UDC 681.3

# 部品合成によるシステム生成及びシステムテスト支援ツール

## Component-based System Generation and System Testing Support Tool

高柳信夫*(1) Nobuo TAKAYANAGI
小谷千里*(2) Chisato KOTANI
久保真哉*(3) Shinya KUBO
野口正浩*(4) Masahiro NOGUCHI

**抄 録**

GUI(Graphical User Interface)を備えるアプリケーションシステムを，短期間かつ容易に作成することを支援するツールとして，システム生成支援ツール及びシステムテスト支援ツールを開発した。システム生成においてはオブジェクト指向技術を積極的に採り入れて部品合成によるアプリケーションシステムの生成を実現し，テスト支援においてはテスト仕様書に基づくシステムテストの自動実行を実現している。これらのツールにより，ユーザはプログラム部品(オブジェクト)を視覚的に組み合わせることで，アプリケーションシステムの作成から仕様書作成，テストまでを効率よく行うことができる。

**Abstract**

Authors have developed an application generator and a system testing tool to support rapid development of application systems with graphical user interfaces. The former tool can generate application systems through component synthesis, adopting object oriented technology, while the later tool can realize automatic system test execution based on automatically generated test specifications. These tools enable software engineers to efficiently build application software, generate its specifications, and test it, by visually combining prepared program components (i.e. objects).

## 1. 緒 言

計算機システムのダウンサイジング化，オープン化の流れは，よりユーザフレンドリなシステムの構築を要請するようになってきた。また，ハードウェアやソフトウェアの進化は目覚ましく，年に数度のバージョンアップは，もはや珍しいことではなくなっている。

このような状況の中で，ソフトウェア開発側への要求は，開発工期の短縮，初期段階での確実な仕様確認，品質の向上，使いやすく分かりやすいGUI(Graphical User Interface)機能の搭載など，開発に必要な技術や負荷は高まる一方である。従って，ソフトウェア開発を支援するツール，中でもGUIベースのアプリケーションシステムに対する開発支援ツールが重要となる。

本報告では，著者らが研究開発した部品合成によるシステム生成*1ツール，システムテスト支援ツール，及びそれらの連携について述べる。前者はGUIベースのアプリケーションシステムを迅速，かつ簡単に構築することを支援するツールである。後者はそれを効率よく網羅的にテストすることを支援するツールである。

## 2. GUIアプリケーションの開発

GUIアプリケーションシステムの開発は，一般に以下のような特徴を有している。

1)GUI部品*2そのものは再利用性がかなり高い。また，システム化対象領域の分析を十分に行えば，再利用の範囲を更に拡大することができる。

2)GUI設計は自由度が大きく，代替案が無数に存在する。そのため，試行錯誤を繰り返すプロセスが必要となる。

3)GUI部品群と，そこから呼ばれるプログラム群との統合の方法はパターン化が可能である。

4)統合作業そのものは複雑で欠陥が混入しやすい。

開発においては，システムを構成する要素をあらかじめ部品化して

*(1) エレクトロニクス・情報通信事業部 システム研究開発センター 研究員
神奈川県相模原市淵野辺5-10-1 〒229-8551 ☎(0427)68-6101
*(2) エレクトロニクス・情報通信事業部 システム研究開発センター 主任研究員
*(3) 大分製鉄所 生産管理部 マネジャー，エレクトロニクス・情報通信事業部 システム研究開発センター兼務
*(4) エレクトロニクス・情報通信事業部 システム研究開発センター 主任研究員

*1 システム生成：ここでいうシステムとは，コンピュータ上で何らかの処理を実現するプログラムの集合体としてのソフトウエアシステムのこと。システム生成とは，何らかのツールを用いてシステム作成を(部分的に)自動化して，システムを生成することを指す。

*2 GUI部品：ボタンやメニューといったGUIを構成する基本的な要素を実現するプログラム部品。

おき，それらの組み合わせ方を開発者に定義させて，その通りに自動合成する手段があれば，アプリケーションシステムをすばやくかつ品質よく組み上げることができ，開発上大変有効である。

