Vol.2019-SE-201 No.9
2019/3/7

自然言語で記載された仕様書からのテストケース自動生成アルゴリズムの構築

村上響一[†1]　青山裕介[†1]　村上神龍[†1]　久代紀之[†1]

概要：システム開発において，テスト設計者は自然言語で記載されたシステム仕様書をもとにテスト仕様書を作成する．テスト仕様書の作成は，システム仕様書の文章を条件文・動作文に分けるという機械的な作業と，記述漏れの補完や論理関係の曖昧さの修正など知識・経験に依る作業がある．テスト設計者は手作業で数百にも及ぶテストケースの作成を行うため，膨大な時間と人為的ミスが発生する．また，テストケースの設計プロセスが暗黙的であるためレビューを行う際，テストケース作成の根拠が分かりにくいという課題があった．

本研究ではテスト設計者へのヒアリング結果をもとにこれまで暗黙的であったテスト設計作業をプロセスとして定義した．さらに，左記プロセスを構成する各ステップの作業を支援するアルゴリズムを開発した．アルゴリズムにより出力した各ステップのレビューを階層的に行うことで，記述漏れの修正・論理関係の修正など段階的にレビューを行うことを可能とした．これらプロセスと一連のツール群の開発により，自然言語で記載された仕様書から最終的なテストケース（ディシジョンテーブル）を生成することを試行した．

1. はじめに

システム開発において，テスト設計者は自然言語で記載されたシステム仕様書をもとにテスト仕様書を作成する．テスト設計者は手作業で数百にも及ぶテストケースの作成を行うため，膨大な時間と人為的ミスが発生する．テスト仕様書の作成は，システム仕様書の文章を条件句・動作句に分けるという機械的な作業と，記述漏れの補完や論理関係の曖昧さの修正など知識・経験に依る作業がある．また，テストケースの設計プロセスが暗黙的であるためレビューを行う際，テストケース作成の根拠が分かりにくいという課題があった．

本研究では実際の企業に所属するテスト設計者複数にヒアリングを行い，これまで暗黙的であったテスト設計作業をプロセスとして再定義した．さらに，左記プロセスを構成する各ステップの作業を支援するアルゴリズムを開発した．アルゴリズムにより出力した各ステップのレビューを階層的に行うことで，記述漏れの修正・論理関係の修正など段階的にレビューを行うことを可能とした．これらプロセスと一連のツール群の開発により，自然言語で記載された仕様書から最終的なテストケース（ディシジョンテーブル）を生成することを試行した．

本論文では，提案プロセスの起点となるセミ形式記述を出力するアルゴリズムの構築及び実装，その評価を行う．

以降，2章にて提案するテストケース作成プロセス，3章にて自然言語からテストケースであるセミ形式記述への変換アルゴリズム，4章にてアルゴリズムの実装，5章にて構築アルゴリズムに対する評価実験，6章にて実験結果に対する考察，7章にて本論文のまとめと今後について述べる．

2. 提案するテストケース設計プロセス

1章で述べた課題を解決するためのテスト設計プロセスを図 1 に示す[1]．提案するテストケース設計プロセスでは，これまでテスト設計者によって暗黙的に行われていた記述漏れの修正・論理関係の修正・前提漏れの修正の3つ（図 1 の1.2, 2.2, 2.3）をテスト設計におけるレビューのステップとして設ける．各ステップでは修正・レビューの

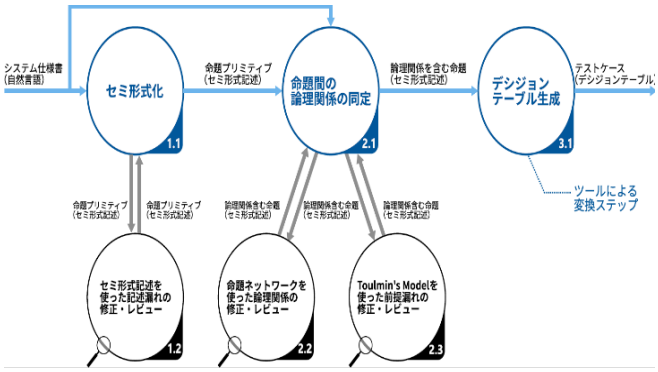

図 1　提案するテストケース設計プロセス[1]

目的に応じたアウトプットに対してレビューを行う．左記修正ステップを経ることで，もとの仕様書記述に含まれる誤りや曖昧性が補完されたテストケースの作成を行うことができる．また，レビュー対象であるアウトプット及びテストケースであるディシジョンテーブルは構築アルゴリズムにより自動的に生成する[1][2]．これにより，テストケース作成時間の短縮と人為的ミスの発生を防ぐ．

2.1 セミ形式記述

図 1 提案プロセスにおけるステップ1.1では，システム仕様書に記載された自然言語記述を「関係詞（主体，対象，制約）」と形式化するセミ形式記述[3][4]に変換する．ステップ1.2では，セミ形式記述を用いてシステム仕様書に記載された自然言語記述の欠落情報の修正を行う．日本語による自然言語記述では，以下のような曖昧性が含まれる．

1. 主語・目的語の省略
2. 表記揺れなど記述の不統一性

テストケースに曖昧性が含まれると手戻りのコストが大きくなるため，テストケース作成時には曖昧性は排除しなければならない．自然言語記述では主語や目的語の漏れや省略がしばしば発生する．レビュー時にセミ形式記述の「??」の箇所を埋めることで，自然言語記述によって生じる曖昧性を修正することができる（図 2 ）．また，動作を表す関係詞の修正・統一によって，表記揺れを統一することができる．

†1 九州工業大学
Kyushu Institute of Technology.

