ソフトウエア工学 16－3
（1980. 11. 17）

ＳＴＥＰＳのプログラミング技術とそのツールＩＳＤＴ

東　基衛　　三野村圭右　　紙谷　進　　高橋雅昭
（日本電気株式会社　情処営支本、応用プログラム部）

はじめに

ソフトウェアの生産性，保守性ならびに品質向上のための手段として登場したソフトウェア工学は，成果を上げつつ対象領域を広げて来ている．すなわちプログラムを対象とする構造的プログラミング技法や，プログラムの正しさの検証技術，さらにはプロジェクト管理技術に至るまでの多岐にわたっている．

しかし，コンピュータユーザの大多数はこれらの技術の恩恵を受けているとは言い難いのではないだろうか．それは，中小型領域のコンピュータにより，数多くの類似の事務処理システムを開発，運用しているユーザである．これらのユーザにおいては，既存システムの保守改造に多くの工数が費やされ，新規システムの開発要求が積滞しているというのが実状である．さらに，必要とする要員の確保も次第に困難になりつつある．したがって，彼らの最大の関心事は，いかに効率よく，かつ容易に事務処理アプリケーションシステムを開発し，保守するかにあるといえる．

ところが，正しさの証明可能なプログラムを作るためには，高いスキルと多くの工数を必要とするし，一方ISDOSのPSL/PSA[3]やSREMのRSL/RSA[1]などはその実現に大規模なコンピュータシステムを必要とする．これらが，ソフトウェア工学が大多数のコンピュータユーザのニーズに必ずしも応えていない理由となっている．

そこで，事務処理システム開発の生産性向上にすぐ効果があり，かつ導入し易いソフトウェア開発支援システムの開発が必要とされる訳であるが，その１つである当社のソフトウェア標準化システムＳＴＥＰＳ[2]は大きな成果を収めている．さらに，最近の支援システムのすう勢からコンピュータによる支援が不可欠となってきている．それも従来のバッチ型ではなく，誰でも手軽に使用できる対話型である必要が出て来ている．　ここに述べる対話型ソフトウェア開発ツールＩＳＤＴは，ＳＴＥＰＳの方法論に基づき，その経験をふまえ，最近のプログラム開発支援技術を取り込んだツールである．

1. 基本概念

1.1 事務処理プログラムの特徴

コンピュータによる事務処理の中核をなすものは，COBOLによるプログラムであろう．そこで事務処理プログラムの開発や保守に真に役立つツールを開発する為には，事務処理プログラムの持っている特徴を把握する必要がある．

その特徴とは，次のようなものである．

(1) 類似のプログラムが大量に作られる．
(2) 比較的単純で，小規模なものが多い．
(3) データの流れが大きな比重を占めている．
(4) データの記述が多く，同一のデータ記述が複数プログラムで共通に使用される．
(5) 長期間にわたり，繰り返し利用される．その間，企業の経営環境の変化に追随してプログラムの変更も多い．
(6) 開発者と利用者が，多くの場合異なる．

これらの特徴は物の生産にたとえれば，類似製品の反復受注生産に見られる．つまり，決して同一製品の大量生産ではないが，似たようなものをユーザの希望により少しづつ変えて作り，必要に応じて手直しもしていくというものである．このような生産形態において，効率化を実現する最良の方法は，部品の徹底的な標準化と，標準部品を利用した設計・製造工程の標準化であり，道具の標準化である．

そこで，ソフトウェアのライフサイクルの全体にわたり，一貫した思想で方法論，ドキュメント体系，プログラムモジュールなどを整備したソフトウェア標準化システムがＳＴＥＰＳである．STEPS によるソフトウェア開発作業は細かく分割されて標準化されており，各作業単位では，フォーム化された用紙を，例を見ながら埋めていけばよい．さらに，事務処理プログラムをパターン化した標準のプログラム ライブラリを用意している．

このＳＴＥＰＳの使用経験をもとに，対話型にソフトウェア開発を支援するツールとして，次のような概念を加えて，ＩＳＤＴが開発された．

(1) プログラムのデータ，制御，処理などへの分解
(2) 複数プログラムによるデータ記述の共用
(3) ワークステーションによるソフトウェアの開発
(4) プログラム設計に必要な情報のデータベース化による再利用

1.2 プログラムの構造と部品の概念

ＩＳＤＴでは，プログラムを形作っている部品への分解と，その部品の共用が大きな特長である．それでは，プログラムを構成している部品とは何であろうか．事務処理プログラムにおいては，それらは次の５つと考えられる．

