B-018

プロダクトライン開発における セミ形式化記述を用いた仕様整合支援ツール

A Design Support Tool to Ensure Consistency of Functional Specifications using Semi-Formal Methods

小宮 紀之[*1]　久代 紀之[*2]　深澤 良彰[*3]
Noriyuki Komiya　Noriyuki Kushiro　Yoshiaki Fukazawa

1．まえがき

複数の機器で共用することを前提としたソフトウェアを開発してプラットフォーム化し，その差分開発を行うことで機能追加や新機種の開発を行っていくプロダクトライン開発[1]が多くの生産現場で採用されている．このような開発形態では，同時並行的な開発が前提となるため，既存機能の設計仕様(以降，本稿で単に「仕様」と記載した場合は設計仕様を指す)と新規開発部分の設計仕様との間の整合性が非常に重要となる．

本研究では，上記課題を解消するために，ソフトウェア設計者が追加機能の仕様設計を行う際，既存部分の仕様の現状と，追加機能の影響による変更・修正箇所を容易に把握することができ，かつ，更に次期開発の設計者に仕様を引き継ぐために，仕様の曖昧性を排除し，設計者毎の解釈の齟齬が発生し難い仕様記述を行うことで，将来に亘って仕様の整合性を保持していくことができる設計支援ツールを提案する．本研究のスコープとする機能追加型開発が行われている空調コントローラ製品に対し，そのソフトウェア仕様の設計工程で本提案ツールを試用し，仕様の不備への気付き，仕様の競合の発見等の効果を確認した．

2．機能追加時の設計手順と必要となる設計支援

機能追加時の設計手順と，必要となる設計支援内容について，作業の流れに沿って述べる（図 1）．

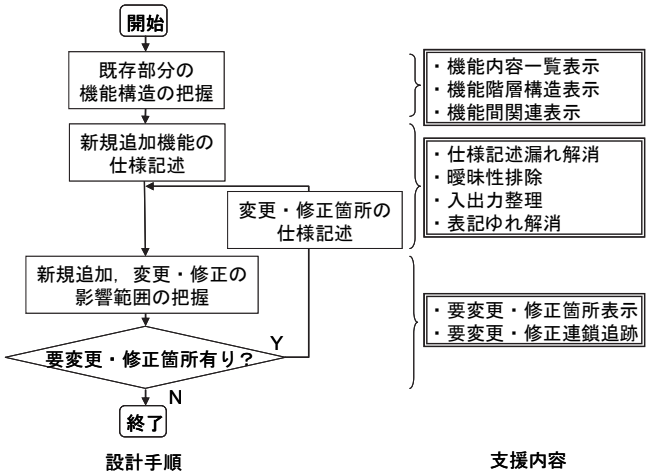

図 1　機能追加時の設計手順と設計支援

(1) 既存部分の機能構造の把握

・全体の機能構造の把握

機能追加の設計を行うにあたり，まず対象となるシステムの機能と全体構造を把握する．既存部分の仕様について，機能の内容，階層構造等を把握する．

この工程で必要となる支援内容は，この把握を促進するためのもので，これら機能内容を一覧し，機能間の関連を含めた機能構造を明示することである．

・新機能の追加箇所の決定

全体の構造を把握した上で，新機能の追加箇所を決定する．機能構造の中で，周囲との関連を考え，適切な位置に新機能を追加する（図 2）．

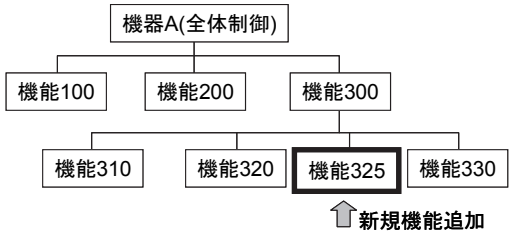

図 2　全体の機能構造と新機能の追加箇所の決定

(2) 新規追加機能の仕様記述

全体の機能構造の中で適切な仕様追加箇所を定めた後，新規追加機能の仕様記述を行う．

この工程で必要となる支援内容は，従来からの仕様記述における問題点を解消するためのもので，下記となる．詳細は次章で述べる．

・仕様記述漏れの解消
・仕様の曖昧性の排除
・各機能で使用する入出力データの整理
・表記ゆれ（同義別表記）の解消

(3) 新規追加，変更・修正の影響範囲の把握

新規機能を追加したことで，例えば機能間の競合を回避するための処理が必要になる等，既存部分の仕様に変更・修正が必要となることがある．また，行った変更・修正により，さらに別の機能に連鎖的に変更・修正が及ぶこともあり得る．