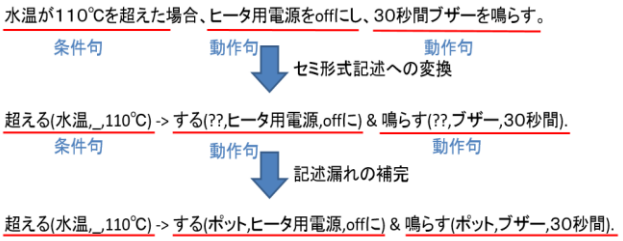

図 3　セミ形式記述例

セミ形式記述における「関係詞（主体，対象，制約）」の単位を命題プリミティブと定義し，命題プリミティブ間の論理関係は論理演算子（Not, And, Or, Imply）を用いて表現することができる．これにより，自然言語記述における条件・動作・論理関係をセミ形式記述上でも表現することができる．

セミ形式記述の条件句命題プリミティブがテストケースの条件，動作句命題プリミティブがテストケースの期待動作に対応する．また，セミ形式記述の関係詞・主体・対象がテストケースの項目，セミ形式記述の制約がパラメータに対応する．左記ルールに則り，セミ形式記述を入力として，テストケースであるディシジョンテーブルを作成することができる．

2.2 命題ネットワーク

図 1 提案プロセスステップ2.1では，ステップ1.1及びステップ1.2で欠落情報の修正を行ったセミ形式記述を論理関係可視化手法である命題ネットワーク[5]に変換する．ステップ2.2では，命題ネットワークを用いて論理関係の同定を行う．自然言語記述の仕様書では，命題間の論理関係に以下の欠陥が含まれる．

1. 命題間の論理関係の記述が誤っている
2. 命題間の論理関係に複数の解釈が存在する

システム仕様書に記載された自然言語を変換したセミ形式記述においても上記欠陥が含まれる．論理関係の把握を行う際，セミ形式記述のままでは並列に作用するAndやOrの構造をテスト設計者とレビュアーが暗黙的に理解するため，両社の間で論理関係の認識の違いが発生することがある．論理関係を可視化する命題ネットワーク（図 3 ）によるレビューを行うことで論理構造暗黙性により生じる齟齬を解消する．

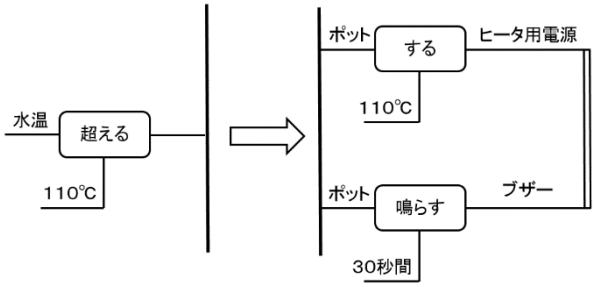

図 4　命題ネットワーク例

2.3 Toulmin's Model

システム仕様書において，記載されている機能には暗黙的な前提が含まれていることがある．これはテスト設計者の製品ドメイン知識や経験によって補完さる．従来のレビューでは，レビュアーはテスト設計者によって暗黙的な前提を補完された結果のみを確認するため，補完の漏れや誤りを発見することが困難であった．このため，ステップ2.3では，Toulmin's Model[6]という命題が成り立つための枠組みを利用することで前提の漏れの可視化及び発見の支援を行う（図 4 ）．

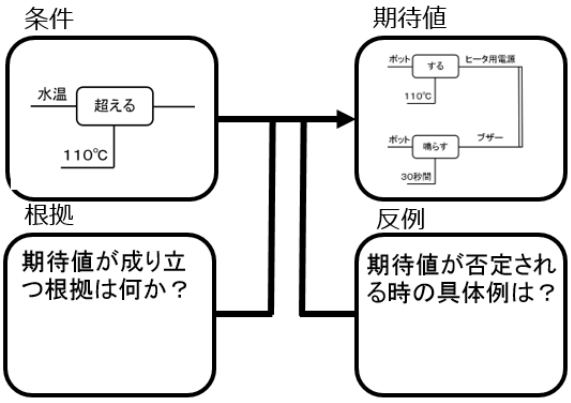

図 2　Toulmin's Model 例

2.4 ディシジョンテーブルテスト

図 1 提案プロセスのステップ3.1では，1.1及び2.1のステップを経て記述漏れ・論理関係の誤り・前提漏れの修正がされたセミ形式記述を入力としてテストケースであるディシジョンテーブル[7]を生成する．このディシジョンテーブルは作成ツールにより自動的に生成することができるため，システムに対するテストはこのディシジョンテーブルの項目を確認する作業となる．ディシジョンテーブル例を表 1 に示す．

命題		テストケース	
		ケース1	ケース2
条件1	超える(水温,_,110℃)	T	F
動作1	する(ポット,ヒータ用電源,offに)	T	ー
動作2	鳴らす(ポット,ブザー,30秒間)	T	ー

表 1　ディシジョンテーブル例

3. セミ形式記述変換アルゴリズム

提案プロセスの各ステップの変換はセミ形式記述が入力となっている．セミ形式記述はシステム仕様書に記述された自然言語を入力として変換される．本章では自然言語記述をセミ形式記述に変換するために構築したアルゴリズムについて述べる．

3.1 構文解析

セミ形式記述への変換には，自然言語処理を用いる．自然言語処理は大きく分けて次の4つのステップに分けることができる[8][9].