この方法によれば，部品同士の統合ロジックの部分は人手が介入しないため，高品質を保証することが可能である。また，部品自身も，再利用を重ねるうちに品質は向上していく。しかしながら，高品質な部品を組み合わせてシステムを構築しても，システム全体としての品質は必ずしも保証されない。これは，組み合わせて初めて発生する類の障害が存在し得るからである。従って，高品質な部品を組み合わせたシステムであっても，システムテストが欠かせない。

ところが，GUIをベースとしたシステムのシステムテストは，困難な作業である。これは，ユーザとの対話処理を中心としたイベント駆動方式であるGUIの特徴によるところが大きい。すなわち，GUIではユーザによる操作の自由度が高いため，それに応じて，必要となるテストケースが指数関数的に膨れ上がってしまうためである。そこで，システムテストを可能な限り自動化するためのテストツールの出現が強く望まれる。

テストの自動化は，テストツールに対し，テスト対象の仕様をいかに正確で詳細に与えることができるかに強く依存している。これをシステム生成ツール側から与えることが可能であれば，テストの自動化は飛躍的に進むことになる。

このように，部品合成によるシステム生成とシステムテスト支援ツールにより，品質の高いシステムを短期間で構築することが可能となる。そこで，著者らは，部品合成によるシステム生成ツール及びシステムテスト支援ツールを開発してきた。これらのツールは，UNIX*3，X Window System*4上で動作するアプリケーションをターゲットにしており，両ツールが連携することで，システム生成からテストまでを効率よく行うことができる。次に，これらのツールについて概説する。

## 3. 部品合成によるシステム生成ツール

### 3.1 概　要

本ツールSoftware assembly in Graphical environment(以下SGと略記する)では，GUI環境の上で部品を組み合わせて目的とするソフトウェアを構築していく。SGでは，GUI環境下でシステムを手早く作成することを可能とするために，部品合成によるアプリケーションシステムの作成を実現するばかりではなく，ソースコード*5や仕様書の自動生成も可能である。プログラム部品*6としては，GUIを構成する部品(GUI部品)とアプリケーションのロジックを構成する部品(アプリケーション部品)の2種類を扱う。GUI部品とアプリケーション部品を同格に扱う点が，GUIビルダとの大きな違いである。また，メッセージリンクと呼ぶ部品間でメッセージを伝搬する機構を備えており，このメッセージリンクを設定することで，システムの動作を定義することができる。この方法では，GUI部品とアプリケーション部品を明確に分離するとともに，その間の接続をメッセージリンクで表現するため，拡張性や保守性が良好な層構造を自然な形で実現できる利点がある。更に，仕様書の生成機能を持つ点が，他の商用ツールにはない特徴である。**図1**に機能と処理フローの概要を示す。

作成可能なアプリケーションの範囲は，用意する部品に依存するため一概には言えない。例えば，FAシステムでは，GUI部品としてメータ部品やチャート部品，アプリケーション部品として信号データI/O部品や制御ロジック部品などが必要であろう。また例えば，OAシステムでは，テキスト入出力部品や高速検索部品が，金融システムでは，日付や金額表示に特化したラベル部品やDB接続部品などが必要となるであろう。適用分野に応じた高品質な部品を豊富にそろえることが重要である。はん用的な部品を数多く用意できる分野の方が，生産性の向上に効果的である。**図2**にSGの画面例