従来，このような変更・修正の波及箇所を調査するには，機能追加を重ねて数百ページにも及ぶことのある既存機能分の仕様書の中から，着目している機能と関連する機能を逐次丹念に手動で検索する必要があった．この工程で必要となる支援内容は，この手間を軽減するためのもので，要変更・修正となった関連機能を逐次追跡し，明示することである．

*1 三菱電機株式会社, Mitsubishi Electric Corporation
*2 九州工業大学, Kyushu Institute of Technology
*3 早稲田大学, Waseda University

（4）変更・修正箇所の仕様記述

新規追加，変更・修正の影響範囲の把握により，変更・修正が必要な箇所が残存すると判断された場合には，当該箇所の仕様の変更・修正作業を引き続き行う．

この工程で必要となる支援内容は，新規追加機能の仕様記述の場合と同様である．

このようにして変更・修正が必要な箇所が無くなるまで繰り返し影響範囲の把握と変更・修正作業を行い，システム全体の仕様を確定する．

3．提案ツールの構成

以上で必要とされた設計支援を行うツールを開発した．ツールの全体構成を図 3 に示す．初期開発では，仕様入力・修正ナビに従って設計者が思い描いている仕様をツールに入力し，新規に仕様記述を行う．これをベースに機能系統図（機能の一覧を階層構造と共に示したもの），機能間関連線（関連する機能同士の結線）をツールが生成，表示する．以降の追加開発では，これら機能系統図，機能間関連線等を用いて機能構造や関連する機能を把握した上で，仕様記述の新規開発分の入力，関連部分の修正を行う．さらに，左記修正等の影響を受け，連鎖的に修正が必要となる箇所を機能間関連線を順次辿ることで追跡調査し，修正が必要な箇所が存在する間修正を繰り返して最終的な仕様記述を完成させる．以上の過程の中で，記述内容・書式をパターン化した仕様記述パターンにより仕様記述漏れチェックが，用語 DB により用語の整合性チェックが行われる．これらについて以下詳述する．

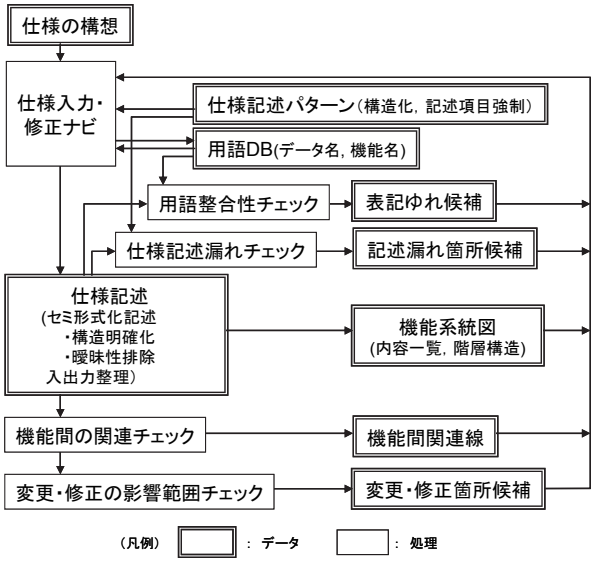

図 3　仕様整合支援ツールの全体構成

（1）仕様入力・修正ナビ

設計者の仕様の入力・修正作業の支援を行う．

初期開発時には，ツールには仕様記述パターンのデータのみ存在する．仕様入力・修正ナビでは，設計者に仕様記述パターンの一覧を提示し，仕様の入力を促す．設計者は記述したい仕様内容に応じて仕様記述パターンを選択し，セミ形式化記述を用いて，データ名・機能名を用語 DB に登録しながら入力作業を行う．

（2）セミ形式化記述

本ツールでは，仕様記述はセミ形式化記述を用いて行う．従来の仕様書では，制御アルゴリズムの仕様も自然言語による文章で記述されることが多く，構造，係り受け，結合優先度等に曖昧さが残り，そのため，仕様の検討漏れに気付かない，仕様の受け取り手により解釈の齟齬が生じる等の問題があった（図 4）．