(1) 外部データ記述
(2) ファイル記述
(3) 内部データ記述
(4) 制御構造記述
(5) 処理モジュール記述

図１は，これらの記述がプログラムを構成するようすを示している．

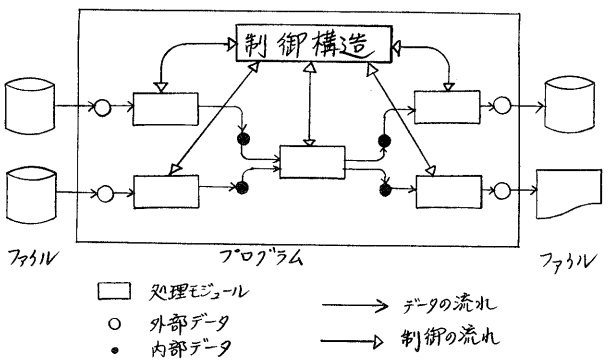


図１ 事務処理プログラムの構造概念図

ＩＳＤＴでは，これらの部品（コンポーネントと呼ぶ）を次のような仕様として記述し，データベース中に管理している．

(1) レコード仕様
　事務処理システムの一組の情報単位の構造を示し，上記の外部データ記述に対応する．
(2) ファイル仕様
　事務処理システムの情報がどのようにファイルされているかを示し，上記のファイル記述に対応する．
(3) 作業場所仕様
　プログラム中の処理モジュール間のインタフェースや，処理モジュール中で使用する為のデータ記述であり，上記の内部データ記述に対応する．

(4) プログラム構造仕様
　プログラム中の処理モジュールの実行順序の制御と，処理の準備および後始末を記述する．前述の制御構造記述に対応する．
(5) 処理モジュール仕様
　プログラム固有の処理，すなわちデータを入力し，加工し，データを出力するプロセスを記述する．前述の処理モジュール記述に対応する．
(6) プログラム仕様
　1つのプログラムを構成するコンポーネントと，そのインタフェースを示す為の仕様である．
(7) ジョブ仕様
　人間から見て意味のある1組のソフトウェアの単位であり，複数のプログラムからなり，プログラムの実行順序を示している．

2. ISDTの機能概要

2.1 ツールの動作環境

ISDTはオペレーティングシステムACOS-2のPWSS(Personal Working Station System)環境下で動作するツールで，1980年7月からリリースされている．

ISDTのユーザはワークステーションに座って画面との対話によりソフトウェア開発作業を進められる．画面はいわゆるTSSとは異なり，ユーザは画面全体に表示されるフォームの必要部分を日本語及びCOBOL言語で埋めればよい．従って画面単位の対話形式が基本であるが，フィールド単位の更新も容易にできる必要がある．また，1つの画面の入力が終了したら，指示を与えることにより，次の画面が表示される．この画面制御がISDTの特徴的なものである．

2.2 仕様の管理

ISDTは前述のように7種類の仕様を管理している．これらの仕様の記述量には増減があるので，1画面に全部を表示することはできない．そのため，各仕様にはいくつかのタイプのフォーマットを持った画面タイプと呼ばれる画面で定義，修正する．そして各画面タイプは，複数の継続画面から構成されている．さらに，画面は蓄えられている仕様の特定の部分に対応しているわけではなく，仕様の記述量の変化に従ってダイナミックに変化するようになっている．すなわち画面イメージそのものが蓄えられているのではなく，蓄えられた仕様を画面という枠を通して更新することになる．仕様と画面の関連を図2に示す．

