データベース・システム 81－7
（1991. 1. 17）

フォーム定義・制御に着目した情報システムの構築について

谷川英和　米田亜旗　中埜善夫　今井良彦
松下電器産業(株)　情報通信関西研究所

本稿では、RDB の 4GL を用いて試作した障害管理システムの問題点を解決する STUFF の構造について述べる。

STUFF はソフトウェア開発に必要な情報を管理する情報管理ツールにおいてデータを処理する実機能部と本体部とを分離し、ユーザインターフェイスを構成するフォームの定義、制御をユーザインターフェイス定義ファイルで定義し、本体部がユーザインターフェイス定義ファイルを読み込み、フォーム定義、制御、実機能部品のハンドリングを行なう。

情報システムの一種である情報管理ツールを STUFF の構造で実現することにより情報管理ツールが拡張性と可搬性を持つようになった。

MAKING AN INFORMATION SYSTEM AIMING AT FORMS DEFINITION AND CONTROL

Hidekazu Tanigawa　Aki Yoneda　Yoshio Nakano　Yoshihiko Imai
MATSUSHITA ELECTRIC INDUSTRIAL CO.,LTD
KANSAI INFORMATION & COMMUNICATIONS RESEARCH LABORATORY
1006,KADOMA,KADOMA-SHI,OSAKA 571 JAPAN

The STUFF structure is designed for solving problems of the bug infomation management system using 4GL of the RDB . In this paper , the STUFF structure is described .

In the STUFF structure function parts that deal with data and a body part are separated . And In the STUFF structure definition and control of the forms that constitute user-interface are defined in user-interface definition files , and the body part defines and contorols forms and deals with the function parts after geting user-interface definition files .

By using the STUFF structure we have much extentsibility and portability in information systems .

1 はじめに

開発手法の確立、高性能のワークステーションやネットワークの普及、開発支援ツールの普及にも拘らず、ソフトウェア開発の生産性の向上はハードウェアのそれと比較して小さいものである。[1]

その原因の一つとしてソフトウェア開発作業の実態が十分に把握されておらず、適切な支援をしていないことが考えられる。

そこで、ソフトウェア開発の実態の調査を行なうと、開発作業はプログラム作成・ドキュメント作成・担当者間情報伝達の3つに分類でき、概ね各々1:1:1の比の作業割合を示していた。そして、ドキュメント作成、担当者間情報伝達において支援ツールが整備されていないことが明らかになった。

この2つの作業を具体的に示すと、システム設計書を書いたり(作成)、蓄えておいた障害情報(蓄積)からプログラムの品質を測ったり、他の開発メンバーに情報を伝達したり(伝達)する作業などがある。よって、ソフトウェア開発作業を作成・蓄積・伝達の3つの技術要素に分けることができる。(図1参照)

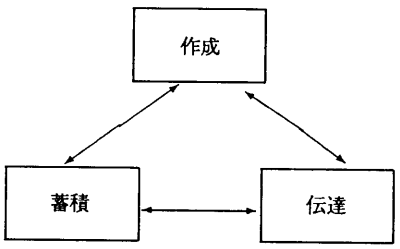

図1: ソフトウェア開発作業3要素

上記のソフトウェア開発作業3要素に示すようにソフトウェア開発作業には情報を蓄積する手段は不可欠である。昨今, ソフトウェア開発支援の立場からデータベース技術に対する新しい要請も出てきている。[3][4][5]

ソフトウェア開発作業におけるデータベース技術に対する要請の例を以下に示す。

- 広い範囲への拡張性を持ったデータモデルの実現
- 対象間関係の管理
- 版管理
- グループ開発支援
- 知識処理
- 複数のデータベースへの対応
- 適切な性能
- データの一貫性保持

以上の8つの要請を挙げたが、それ以外にも多々論じられている。

上記の背景により我々はソフトウェア開発を行なうために必要な情報の生成・利用を効率的に行なう情報システムの構築のための一例として障害管理システム(治虫君)を市販のRDBの4GLを用いて試作した。この治虫君-4GL版については報告済みである。[2]

しかし、治虫君-4GL版には拡張性や可搬性に問題があることが明らかになった。そこで、拡張性や可搬性にすぐれた情報システムのプラットフォームとして、STUFF(System structure with information Transmitted by tag file between UserinterFace and Function parts)を考えた。本稿では、試作した治虫君-4GL版の問題点を示し、その問題点を解決するSTUFFの構造およびその応用例である障害管理システム(治虫君-STUFF版)について述べる。