例）機器が停止かつ下限温度＞室温の状態となった場合，
制御 A を実行する．
↓
【構造】「そうでない場合」が意識されない（漏れの元）．
【係り受け・結合優先度】「停止」も「なった」のか，
予め「停止」状態であったのか，いずれでも良いのか，
不明．

図 4　自然言語による従来の仕様記述とその問題例

このような問題を解消し，構造の明確化，曖昧表現の排除を目的として，形式化記述の導入を検討した．しかし，通常の形式化記述は，集合論や論理学の数学をベースにして厳密に定義されたものであるため，正確かつ詳細な記述・検証ができる一方で，記述の負荷が高くなってしまう．従来の仕様書記述に慣れた設計者にも書き易く，導入に対する心理的抵抗を低く抑えるものとして，馴染み易い自然言語を残しつつ，曖昧さを排除する形式化記述方式（セミ形式化記述）を考案した．後述する仕様記述パターンも併用することで，図 4 に掲げた例は図 5 のように記述することができる．

```
((<$運転状態$> == 停止) && (<$下限温度$> > <$室温$>)) ?:
  YES:
    実行:
      <<制御 A>>を実行する
    非実行:
      #(何もしない)
  NO:
    実行:                          ---- (A)
      #(何もしない)
    非実行:
      #(何もしない)
```

図 5　セミ形式化記述の例

仕様記述の構造を見える化することにより，図 5 の破線囲み部分（A）のケースが未検討であることに容易に気付くことができるようになる．例えそのケースでは何もする必要がない場合でも，図 5 のように記述しておくことで各ケースを網羅的に検討済であることを記録に残すことができる．また，冒頭の条件判定部分についても，係り受け・結合優先度について曖昧さが解消され，解釈に齟齬が発生しないように改善することができる．

さらに，図 5 に示すように，参照・操作する外部データ名・呼び出す外部機能名にはタグ（<$～$>, <<～>>）付けすることにより仕様記述から明確に切り出しができるようにした．設計者が外部データ・外部機能を意識し，明示することで，当該機能の入出力を整理し，他の機能との関連

を抽出することができるようにすると共に，後に機能追加を行う設計者に対しても設計情報を引き継ぎ易くすることができる．

（3）仕様記述パターン

セミ形式化記述の導入目的の一つ，仕様記述の構造化を推進するために，仕様記述パターンを定めた．仕様記述パターンの一部を図 6 に示す．パターンの抽出にあたっては，本ツールのメイン・ターゲットとなる空調コントローラ製品の仕様書記述の分析結果およびアルゴリズムの記述パターンとして広く使われているものを参考にした．

このようにパターン化し，記述項目を強制することで，設計者の当初の着目点以外のケースも網羅的に検討することを促すことができる．さらに，書式・記述すべき内容を定め，提示することで，設計者の記述作業の容易化を図ることもできる．

```
P1:      #(通常処理)
    [(元データ)を用いて、] 処理 [(結果)を得る]

P2:      #(条件分岐)
    (条件)？:
      [ YES:
        [ 実行:
            Pn ]
        [ 非実行:
            Pn ] ]
      [ NO:
        [ 実行:
            Pn ]
        [ 非実行:
            Pn ] ]

P3:      #(繰り返し)
    毎( 項目, 継続条件 ):
      Pn
      ・・・

P5:      #(順次実行(実行順序規定))
    SEQ:
      Pn
      ...
      Pn
      ・・・

[ ]：省略可(ただし省略は非推奨)
```

図 6　仕様記述パターン（抜粋）

（4）用語 DB（データベース）

セミ形式化記述では，データ名・機能名等に自然言語記述を容認している．そのため，仕様追加のタイミングや設計者の違いによって同一物理量・機能に対して異なる用語を割り当ててしまったり，例え同一のタイミングで同一の設計者が記述したとしても表記の省略や単純なかな漢字変換の差異などが発生したりする表記ゆれの問題が起こり得る（図 7）．このような表記ゆれは，設計者の仕様の理解の妨げとなり，仕様間の整合を取る上での阻害要因となる．また，本研究のツールのように仕様記述を機械処理する場合にも一致・不一致の判定処理が重くなるなどの障害となる．

