ソフトウェア工学 58－10
（１９８８．２．４）

プログラム自動生成システムＳＡＧＥ

上原憲二，黒田清隆，土井日輝，行徳孝彦，鈴木由美子
三菱電機㈱情報電子研究所

疑似日本語で書かれたプログラム仕様書から手続き型言語のソースコードを生成するプログラム自動生成システムＳＡＧＥ(Sysytem for Automatic program GEneration)について記す。対象分野は事務処理のバッチ型ファイル操作であり、事務処理用簡易言語のソースコードを生成する。仕様獲得の方式は、処理パターンに基づいて仕様書から必要な情報を、仕様書の章構成を利用して、取り出すものである。ＡＩワークステーションＭＥＬＣＯＭ－ＰＳＩ上でプロトタイプ開発・評価を行った結果、十分な速度性能を得ることができた。

SAGE：A SYSTEM FOR AUTOMATIC PROGRAM GENERATION

Kenji UEHARA, Kiyotaka KURODA, Hideru DOI, Takahiko GYOHTOKU, Yumiko SUZUKI

Information Systems & Electronics Development Laboratory, MITSUBISHI ELECTRIC CORPORATION
5-1-1 Ofuna, Kamakura, Kanagawa, 247 JAPAN

We present a system for automatic program generation SAGE, which generates source codes in a procedural language from program specification document in Japanese language. SAGE is for batch type file processing in business application, and generates source codes in a problem oriented language. The method to understand the specification is to take information out from program specification document to the processing pattern which is inferred from the document. We developed the prototype system and estimated it to be of good performance.

１．はじめに

近年、知識情報処理技術の発展がめざましく様々な分野に応用されている。ソフトウェア生産技術においても、それを応用した技術の開発に期待が寄せられている。プログラム自動生成システムＳＡＧＥ(System for Automatic program GEneration) もそのようなシステムであり、特に自然言語処理技術を応用している。

ＳＡＧＥは次のような特徴をもつ。

①対象分野を事務処理のバッチ型ファイル操作とする。

②日本語（疑似日本語）で記述したプログラム仕様書より事務処理用簡易言語のソースコードを生成する。疑似日本語は上記分野特有の用語、言い回しを用いた自然な仕様記述を可能とする。

③仕様獲得の方式としては、仕様書の章構成を利用する。まず、仕様書の概要記述からプログラムの処理パターンを推定し、次に処理記述から具体的な仕様をその処理パターンへ取込む。

以下、２章ではシステムの概要を示し、３、４、５章ではシステムの中心部分について記述し、６章ではプロトタイプによる評価について述べる。

２．システムの概要

ＳＡＧＥはソフトウェア開発のウォーターフォール・モデルに基づく製造段階を機械化するものであり、設計段階の生産物である仕様書を入力として、ソースコードを生成する（図１参照）。これにより仕様書が書ければプログラム言語を知らなくてもプログラムを作成することができる（プログラマ人口の拡大）。また、仕様書の修正がすぐにソースコードに反映できるので保守が容易になる。

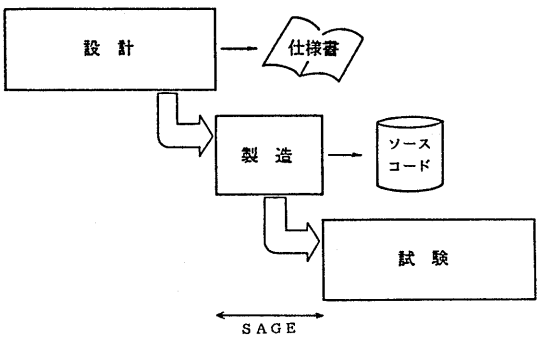

図１．ＳＡＧＥの位置付け

（１）入力仕様書の構成

ＳＡＧＥの入力となる仕様書としては、全く新しいものを採用するのではなく、開発プロセスの製造段階を機械化する意味において、またユーザにとって書き易く読み易いものとするため、実際に使われている事務処理用プログラムの仕様書を調査して、次のような構成とした。

第１章　概要
　主な処理内容の記述。
第２章　ファイル記述
　入出力ファイル、参照ファイル、サマリリスト、エラーリストなど処理に用いるすべてのファイルの名前、属性およびファイルレコードの構成要素であるデータ項目の名前、属性の記述。
第３章　処理
　処理内容全体の記述。更に、３章では節を設けて詳細な部分処理を記述する。