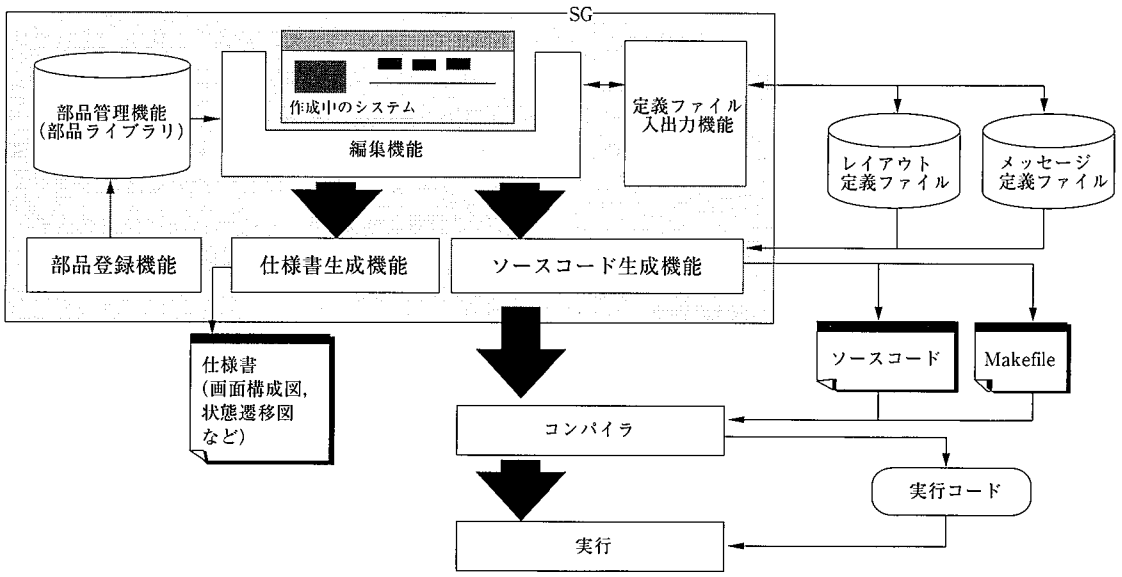


**図1**　SGの機能と処理フローの概要

*3　UNIXは，X/Open Company Limitedが独占的にライセンスしている米国並びに他の国における登録商標。

*4　X Window Systemは，The Open Groupの商標。

*5　ソースコード：プログラムをプログラミング言語を用いて表現したもの。基本的には，プログラム開発者によって書かれる。一般的には，ソースコードそのものは実行することができず，実行形式に変換することによって，実際に動作するプログラムとなる。

*6　プログラム部品：プログラム中の汎用的な要素または部分を部品としたもの。

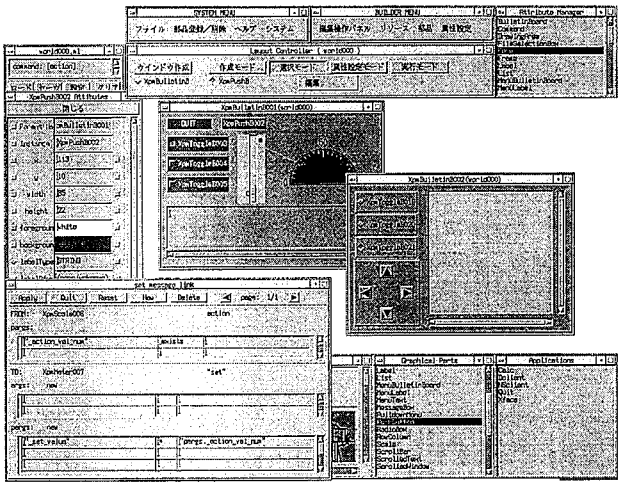

**図2**　SG画面例

を示す。なお，SG自体もSGで作成されている。

### 3.2 機　　能

SGの主要な機能としては，部品を選択しメッセージを設定する編集機能，部品の登録機能，ソースコードの自動生成機能，仕様書の自動生成機能の四つがある。以下に各機能と特徴について述べる。

(1)部品合成と動作定義

前述の通り，GUI部品とアプリケーション部品を組み合わせ，部品間でやりとりされるメッセージを設定することで，システムの構成と動作を定義する(**図3**)。これにより，既存の部品を組み合わせるだけで，実際に動作するシステムを構築することが可能となる。

