文についての曖昧さは, ユーザの視点から記述した要求文なのか, システムが提供するサービスを記述した要求文なのかが不明確になることに起因すると考える．そのため，記述構文パターンとしては，ユーザを主語とするパターンとシステムを主語とするパターンの 2 パターンを定義した（表 3）．

表 3　要求文の記述構文パターン

記述内容	記述パターン
ユーザの視点からシステムとの界面の仕様	(ユーザ)は，～できる
ユーザからみたシステムが提供するサービス	(システム)が，～を提供する (してくれる，ほしい)

3.2.2 理由文のパターン

要求文を前節のように定義した場合，理由文で記述されるべき「必要な理由や背景」[10]は，ユーザがその要求を

満足するシステムを利用することによって得られる価値，または，その要求が規定されることになった何らかの制約であると考えられる．そこで，USDM の様式における理由文の記述内容は，利用時の品質[11]，あるいは，制約事項であると定義する．

表 4 に理由文の記述構文パターンを示す．ここで，利用時品質は，システムから得る成果ならび受ける影響，業務上の効果・効率，リスク回避として定義する．制約は，利用する基盤，法律，商習慣，標準への準拠として定義し，記述構文パターンとしては，「(根拠）により」，「〜ため」の 2 つを定義した（表 4)．

表 4　理由文の記述構文パターン

記述内容	記述パターン
①利用時品質： システムから得る成果／受ける影響 業務上の効果・効率，リスク回避	〜を〜するため [(要求目標値の記述）] XX業務の効率を10%向上させるため
②制約： 利用する基盤，法律，商習慣，標準への準拠	（根拠）により，〜ため Xシステムと接続により，通信をXXで行うため

3.2.3 説明文のパターン

説明文の記述内容は，概念の定義，主に新出の名詞・動詞あるいは，要求の理解を助けるものの 2 つである．記述パターンは用語の説明であり，パターンとしては定義せず，自由度を持たせた（表 5)．

表 5　説明文の記述構文パターン

記述内容	記述パターン
①「概念」の定義 （新出の名詞と動詞）	用語：用語の定義 エンドユーザ：当該システムをインターネットを通じて利用するユーザ
②要求の理解を助けるもの	特に定義しない

3.2.4 仕様文のパターン

仕様文の記述内容は，上位の要求を実現するシステムの振る舞いおよび制約事項の 2 つを記述するものと定義する．システムの振る舞いおよび制約事項を記述するための記述構文パターンとしては，コンパクトで網羅性のある最小限のパターンにするために，EARS パターンをベースとした上で，日本語の自然さを失わないように以下の点に留意し，表 6 のように設定した．

- 日本語要求仕様文の条件記述に頻出する「とき」「場合」「ら」「と」に対応できるようにする．
- 契機型と不測型を区別しない（構文レベルで分けることができる簡潔な日本語構文ができなかったため）

表 6　仕様文の記述構文パターン

種別	パターン		
	型	構文	備考
システムのふるまい	遍在型	<システム>は、<応答>する	
	契機/不測型	<システム>は、**〈契機〉とき/場合**、<応答>する	※**〈契機〉**→過去形
	状態型	<システム>は、**〈条件〉間**、<応答>する	※**〈条件〉**→現在形
	選択型	<システム>は、<タイプ別>と、<応答>する	
	組合せ型1	<システム>は、**〈契機〉とき**、**〈条件〉場合**、<応答>する	※**〈契機〉**→過去形, **〈条件〉**→現在形
	組合せ型2	<システム>は、**〈条件1〉かつ〈条件2〉とき/場合**、**〈契機〉と/ら**、<応答>する	※**〈条件1,2〉**→現在形, **〈契機〉**→現在形
制約事項	制約事項	(集合の特性)は，<特徴値・量>とする	

仕様文の曖昧さは，主部に起因するものと，条件部に起因するものがある．主部に起因する曖昧さについては，そのシステムに特有の<応答>の記述方法を除けば，否定語の利用､断定しない表現など文末の表現に関するものであり，それらは 表 6 で対応できていると考えてよい．一方，条件部に起因する曖昧さについては，条件部の記述方法はより詳細にパターン化することができると考えられ，その記述方法を記述構文パターンとすることで，EARS パターンをベースとして条件部の最後の語のみを規定した　表 6 の記述構文パターンよりも，曖昧さを除去できるものと考えられる．そのため，契機/不測型における<契機>の記述，および，状態型における<条件>の記述について詳細に分析した．