1. 形態素解析：文章を単語に分割する．単語が語形変化している場合は，原型へ戻す．単語の品詞を決定する．
2. 構文解析：単語間の構文的関係を決定する．
3. 意味解析：単語，文の意味を決定する．
4. 文脈解析：複数の文にまたがる処理を行う．

本研究では，文章の形態素解析を形態素解析ツールjuman[10]，構文解析を構文解析ツールknp[11]により行う．

3.2 セミ形式記述変換アルゴリズム概要

日本語における自然言語記述は図 5 のように階層的な構造をしている．文は句の単位，句は文節の単位，文節は詞と辞の単位に分けられる．句はそれぞれテストケースにおける条件または動作の性質を持ち，句の単位はセミ形式記述の命題プリミティブの単位に相当する

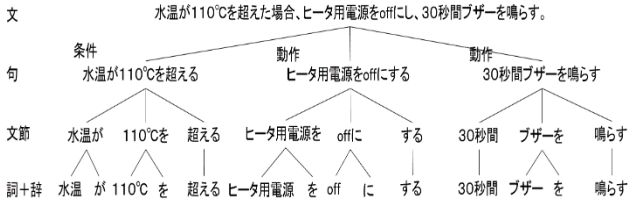

図 5　文の構造例

構築アルゴリズムでは，句の判定，句間の倫理関係の同定を行い，句ごとに命題プリミティブへの変換を行うことでセミ形式記述へ変換する．句の判定，句間の論理関係同定，句の命題プリミティブへの変換はそれぞれ構文解析結果を用いたルールベースで行う．に自然言語記述を入力としてセミ形式記述に変換するアルゴリズムのフローチャートを示す（図 6 ）．

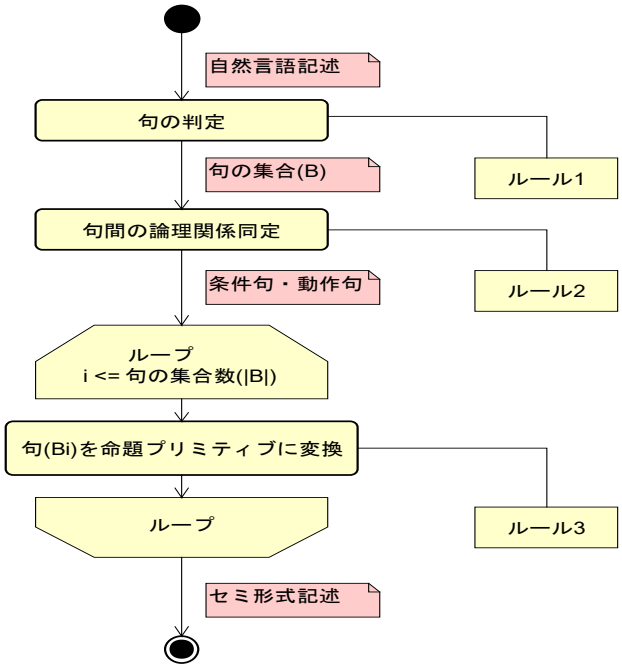

図 6　セミ形式記述変換アルゴリズムフローチャート

文を juman により形態素解析した各分節に対して，knp による解析で得られる feature 情報をもとに文節ごとに変換アルゴリズムのルール 1 からルール 3 で用いる属性の決定を行う．属性は論理関係を表す logic，関係詞を表す relation，主体・対象・制約を表す parameter の 3 種類を決定する．

3.2.1 句の判定ルール

図 6 のルール 1 である句の判定ルールを図 7 のように定める．句の判定は関係詞ごとに行う．関係詞（relation）は，動作を表す action と状態を表す state の 2 種類とする．関係詞の決定は knp の解析で得られる feature と表 2 のように対応している．

文を S とする
文 S の文節を$m(i)$とする
文末の文節番号をTとし、文末文節を$m(T)$とする
$m(i)$が関係詞 relation であれば$m(i)$の子文節の集合をD_iとする
$m(i)$とD_iの集合を句B_iとする（$m(i)$は句B_iの終端文節となる）

図 7　句の判定ルール

表 2　feature と relation 属性の対応

relation	feature
action	<用言：動>
state	<用言：判>, <用言：形>かつ<文末：True>

3.2.2 論理関係同定ルール

図 6 のルール 2 である論理関係同定ルールを図 8 に示す．

1.入力文章 S を形態素・構文解析(juman/knp)
2.文節$m(i)$ごとに属性決定
3. for $m(1)$: $m(T)$ [$m(1)$から $m(T)$まで文節の logic 属性を見ていく]
4.　if logic 属性が Not
5.　　then $m(i)$を終端文節として持つ句B_iに Not(!)を付ける
6.　if logic 属性が imply
7.　　then $m(i)$を終端文節として持つ句B_iを条件句とする
8.　　else $m(i)$を終端文節として持つ句B_iを動作句とする
9.　else if logic 属性が Not または And または Or
10.　　then $m(i)$を終端文節として持つ句B_iを And(&)または Or(|)を付ける
11.　end if
12.end for

図 8　論理関係同定ルール

論理関係は句間の関係であり，関係詞（句の終端文節）の feature から決定する．論理関係 logic は，否定を表す Not，論理積を表す And，論理和を表す Or，論理包含を表す Imply の 4 種類とする．論理関係の決定は knp の解析で得られる feature と表 3 のように対応している．

表 3　feature と logic 属性の対応

logic	feature
Not	<否定表現：True>
And	<並列タイプ：AND>, <ID：～て(用言)>, <ID：～で(判)>, <ID：～ても>, <ID：～ながら>, <ID：～て>かつ<用言：動>
Or	<並列タイプ：OR>, <ID：～たり>
Imply	<外の関係：True>かつ「場合」，「時」，「際」，<外の関係：True>かつ<ID：～ため>, <外の関係：True>かつ<時間：True>かつ<相対名詞：True>, <係：連用>かつ<ID：～ば>, <ID：～たら>, <ID：～ので>, <ID：～が>, <ID：～ように>, <ト：True>