【同一物理量・機能に対する異なる用語の割当て】
予冷・予熱制御　⇔　最適起動制御
【省略，単純表記ゆれ】
温度下限値，上限値　⇔　温度下限値，温度上限値
吸い込み温度　⇔　吸込み温度

図 7　表記ゆれの例

このような問題を解消するために，用語 DB を導入する．仕様を記述する際，データ名・機能名は予め用語 DB に存在するものを利用するか，新しいものは登録してからでないと使えないようにすることで，用語の統一化を図る．用語 DB への登録の際には当該用語が意味する内容も併せて登録するようにし，同一物理量・機能に対する異なる用語の割当ての検出に利用する．用語の DB 化により，仕様記述時にどのような用語を用いるべきか分からなくなった場合にも，DB のルックアップにより解決できるという効果も期待できる．

（5）機能系統図

機能の一覧を階層的に表現したもので，これに対して後述する機能間関連線を追記することで，システム全体から見た各機能の位置付けや，機能間の関連を見通しよく把握することができるようになる（図 8）．

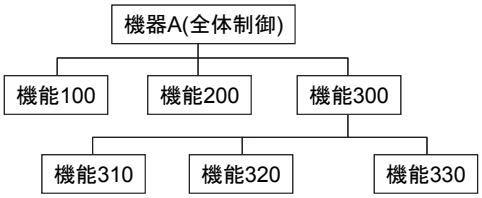

図 8　機能系統図

（6）機能間関連線，影響箇所強調

機能間の依存・制約関係を機能間の結線で表現したものが機能間関連線である（図 9）．この関連を辿ることで，機能の新規追加や変更・修正が行われた際に，どの機能に影響が及ぶ可能性があるかを容易に把握することができる．すなわち，さらに変更・修正が必要となるか否かについては，関連先の機能を候補として仕様の見直しを行えばよい．結線の本数を関連するデータ・機能の数に原則比例させることで，関連の複雑度を視覚的にも把握し易くしている．同図に示すように，機能間の関連線は機能間のわたりの部分を見易さのため 1 本に集約するが，着目しているデータや機能に対し，関連線の経路や関連先の機能の枠線を強調表示（線の太さや着色を変える等）することで，関連先を追跡可能としている．

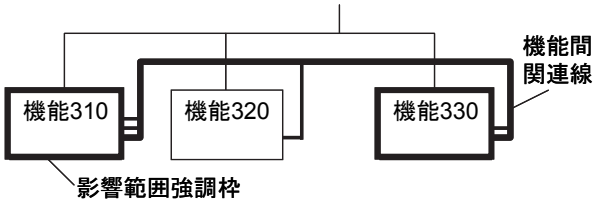

図 9　機能間関連線と影響範囲強調枠

機能間の依存・制約関係には，一方の処理結果に他方が依存する，一方の処理中には他方の処理が行えない（排他）など様々なものがあるが，本ツールでは，データ参照や機能呼び出しの関係を元に機能間の関連を求めるものとする．同一データへのアクセスを行う機能を検出して機能間の関連を抽出している例を図 10 に示す．

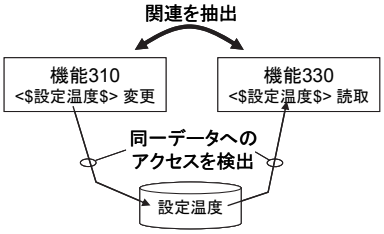

図 10　機能間の関連の抽出例

以上の構成により，セミ形式化記述を用いて各機能仕様を漏れなく記述し，機能間の関連を効率よく把握しつつ全体として整合性の取れた仕様書が作成可能となる．

4. 評価

本ツールを空調コントローラ製品のソフトウェア仕様の設計工程で試用し，仕様記述パターンによる仕様記述可否，仕様の不備への気付き，仕様の競合の発見の効果を確認した．さらに，現行仕様書に潜む表記ゆれを推定することで，表記ゆれ解消効果の見込みも得た．