<契機>：「when 節」と「イベント」の 2 つに分類．

- 「when 節」は，明確なイベントがなく，物理量，時間などがある閾値を超えた時を表現するもので，常に過去形で表現することとした．
- 「イベント」: 明示的なイベントがある場合を表現するものとし，そのイベントが内部的な処理によるものか，何らかの操作によるものか，時間経過を伴うものか，割り込みによるものかによって記述構文パターンを分類した．

定義した記述構文パターンを表 7 に示す．

<条件>：「状態変数型」と「変数論理型」の 2 つに分類．

- 「状態変数型」: なんらかのオブジェクトの状態変数がある状態値をとることを表現するもので，自明なオブジェクトや状態変数を省略して記述することを許容するように記述パターンを構成した．
- 「変数論理型」: 変数に関して論理的に記述するためのパターンとして定義した．

定義した記述構文パターンを表 8 に示す．

表 7　契機/不測型における
<契機>に関する記述構文パターン

種別		例文
when節		・サーミスタが100℃になったとき
		・水温が110℃を超えた場合
イベント	done型	・PCからの指令受信が完了したとき
	オブジェクト+操作	[「とき」の場合] ブレーキペダルを踏んだとき(能動態), ブレーキペダルが踏まれたとき(受動態)
		[「場合」の場合] ブレーキペダルを踏んだ場合(能動態), ブレーキペダルが踏まれた場合(受動態)
		[「と」の場合] 楽曲名を選択すると(能動態), 楽曲名が選択されると(受動態)
	after型	・蓋センサのON状態が3秒以上続いたとき
		・ブレーキペダルを踏んでから5秒後
	内部イベント(割込)	・ハードウェアエラーが生じたとき

表 8　状態型における<条件>に関する記述構文パターン

構文	例文
状態変数型	カーナビの**検索画面**がアクティブ状態である場合
	カーナビが起動中のとき
	シートベルトを装着していない場合
	運転モードが冷房のとき
	動画が表示中であるとき
変数論理型	(室内温度≧設定温度+1)である間
	室内温度より1℃以上高い場合

4. ツール実装

4.1 SE Suite

SE Suite は，スペインの The REUSE Company によって開発された要求文書の品質向上を目的としたソフトウェアである[12]．自然言語で記述された要求文書が期待する品質評価基準に沿った記述になっているかどうかを，語句，記述構文パターンなどの面から解析し，要求文の問題点と品質評価結果を出力したり，得られた問題や品質評価結果，記述構文パターンをもとに要求文を編集・修正したりすることができる（図 1）．品質評価基準は，INCOSE のガイドライン[13]をベースとしたものが標準であるが，利用者独自のものを利用することも可能である．日本語で記述された要求文書にも対応しており，INCOSE のガイドラインの内容を日本語での評価基準として評価することも実現している[14].

4.2 USDM に関する提案パターンの実装

記述構文パターンは，文全体または一部における語順を規定するものである．SE Suite において，記述構文パターンにおける語句の指定は，特定の語句，品詞の種類，または，別途定義した語句の集合を構文に沿って配置することで定義することができる. また，記述構文パターンは階層化することもできるため，記述構文パターンの一部を他の記述構文パターンによって表現することも