2 障害管理システム(治虫君-4GL版)

2.1 従来の障害管理

従来の障害管理は、障害の発見者が障害票(帳票)に手書きで障害状況を記入して障害管理者に手渡し、障害管理者が障害状況から対処者を判断して対処者に障害票を手渡し、対処者が障害を対処した後に障害票に対処記録を書き込み、障害の発見者に障害対処を報告していた。

しかし、このように紙ベースで障害票を記述し、手渡しで情報を伝達していたプロジェクトにおいて以下の問題点が生じていた。(図2参照)

- 二重の情報記述による非効率
- 障害票検索の非効率
- 情報伝達の遅延、洩れ

上記の問題点を早急に解決するために、4GLを用いて障害管理システム(治虫君-4GL版)を開発した。

以下に治虫君-4GL版の概要、特徴、および治虫君で利用されている伝達制御モデルについて述べる。

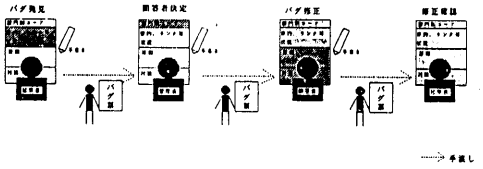

図 2: 従来の障害管理

2.2 治虫君-4GL 版

2.2.1 治虫君-4GL 版の概要

電子メールやニュースシステムの普及により、人手による担当者間の情報伝達から機械的な情報伝達が可能になってきた。しかし、まだ電子メールやニュースシステムを有効利用しているとは言えない。

治虫君-4GL 版はデータベースを用いて障害情報を一括管理し、電子メール・ニュースシステムを用いて障害情報の自動伝達を行なう障害管理システムである。(図3参照)

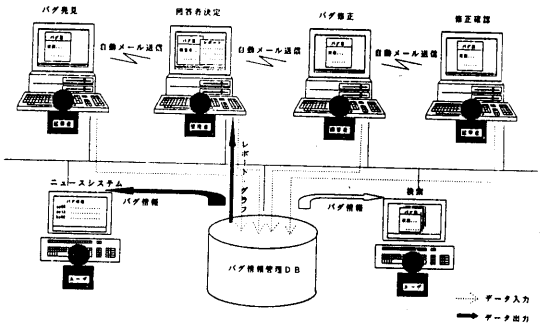

図 3: 治虫君-4GL 版による障害管理

2.2.2 治虫君-4GL 版の特徴

治虫君-4GL 版は図 2に示す従来の障害管理と比較して以下の特徴を持つ。

- 障害情報伝達の確実、迅速、広範化
- 障害レポート作成の効率化
- 障害情報の高速検索

2.2.3 伝達制御モデル

伝達制御モデルは、入力される情報(入力情報)を予め決められた情報(伝達制御情報)と照らし合わせて伝達先を決定する伝達制御方法をモデル化したものである。(図 4参照)

伝達制御モデルにより自動的に目的の伝達先に情報が伝達される。

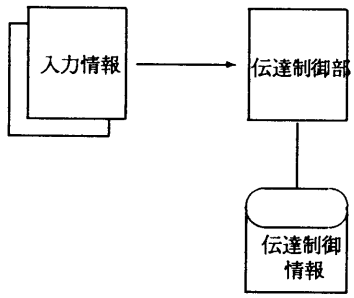

図 4: 伝達制御モデル

以下に、伝達制御モデルにおいて伝達制御部が行なう伝達制御方法と伝達制御情報として格納されている伝達先指定方法について述べる。

1. 伝達制御方法
   入力情報には、伝達先に送信される情報(伝達情報)と伝達先に送信されない情報(非伝達情報)がある。入力情報の種類から情報伝達の制御方法は、以下の3つに分類することができる。