「第１章　概要」においては、主要な入出力ファイルの名前、主要な処理を表わす動詞や名詞を用いて、主な処理内容が記述される。対象分野のファイル操作プログラムでは、処理はいくつかの定形的処理パターンに分類される。この章の記述からその処理パターンを予測できる。

一般に仕様書は、大まかな機能の記述の後にその入出力データを明確にする。「第２章　ファイル記述」ではすべてのファイルについて、ファイル、レコード、データ項目の名前、属性を記述する。使用されるファイルは通常プログラム毎に異なるが、一定の業務あるいは組織に限定すると使用するファイルは固定化する傾向にある。そのような場合は、固定化したファイル名を記すだけで仕様書として十分な情報を伝達できることがある。

「第３章　処理」においては、プログラムの処理内容の全体に渡って記述される。ファイル操作プログラムではこの記述は主に２レベルに分けられる。１つは処理全体のアルゴリズムに対応するもので、ファイルやレコードに関する名詞やそれを目的語とした動詞で記述される。例えば、「ファイルからレコードを入力する」、「レコードの検査をする」、「レコードをマッチングする」、あるいは検査やマッチングの結果による制御などの記述である。他の

1つはレコードに含まれるデータ項目などの演算に関する記述である。例えば、入力レコードの各データ項目から出力レコードの各データ項目をどのように計算するかを記述する。そこで、全体のアルゴリズムに対応する記述は「第３章」として記述することとし、レコード内のデータ項目の演算に関する記述は詳細な部分処理として「第３章」内の節として記述することとした。レコード内のデータ項目の演算としては、入力レコードのデータ項目から出力レコードのデータ項目を求めるレコード編集、レコードのデータ項目の値に基づいて検査、判定を行うレコード検査、レコード判定などがある。

図２に仕様書の例を示す。

```
1　概要
　取引ファイルを入力し項目検査を行い、売り上げファイルを作成する。

2　ファイル記述
　ファイル名　　　ＦＤ名　　モード
　取引ファイル　　T1TORI　　　Ｉ
　売り上げファイル　S1URI　　　Ｏ　　・・・
　エラーファイル　　E1ERR　　　Ｏ
2.1 取引ファイルレコード
　項目名　　識別子名　　属性
　区分　　T1-KUBUN　　X(1)
　商品名　　T1-SNAME　　X(20)
　単価　　T1-TANKA　　9(6)
　数量　　T1-SURYO　　9(3)
2.2 売り上げファイルレコード
　項目名　　識別子名　　属性
　商品名　　S1-SNAME　　X(20)
　単価　　S1-TANKA　　9(6)
　数量　　S1-SURYO　　9(3)
　金額　　S1-KINGAKU　　9(9)
　値引額　　S1-NEBIKI　　9(9)
2.3 エラーファイルレコード
　項目名　　識別子名　　属性
　区分　　E1-KUBUN　　X(1)
　商品名　　E1-SNAME　　X(20)
　単価　　E1-TANKA　　9(6)
　数量　　E1-SURYO　　9(3)

3　処理
　取引ファイルから入力条件を満たすレコードを入力し、項目検査を行う。エラーがない場合、売り上げ金額計算を行い売り上げファイルに出力する。エラーの場合、レコードをエラーファイルへ書き出す。
3.1 入力条件，１’
　区分　＝　１’
3.2 項目検査
　単価　＞　３０００
　数量　＞　０
3.3 売り上げ金額計算
　単価　＊　数量　－＞　金額
　金額　－　金額　＊　０．２　－＞　値引額
```

図２．仕様書の例

例からわかるように、「第１章」、「第３章」は疑似日本語で記述するものとし、これらの記述を仕様文と呼ぶ。「第３章」中の節はデータ項目の演算の記述容易のため疑似コードで記述する。また、「第２章」は表形式の記述である。

（２）処理パターンによる仕様獲得の方式

上記のような特徴をもつ仕様書から仕様を獲得する方式を示す。

まず、「第１章　概要」から処理パターンを推定する。処理パターンはアルゴリズムの記述のレベルに対応して、ファイルやレコードを表わすノード、および入力、出力、検査、判定などの動詞を表わすノードをもち、それらの間をリンクで結合したものである（図３参照）。