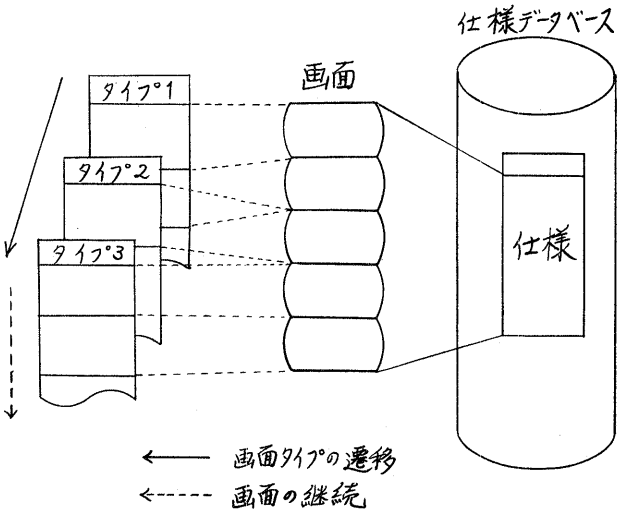


図2 仕様と画面の関連

2.3 ISDTの機能

ISDTの各機能を選択して実行の指示を与えるには，画面上で表1のISDTのコマンドを選択すればよい．主要な機能のいくつかについて詳しく説明する．

(1) 仕様の定義・修正機能

新たに仕様を定義する場合は，システムで提供されている標準の仕様をベースに修正を行えばよく，無駄なインプット作業は省くことができる．又標準仕様のかわりに，ユーザが作成した仕様をベースとすることも可能になっている．このようにISDTでは既にあるものを置き換えたり，追加したりする機能に重点が置かれる為，画面上の修正機能が充実している．

(2) プログラム生成機能

すべての仕様定義が終わると，ISDTは必要な仕様を組み合わせて標準化，構造化され，解読しやすいCOBOLソースプログラムを生成する．

(3) ドキュメンテーション，仕様の解析機能

ISDTではデータベースに蓄えられている仕様を，ドキュメントとして自動的に編集して出力する．これにより手書きのドキュメントがほとんど不要になり，ドキュメンテーションの省力化，標準化がはかれる．又仕様の不備や矛盾があればこのとき通知される．

ISDTの機能の概念図を図3に示す．

項番	コマンド名	コマンドの機能
1	DEFINE	仕様の定義・修正機能
2	ADD	定義された仕様に定義名を付与してDBに新規登録する機能
3	REPLACE	修正された仕様で，登録済の仕様を置換する機能
4	CANCEL	DEFINEコマンドの結果を取り消す機能
5	DELETE	登録済の仕様を削除する機能
6	RENAME	登録済の仕様の定義名を変更する機能
7	REFER	仕様の問い合わせ機能
8	GENERATE	プログラム生成機能
9	DOCUMENT	ドキュメンテーション・仕様の解析機能
10	GENJCL	実行用JCL作成機能

表1 ISDTのコマンド一覧

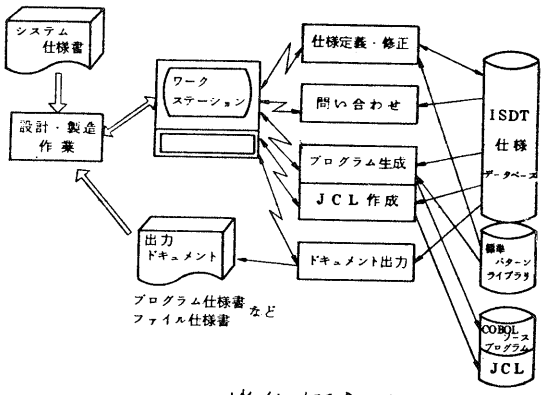


図3 ISDTの機能概念図

3. ISDTの利用

3.1 ISDTの位置づけ

ISDTは，STEPSのシステム開発標準に基づいてシステムを開発する際の，プログラミングまわりの作業をツール化したものである．ISDTの位置づけは図4のようになる．

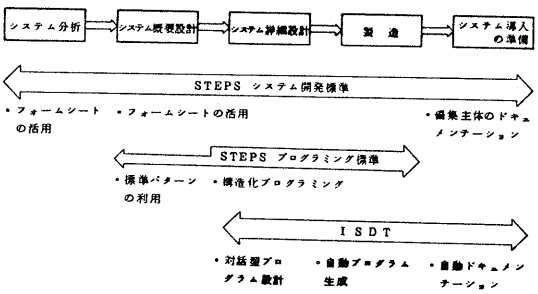


図4 ISDTの位置づけ

このように，ISDTではシステム概要設計フェーズの結果を受けて，それをプログラムの要求仕様として画面から対話型に入力するところから作業が始まる．続いて，対話型プログラム設計，対話型コーディング，仕様の解析，自動プログラム生成，自動ドキュメンテーションの順に作業を進めていく．