   (a) 伝達情報パラメータ制御……ある伝達情報の値により伝達先を決定

   (b) 非伝達情報パラメータ制御……伝達情報以外の値(動作名など)により伝達先を決定

   (c) 複合パラメータ制御……上記の伝達情報・非伝達情報の値により伝達先を決定

2. 伝達先指定方法

   (a) 静的アドレス指定……ある人のアドレスやあるグループのアドレスなどを直接に指定する伝達先指定方法

   (b) 動的アドレス指定……ある条件などにより動的にアドレスが変わる伝達先指定方法

2.2.4 治虫君-4GL 版での伝達制御

治虫君は2.2.3の伝達制御方法のうち、複合パラメータ制御を行なっている。伝達先指定方法は、動的アドレス指定である。表1を用いることにより、各動作時に自由に障害情報を伝達することができる。例えば、入力障害情報のデータから「動作ID=1」と判断できる場合は新規登録のレコードを選択する。(動作という非伝達情報による非伝達情報パラメータ制御) また、「動作ID=3」と判断し「解決属性='y'」という情報が与えられた時は、起票者管理表を「select アドレス from 検索表 where 検索表.バグ票ID=入力バグ票ID」の検索文により検索を行なって障害情報の伝達先を決定する。(複合パラメータ制御)

表1に伝達制御表の構造を示す。

表 1: 伝達制御表

ID	動作	属性名	条件	検索表	検索式
1	新規登録			バグ管理者管理表	select アドレス from 検索表
⋮	⋮	⋮	⋮	⋮	⋮
3	解決	解決	='y'	起票者管理表	select アドレス from 検索表 where 検索表.バグ票ID= 入力バグ票ID
		解決	='n'	回答者管理表	select アドレス from 検索表 where 検索表.アドレス= 入力回答者アドレス

2.2.5 治虫君-4GL 版の結果

治虫君-4GL 版を使用してソフトウェア開発を行なった場合、表2に示すような作業効率向上が測定できた。

効率向上は主に障害情報の電子化と障害情報の自動伝達による。

表 2: 効率評価

比較項目	導入前	導入後
障害票作成	940 分	1128 分
レポート作成	1920 分	44 分
検索	2380 分	79 分
障害情報伝達	864 分	0 分
合計	6104 分	1251 分

[注意]
表2は188件/月の障害発生、476回/月の検索回数のプロジェクトのデータである。

2.2.6 治虫君-4GL 版の問題点

治虫君-4GL 版は、RDB の 4GL で開発した。

4GL を用いたことにより 3GL を用いるより短期間で開発できたと思われる。しかし、多くのプロジェクトで利用されるに従って以下のような問題点が明らかになってきた。

- 他のデータベースへの移植が困難である
- 入力項目の変更が困難である
- 機能追加が困難である
- 同様の機能を持つ別のアプリケーションを開発する時に、開発済みのモジュールを利用しにくい

上記の問題点は、プロジェクトごとに DB 利用目的が違うために利用する DB も異なっており、障害管理システムの利用形態も多少なりとも違ってくることから出てくる問題点である。システムの側から見ると、複数のデータベースへの対応という課題と、プロジェクト毎の各種カスタマイズの容易性という課題が出てきたわけである。

3 フォーム定義・制御システム構造 (STUFF)

2.2.6で述べた問題点を鑑みて、ここで説明する STUFF のシステム構造を考えた。

3.1 STUFF の構造

STUFF は図 5に示すように、本体部と実機能部 (実際に処理を行なう部分) に分かれている。

本体部は、実機能部のハンドリングを行なう部品ハンドリング部とユーザインターフェース定義ファイルで定義されている内容を読み取り、フォーム作成、フォーム制御を行なうユーザインターフェース部からなる。

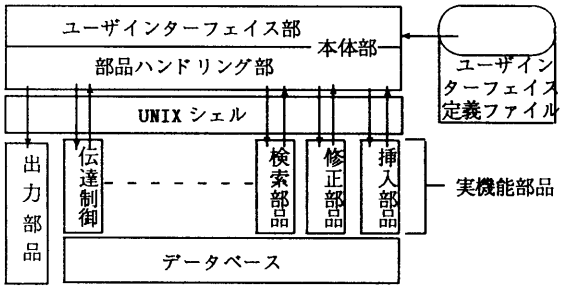

図 5: STUFF 構成図

以下に、本体部、ユーザインターフェース定義ファイル、および実機能部について述べる。

3.2 本体部

本体部のうちのユーザインターフェイス部は、ユーザインターフェイス定義ファイルに記述された定義を読み込み、フォーム定義、フォーム制御を行なう。また、部品ハンドリング部は、実際に作業をする実機能部品を呼びだし、実機能部品の入力を与え、出力を受け取り処理する。