次に、「第３章　処理」には処理全体に渡る記述があるので、そこから、推定した処理パターンの各ノードへ具体的なファイルやレコードの名前、動詞に対応する部分処理の名前（図２では「入力条件」、「項目検査」、「売り上げ金額計算」）や節番号を取込む。

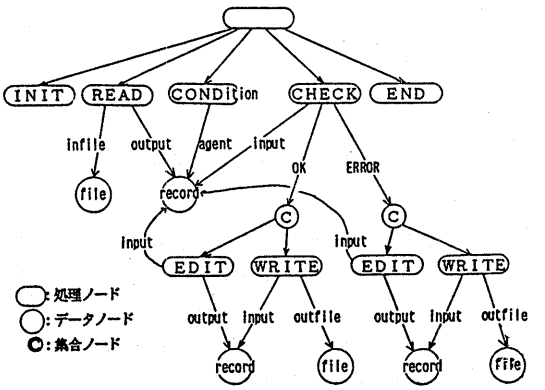

図３．処理パターンの例

（３）システムの構成

上記のような仕様書を入力し、処理パターンを用いた仕様獲得方式によるシステムの構成図を図４に示す。

①仕様書編集部

プログラム仕様書を編集し、表示する。

②単語登録部

プログラム仕様書から、その仕様書の中で独自に使われている単語（ファイル名、データ項目名、節名など）を抜き出し、ユーザ辞書に一時的に（コード生成が終わるまで）登録する。

③ファイル記述登録部

固定化したファイルのファイル記述をユーザ辞書に登録する。

④仕様文解析部

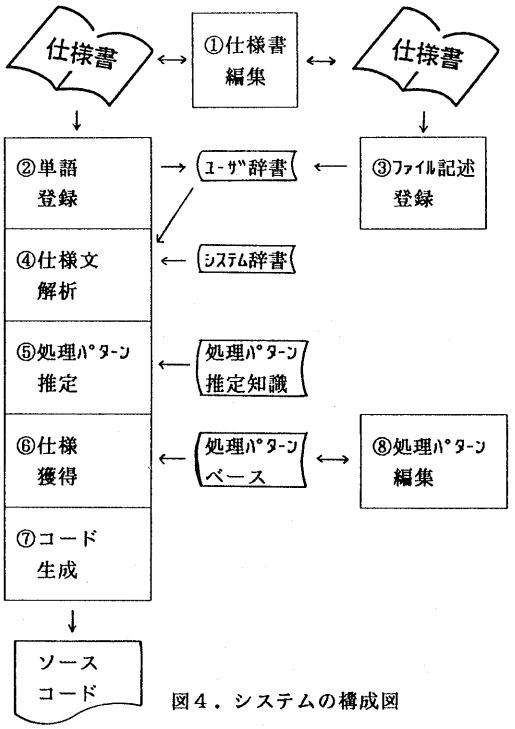

図４．システムの構成図

プログラム仕様書中の仕様文を解析して、ＡＤＳ（A Dependency Structure: ノードとリンクからなる依存構造）による仕様記述のネットワーク表現を作成する。仕様文は処理の概要の部分「第１章」と処理全体のアルゴリズムの部分「第３章」に分かれており、それぞれに対して概要ＡＤＳ、処理ＡＤＳを作成する。

⑤処理パターン推定部

処理パターン推定知識を用いて、概要ＡＤＳから求めるプログラムの処理パターンを推定する。

⑥仕様獲得部

推定された処理パターンへ該当する情報を処理ＡＤＳから取込んで、プログラム全体の情報をもつプログラムＡＤＳを作成する。

⑦コード生成部

プログラムＡＤＳからソースコードを生成する。

⑧処理パターン編集

処理パターンを編集し、ＡＤＳ表現に変換する。これはシステムの変更容易性のためのものであるが、将来はユーザ定義処理パターンの登録のためのものとする。その際は、仕様文解析の文法、システム辞書、処理パターン推定知識などの自動的な更新機能が必要となる。

３．仕様文解析

仕様文解析部は、プログラム仕様書の処理の概要の仕様文、全体アルゴリズムの仕様文を、辞書を参照しながらボトムアップ・パーザを用いて解析し、それぞれ概要ＡＤＳ、処理ＡＤＳにまとめる。