次節以降に，ISDTを用いたソフトウェア開発プロセスを，順を追って示す

3.2 対話型プログラム要求定義

システム概要設計フェーズの結果得られるシステム仕様書中からプログラムに必要な情報を，プログラムの要求定義として仕様毎に入力する．定義する仕様の順序や，仕様内での画面の順序は任意である．また，空欄にしておいて後から埋めることも可能である．この場合，仕様の解析機能によって未記入の部分をチェックすることができるので，空欄のまま忘れる心配はない．図5，図6はプログラム要求定義画面の例である．例えば図6のように，レコードの項目名称はこの段階ではプログラムと独立に日常語で記述すればよい．

3.3 対話型プログラム設計

要求定義の結果をさらに詳細化し，プログラム内部のデータおよび処理の方式を検討する．レコード仕様では図7のように，この段階でCOBOLプログラムで扱う項目標識や属性，レコード上でのレイアウトを決定する．

このようにISDTでは，プログラム設計でもCOBOL言語をそのまま用いる場合がある．なぜなら設計用の別の言語を用意するのは，ユーザの負担になるだけだからである．

プログラム構造仕様の検討では，ISPL(ISDT標準パターンライブラリ)としてシステムが提供する標準パターンの中から最も適したパターンを選び，そのプログラム構造仕様と作成しようとするプログラムの要求仕様との比較検討により，必要に応じて修正をほどこす．プログラム構造仕様には，制御構造の記述に，構造的プログラミングの基本制御構造の考え方をとり入れているので，プログラムの処理の流れの理解が容易になっている．

プログラム仕様では，独立に設計された各仕様，例えばファイル仕様や処理モジュール仕様のどれとどれを組み合わせて使用するかを記述すればよい．

図8は，プログラム仕様の一部として記述する，入出力レコード一覧である．

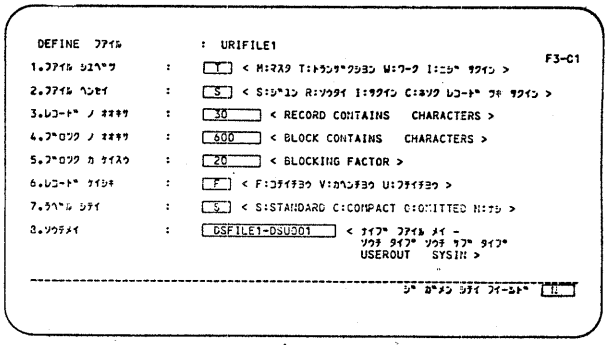


図5 ファイル定義オ3画面の例

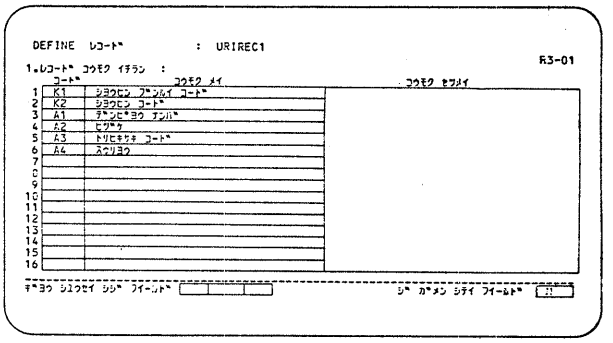


図6 レコード定義オ3画面の例

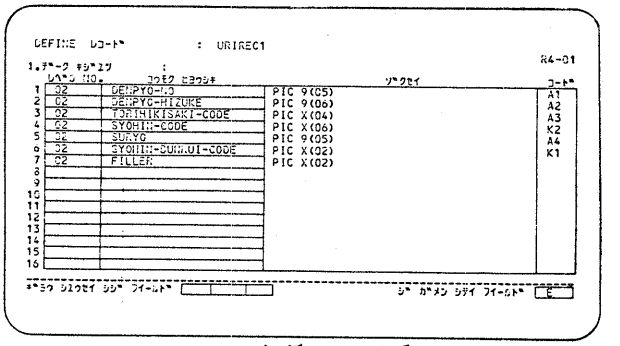


図7 レコード定義オ4画面の例

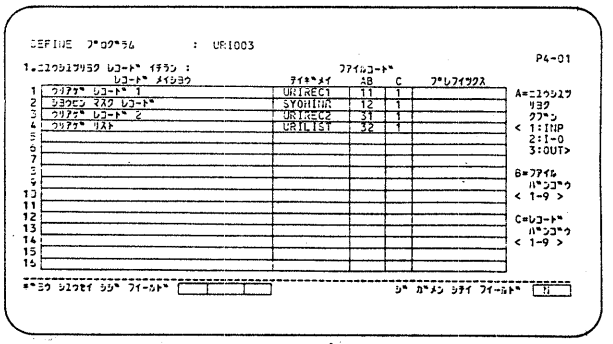


図8 プログラム定義オ4画面の例

3.4 対話型コーディング

プログラムの製造フェーズで行なう作業は，既にプログラム構造やレコードの記述はできているので，業務固有の処理と，その処理で用いる作業場所のコーディングだけを行なえばよい．プログラム設計時に，処理モジュール仕様や作業場所仕様の概略は定まっているので，それに従って画面に表示されるコーディングフォームに入力すればよい．