実際の作業としては，画面上でマウスを使ってGUI部品をレイアウトし，更に，画面上の部品間を線で結ぶことで部品間のメッセージを設定する。ほとんどの作業は，マウスのみで可能になっている。

(2)部品の追加登録

ユーザが自ら部品を作成し追加していくことにより，対象とするドメインの部品を蓄積することが可能となる。また，部品作成支援として，GUI部品作成については，専用の作成支援ツールを用意している。アプリケーション部品については，ユーザが用意した関数を部品として扱えるようにするために，インタフェース部分を自動生成し部品化する機構を備えている。

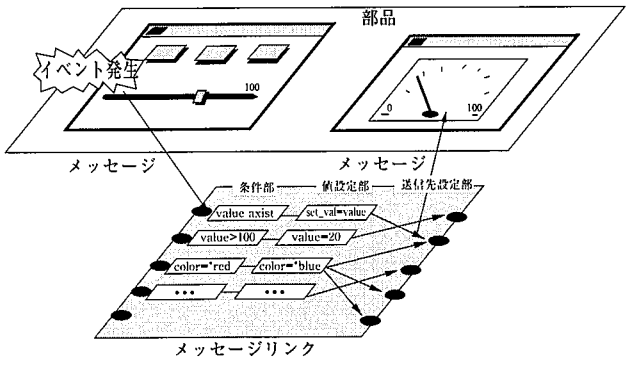


**図3**　部品間のメッセージの伝播

(3)作成中の動作確認

プロトタイプの作成時には，まさに"試しながら作る"ことが要求されるため，アプリケーションシステム作成中にいつでもその動作を確認可能としている。

(4)ソースコードの自動生成

一般的なGUIビルダでは，ユーザが必要な部分を書き加えることを前提としたソースコードスケルトンのみを生成する機能しか持たないものが多い。SGでは，スケルトンではなく，そのままコンパイルして実行することが可能なソースコード(C++[*7])をすべて生成する。また，他では見られないユニークな機能として，GUI部分を剥ぎ取ったソースコードの生成も可能にしている。この機能は，制御システムの開発に主に利用される。制御システムは，制御ロジック部品の再利用性が高いこと，部品の統合を決まったパターンで行えること，設計時のプロトタイピングが重要であることから，部品合成が効果的な分野である。そこで，SG上で制御ロジックの設計から実装までを行い，本番では不要なGUIを捨て去ってコンパクトで性能の良いシステムを生成することを狙っている。

(5)仕様書の自動生成

SGでは作成したシステムの仕様書を自動生成する機能を備えている。これにより，試行錯誤しながら作成したシステムについても，保守性が向上すると考えられる。更に，システムテストツール向けに，テスト仕様書を生成する機能も備えている。これにより，次章で述べるシステムテストツールにおいて，テストスクリプトを自動生成することが可能となった。

## 4. システムテスト支援ツール

### 4.1 概　　要

本ツールSystem Testing tool(以下STと略記する)は，X Window System上で稼働するクライアントプログラムに対するGUIテスト及びシステムテストを効果的に実施することを支援し，これらテスト工程の生産性及び品質の向上を促進するためのツールである。

STは，それぞれの目的に合わせた種々の形式のGUIテスト及びシステムテストに対し適用可能であるが，特にストレステスト[*8]，リ

[*7] C++：プログラミング言語の一つ。

[*8] ストレステスト：テスト対象システムが正常に稼働できなくなる最大負荷条件を明確にすることを目的としたテスト。例えば，同時に100端末からアクセス可能な発券システムに対し，実際に100以上の端末から同時にアクセスしてテストする。

[*9] リグレッションテスト：修正を施されたテスト対象に対し，修正前に実現できていた機能が，修正によって縮退していないかどうか確認することを目的としたテスト。

グレッションテスト*9において，その効果を発揮することができる。通常の商用ツールには，特定のGUIライブラリを利用するか，あるいは，テスト支援用の拡張を施したXサーバでしか機能しないという制約がみられるが，STでは，それらの制約を排除したことに特徴がある。また，仕様検証用のテストスクリプトを，テスト対象の仕様書から自動的に生成することが可能であることも，他の商用ツールには見られない大きな特徴となっている。