（１）辞書

システム辞書およびユーザ辞書の２つの辞書を用いて解析を行う。

システム辞書には、助詞、助動詞などの基本的な単語と対象分野における基本的な動詞、名詞が納めてある。これらの単語は、実際の仕様書を調査して選び出したものである。

ユーザ辞書は、システムにとって特定できない単語を納めるものである。業務あるいは組織に固定化した単語は予め登録する。入力仕様書独自に使われている単語は仕様文の解析を始める前に登録する。これは仕様書のもつ構成を利用して、ファイル名、レコード名、データ項目名、部分処理名、条件名などを抜き出して登録するものである。

（２）解析処理

仕様文の解析処理は、ユーザ辞書、システム辞書の順に参照しながらボトムアップ・パーザを用いて行う。

システム辞書には用言に関し格情報をもつ。これも実際の仕様書の調査に基づくもので、対象分野向けになっている。例えば、「～ｖ～場合」において通常ｖは一般の用言だが、この分野ではアルゴリズムを記述するため検査、判定後などの状態を表わす用言であることが多い。従って、このような用言にだけこの「場合」を表わす格をもたせている（図５参照）。

解析処理の結果として作られたＡＤＳは、入力の仕様文の関係節や重文の構造を反映したものとなっている。後の仕様獲得の便宜のために、このＡＤＳを「入力する」、「編集する」などの処理を表わす

深層格	格助詞	意味素性
agent	が	record
condition	を	condition
action	—	case（場合）

図５．動詞「満たす」の格情報

動詞を中心とした単文に対応した構造をもつＡＤＳに変換する（図６参照）。

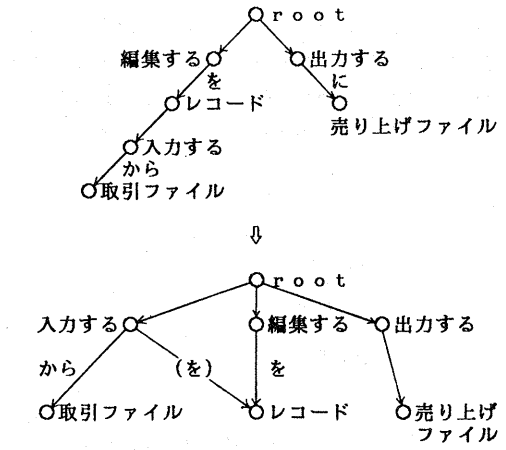

図６．関係節の単文化

処理の記述が長くなるような場合、部分を表わす名詞を文章中で導入して、階層的に記述することがある。例えば、「～エラーのない場合、Ａ処理を行う。エラーの場合、・・・。Ａ処理では、・・・」における「Ａ処理」である。このような単語は解析処理で一旦は未知語となるが、処理を表わす名詞としてユーザ辞書に登録して解析を続ける。

図７に図２「第３章」の仕様文の解析結果のＡＤＳを示す。

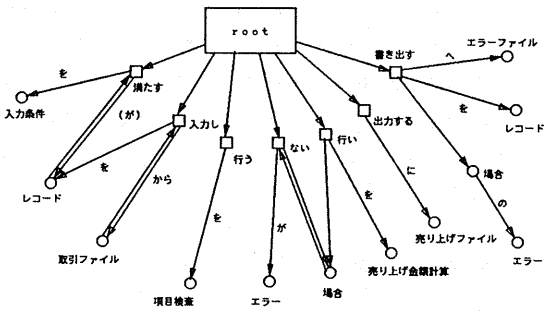

図７．仕様文解析結果のＡＤＳの例

４．処理パターン推定

対象分野のファイル操作プログラムの処理パターンの例をあげると、

ａ…ファイルの編集処理

ｂ…ファイルのレコード検査処理

ｃ…マスタファイルのメンテナンス処理

などがある。

（１）キーワードによる処理パターン推定

疑似日本語は自然な仕様記述を可能とするようなものなので、同じ処理パターンに対応する「概要」の仕様文には様々な言い回しがある。逆に、類似した仕様文でも異なった処理パターンに対応することもある。従って、言い回しを反映したかたちの概要ＡＤＳの構造に基づいて処理パターンを推定することはできない。そこで、処理を特徴付ける単語（キーワード）の有無に着目して推定する。このようなキーワードとしては処理を表わす動詞、処理対象を表わす名詞などが挙げられる（２．（１）参照）。