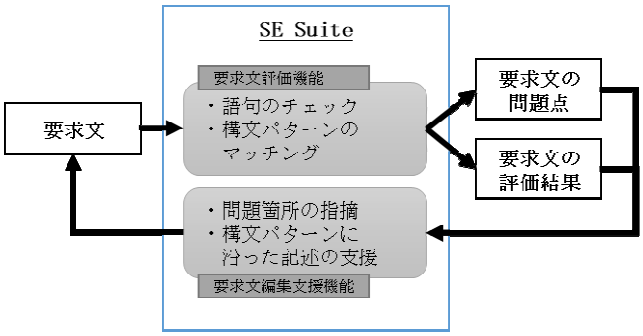

図 1 SE Suite

可能である．なお，評価，編集支援を実現するには，事前に用語や用語の関係性，記述構文パターンの定義が必要である．

図 2 に，状態型の記述構文パターンについて，SE Suite を用いて定義した例を示す. 各部について以下に説明する．

<オブジェクト>はあらかじめ設定しておいたオブジェクトとして利用可能な用語の集合，「は」は助詞の「は」のみ，「記号-読点/SYMBOL-COMMA」は読点を品詞の一種として指定している．このように単独の語句だけ許容する箇所や語句の集合全体を許容する箇所を定義することで，パターンの中で厳密に指定したい部分，自由度を持たせたい部分を柔軟に指定することが可能である．

次の［条件］は，表 8 で定義した条件に関する記述パターンをパターンの一部として指定している．他にも，［応答（繰り返し）］では，応答が複数ある場合に応答に関するパターンが何度か現れることを許容する指定をしている．また，指定の一部をオプションにして指定した用語や品詞があってもなくてもよい，というパターンをつくることもできる．このように，サブパターン，繰り返し，オプションを使うことで，パターンを厳密に指定しながらいくつかの選択肢を提示することが可能になる．

5. 評価実験

5.1 実験目的

本研究における評価実験の目的は，定義した記述構文パターンを用いて要求仕様を記述することができるかどうかを確認した上で，変換によって要求仕様の曖昧さを軽減できるかどうかを評価することである．この評価実験は，SE Suite を利用して行った．

5.2 実験対象

本評価実験では，①「話題沸騰ポット（GOMA-1015 型）

図 2 状態型の記述構文パターン

要求仕様書 第 7 版」[15]のうち USDM 形式の部分（2 章～5 章），②「LED キューブ装置 要求仕様書（USDM 版）」[16]，および，③弊社で独自に作成した EPB 要求仕様書を実験対象とした．なお，今回定義した要求文，理由文，説明文の記述構文パターンは文型の大枠の規定にとどまっており曖昧さの軽減に大きくは寄与しないと判断し，評価対象としては，仕様文のみとした．

5.3 実験 1 仕様文のパターン適合度評価

(1) 実験手順

「各要求仕様書において，意味を変更することなく記述構文パターンに沿った記述形式に変換できる仕様文の割合」を評価することで，定義した記述構文パターンの適用可能性を評価した．各要求仕様書に含まれる仕様文が定義した記述構文パターンのいずれかに合致し得るかを分類した上で，SE Suite の記述支援機能を用いて，仕様文を記述構文パターンに沿った記述形式に変換した．

(2) 評価方法

SE Suite において，記述構文パターンに合致しているかどうかを評価する品質評価基準を定義し，各要求仕様書に含まれるすべての文に対して，変換後でそれぞれいくつの文が記述構文パターンに合致させることができたかを計数した．

(3) 実験結果

各要求仕様書①～③に対して，文書に含まれる仕様文の数に対して，変換後に記述構文パターンに適合させることができた仕様文の数は表 9 のようになる．参考までに変換前の段階で記述構文パターンに適合している仕様文の数も掲載した．

表 9　実験 1 の評価結果

対象	仕様文数	変換前適合数	変換後適合数
①	57	0	35
②	65	22	50
③	17	0	14

5.4 仕様文の曖昧さ除去度合いの評価

(1) 実験手順

実験 1 において，変換が行われた仕様文に対し，変換前の仕様文と変換後の仕様文を比較して，変換前にあった仕様の曖昧さが変換後の仕様文，すなわち，記述構文パターンに沿った記述がなされた仕様文において，改善されているかどうかを評価した．

(2) 評価方法

この評価実験において，要求文書の曖昧さとは，要求文書に複数の解釈が可能になることであると定義する．すなわち，今回評価するのは，記述構文パターンを利用して仕様文を変換することで仕様文の解釈の仕方が削減されることを確認することである．この評価は要求分析の経験がある単独の開発者のレビューによって行った．