部品ハンドリング部と実機能部品との間の情報伝達は、タグ付き情報の集合であるタグ付き情報ファイルで行なう。

タグ付き情報ファイルで部品ハンドリング部と実機能部品間の情報伝達を行なうことにより、情報の処理に対する自由度が増す。例えば、実機能部品中の出力部品は、タグ付き情報ファイルを出力する場合に、レイアウト定義ファイルによりプロジェクト毎に出力フォーマットを定義できるようにして、同一のタグ付き情報ファイルをプロジェクト毎に違う帳票フォーマットで出力できる。レイアウト定義ファイルでは、タグ付き情報のレイアウトを簡単に記述する。

以下に部品ハンドリング部と実機能部品間で情報伝達に使用されるタグ付き情報ファイルの例を示す。

```
----------------------------------------
<処理>shinki
<バグ票>
<fieldValue id=処理番号>zz00
<fieldValue id=ランク>A
<fieldValue id=要約>日本語変換のミス
<fieldValue id=状況>漢字端末において日本
                    語変換ができない
        :
----------------------------------------
```

3.3 ユーザインターフェース定義ファイル

ここでは、ユーザインターフェイスを定義する各種定義ファイルの仕様について説明する。

3.3.1 ファイル構成

ソフトウェア開発を支援する情報管理システムを構築する要素として、ユーザインターフェイスの1つの画面を定義するフォームと、フォームの構成を示すメニューと、フォームを構成する1つの入力項目を定義するフィールドと、フォームの制御と情報の処理をインタラクティブに行なうためのボタンなどがあげられる。

よって、本体部の中のユーザインターフェイス部がフォーム定義、フォーム制御を行なうために以下の定義ファイルを用意する。

ファイル名	内容
MenuInit	メニュー構造の定義を行なう
FormInit. *	フォームの定義を行なう
FieldInit	フィールド情報の定義を行なう
ExecInit	実行ボタンの定義を行なう

*は任意の文字列

上記の4つのファイルは、タグ付き情報で記述する。ユーザインターフェイス定義ファイルをタグ付き情報で記述することによりユーザインターフェイスの構造、制御が情報記述の順序によらないなど、ユーザインターフェイスに関する情報の追加、変更が容易になる。

メニュー、サブメニュー、フィールド、実行ボタンを以下の図6の画面イメージに示す。

3.3.2 メニュー定義ファイル

メニュー定義ファイルは通常、<mainMenu>のタグの下に、<subMenu>と<form>が対になったものをサブメニューの個数分並べる。これを、メインメニューの個数分だけ繰り返す。以下にメニュー定義ファイルの例を示す。

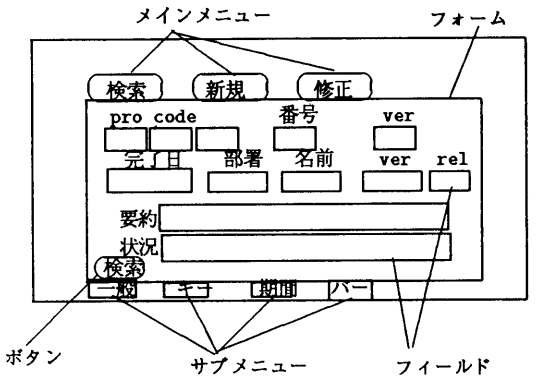

図 6: 画面イメージ

[例]

```
-----------------------------------------
<mainMenu> 障害管理システム
  <subMenu>新規登録 <form> 新規登録フォーム
  <subMenu>完了登録 <form> 完了登録フォーム
  <subMenu>検索     <form> 検索フォーム
-----------------------------------------
```

ユーザインターフェイス部はメニュー定義ファイルを読み込みシステムの枠を作る。

3.3.3 フォーム定義ファイル

フォーム定義ファイルは、フォームに属するフィールドおよびボタンの定義、フィールドおよびボタンの配置定義、ボタンを押した時に実行する実行機能部品名、フォーム制御などを記述しているファイルである。