3.5 仕様の解析

画面から入力された仕様は仕様データベース中に蓄えられている．ISDTは，各仕様の未定義の部分を検出したり，仕様間に矛盾がないかを検査し，エラーメッセージとして出力する．例えば，次のような メッセージが出力される．

未定義の例： このプログラムには，ファイルが1つも定義されていない．

記述ミスの例： 上記のアスタリスク(*)の処には，指定してはいけないものが入っている．

関連チェックの例： 次の項目標識は，どのレコードにも関係しない．

3.6 自動プログラム生成

ISDTの特長の1つに，仕様の記述とドキュメンテーションだけでなく，実際COBOLのプログラムを作ってくれることがある．この時，標準パターンに対応する作業場所や処理モジュールは，システム提供の標準パターンライブラリ(ISPL)から自動的に引用され，直接COBOLコンパイルできるソースプログラムが生成される．生成されたプログラムは抽象化レベルに合わせた階層構造をもち，標準処理と，ユーザ定義の処理の分離が なされている．従ってSTEPSを用いて作成されたプログラム同様，見やすいものに なっている．図9に生成されたCOBOLソースプログラムの例を示す．

```
SEQNO.C...1....+....2....+....3....+....4....+....5....+....6....+....7..IDENT...
001700/                                                                 ISDT
001710 PROCEDURE DIVISION.                                              ISDT
001720 MAIN-PROCESS.                                                    ISDT
001730     PERFORM MAIN-INIT-PROC          THRU EXIT-MAIN-INIT-PROC     G1SU01
001740     PERFORM STD-MAIN-INIT           THRU EXIT-STD-MAIN-INIT      G1SSXX
001750     PERFORM PR-INIT-PROC            THRU EXIT-PR-INIT-PROC.      G1SU12
001760     PERFORM COLLATE                 THRU EXIT-COLLATE            G1PS01
001770     UNTIL   INP1-R1-KEY = HIGH-VALUE.                            G1PS01
001780     PERFORM STD-MAIN-FINISH         THRU EXIT-STD-MAIN-FINISH    X1ESX6
001790     STOP RUN.                                                    ISDT
001800**********************
001810 COLLATE.                                                         G1PS01
001820     PERFORM POSITION-INP2           THRU EXIT-POSITION-INP2      G2SS01
001830     UNTIL   INP2-R1-KEY  NOT <  INP1-R1-KEY.                     G2SS01
001840     IF      INP1-R1-KEY = INP2-R1-KEY                            G2PS01A
001850     PERFORM MATCH                   THRU EXIT-MATCH              G2PS01A
001860     ELSE                                                         G2PS01B
001870     PERFORM UNMATCH                 THRU EXIT-UNMATCH.           G2PS01B
001880     PERFORM INP1-READ-RTN           THRU EXIT-INP1-READ-RTN.     INP1RDR1
001890 EXIT-COLLATE.
001900     EXIT.
001910**********************
001920 POSITION-INP2.                                                   G2SS01
001930     PERFORM INP2-READ-RTN           THRU EXIT-INP2-READ-RTN.     INP2RDRA
001940 EXIT-POSITION-INP2.
001950     EXIT.
001960 MATCH.                                                           G2PS01A
001970     PERFORM EDIT-MATCH-PROC         THRU EXIT-EDIT-MATCH-PROC    G3PU01
001980     PERFORM OUT1-WRITE-RTN          THRU EXIT-OUT1-WRITE-RTN     OUT1WR01
001990     PERFORM EDIT-MATCH-PR-PROC      THRU EXIT-EDIT-MATCH-PR-PROC. G3PA01
002000 EXIT-MATCH.
002010     EXIT.                                                        G2PS01B
002020 UNMATCH.                                                         HEAD-RTN
002030     PERFORM HEAD-RTN                THRU EXIT-HEAD-RTN           G3PU04
002040     PERFORM EDIT-UNMATCH-PROC       THRU EXIT-EDIT-UNMATCH-PROC  OUT2WR01
002050     PERFORM OUT2-WRITE-RTN          THRU EXIT-OUT2-WRITE-RTN.
002060 EXIT-UNMATCH.
002070     EXIT.
```

