GUI とプログラミング言語の利点を兼ね備えたスクリプト言語: guript

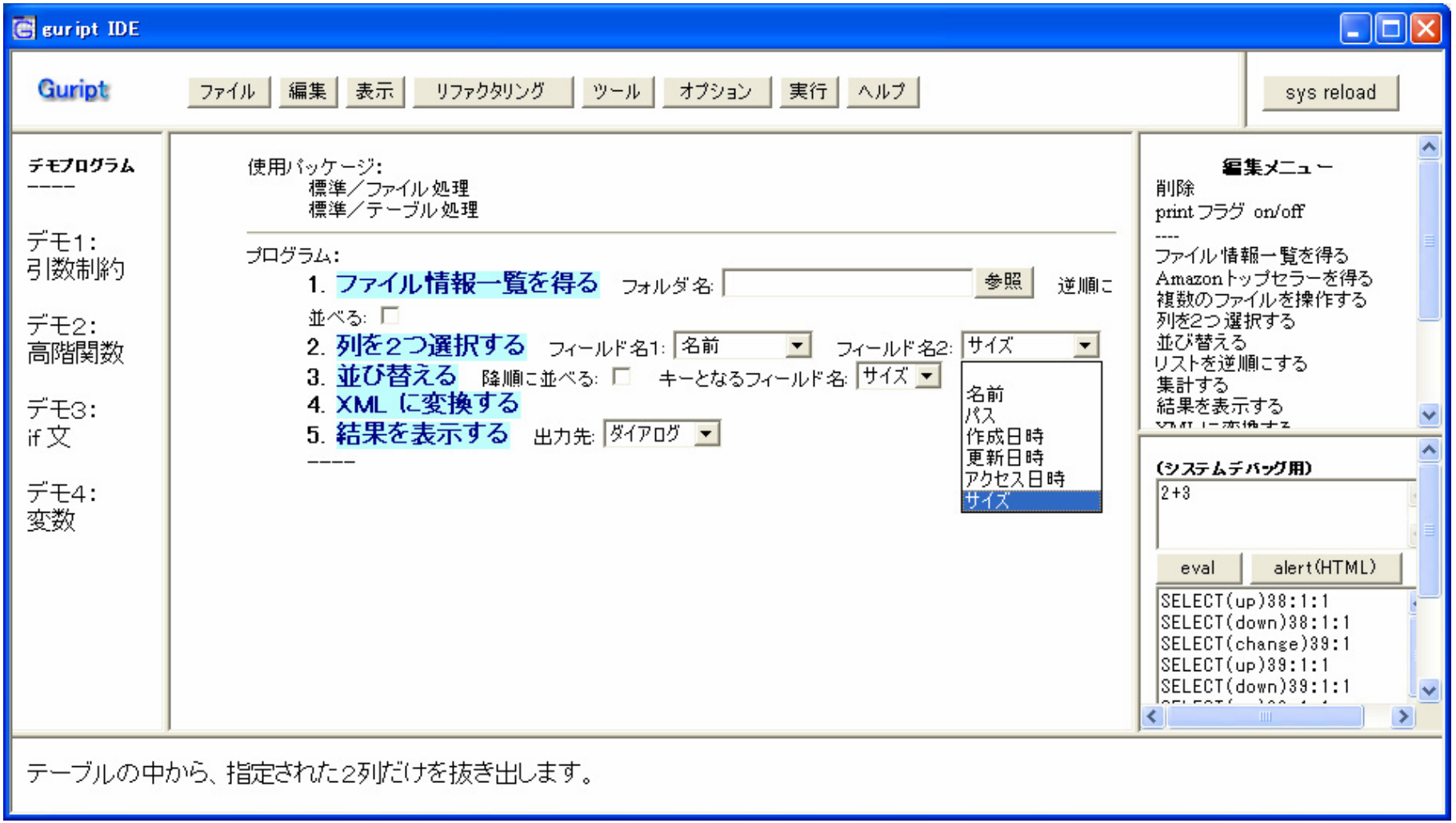
日本ソフトウェア科学会第21回大会ポスター・デモセッション資料

産業技術総合研究所
情報技術研究部門
一杉裕志、古川浩史

2004年9月16日

guript = GUI + script

- guript IDE（統合開発環境）のスクリーンショット

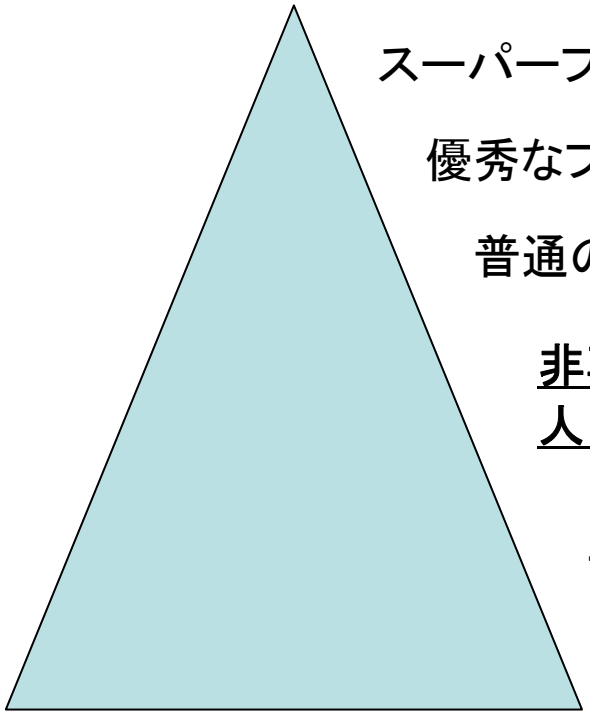


guript の特徴

- 非プログラマでも使える「スクリプト言語」