3.2.1 命題プリミティブ変換ルール

図 6 のルール 3 である命題プリミティブ変換ルールを図 9 に示す.

```
1.入力文章 S を形態素・構文解析(juman/knp)
2.文節m(i)ごとに属性決定
3. for m(1): m(T) [m(1)から m(T)まで文節の relation 属性を見ていく]
4.   if m(i)の relation 属性が action
5.     then m(i)を関係詞としてセミ形式記述化
6.     セミ形式の初期記述を「m(i)(?-main, ?-object)」とする
7.       for all D_i [m(i)の子文節の集合について parameter 属性を見ていく]
8.         if D_i(j)の parameter 属性が main(主格)または object(目的格)
9.           then セミ形式記述の「?-main」または「?-object」の位置にD_i(j)を挿入
10.          else セミ形式記述の最後の位置にD_i(j)を restriction(制約)として追加
11.        end if
12.      end for
13.  else if m(i)の relation 属性が state
14.    then m(i)を制約としてセミ形式記述化
15.    セミ形式の初期記述を「is(?-main, _, m(i))」とする
16.      for all D_i [m(i)の子文節の集合について parameter 属性を見ていく]
17.        if D_i(j)の parameter 属性が main(主格)
18.          then セミ形式記述の「?-main」の位置にD_i(j)を挿入
19.          else セミ形式記述の最後の位置にD_i(j)を restriction(制約)として追加
20.        end if
21.      end for
22.  end if
23.end for
```

図 9　命題プリミティブ変換ルール

命題プリミティブへの変換は，ルール 1 で決定した句ごとに行う．命題プリミティブ間の論理関係はルール 2 で決定したものを用いる．命題プリミティブの項（parameter）である主体・対象・制約は knp の解析により関係詞の子文節として取得できる．命題プリミティブの関係詞はルール 1 により決定する．命題プリミティブの項は，主体を表す main，対象を表す object，制約を表す restriction の 3 種類とする．命題プリミティブの項の決定は knp で取得できる feature と表 4 のように対応している．

表 4　feature と parameter 属性の対応

parameter	feature
main	<解析格：ガ>
object	<解析格：ヲ>
restriction	<係：ニ格>, <係：ヘ格>, <係：カラ格>, <係：ヨリ格>, <係：デ格>, <係：マデ格>, <係：ノ格>, <係：ト格>,<ニテ：True>,<用言：形>,<副詞：True>

4. アルゴリズムの実装

3 章で述べたセミ形式記述変換アルゴリズムをプログラミング言語 Python[12]により，自然言語記述を入力とし，セミ形式記述を出力する機能として実装した．システム仕様書に記載される文は，テストケースとして扱う文の単純な形として以下の 4 種に分類することができる．

1. 宣言的な文
2. 状態を表す文
3. 並列な動作を表す文
4. 条件・動作に分かれる文

話題沸騰ポット[13]の文章を例として実装機能による出力結果の例を示す．

1. 宣言的な文

「サーミスタはポット内の水温を検出します」という文を入力すると「検出する（サーミスタ，ポット内の水温）」というセミ形式記述を出力する．

2. 状態を表す文

「ポットは沸騰状態である」という文を入力すると「is（ポット，_，沸騰状態）」というセミ形式記述を出力する．

3. 並列な動作を表す文

「ポンプは，ポット内の水を吸い上げて，給湯口から排出します」という文を入力すると「吸う（ポンプ，ポット内の水）＆排出する（??, ??，給湯口から）」というセミ形式記述を出力する．

4. 条件・動作に分かれる文

「給湯ボタンを押すと，ポンプを動作させる」という文を入力すると「押す（??，給湯ボタン）->動作する（??，ポンプ）」というセミ形式記述を出力する．

システム仕様書の文は基本的に上記 4 種を組み合わせて記述されているため，構造的に複雑な文であってもこれらの変換を組み合わせてセミ形式記述へと変換ができる．

5. アルゴリズム評価実験

4 章で述べた自然言語記述をセミ形式記述へと変換する機能を実際の企業が使用しているシステム仕様書に適用し，アルゴリズムの評価を行った．実験に使用するシステム仕様書は，共同研究企業 A の仕様書 A1（34 文），仕様書 A2（52 文），共同研究企業 B の仕様書 B1（21 文），仕様書 B2（20 文），横浜市健康福祉局の仕様書 C1（38 文），仕様書 C2（52 文）[14]の 6 つであり，それぞれシステムの機能について記述されている．

評価は，出力されるセミ形式記述が理想的なセミ形式記述にどれだけ近いかという観点で行う．理想的なセミ形式記述とは，関係詞・主体・対象がテストケースの項目，制約がテストケースのパラメータとなる形のもとのとし，手作業で作成した．実装機能により出力したセミ形式記述と理想的なセミ形式記述の差異を誤りとして数える．上記 6 種類の仕様書に対し，アルゴリズム・ルールの修正・追加を全 8 回実施し誤り数の遷移を確認した．仕様書ごとの誤り数の遷移比較にあたり，評価に使用した仕様書の文の数にばらつきがあるため各仕様書で生じた誤りの数を各仕様書の文の数で正規化した値を用いる．また，誤りにはアルゴリズムに起因する誤り，もとの文に起因する誤り，構文解析器 knp に起因する誤りの 3 種類がある．

5.1 セミ形式記述修正エディタ

理想のセミ形式記述作成にあたり，実装機能により出力

されたセミ形式記述の修正を手作業で行った．セミ形式記述の修正作業は以下の 3 項目がある．

1. 論理関係の修正
2. 変換に際して欠落する語の補完
3. 「??」の補完

上記修正を行うための補助ツールとして，Visual Studio Code[15]の拡張機能を用いてセミ形式記述修正エディタを作成した．セミ形式記述修正エディタは，ハイライト機能，ジャンプ機能，マージ機能の 3 機能を持つ．