(3) 実験結果

各要求仕様書①～③に対して，変換後に記述構文パターンに適合させることができた仕様文に対して，曖昧さが除去されたと判断できた仕様の数は表 10 のようになる．

表 10　実験 2 の評価結果

対象	変換後適合数	曖昧さが除去された仕様数
①	35	23
②	50	9
③	14	7

6. 結果の分析と考察

6.1 実験 1 の結果の分析

各要求仕様書における仕様文は，変換前から記述構文パターンに適合している仕様文、変換によって記述構文パターンに適合することができる仕様文，記述構文パターンに適合するように変換することができない仕様文に分類できる．ここでは，特に変換前から記述構文パターンに適合している仕様文、記述構文パターンに適合するように変換することができない仕様文について分析する．

変換前から記述構文パターンに適合している仕様文は，要求仕様書②に存在する．これらの仕様は，すべてが遍在型，制約事項型のいずれかに該当するものである（表 11）．遍在型は，主語と応答部分があればよいため，適合しやすいものと考えられる．同様に，制約事項型は，主語と特徴値または特徴量さえあればよく，一般に主語を明示しないと文が成立しないため主語が省略されることも少なく，適合しやすいものと考えられる．

表 11　変換前に適合した仕様文の内訳

対象	変換前適合仕様文		
	遍在型	制約事項型	その他
②	9	13	0

記述構文パターンに適合するように変換することができない仕様文は，すべての要求仕様書に存在する．これらは，記述構文パターンに適合するように変換を適用しようとしても定義した記述構文パターンでは変換が困難なものである．それらを分類すると以下のようなものがあることがわかる．それぞれ，記述構文パターンとして追加すべきかどうかについても検討した．

(1) 無条件型

仕様文「操作量を 0%とする．」を例に挙げると，いつい

かなる時でも操作量を 0%とすることを期待しているわけではないと考えらえるため,遍在型ではないにも関わらず,仕様文そのものには操作量を 0%とする条件が記載されていない．そのため，記述構文パターンには適合しないと判断できる．

このような無条件型の記述は，USDM において，連続する仕様文が処理の一連の流れを示す仕様であることを前提に記述したり，仕様文をグループ分け[17]する際に仕様の条件ごとにグループ分けしたりする場合に見られる．連続する仕様文が一連の流れを示すかどうかは明確ではないし，グループ名に条件が書かれていても，見過ごしやすい．そのため，条件にあたる部分は，仕様文本文中に明記することで，曖昧さが削減されると考えられる．すなわち，無条件型の仕様文は曖昧さの軽減の観点からは記述されるべきではなく，記述構文パターンとしては追加されるべきではないと考える．

(2) 否定応答型

仕様文「操作パネルの温度／モード表示窓に，サーミスタの温度の数値は表示しない.」を例に挙げる．この仕様文は，上記の無条件型にも分類できるだけでなく，応答部分が否定文で記述されていることにより，記述構文パターンに適合しないと判断できる．

否定文の要求は，解釈の可能性を増やすことが多い[18]．上記の例では,「何か別の数値を表示するのか，それとも何も表示しないのか」「他の箇所に表示するのか」といった多くの疑問が湧き上がってくる．また，否定文の要求は，有限時間内に検証することが難しいという課題もある[19]．このように，否定文の要求は，曖昧さの軽減の観点からは記述すべきではなく，記述構文パターンに追加すべきではない．

(3) 括弧を利用した補足説明型

仕様文「タイムアップした（残り時間が 0min0sec にカウントダウンでなった）場合，ブザーを 100msec 間隔で 100msec を３回鳴らす.」の場合,「タイムアップする」という内容を「残り時間が 0min0sec にカウントダウンでなった」という括弧の中の文で補足説明している．これは括弧の部分を削除したり，逆に括弧の中の内容のみを記述したりすることで，記述構文パターンに沿った記述にすることは可能であるが，仕様文の記述者以外にはどちらを残すべきか判断が難しいため，記述構文パターンには適合しないと判断した．また，括弧を利用した補足を許容したとしても曖昧さを軽減することはなく，記述構文パターン構文パターンとして追加すべきではない．