  - Unix のコマンドラインのような処理がマウスだけでできる
- 日本語化された構造エディタ
- 手続き型言語（⇔ Visual language）
- IDE（統合開発環境）
- 使いやすい標準ライブラリ
- 高い拡張性

対象とするユーザ

- プログラマの階層



スーパープログラマ（人口100万人に1人）

優秀なプログラマ（人口1万人に1人）

普通のプログラマ（人口1000人に1人）

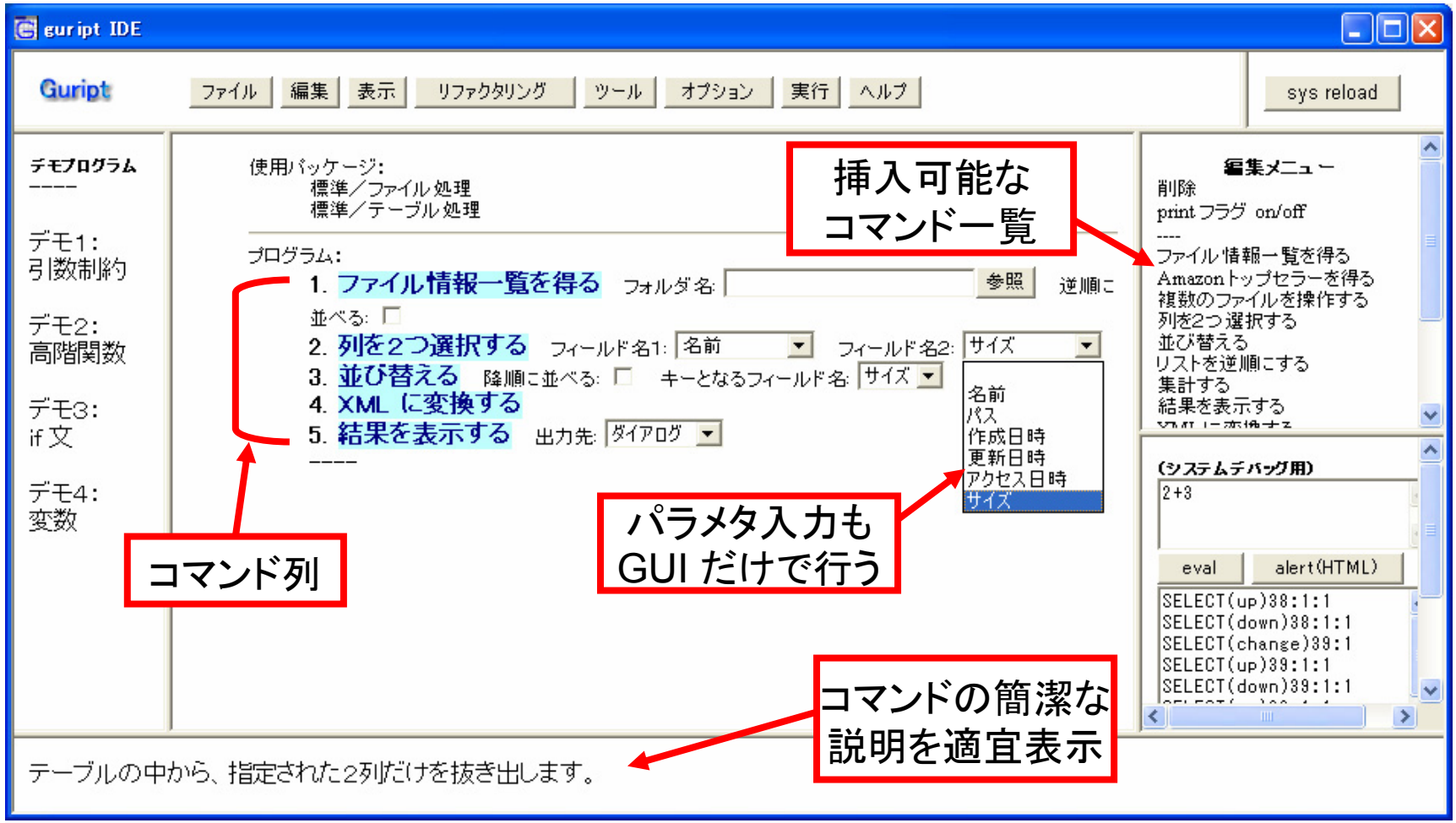
非専門家プログラマ（JavaScript、エクセルマクロ、人口100人に1人） **このあたりが対象**