5.1.1 ハイライト機能

ハイライト機能は，セミ形式記述の修正 1. 論理関係の修正，2. 変換に際して欠落する語の補完を行うために作成した．ハイライト機能を使用すると，論理記号を赤帯でハイライトする．また，もとの文には存在するが，セミ形式記述には存在しない語（変換に際して欠落した語）を青枠でハイライトする．セミ形式記述を修正する際，修正箇所を探すことに時間がかかるため，ユーザーはハイライトされた箇所に着目することで修正の手間を減らすことができる．図 10 にハイライト機能を使用した際のエディタを示す．

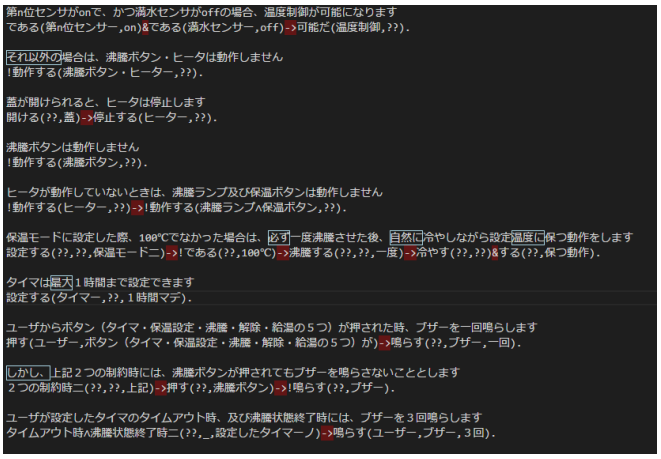

図 10　セミ形式記述修正エディタハイライト機能

5.1.2 ジャンプ機能

ジャンプ機能は，セミ形式記述の修正 3. 「??」の補完を行うために作成した．ジャンプ機能を使用すると，セミ形式記述内の「??」に飛ぶことができる．セミ形式記述を修正する際，「??」の補完または修正の数が多く存在するため，ユーザーはジャンプ機能を使用することで，「??」の修正の手間を減らすことができる．図 11 にジャンプ機能を使用した際のエディタを示す．

5.1.3 マージ機能

マージ機能は，セミ形式記述変換アルゴリズムの修正に使用する．上記自作エディタを用いてセミ形式記述を修正すると，もとの出力と修正による差分を別ファイルに保存

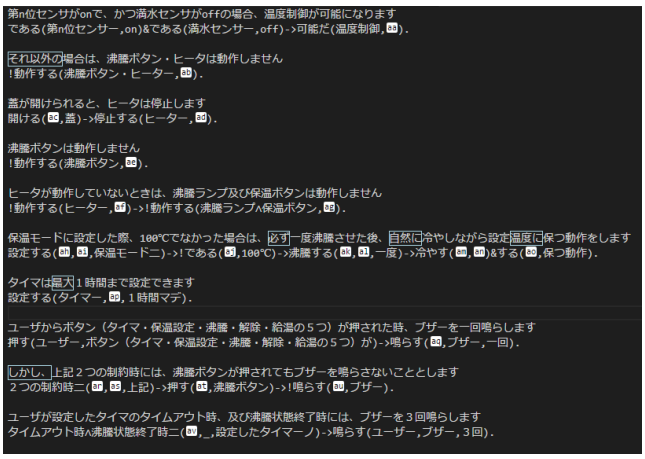

図 11　セミ形式記述修正エディタジャンプ機能

する．修正を重なることで，最終的にはアルゴリズムにより出力したセミ形式記述と手作業により作成した理想のセミ形式記述を並べて記載したファイルを出力する．出力されたファイルを確認することで，現在の出力と理想的な出力を比較しながらセミ形式記述変換アルゴリズムの修正を行うことができる．また，理想的なセミ形式記述を作成する過程を保存することができるため，ユーザーがどの部分の修正に手間取ったか，修正が困難な箇所を確認することができると考えている．図 12 はマージ機能により出力したファイルである．

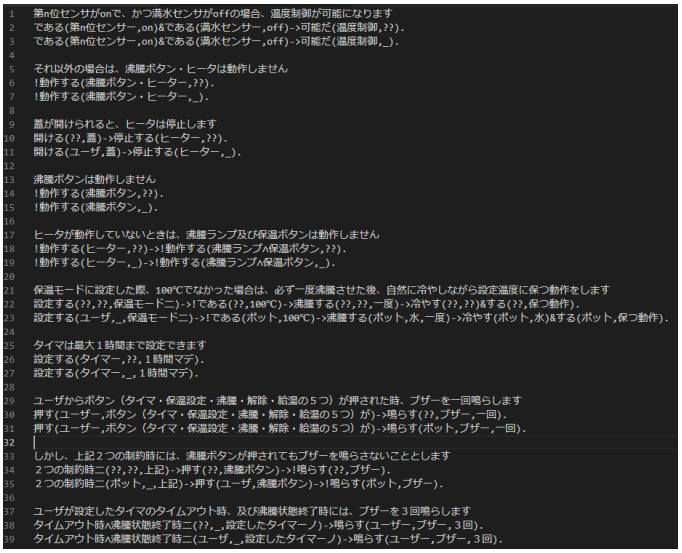

図 12　セミ形式記述修正エディタマージ機能により出力したファイル

5.2 命題プリミティブに関する誤り

変換アルゴリズムにより出力したセミ形式記述と，理想的なセミ形式記述の命題プリミティブに関する差異を誤りとして数えた．命題プリミティブに関する誤りは以下の 5 種類がある．

1. 関係詞
2. 主体
3. 対象

4. 制約
5. 変換に際して欠落した語

図 13 に修正回数と命題プリミティブの誤り数の遷移を仕様書ごとに比較したグラフを示す.