図9 生成COBOLプログラムリストの例

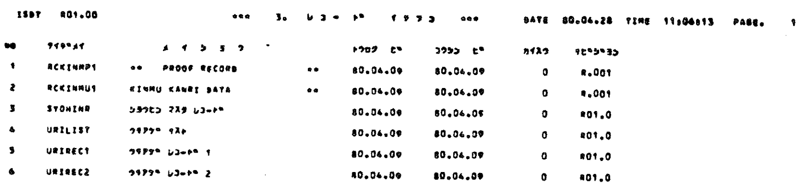
3.7 自動ドキュメンテーション

ソフトウェアはプログラムだけで なく，ドキュメントがあってこそ一人立ちできるものである．ドキュメントの目的は大きく分けて，設計の手助けをするワークドキュメントと，開発作業の各工程内および各工程間のコミュニケーションの為に作業の区切りにつくる出力ドキュメントがある．ISDTでは，ワークドキュメントに対応するものとして画面があり，入力された情報は仕様データベースに蓄えられている．ワークドキュメントを編集し，出力ドキュメントを作成する作業を機械化したのが自動ドキュメンテーションである．得られるドキュメントは1つの仕様に対応するものだけで なく，あるプログラムやジョブの視点から関連する仕様を集めて1つのドキュメントにすることも行われる．又，プログラムとドキュメントが同じ情報源から作成されるので，プログラムとドキュメントの不一致という問題は決して起り得ないことになる．必要な ドキュメントを画面で選択することにより，表2の16種類のドキュメントが得られる．

図10，図11に出力ドキュメントの例を示す．

項番	出力ドキュメント名	出力範囲				
		全体	マスク	単一	1ジョブ	1プログラム
1	ジョブ一覧	○	○			
2	ファイル一覧	○	○		○	
3	レコード一覧	○	○		○	○
4	プログラム一覧	○	○		○	
5	作業場所一覧	○	○		○	
6	プログラム構造一覧	○	○		○	
7	処理一覧	○	○		○	
8	定義名一覧	○	○			
9	ジョブ仕様書	○	○	○		
10	ファイル仕様書	○	○	○	○	
11	レコード仕様書	○	○	○	○	○
12	プログラム仕様書	○	○	○	○	
13	ファイル レコード関連表	○	○		○	
14	レコード ファイル関連表	○	○		○	
15	プログラム レコード関連表	○	○		○	
16	レコード プログラム関連表	○	○		○	

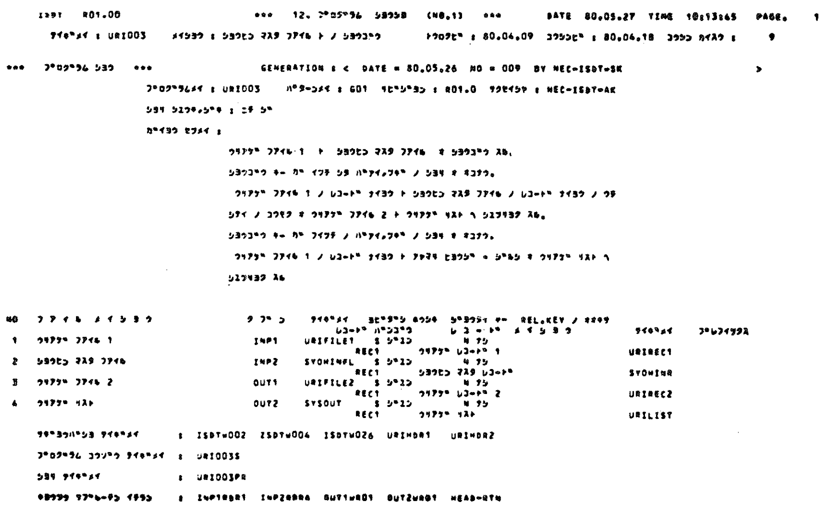
表2　出力ドキュメント一覧



```
ISDT   R01.00                ***   3. ﾚ ｺ ｰ ﾄﾞ  ｲ ﾁ ﾗ ﾝ   ***      DATE 80.04.28  TIME 11:04:13  PAGE.   1

NO   ﾃｲｷﾞﾒｲ          ﾒ ｲ ｼ ｮ ｳ                 ﾄｳﾛｸ ﾋﾞ    ｺｳｼﾝ ﾋﾞ    ｶｲｽｳ   ﾘﾋﾞｼﾞｮﾝ
 1   RCKINMP1    **  PROOF RECORD       **     80.04.09    80.04.09     0     R.001
 2   RCKINMU1    KINMU KANRI DATA       **     80.04.09    80.04.09     0     R.001
 3   SYOHINR     ｼｮｳﾋﾝ ﾏｽﾀ ﾚｺｰﾄﾞ               80.04.09    80.04.09     0     R01.0
 4   URILIST     ｳﾘｱｹﾞ ﾘｽﾄ                     80.04.09    80.04.09     0     R01.0
 5   URIREC1     ｳﾘｱｹﾞ ﾚｺｰﾄﾞ 1                 80.04.09    80.04.09     0     R01.0
 6   URIREC2     ｳﾘｱｹﾞ ﾚｺｰﾄﾞ 2                 80.04.09    80.04.09     0     R01.0
```