このようなキーワードの有無も言い回しの影響を受ける。これに対しては、記述の意味を考慮して暗黙のキーワードを推論する。例えば、

Ａ：「取引ファイルより売り上げファイルを作成する。」

Ｂ：「取引ファイルを入力し、売り上げファイルを作成する。」

は同じ事を記述していると考えられるが、Ａの記述には「入力する」というキーワードはない。しかし意味を考慮すれば記述Ａに対して「取引ファイルを入力する」という処理があることを推論でき、キーワードの有無についてＡ、Ｂとも同等になる（図８参照）。

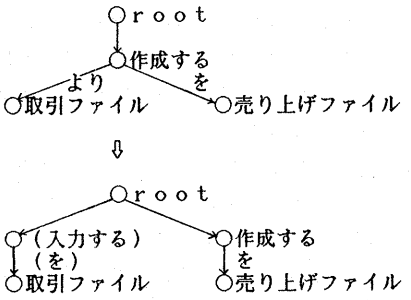

図８．キーワードの推論

（２）処理パターン推定知識

処理パターンを推定するための知識としては、「ある処理パターンを記述するためには、どのようなキーワードが必要で、また、そのキーワードはど

のくらいその処理パターンを特徴付けるか」を表にしたものである。

例えば、上記ａ、ｂ、ｃの処理パターンを想定した場合、「入力」のそれぞれを特徴付ける度合は１／３である。「メンテナンス」はｃに対して１であり、ａ、ｂに対してはその処理パターンであることを否定する（表１参照）。

表１．処理パターン推定知識の例

処理ﾊﾟﾀｰﾝ / ｷｰﾜｰﾄﾞ	a	b	c
入力	1/3	1/3	1/3
編集	1/3	1/3	1/3
検査	*	1/2	1/2
メンテナンス	*	*	1

＊：その処理パターンであることを否定することを示す。

５．仕様獲得

処理パターンは、処理全体のアルゴリズムを構成する基本的処理単位（以後処理単位と呼ぶ）を処理単位間の関係を示すリンクで結合し、また、処理対象であるデータのノードをリンクで結合したものである（図３参照）。これに対して、動詞を中心とした単文の列の形をした処理ＡＤＳ（３．（２）参照）から、処理単位ごとに情報を取り込む。これらのノードとリンクは、スロット（属性名と属性値の対）の集りから成り、属性値が未定義のスロットの値を求めることで仕様を取り込む。

仕様獲得は、まず処理ＡＤＳに含まれる各単文についてその動詞の格の整理を行い、その文が意味する概念を意味ユニットとしてまとめる。このようにして得られた意味ユニットの列について、意味ユニット間の関係を推論する。これによって、処理全体のアルゴリズムにおける各意味ユニットの文脈を明確にする。次に、処理パターンを構成する各処理単位について、対応する文脈をもつ意味ユニットを取り込むことで、全意味ユニットの関係付けを行い、処理対象の具体的なファイル名、レコード名や対応する部分処理の具体的な節名、節番号（２．（２）参照）を明らかにする。最後に、部分処理を表わす節の疑似コードを解釈する。

（１）意味ユニット作成

意味ユニットは処理単位と同様に処理のノードと処理対象のデータのノードをリンクで結合したものである。これらノードとリンクのスロットにその単文がもつ情報を、動詞の格情報に従って整理し、それの意味する概念を明確にする（図９参照）。

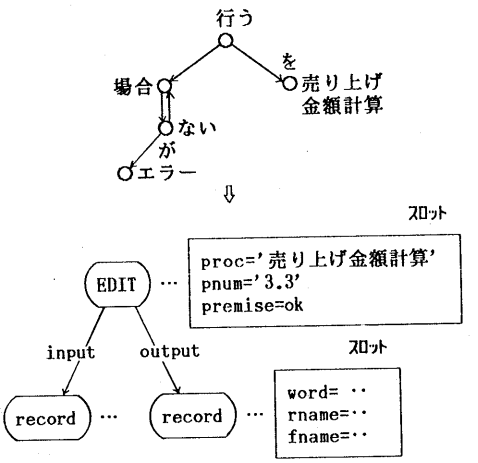

図９．意味ユニット作成の例
proc,pnum は節名、節番号を、premise=okは「エラーがない場合」を示している。レコードに関する記述が含まれていないので、レコードのスロットの属性値は未定義のままである。