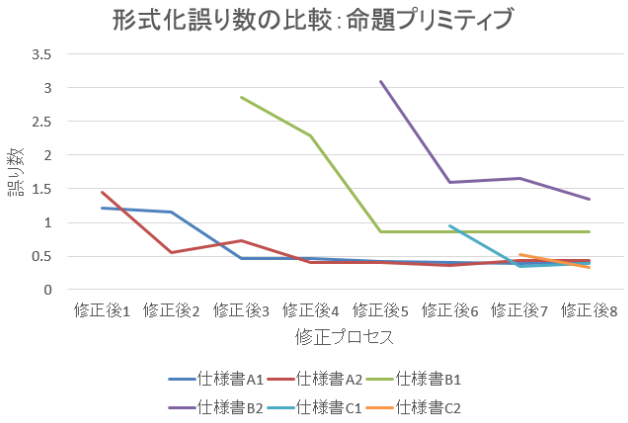

図 13 仕様書間の命題プリミティブ誤り数の比較

修正内容としては, 変換アルゴリズムのルール 1 において, <時間 : True>かつ<相対名詞 : True>の条件のとき関係詞を時間に関する状態とするルールの追加や, ルール 3 において, <ID:〜によって>, <ID:〜に対して>, <ID:〜にとって>, 文末の括弧など, これまで制約（restriction）としていなかった feature を制約とするためのルールの追加を行った.

5.3 論理関係に関する誤り

変換アルゴリズムにより出力したセミ形式記述と, 理想的なセミ形式記述の論理関係に関する差異を誤りとして数えた. 論理関係に関する誤りは以下の 4 種類がある.

1. Not
2. And
3. Or
4. Imply

図 14 に修正回数と論理関係の誤り数の遷移を仕様書ごとに比較したグラフを示す.

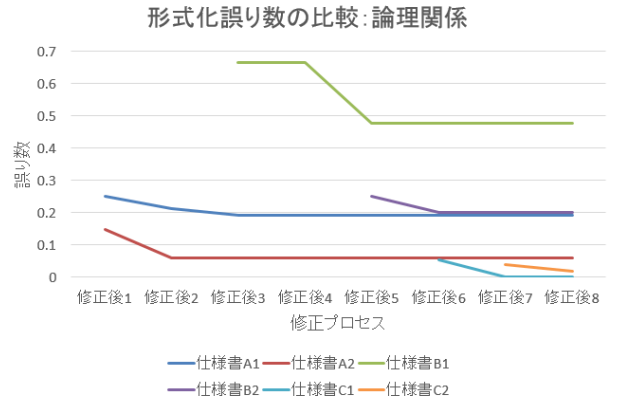

図 14 仕様書間の論理関係誤り数の比較

修正内容としては, 変換アルゴリズムのルール 2 において, <ハ : True>かつ<正規化代表表記 : 〜時>を Imply とするルールの追加や<用言 : 動>かつ<体言 : True>かつ<ID : (サ変)読点>の条件のとき論理関係を And とするルールの追加などを行った.

6. 考察

5 章の実験結果について考察を行う. アルゴリズムの修正内容としては, 基本的には構築アルゴリズムの各ルールに条件を追加する形式で修正を行った. 図 13 及び図 14 から様々な仕様書の文章を用いてアルゴリズムの修正を重ねることで, 構築アルゴリズムの精度が命題プリミティブ変換・論理関係の判断ともに向上することが分かった.

ここで, 仕様書の記述特徴により変換アルゴリズムがどのような影響を受けるかを分析するため, 最終的なアルゴリズムによって出力したセミ形式記述で生じた命題プリミティブの誤りの数と論理関係の誤りの数を各仕様書の文の数で正規化したグラフを図 15 に示す. すなわち, グラフの値は構築アルゴリズムによりセミ形式記述に変換する際に 1 文ごとに生じる誤りの個数である.

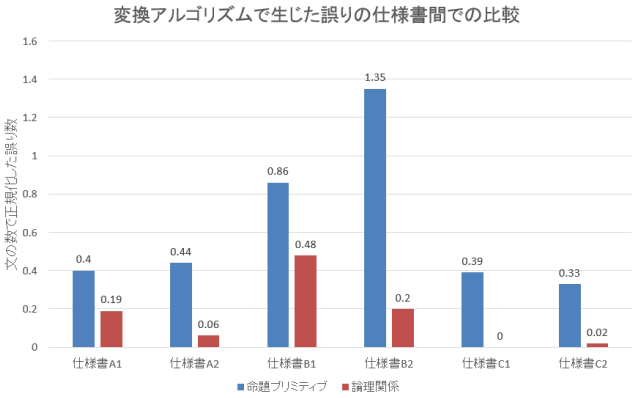

図 15 最終的な変換アルゴリズムで生じた誤りの仕様書間での比較

図 15 より, 仕様書 A, 仕様書 C に比べて仕様書 B の誤りが突出して多いことが分かる. 仕様書 A, 仕様書 C はシステムの機能が 1 文単位で簡潔に書かれており, 構文解析の誤りやアルゴリズムによる誤変換が少なかった. 一方, 仕様書 B は冗長な記述が多く, 構文解析での誤りやアルゴリズムの修正を重ねても対応できない形式の文章が多かった.