図10　レコード一覧の例



```
ISDT   R01.00               ***  12. ﾌﾟﾛｸﾞﾗﾑ ｼﾖｳｼｮ  (NO.1)  ***      DATE 80.05.27  TIME 10:13:45  PAGE.   1
  ﾃｲｷﾞﾒｲ : URI003    ﾒｲｼｮｳ : ｼｮｳﾋﾝ ﾏｽﾀ ﾌｧｲﾙ ﾄ ﾉ ｼｮｳｺﾞｳ      ﾄｳﾛｸﾋﾞ : 80.04.09  ｺｳｼﾝﾋﾞ : 80.04.18  ｺｳｼﾝ ｶｲｽｳ :    9

***   ﾌﾟﾛｸﾞﾗﾑ ｼﾖｳ   ***            GENERATION : <  DATE = 80.05.26  NO = 009  BY NEC-ISDT-SK              >
                 ﾌﾟﾛｸﾞﾗﾑﾒｲ : URI003    ﾊﾟﾀｰﾝﾒｲ : G01  ﾘﾋﾞｼﾞｮﾝ : R01.0  ｻｸｾｲｼｬ : NEC-ISDT-AK
                 ｼﾖｳ ｼﾕﾍﾞﾂ,ｼﾞｷ : ﾆﾁ ｼﾞ
                 ｶﾞｲﾖｳ ｾﾂﾒｲ :
                        ｳﾘｱｹﾞ ﾌｧｲﾙ 1  ﾄ  ｼｮｳﾋﾝ ﾏｽﾀ ﾌｧｲﾙ  ｦ ｼｮｳｺﾞｳ ｽﾙ。
                        ｼｮｳｺﾞｳ ｷｰ ｶﾞ ｲｯﾁ ｼﾀ ﾊﾞｱｲ,ﾌｷﾞ ﾉ ｼｮﾘ ｦ ｵｺﾅｳ。
                         ｳﾘｱｹﾞ ﾌｧｲﾙ 1 ﾉ ﾚｺｰﾄﾞ ﾅｲﾖｳ ﾄ ｼｮｳﾋﾝ ﾏｽﾀ ﾌｧｲﾙ ﾉ ﾚｺｰﾄﾞ ﾅｲﾖｳ ﾉ ｳﾁ
                        ｼﾃｲ ﾉ ｺｳﾓｸ ｦ ｳﾘｱｹﾞ ﾌｧｲﾙ 2 ﾄ ｳﾘｱｹﾞ ﾘｽﾄ ﾍ ｼｭﾂﾘｮｸ ｽﾙ。
                        ｼｮｳｺﾞｳ ｷｰ ｶﾞ ﾌｲｯﾁ ﾉ ﾊﾞｱｲ,ﾌｷﾞ ﾉ ｼｮﾘ ｦ ｵｺﾅｳ。
                         ｳﾘｱｹﾞ ﾌｧｲﾙ 1 ﾉ ﾚｺｰﾄﾞ ﾅｲﾖｳ ﾄ ｱﾔﾏﾘ ﾋｮｳｼﾞ ｦ ｼﾞｷﾝ ｦ ｳﾘｱｹﾞ ﾘｽﾄ ﾍ
                        ｼｭﾂﾘｮｸ ｽﾙ

NO   ﾌ ｧ ｲ ﾙ  ﾒ ｲ ｼ ｮ ｳ            ｸ ﾌﾞ ﾝ    ﾃｲｷﾞﾒｲ    ｺﾋﾞｸﾞｼ ｶﾂｼｷ  ｼﾞｮｳﾀｲ ｷｰ  REL.KEY ﾉ ｷｵｸｲ
                                                 ﾚｺｰﾄﾞ ﾊﾞﾝｺﾞｳ      ﾚ ｺ ｰ ﾄﾞ  ﾒ ｲ ｼ ｮ ｳ          ﾃｲｷﾞﾒｲ    ﾌﾟﾚﾌｲｯｸｽ
 1   ｳﾘｱｹﾞ ﾌｧｲﾙ 1                  INP1     URIFILE1    S ｼﾞｭﾝ           N ﾅｼ
                                                       REC1      ｳﾘｱｹﾞ ﾚｺｰﾄﾞ 1                   URIREC1
 2   ｼｮｳﾋﾝ ﾏｽﾀ ﾌｧｲﾙ                INP2     SYOHINFL    S ｼﾞｭﾝ           N ﾅｼ
                                                       REC1      ｼｮｳﾋﾝ ﾏｽﾀ ﾚｺｰﾄﾞ                 SYOHINR
 3   ｳﾘｱｹﾞ ﾌｧｲﾙ 2                  OUT1     URIFILE2    S ｼﾞｭﾝ           N ﾅｼ
                                                       REC1      ｳﾘｱｹﾞ ﾚｺｰﾄﾞ 2                   URIREC2
 4   ｳﾘｱｹﾞ ﾘｽﾄ                     OUT2     SYSOUT      S ｼﾞｭﾝ           N ﾅｼ
                                                       REC1      ｳﾘｱｹﾞ ﾘｽﾄ                       URILIST

   ｻｷﾞｮｳﾊﾞｼｮ ﾃｲｷﾞﾒｲ         :  ISDTW002  ISDTW004  ISDTW026  URIHDR1   URIHDR2
   ﾌﾟﾛｸﾞﾗﾑ ｺｳｿﾞｳ ﾃｲｷﾞﾒｲ     :  URI003S
   ｼｮﾘ ﾃｲｷﾞﾒｲ               :  URI003PR
   ﾃﾞｰﾀﾒｲ ｻﾌﾞﾙｰﾁﾝ ｲﾁﾗﾝ      :  INP1RDR1  INP2RDRA  OUT1WRD1  OUT2WRD1  HEAD-RTN
```