(4) 複雑な契機・条件やその組み合わせを持つ型

仕様文「水の温度特性により決定される比例係数 Kp，微分係数 Kd，積分係数 Ki を使って，以下の式で時間 t0 における操作量 M（%）を計算する．この数式を展開すると，前回の操作量 M1 と今回の操作量 M0 の差⊿M（%）は，⊿M = M0 - M1 = Kp （T1 - T0） + Ki （Tg - T0） + Kd （2 T1 - T0 - T2）で表わされるので，今回の操作量 M0 は，M0 = M1 + ⊿M （ただし，0≦M0≦100%）となる.」や「以下のいずれかの状態となった時，沸騰行為を止める.・エラーを検知した時（５章の「エラー検知」を参照.）・蓋センサ off・全ての水位センサが off」のように，構文が複雑で記述構文パターンに適合しないものがあることも分かった．複雑な契機・条件は仕様文としては必要であり，組み合わせの記述構文パターンとして追加すべき内容である．ただし，どのような記述構文パターンを定義すべきかは，今後検討する必要があるだろう．

上記以外にも，意味内容が不明瞭であるため記述構文パターンに当てはめることができないものなど，上記の分類に当てはまらないが、記述構文パターンに適合させることができない仕様文も存在した（表 12）．

表 12　記述構文パターンに適合するように変換することができない仕様文の内訳

対象	変換不可能仕様文				
	(1)	(2)	(3)	(4)	その他
①	1	2	2	15	2
②	2	0	0	6	7
③	0	0	0	3	0

なお,よく見られる構文として「タイマボタンを 100msec 以上押される度に」のようにイベントが繰りされることを表現していると考えられるものがある．本研究では，契機の一つとして「タイマボタンを 100msec 以上押されると」のように変換しており，これでも意味の変化はないと考えられる．

6.2 実験 2 の結果の分析

各要求仕様書において，記述構文パターンにより曖昧さが軽減されていることが分かる．②の仕様書については，曖昧さが除去された仕様文の割合が極端に少ないが，これは②において記述構文パターンに適合した仕様文は遍在型や制約事項型が多いことによるものである．すなわち，遍在型では明らかな主語の補完によって記述構文パターンに適合するものが多く，制約事項型では語順の変更や動詞の補完によって記述構文パターンに適合するものが多い．これらは，解釈の幅を狭めることにはつながらないため，曖昧さの軽減にはつながらないことになる．

曖昧さの評価は，各仕様文の変換前，変換後を一つ一つ確認して，複数の解釈の可能性が削減されているかどうかを確認している．実際に変換し，曖昧さが軽減された例をいくつか示す．

(1) 仕様文「コンセントに初めて繋いで直ぐは，一度アイ

ドルとなる.」

この仕様文を記述構文パターンに適合するように変換すると「電源プラグをコンセントに繋ぐと，アイドルとなる.」となる．この仕様では，「コンセントに繋ぐ」ことが「操作」であることが想定できるので，イベント型の「オブジェクト＋操作」に該当すると判断できる．常識的に考えれば, コンセントに繋ぐのは電源プラグであることから，対象となるオブジェクトは「電源プラグ」であると想定する．

変換前の仕様では，「初めて」という語を使うことにより．電源プラグをコンセントに繋いだ当初という意味合いを付与しようとしたのかもしれないが，一回目にコンセントを繋いだ場合のみアイドル状態となることを意図しているようにも読める．例えば，二回目以降にコンセントを繋ぐ場合は，以前の状態を記憶しており記憶した状態になることを想定しているなどである．また,「直ぐは」という語からは, コンセントに繋いだ直後のみアイドル状態になり，その後アイドル状態でない状態に切り替わる，という時間範囲について言及しているようにも，コンセントに繋いだ直後にアイドル状態になる，という「イベント」について言及しているようにも解釈が可能である．

一方で, 変換後の仕様では, コンセントを繋ぐという「イベント」により，アイドル状態になることを期待していることが明確である．

(2) **仕様文「蓋センサが on になって 3sec 経過した時点で，満水センサが on を検出した時，このポットの許容上限を超えていると判断する.」および「蓋センサが on になって 3sec 経過した時点で，全ての水位センサが off を検出した時，このポットは空と判断する.」**