（２）意味ユニット間の関係の推論

処理の前提条件を示す記述（例「エラーの場合、Ａ処理する」）、処理の階層化の記述（例「Ａ処理では、・・・Ｂ処理する」）に対して、前提条件の及ぶ範囲、意味ユニット間の階層性を明確にし、個々の意味ユニットのスロットに設定する（図１０参照）。図９中premise は前提条件を表わすスロットである。

＜条件１＞の場合、＜処理１＞を行う。
＜条件２＞の場合、＜処理２＞を行う。
＜処理１＞では、＜条件３＞の場合
＜処理３＞を行い、＜処理４＞を行う。
＜条件１＞かつ＜条件３＞
＜条件４＞の場合＜処理５＞を行う。
＜条件１＞かつ＜条件４＞

図１０．意味ユニット間関係推論の例

（３）処理パターンへの意味ユニットの取り込み

処理パターンを構成する処理単位に対する意味ユニットの選択は、処理単位の処理の名前と意味ユニット名、処理パターンの構造から得られる前提条件と意味ユニットの前提条件を表わすスロットの値の合致性をもとに行う。

仕様の取り込みは、処理単位がもつスロットの属性名に対する属性値を意味ユニットから取り出すことで行う。このとき既に値が定まっている場合は、意味ユニットがもつ値と矛盾がないか調べる。矛盾した場合は、選択した意味ユニットは不適当だとして、意味ユニットの再選択を行う（図１１参照）。

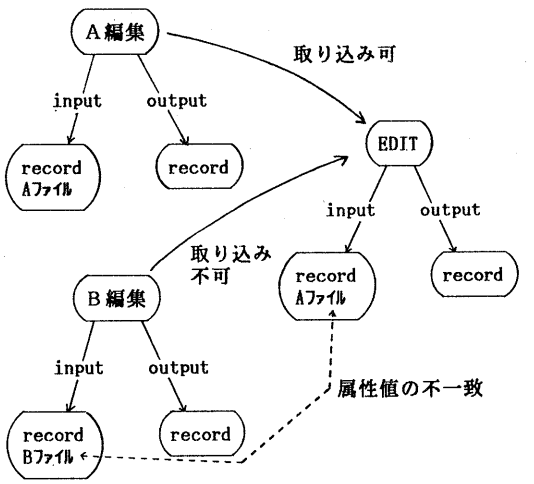

図１１．意味ユニット取り込みの例

図９の意味ユニットは、上記のような選択により図３における CHECKノード下のEDITノードに取り込まれる。この意味ユニットにおいてはその処理（売り上げ金額計算）の入力レコードは不明だったが、この取り込みによって取引ファイルのレコードであることがわかる。これは、この入力レコードのノードを出力レコードのノードとして共有するREADノードへの、「入力条件を満たすレコードを取引ファイルから入力し」から得られた意味ユニットの取り込みの時点で、取引ファイルのレコードであることがその共有ノードのスロットに設定されることによる。

６．評価

ＳＡＧＥのプロトタイプをＭＥＬＣＯＭ－ＰＳＩ上に開発した。プロトタイプは図４における仕様書編集部、ファイル記述登録部、処理パターン編集部を除いた構成となっている。また、システム辞書、扱える処理パターンは制限したものになっている。

プロトタイプをいくつかのプログラム仕様書に適用した結果、十分な速度性能を得た。図２に示した仕様書例に対しては２～３分費やした。

また、処理時間の測定結果によれば、仕様文解析に約６０％の時間を費やしており、仕様文の解析処理の効率化が必要であることが判明した。

７．おわりに

疑似日本語で書かれたプログラム仕様書から、事務処理分野のバッチ型ファイル操作プログラムのソースコードを生成するプログラム自動生成システムＳＡＧＥについて述べた。その仕様獲得の方式は、処理パターンに基づいて仕様文から情報を取り出すものである。プロトタイプ開発・評価では十分な速度性能を得ることができた。

今後の課題としては、仕様文解析処理の効率化を考慮して、システム全体を開発することが挙げられる。

参考文献

〔1〕 土井、他：プログラム自動生成システムにおける仕様書理解の一手法、情処第３４回全国大会、２Ｔ－５．

〔2〕 行徳、他：プログラム自動生成システムＳＡＧＥにおける仕様文解析法、情処第３５回全国大会、３Ｘ－４．

〔3〕 鈴木、他：プログラム自動生成システムＳＡＧＥにおける仕様獲得法、情処第３５回全国大会、３Ｘ－５．