図11　プログラム仕様書の例

4. まとめ

4.1 ＩＳＤＴの効果

ＩＳＤＴを用いると，次のような効果が期待される．

(1) 画面を通じてチュートリアルに誰にでも，容易に高品質のプログラムが開発できる．

(2) プログラムのモジュール化，構造化が促進され，パターン化，標準化も促進されることにより，信頼性の高いプログラムが開発できる．

(3) プログラムの自動生成，コンポーネントの共通利用，ドキュメントの自動作成などにより，プログラムの開発効率が向上する．

(4) 構造化されたプログラム，問い合わせ機能や対話型修正機能，ドキュメンテーション支援により，保守性が向上する．

(5) ワークステーションの活用により，仕様書，コーディングシート，カードなどの資源が節約される．

4.2 今後の課題

ＩＳＤＴはまだ第１版をリリースしたばかりで，現在フィールドテスト中であり，大規模なシステムの開発の場に実際に適用しての評価はまだ不十分である．

今後の課題としては，次のようなことがあげられる．

(1) ＩＳＤＴのような対話型支援ツールは，マンマシンシステムとしての操作性が重要なポイントであろう．この操作性の問題は，人間の主観によるところが大きいので，実際の利用結果を通して改善していく必要がある．

(2) ＩＳＤＴの支援範囲を，ソフトウェア開発のより前の工程，すなわちシステム概要設計フェーズにまで拡大することである．

参考文献

(1) Alford, M.W., et al., Software Requirements Engineering Methodology, SREM Final Report, Vol. I, II, III, 1977

(2) 東，水野，"ソフトウェアの標準化"，日本経済新聞社，1978

(3) Teichrow, D. et al., "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing System", IEEE Trans. on Software Engineering, Vol. SE-3, No.1, Jan. 1977, pp. 41-48

(4) Wirth, N., Algorithm + Data Structure = Programs, Prentice-Hall Inc., 1976