この仕様文を記述構文パターンに適合するように変換すると，それぞれ「ポットは，蓋センサの ON 状態が 3 秒以上続いているかつ満水センサが on 状態であるとき, ポットの許容上限を超えていると判断する.」および「ポットは，蓋センサの ON 状態が 3 秒以上続いているかつ全ての水位センサが off 状態であるとき，空であると判断する.」のようになる．もとの仕様文は，仕様文に記述された最初の条件の時点なのか，2 番目の条件の時点なのか，その両方が成立したときだけなのかが曖昧である. そして, 実際には，最初の条件は，蓋センサが on になって 3sec 経過した時点以降であればよい，というのが記述したかった内容でないだろうか.

この仕様の記述に関する課題は，「時点で」「時」という時間に関する類似表現が複数あるにもかかわらずその整合が取れていないことである．組み合わせの記述構文パターンに定義したように，時間に関する表現を限定し，それ以外は「場合」や「と」のような表現で，状態，契機を表現することで，曖昧さが軽減される．

6.3 考察

実験手順において，要求文を記述構文パターンに沿った形式に変換する際に，変換が困難な場合があることを指摘した．変換が困難な理由をいくつかの型に分類して説明したが，無条件型と否定型は必要な情報の欠如が原因で記述構文パターンへの変換が困難になっているもの，括弧を利用した補足説明型は, 適切な用語の定義がなされておらず, 仕様文中で用語の定義もあわせて行っていることにより，仕様文がわかりにくくなっているものと考えられる．前者は，適切な情報が要求仕様書に記述されていれば変換可能であり，後者は，用語の定義が適切になされていれば変換可能になる．そのことを考慮すれば，今回定義した記述構文パターンは大部分の仕様文に対して適用可能であると言える．

また，本研究で実施した変換は，実際にはいくつかの課題がある．課題の一つは，当初の要求文の曖昧さが記述の変換によって軽減されているということは，実際には解釈の幅を狭める選択がこの手順において行われているということであり，意味の変更が行われている可能性がある，ということである．二つ目の課題は，その意味の選択が作業者の主観に依存してしまう可能性があり，変換の作業者が要求仕様を記述した者と異なる場合に，その解釈の違いが生じうるということである．これらの課題は，実際の開発における要求分析プロセスにおいては，要求仕様を記述した者が今回定義した記述構文パターンを使って当初から要求仕様を記述することであらかじめ曖昧さを軽減した形での要求仕様を記述したり，変換作業をした上で要求仕様の記述者からレビューを受けたりすることにより，要求仕様の曖昧さの軽減を実現することができると考える．

また，本研究で仕様文の記述構文パターンとして定義した記述構文パターンを使って記述すると，日本語としての不自然さは多くの場合解消されるが, 組み合わせ型の場合, 不自然さが増してしまうものが存在した．日本語としての不自然さがあると，それだけで解釈の幅が広がってしまう可能性があるため，さらに日本語として自然な記述構文パターンを検討していく必要があるだろう．

本研究において定義した記述構文パターンへの変換ができなかった文も存在するため，それらに対する記述構文パターンの定義も必要になる. とりわけ仕様文に関しては, 実際の開発現場ではより詳細な内容を複雑に規定するような仕様が記載されていることもあり，そこに多くの曖昧さが含まれることも少なくない．これらの記述の曖昧さを軽減する記述構文パターンを定義することができれば，より曖昧さの少ない要求仕様を記述することが可能になると考える．

また，今回評価対象としなかった要求文，説明文，理由文に対して，より詳細な記述構文パターンを検討することで評価可能にすることも考えられる．ただし，USDM 形式

の要求仕様において，仕様文と比較して，要求文，説明文，理由文の記述内容は記述の幅が大きくなることが考えられるため，記述構文パターンの検討と並行して，書くべき内容の詳細化，検討なども考慮に入れる必要がある．

7. まとめと今後の課題

今回の評価で，USDM 形式の要求仕様についても記述構文パターンを定義し，要求仕様を記述構文パターンに沿って記述することで，要求仕様の曖昧さの軽減を実現できることが確認できた．