表 5 に knp で文を解析した際に取得した文節の数を各仕様書の文章の数で正規化した値を示す. この値が大きいほど 1 文に含まれる文節の数が多いため, 記述が冗長であると言える. 仕様書 C は文末が「〜こと」で終わる形式の文章であり, 構築アルゴリズムでは文末の「こと」は変換の際に削除するため, 表 5 では差し引いた値を使用している.

1 文に含まれる文節の数と誤りの数の関係を調べるために, 1 文に含まれる文節数と命題プリミティブ誤り数の相

関を図 16，文節数と論理関係誤り数の相関を図 17 に示す．

表 5　各仕様書の文の数で文節の数を正規化した値

	A1	A2	B1	B2	C1	C2
文節数	4.81	4.50	5.71	7.15	3.66	4.90

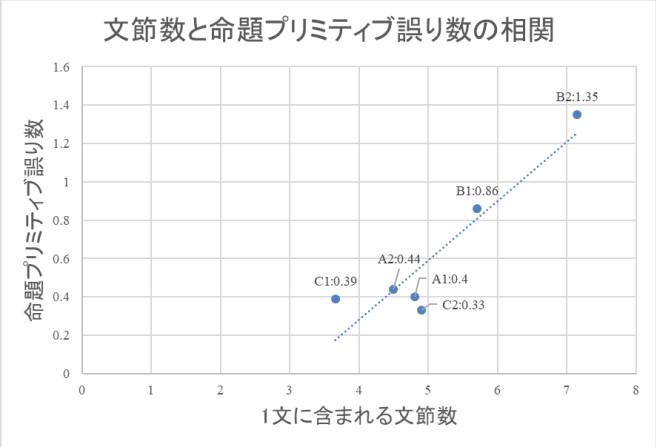

図 16　文節数と命題プリミティブ誤り数の相関

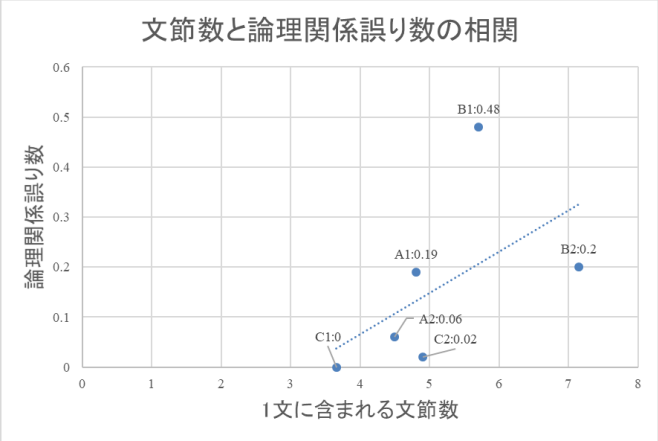

図 17　文節数と論理関係誤り数の相関

図 16，図 17 より文節数と命題プリミティブ誤り数の相関係数は 0.92 で強い正の相関があるが，文節数と論理関係誤り数の相関係数は 0.55 で強い正の相関がないことが分かった．1 文に含まれる文節数が多くなると，構文解析をする際に係受け関係や品詞の種類の判断に誤りが多くなるため，命題プリミティブに変換する際の誤りが増加すると考えられる．そこで，1 文に含まれる文節数と構文解析器 knp によって生じた命題プリミティブの誤りの数の相関を図 18 に示す．

図 18 より，1 文に含まれる文節数と knp による命題プリミティブの誤り数の相関係数は 0.71 であり，比較的強い正の相関がある．したがって，1 文に含まれる文節数が多く冗長な記述だと構文解析時に生じる誤りの数が増え，命題プリミティブに変換する際の誤りが増加するということが言える．

一方，図 17 より文節数と論理関係の誤り数には相関がなく，誤りはアルゴリズムに起因するものが多い．図 19 に論理関係の誤り数とアルゴリズムに起因する誤り数の相