[例]

```
-------------------------------------
<form>登録結果フォーム
        <formtype>normal
        <input>new
        <v>
           <h><heading>処理番号は
                 <field>プロジェクト
                 <field>コード
                 <field>サブコード
              <heading>-
                 <field>番号
              <heading>です。
           <\h>
           <h><heading>バグ票出力:CTRL/V
              <heading>出力先選択:CTRL/C
              <heading>登録に戻る:TAB
           </h>
        <\v>
-------------------------------------
```

ユーザインターフェイス部はフォーム定義ファイルによりフォームを構成し、フォームを構成するフィールドへのデータの取り込み制御を行なう。

以下にフォーム定義ファイルの内容について説明する。

1. フォーム名定義部

   フォーム名はユーザが任意につけられるフォームの識別子であり、<form>のタグの後に記述される。

2. フォームタイプ定義部

   フォームタイプは、フォームのタイプを<formtype>のタグの後に記述される。以下のタイプが存在する。

タイプ名	内容
normal	メインメニューとサブメニューによって選択される通常のフォーム。
panel	パネルの形で出力されるフォーム。
dummy	実際には画面に出力されないダミーのフォーム。変数値の確保などに用いる。

3. 入力元定義部

   入力元は、フォームを表示した時にそのフォームに属するフィールドの内容をどこから持ってくるかを<input>タグの後に記述する。

   以下に入力元定義部で用いられるデータの例を示す。

タイプ名	内容
new	どこからも持ってこない。(内容はクリアされる。)
get	実機能部品の結果ファイルを取り込む。
copy	他のフォームの内容をコピーする。

4. フィールド配置定義部

   前記した種々のフォーム定義のあとで、そのフォームが持つフィールド、実行ボタン、コメント文字列

のレイアウトと表示法を指定する。

[定義部例]

```
------------------------------------
<v>アイテム1
          <h>アイテム2 アイテム3<\h>
   アイテム4
<\v>
------------------------------------
```

[表示例]
アイテム1
アイテム2 アイテム3
アイテム4

フィールド配置定義部では各項目のレイアウトを座標値で指示するのではなく、位置関係のみで指定することができる。このため、表示を行なう機器に依存しない形でフォーマットを定義することができる。

3.3.4 フィールド定義ファイル

フィールド定義ファイルは、フォームを構成するフィールドの型、長さ、候補データを示す。

[例1]

```
------------------------------------
<field>ソフト名<dataType>char<length>10
<field>受付日<dataType>date
<field>氏名<dataType>kanji<length>4
------------------------------------
```

[例2]

```
------------------------------------
<field>ランク<dataType>char<length>1
  <value>A<comment>重大バグ
  <value>B<comment>やや軽いバグ
  <value>C<comment>軽いバグ
  <value>S<comment>仕様変更
------------------------------------
```

例2はポップアップメニューで候補データからデータを選択して入力するフィールドの指定方法である。(図7参照)

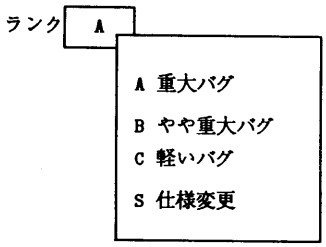

図7: ポップアップメニュー

ユーザインターフェイス部はフィールド定義ファイルによりフォームを構成するフィールドのチェックを行なう。

3.3.5 実行ボタン定義ファイル

実行ボタン定義ファイルは、実機能部品を動作させるための実行ボタンの定義を行なう。

[例]

```
------------------------------------------
<button>新規登録<string>登録
 <execCommand>shinki<message>ただ今バグ
                        票を新規登録中です
 <form>登録結果フォーム
 <afterService>出力フォーム表示
   <resultForm>出力フォーム
   <fieldValue id=処理番号>処理番号フィールド
------------------------------------------
```

ボタンの識別名を<button>タグの後に、画面上でのボタン文字列を<string>タグの後に、起動する実機能部品名(コマンド名)を<execCommand>タグの後に、起動中に画面に出すメッセージ文字列を<message>の後に記述する。

<form>タグの後に指定されたフォームのデータをタグ付きデータにして実機能部品(<execCommand>タグの後に記述されたコマンド)に渡して実行する。

<afterService>タグの後には、実機能部品実行後の処理を記述する。上記の例では、<resultForm>タグの後の「出力フォーム」が有する<fieldValue id=処理番号>タグの後の「処理番号フィールド」に<fieldValue id=処理番号>タグで指定された実機能部品の実行結果が代入される。