4.1 仕様記述パターンによる仕様記述可否確認

本研究で定めた仕様記述パターンを用いて，空調コントローラ製品の既存ソフトウェアの全制御仕様を記述した．

(1) 下記一部の修正を経て，全仕様の記述が行えることを確認した．記述したロジックの，元の仕様との整合性はレビューにより確認した．

(2) ループ脱出方法の記述の仕方の定義が欠落していたため，設計者によりまちまちな記述となり，分かり難くなりがちであった点を修正した．

① 通常ループの継続条件の記述方法を定義

② 条件判定やループ等の構造の入れ子を一気に複数脱出する場合（例外）の記述方法を定義

4.2 仕様不備への気付き

上述の仕様記述作業を通じて，既存部分の仕様不備の指摘が行われた．これらについてその内容と件数を評価し，ツールによる気付きの効果を確認した．

(1) 仕様不備の指摘結果

表 1 仕様不備の指摘

仕様不備の指摘内容	件数
検討漏れケース有り	7
条件未記載	4
定義不明瞭	2
合計	13

(2) 仕様不備への気付き効果

指摘内容の分析の結果，表 1 に示した指摘内容のうち，表 2 に示すものはツールによる気付きの効果によるものと考えられることが分かった．

表 2 ツールによる気付き効果

仕様不備の内容	気付き件数
検討漏れケース有り	7
条件未記載	2
合計	9

これらは，従来手法では検討から漏れてしまっていた，設計者が主眼を置いているメインのケース以外のケースについての検討が，本ツールで仕様記述パターンに当てはめて仕様記述を行うことで促進され，不備の気付きに至ったものと考えられる．

4.3 仕様の競合の発見

新規機能の追加の際，従来通りの手法で仕様書記述を行った場合と，本ツールを用いて行った場合とで，既存の仕様と新規追加する仕様との整合を取る過程を比較した．

表 3 既存仕様と新規仕様との整合を取る過程の比較

	従来手法	ツール利用
関連仕様抽出方法	記憶想起＋検索	関連線追跡
競合有無判定結果	無し	有り

(1) 関連仕様の抽出は，従来手法では，以前の仕様書作成時の記憶，あるいは想起される関連単語の仕様書からの検索によって行われた．ツール利用の場合はツール上に提示される機能間の関連線を辿ることで容易に短時間で実施可能であった．

(2) 競合の有無の判定については，従来手法では無し，ツール利用では有りとされた．正しい結果は有りである．ツールを利用することで関連先の仕様が容易に抽出可能なため本来の仕様の見直し作業の方に集中できること，セミ形式化記述により構造が明確かつ曖昧さが排除されているために詳細まで見落とし難く理解し易いことなどが要因と考えられる．

以上から，ツールの仕様競合の発見効果を確認した．

4.4 表記ゆれの解消

仕様記述に際して，予め用語 DB に用語（データ名，機能名）を登録してから記述に用いるのではなく，既存仕様書をベースに先に全ての仕様記述を行って，そこから用語をピックアップすることで，既存仕様書内にどの程度の表記ゆれが潜んでいるかを調査した．

表 4 表記ゆれ

	総数	表記ゆれ発生個数
機能名	54	0
データ名	348	9

本ツールを適用し用語 DB を活用することで，用語を記述する毎に DB から用語を選択したり，DB への用語の新規登録の妥当性を検証したりするため，これら表記ゆれの解消が行えると見込んでいる．

5. まとめ

本研究では，ソフトウェア設計者が追加機能の仕様設計を行う際，既存機能の仕様および機能追加の影響による変更・修正箇所を容易に把握でき，かつ曖昧性を排除することで設計者毎の解釈の齟齬を防止し，将来にわたって仕様の整合性を保持していくことが可能な設計支援ツールを提案した．本提案ツールを空調コントローラ製品のソフトウェア設計作業で試用し，提案した仕様記述パターンの妥当性，仕様設計における仕様の不備への気付き，仕様の競合の発見の支援効果を確認した．また，表記ゆれ解消効果の見込みも得られた．

一方で，ツール使用時の作業分析から，下記のような問題も抽出された．

・仕様記述パターンを用いた記述作業において，パターン選択に戸惑いが見られた．

・仕様記述に用いる用語の選択に迷いが見られた．

設計者が考えている仕様に対応する仕様記述パターンを効率よく選択し，仕様記述を行わせるためのナビゲーション，および類似の用語を多数含むデータベースの中から目的の用語を効率よく検索・選択可能とするための方策が必要である．

参考文献

[1]クラウス・ポール, 他 ”ソフトウェアプロダクトラインエンジニアリング”, エスアイビーアクセス (2009).