### 4.2 システム構成

STは，テスト対象システムに対する，マウス及びキーボード操作の記録／再現と仕様検証を行う記録／再現部と，テスト対象仕様を基に仕様検証用スクリプトを自動生成するスクリプト生成部，及び，テスト対象仕様をディスプレイ画面上にグラフィカルに表示する仕様表示部，そして，スクリプトの作成，編集，及び，インタラクティブな実行を可能にするスクリプト編集部から構成される。また，記録／再現部は操作記録機能を受け持つ部分と，操作再現機能を受け持つ部分，及び，ツール全体のGUI部から構成される。

STに対する入力としては，テスト対象に関する3種類の仕様ファイル及びテストスクリプトファイルがあり，STからの出力としては，テスト結果レポートがある。また，ST内部における中間生成ファイルとして，状態遷移グラフ*10データファイル，構造グラフデータファイルが存在する。テストスクリプトファイルには，操作記録によるもの，自動生成によるもの，及びプログラミングによるものが用いられる。これらの関係を**図4**に示す(スクリプト編集部及びGUI部は図中に明記していない)。

### 4.3 機　能

STに具備された主要機能について以下に述べる。

(1)GUI操作記録/再現機能

テスト実行作業を支援するためのものであり，GUIテストの自動化においてまず第一に必要とされるものである。具体的には，マウス及びキーボードを利用してテスト対象に対して行った操作をテストスクリプトとして記録し，随時その内容を人手によらず自動的に再現するための機能である。また，再現時には，記録ではなくプログラミングによって直接作成したテストスクリプトを用いることも可能である。

(2)仕様検証用スクリプト自動生成機能

テスト設計及びテストケース作成作業を支援するためのものであるが，その実現の困難さからGUIテスト及びシステムテストの自動化においては最後まで残されてきたものである。STでは，テスト対象の構造仕様*11，状態遷移仕様*12，画面仕様に着目し，仕様で記述されたすべての状態と状態遷移を洩れなく網羅させるテストスクリプトを自動生成することを実現した。具体的には，テスト対象となるアプリケーションの仕様を基にして，アプリケーションに対して状態遷移を網羅するような操作を行わせ，その実行中にアプリケーションの状態を調査し，仕様に合致するかどうかを検証することが可能である。

(3)テスト対象仕様表示機能

テスト対象のアプリケーションシステムに対する構造仕様と状態遷移仕様をグラフィカルに表示し，テスト実施者の理解を容易にするためのものである。具体的には，構造仕様及び状態遷移仕様を，画面上にグラフィカルなビューとして表示する機能である。これにより，テスト実施者は，アプリケーションがどの部品でどのように構成されており，どのように状態を遷移して行くかをイメージしやすくなる。

(4)インタラクティブ実行機能

テストデータ作成作業，テスト実行作業及びテスト結果判定作業を支援するものであり，視覚的な情報を提供することにより進行中のテスト内容に対するテスト実施者の理解を深めさせるものである。専用のテストスクリプトエディタを用い，編集中のスクリプト内容を任意の単位で抽出し，インタラクティブにテスト対象への操作を行うことが可能となる。

### 4.4 テストスクリプト言語

本ツールは，テストスクリプトに記載されたコマンド列を理解し，その内容に従って，GUIテスト及びシステムテストを自動的に実行するものである。準備された数十種類のコマンドを適切に組み合わせることによって，テスト対象システムに対し，人間が行うことのできるすべての操作を，STによって自動的に実施することが可能となる。