ユーザインターフェイス部がボタン定義ファイルを参照してボタンをフォーム上に表示したりフォーム制御を行ない、部品ハンドリング部が実機能部品の実行および実機能部品との情報の受渡しを行なう。

3.4 実機能部

実機能部の各部品は、タグ付き情報ファイルを受け取り、結果をタグ付き情報ファイルに書き込み、部品ハンドリング部にタグ付き情報ファイルを渡す。

結果のタグ付き情報ファイルの例を示す。

[例]

```
--------------------------------
<処理内容>shinki
<結果>成功
<fieldValue id=処理番号>zz00-123
--------------------------------
```

4 STUFF の応用例

障害管理システム (治虫君-STUFF 版) は、2.1で述べた治虫君-4GL 版の機能を 3で説明した実機能部と本体部を組み合わせることにより実現したものである。

以下に治虫君-STUFF 版の概要、特徴、および開発結果について述べる。

4.1 治虫君-STUFF 版の概要

4.1.1 機能概要

治虫君-STUFF 版はユーザの立場から見た場合には、2.1で述べた機能と同等の機能を持つ。

4.1.2 実機能部品

治虫君は、以下の実機能部品を備えている。

- データ挿入部品
- データ修正部品
- データ削除部品
- データ検索部品
- レポート作成部品
- 出力部品
- 伝達制御部品

以下に詳細に実機能部品について述べる。

1. データ挿入部品・修正部品・削除部品
   本体部の中の部品ハンドリング部からタグ付き情報ファイルを受け取り、結果をタグ付き情報ファイルで返す。部品はデータベースに依存しない埋め込み型 SQL を用いて実現されている。

   タグと属性との対応表を参照してデータベースにアクセスする。

   入力のタグ付き情報ファイルの例を以下に示す。

```
----------------------------------------
<処理>shinki
<バグ票>
<fieldValue id=処理番号>zz00
<fieldValue id=ランク>a
<fieldValue id=要約>日本語変換のミス
<fieldValue id=状況>漢字端末において日本語変換ができない
                        :
                        :
----------------------------------------
```

   出力のタグ付き情報ファイルの例を以下に示す。

```
----------------------------------
<処理内容>shinki
<結果>成功
<fieldValue id=処理番号>zz00-123
----------------------------------
```

2. データ検索部品
   以下に示すような条件が記述された情報を入力ファイルとして検索部品を実行する。

```
--------------------------------
<処理>hanyou_sch
<検索条件>
<or>
<opr>
      <and>
      <opr>
        <属性値>pro
        <演算子>!=
        <値>zz
      <opr>
```

```
        <属性値>code
        <演算子>=
        <値>00
      </and>
<opr>
      <属性値>rank
      <演算子>=
      <値>a
</or>
--------------------------------
```

結果の例は以下のようである。

```
--------------------------------
<処理内容>hanyou_sch
<検索結果>
<バグ票>
<fieldValue id=処理番号>zz00-1
<fieldValue id=受付日>1990/9/3
<fieldValue id=要約>障害管理システム
                              の動作不良
<fieldValue id=ランク>a
                    :
                    :
<バグ票>
<fieldValue id=処理番号>zz00-3
                    :
                    :
--------------------------------
```

3. レポート作成部品
   レポート作成部品も検索部品などと同様にタグ付き情報入力ファイルを受け取り、タグ付き情報出力ファイルを作成する。

4. 出力部品
   上記のデータ検索部品の出力ファイルやレポート作成部品の出力ファイルなどのタグ付き情報ファイルをレイアウト定義ファイルの設定に基づいて自由なフォーマットで出力する。レイアウト定義ファイルにはタグの付いた情報を出力するレイアウトが平易に記述されている。

5. 伝達制御部品
   伝達経路を記述したデータベース中の伝達制御表を参照して、伝達先を決定して与えられた情報を決定した伝達先に伝達する。

4.1.3 本体部

本体部として、漢字端末版とX-Window版と電子メール版を用意する。この3種類の本体部のいずれを用いても治虫君-STUFF版を使用することができる。

1. 漢字端末版
   ホストマシンに漢字端末を接続して開発を行なうプロジェクトを対象としている。
   画面例を以下の図8および図9に示す。

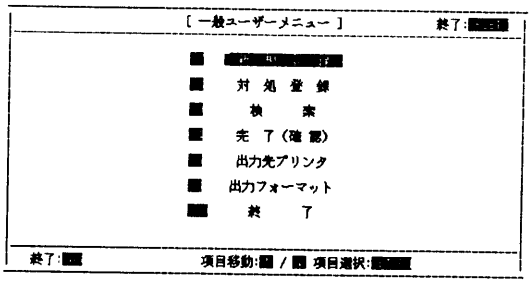

図 8: 一般ユーザーメニュー (漢字端末版)