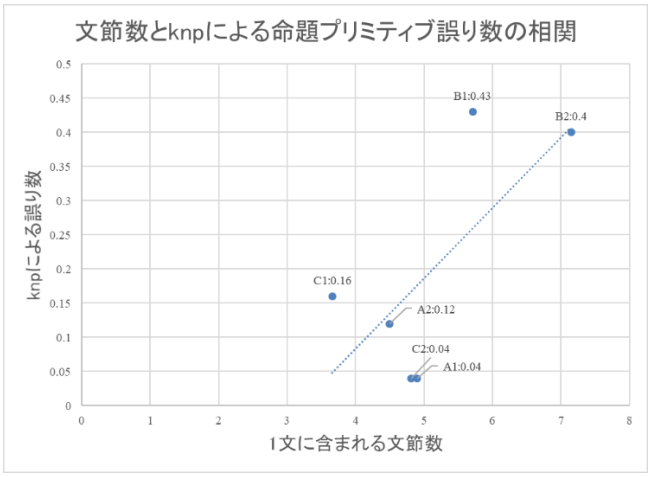

図 18　文節数と knp による
命題プリミティブ誤り数の相関

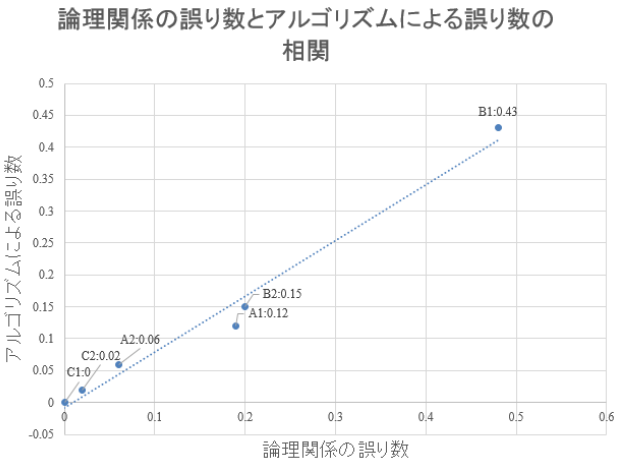

図 19　論理関係の誤り数と
アルゴリズムによる誤り数の相関

関を示す．相関係数は 0.99 であり，強い正の相関があることが分かる．現状の論理関係同定アルゴリズムだと，命題プリミティブに対応する論理演算子が同定ルールと 1 対 1 でしか対応しておらず，文意に沿った論理関係の判断を行うことができない．例えば，「沸騰中に解除ボタンを押すと，沸騰を止める」という文から理想のセミ形式記述を作成すると，「is (??, _, 沸騰中に) & 押す (??, 沸騰ボタン) ->止める (??, 沸騰)」となる．しかし，同じ文章をアルゴリズムにより変換すると，「is (??, _, 沸騰中に) ->押す (??, 沸騰ボタン) ->止める (??, 沸騰)」と変換されてしまい，状態 (is) の後の論理関係に誤りが生じてしまう．テストケースの作成において，条件に状態が来る場合は前提条件となる．変換の際に条件句に状態句しかなく，条件が成り立つ動作句が存在しない場合は，条件の状態句と動作句の論理関係を And にするなどの修正が必要である．さらにこれらテストケース特有の記述パターンに対応した変換を行

うため，セミ形式記述の蓄積などを行い，記述パターンを学習することで記述パターンに対応した論理関係や命題プリミティブを予測するアルゴリズムの追加も考えられる．

7. まとめ

システム仕様書からテストケース仕様書を作成する際，テスト設計者は知識や経験に基づいてシステム仕様書に含まれる記述漏れや前提漏れなどの欠陥を暗黙的に修正していた．そのためレビュアーは作成されたテストケースへのレビューを行う際テストケースの作成根拠が分かりづらく，本質的なレビューを行うことが困難であった．また，テストケースの作成は手作業で行うため，テスト仕様書の作成には膨大な時間がかかっていた．

これまで暗黙的に行われていたテスト設計者の作業をプロセス化し，各ステップで出力される結果に対してのレビューを行うことでテスト設計者とレビュアーの齟齬を減らし，適切なレビューが支援するプロセスの提案及びツール群の実装を行った．

本研究では，これまでテスト設計者が手作業で行っていた自然言語から条件・動作を取り出すという機械的な作業や上記プロセスのテストケース自身である自然言語からセミ形式記述への変換を，構文解析を用いたアルゴリズムを設計することで自動化した．これにより，手作業でテストケースを作成することで生じる膨大な時間の削減および人為的ミスの発生を解決する．

本論文では，提案するテスト設計プロセスの起点となる自然言語記述からセミ形式記述を生成するアルゴリズムの構築及び評価を行った．評価実験により，アルゴリズムの修正を重ねることでその誤りの数を 1 文あたり 1~2 個にまで減らすことができた．また，仕様書に記述される文が冗長であると構文解析結果による誤りが増え，セミ形式記述での誤りが増加することも確認できた．

今後は，セミ形式記述の修正プロセスにおいて，テスト設計者がよりセミ形式記述の修正を手軽に行えるよう，セミ形式記述修正エディタの拡張，アルゴリズムの精度向上及びツール群の統合化を目指したい

謝辞

本研究は科研費 16K00100, JST CREST JPMJCR1304 の支援を受けて実施されたものである．

参考文献

[1] 青山裕介，久代紀之，村上響一，仕様書からのテストケース設計プロセスの定義とテストケース生成支援ツール，FITPaper, 2018
[2] 青山裕介，黒岩丈瑠，久代紀之，自然言語使用からの機能間の並列・順序動作の抽出と左記テスト環境, FITPaper, 2017.
[3] 村上響一，青山裕介，村上神龍，久代紀之，牧茂，田畑一政，神代勉，中村潤，自然言語仕様書からの試験ケース生成のための条件・動作の同定手法，研究報告ソフトウェア工学(SE)，Vol.2018. No.7, pp.1-7, 2018.
[4] C. Rolland, C.B.Achour: Guiding the construction of textual use case specifications, Data & Knowledge Engineering, Vol.25, Issues 1-2, pp.125-160, ELSEVIR, 1998.
[5] 久代紀之，藤田裕司，村上響一，青山裕介，リスク認知における知ってる/知らない知識の表出化と可視化，電子情報通信学会, 2018.
[6] Kirschner, P.A., Buckingham-Shum, S.J. and Carr, C. S.: Visualizing argumentation: Software tools for collaborative and educational sense-making, pp. 8-9. Springer Science & Business Media, 2012.
[7] J. マイヤーズ, M. トーマス：ソフトウェアテスト・テストの技法 第2報，近代科学社, 2006.
[8] 増田聡，松尾谷徹，津田和彦，試験ケース作成自動化のための意味的役割付与方法，特集ソフトウェア工学の基礎，Vol.34, No.2, pp.16-27, May 2017.
[9] 奥村学，自然言語処理の基礎，コロナ社, 2010.
[10] 京都大学黒橋・河原研究室，自然言語処理のためのリソース，http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN
[11] 京都大学黒橋・河原研究室, knpで付与されるfeature一覧，http://nlp.ist.i.kyoto-u.ac.jp/index.php?plugin=attach&refer=KNP&openfile=knp=feature.pdf.
[12] http://www.python.org
[13] 話題沸騰ポット(GOMA-1015型) 要求仕様書[第6版]，SESSAME, 2004.
[14] 横浜市健康福祉局健康安全課，予防接種業務支援システム開発業務委託仕様書, 2013.
[15] https://code.visualstudio.com/