ST用に開発したテストスクリプト言語は，Tcl言語[1)]を拡張した

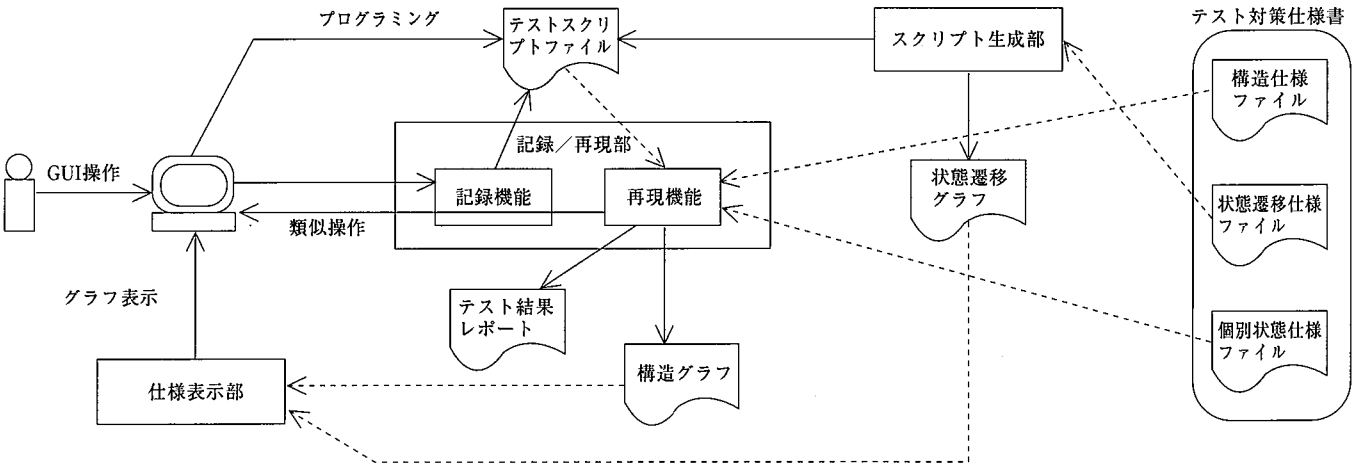


**図4**　ST全体概要図

*10 状態遷移グラフ：状態遷移を，状態をノード，状態間の遷移をエッジで表現したグラフ。

*11 構造仕様：テスト対象のアプリケーションシステムが，どのプログラム部品をどのように組み合わせて作成しているかという構造上の仕様。

*12 状態遷移仕様：テスト対象のアプリケーションシステムが，どの入力(イベント)により，どの状態からどの状態に遷移するかという仕様。

```
#ファイルからデータを読みとり，テキストエリアへ入力する。
MoveWin @(Sample) @(100,0)  t100      #GUI初期レイアウト
Raise   @(Sample)           t1000     #の設定
set fd [open Sample.dat r]            #データファイルを開く
set WTIME 100                         #イベント発生時間間隔
While ¦¦ [eof $fd] ¦ ¦                #ファイル処理のループ
  for ¦set i 1¦ ¦$i<10¦ ¦incr i¦       #1行をdataに読み込む
      gets $fd data
      if ¦$data == "Right"¦ ¦set product "R"¦
      if ¦$data == "Left"¦ ¦set product "L"¦
      #Sampleウインドウのi番目のボタンをクリック
      Click          @(Sample.button$i) t$WTIME
      #Sampleウインドウのi番目のテキストに文字列dataをタイプ
      Type    '$data'   @(Sample.test$i) t10
  ¦
  •••
¦
Closed $fd                            #ファイルを閉じる
exec echo Sample_finished >@ stdout   #終了メッセージ出力
```

**図5**　テストスクリプトファイル例

ものであり，GUI操作及びテスト作業に特化したコマンドを組み込んだこと，及びオブジェクトベースの表記を可能とし，テストスクリプトの可読性及び保守性を高めているところに特徴がある。