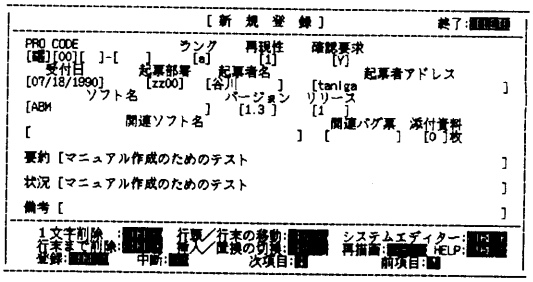

図 2.2 新規登録画面

図 9: 新規登録フォーム (漢字端末版)

2. X-Window版
   ワークステーションが普及しているプロジェクトを対象としている。

3. 電子メール版
   手軽にシステムを使いたいプロジェクトを対象としている。電子メール版を使うプロジェクトは電子メールが普及しているプロジェクトである。
   エディタでタグ付き情報ファイルを作成して、作成したファイルをあらかじめシステムが動作するように設定されたアカウントに送信することにより、障害管理システム (治虫君) を利用できる。処理結果もタグ付き情報ファイルが電子メールで送信者に送られるようになっている。

4.2 治虫君-STUFF 版の特徴

STUFF の構造に基づいて実現された治虫君-STUFF 版の特徴を以下に列挙する。

- ユーザインターフェイスのレイアウトが容易に変更できる
- 障害票の項目が容易に変更できる
- 機能追加が容易である

4.3 治虫君-STUFF 版の開発結果

STUFF の構造を採用することにより、ツールを提供する側の作業が軽減される。つまり、簡単にプロジェクトによるユーザインターフェイスの構成、制御の違いを吸収でき、機能追加が容易に行なえる。また、治虫君開発時に作成した実機能部品と本体部を用いて他の情報管理システムを容易に構築できることが予想される。しかし、ユーザインターフェイス変更および他システム構築時の作業低減を定量的には評価できていない。

5 おわりに

本稿では、ソフトウェア開発支援のための情報管理システム構築にデータベースを利用する立場で、STUFF の構造について述べた。

そして、STUFF の応用例として障害管理システム (治虫君-STUFF 版) を紹介した。

また、グループ開発支援モデルの一例として伝達制御モデルを治虫君に適応してその有効性を述べた。

本体部と実機能部とを分離してアプリケーションを実現する STUFF の構造を採用することにより、ユーザインターフェイスの構成および制御の変更が容易になり、機能追加および変更も簡単になった。

しかし、STUFF で実現したシステムはソフトウェアドキュメントの中の帳票を管理、利用するシステムだけである。しかも、対象間関係管理や版管理などの複雑なデータ管理は行なっていない。

さらに属性が決まらない上位工程のドキュメントの管理などもこれからの課題として挙げられる。

今後、障害管理システム (治虫君) で作成した実機能部品および本体部を用いて他のドキュメントおよびプロジェクト管理ツールを構築して STUFF の有用性を評価したい。そして、そのときに現在利用しているリレーショナルデータベースで管理可能かどうか、現在話題になっているオブジェクト指向データベースなどが必要になってくるのかどうか見極めていきたい。

参考文献

[1] 菅野：ソフトウェアの品質管理日科技連ソフトウェア品質管理シリーズ

[2] 谷川他：グループ開発を支援する障害管理システム．平成 2 年電気関係学会関西支部連合大会 S8-9

[3] 落水:データベース技術とソフトウェア技術．データベースシステム研究会 90-DBS-76

[4] 松本：CASE 環境の基礎技術．情報処理 VOL.31 NO.8 1990

[5] 松本：CASE 環境のためのデータベース．Advanced Database System Symposimu'90

[6] 久保監修：富士通におけるソフトウェア品質保証の実際

[7] Ed Lee：User-Interface Development Tools IEEE software May 1990