今後の課題として，図的言語と本研究の手法との比較がある．USDM 形式の要求仕様書であっても，状態遷移図などの図的言語を一部に取り入れて記述することが，実際の開発現場では数多く行われている．図的言語を採用する目的は，曖昧さの軽減や視認性の向上がその目的であると考えられるが，本研究の手法を用いれば，図的言語の一部を自然言語での記述で置き換えることも可能かもしれない．自然言語での記述の方が機械的な処理に向いていることも多く，これにより，要求仕様の更なる品質向上を実現できる可能性もある．

また，様々な記述構文パターンを詳細に定義した場合，記述構文パターンに沿って要求仕様を記述することが煩雑な手続きになってしまう可能性がある．利用すべき記述構文パターンを選択し，記述構文パターンに沿った文章を記述することを容易にするためには，その作業を支援するソフトウェア等の役割が重要になってくる．今回活用した SE Suite は記述構文パターンを事前に選択すれば，記述構文パターンに沿った要求仕様記述を支援する機能があるが，要求仕様の曖昧さをさらに軽減するためには，さらなる支援機能を持ったソフトウェアの検討および開発も進めていくことも重要な活動であろう．

参考文献

[1] ISO/IEC/IEEE 29148:2011, Systems and software engineering —Life cycle processes—, 6.1 Requirement processes.
[2] ISO/IEC 12207:2008, Systems and software engineering — Software life cycle processes, 6.4.2 System requirements analysis process.
[3] ISO/IEC/IEEE 29148:2011, Systems and software engineering —Life cycle processes—, 5.2.4 Requirements construct.
[4] INCOSE, INCOSE SYSTEMS ENGINEERING HANDBOOK, 9.2 MODEL-BASED SYSTEMS ENGINEERING, 2015.
[5] 清水吉男.【改訂第 2 版】[入門+実践] 要求を仕様化する技術・表現する技術. 技術評論社, 2010, p160.
[6] 宮本陽子. 形式仕様記述言語の利用による仕様書の改善.株式会社メタテクノ, 2011.
[7] ISO/IEC/IEEE 29148:2011, Systems and software engineering —Life cycle processes—, 5.2.4 Requirements construct.
[8] Marvin, Alistair, et.al, “Easy approach to requirements syntax (EARS)”, Requirements Engineering Conference 2009. RE’09. 17th IEEE International. IEEE, 2009.
[9] 小林展英, 岡本惇一郎, 岡戸真一郎. EARS を拡張した車載ソフトウェア使用記述言語セットの提案. 人工知能学会 第 17 回知識流通ネットワーク研究会, 2017.
[10] 清水吉男.【改訂第 2 版】[入門+実践] 要求を仕様化する技術・表現する技術. 技術評論社, 2010, p170.
[11] ISO/IEC 25010:2011, Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)-System and software quality models
[12] Fraga, Anabel, et al., "Ontology-assisted systems engineering process with focus in the requirements engineering process." Complex Systems Design & Management. Springer, Cham, pp. 149-161, 2015.
[13] INCOSE, Guide for Writing Requirements, 2017.
[14] 佐野町友紀, 中島毅, 南伸二, 疋嶋英理子, 日本語要求仕様の品質向上のための記述ルールデータセットの提案と評価, 情報処理学会研究報告, 2016.
[15] 話題沸騰ポット GOMA-1015 型　要求仕様書 第 7 版,組込みソフトウェア管理者・技術者育成研究会（SESSAME), http://www.sessame.jp/workinggroup/WorkingGroup2/POT_Specification_v7.PDF
[16] LED キューブ装置 ソフトウェア要求仕様書（USDM 版）, https://www.cqpub.co.jp/interface/download/2011/03/SpecOfLedcube_Spec_USDM_v1.PDF
[17] 清水吉男.【改訂第 2 版】[入門+実践] 要求を仕様化する技術・表現する技術. 技術評論社, 2010, p234.
[18] 清水吉男.【改訂第 2 版】[入門+実践] 要求を仕様化する技術・表現する技術. 技術評論社, 2010, p260.
[19] INCOSE, Guide for Writing Requirements, 2017, 4.3.5.