STで実行可能なテストスクリプトの一例を**図5**に示す。

## 5. 部品合成によるシステム生成ツールとシステムテスト支援ツールの連携

前述の通り，部品単体の品質がある程度保証されていても，それらを組み合わせた結果の品質は保証されない。従って，組み合わせたことによって初めて起こる不具合を検出するために，システムテストが必要となる。

SGによるシステム生成では，SG側でGUIの構成ばかりでなく，状態遷移に関する情報も保持しているため，システムに関する正確な情報をテストツールに伝えることができる。そこで，SG側でテスト仕様書を自動生成し，ST側でこれを元にテストスクリプトを自動生成することを可能にした。具体的には，SG側で，**図6**に示すように，構造仕様，状態遷移表，個別状態仕様を自動生成してテストツール側に渡している。これにより，状態遷移を網羅する形で仕様検証を行うことを目的としたシステムテストの自動化が実現している。

## 6. 適用状況

SGは，これまでいくつかのプラットフォームに移植され，複数の実システムの構築に利用されている。例えば，新日本製鐵八幡製鉄所では，SGをその開発初期から利用してきた。そして現在では，SGをコアとした制御システム構築ツールGOOD[2]にまで発展させ，制御システム開発の有力な武器として役立てている。また，新日本製鐵大分製鉄所では，厚板の出荷命令業務の負荷削減，出荷命令の内容改善を目的に，厚板出荷命令作成システム[3]をSGを使って開発し，1995年11月より順調に実機稼動させている。

STは，開発中の頃より著者らのグループでソフトウェア開発に利用してきており，当該ソフトウェアの品質保証はもちろんのこと，商用GUIライブラリの欠陥の検出に至るまで威力を発揮してきた。実プロジェクトへの典型的な適用例としては，エレクトロニクス・情報通信事業部　金融システムソリューション部が請け負った200万ステップにも及ぶ大規模システム開発で"住友銀行オフバランスプロジェクト"[4]がある。このプロジェクトでは，GUIテスト及びシステムテストでSTを繰り返し利用し，故障発見及びテスト工期の短縮に大きく貢献した。

## 7. 結　　言

本報告では，GUIベースのアプリケーションシステムの構築を支援するツールとして二つのツールについて述べた。SGでは部品合成によりシステムを生成し，STではシステムテストを部分的に自動化している。また，両者の連携により，システム作成からテストまでの支援をある程度まで実現できた。これらのツールは，開発中から新日本製鐵社内ユーザで試用し，その声を反映してきた。その結果，実システムの構築に利用され，評価を得ている。

### 参照文献


1) Ousterhout, J.：Tcl Overview. University of California at Berkeley, 1993
2) 関口修　ほか：新日鉄技報，364, 19 (1997)
3) 宮永義英，川邊利一：材料とプロセス，9 (5), 957 (1996)
4) 日経コンピュータ，(412), 156 (1997)


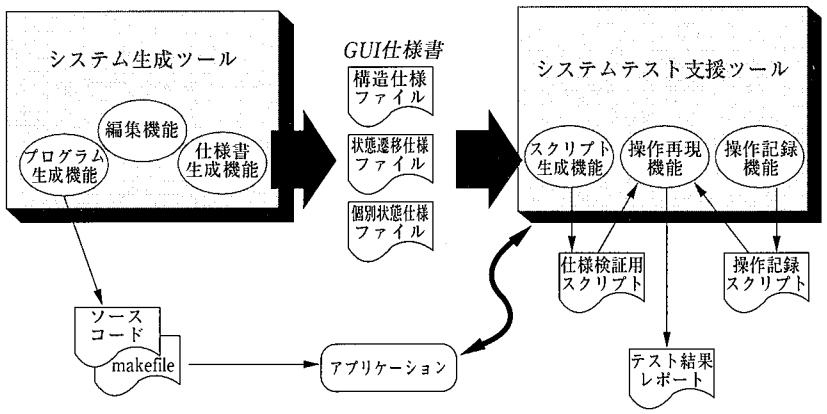


**図6**　システムテストの自動化