一般人（情報家電ユーザ、人口10人に1人）

小・中学生（教育用、全員）

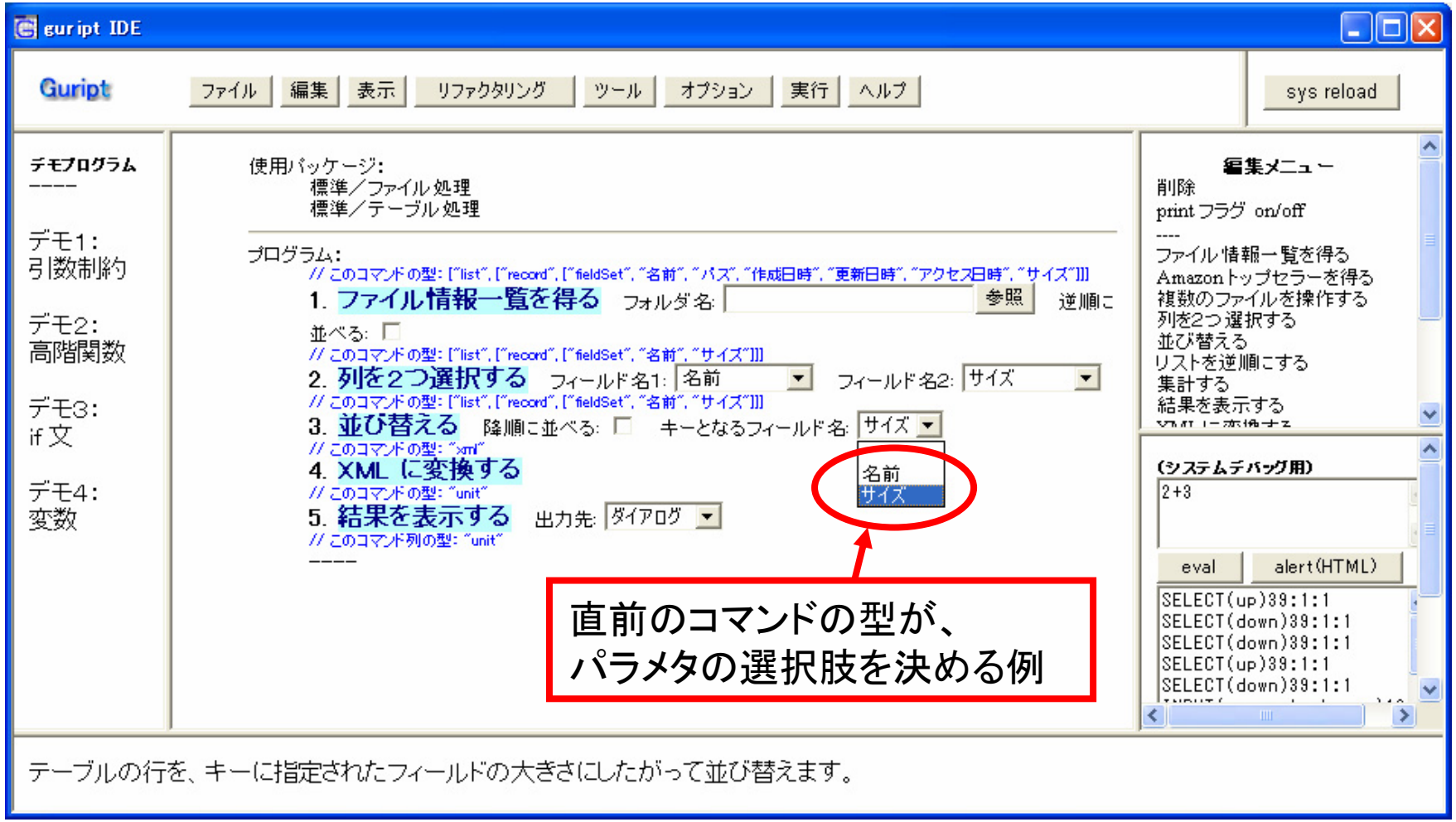
プログラム例

- Unix のパイプのように、データが上から下に流れる



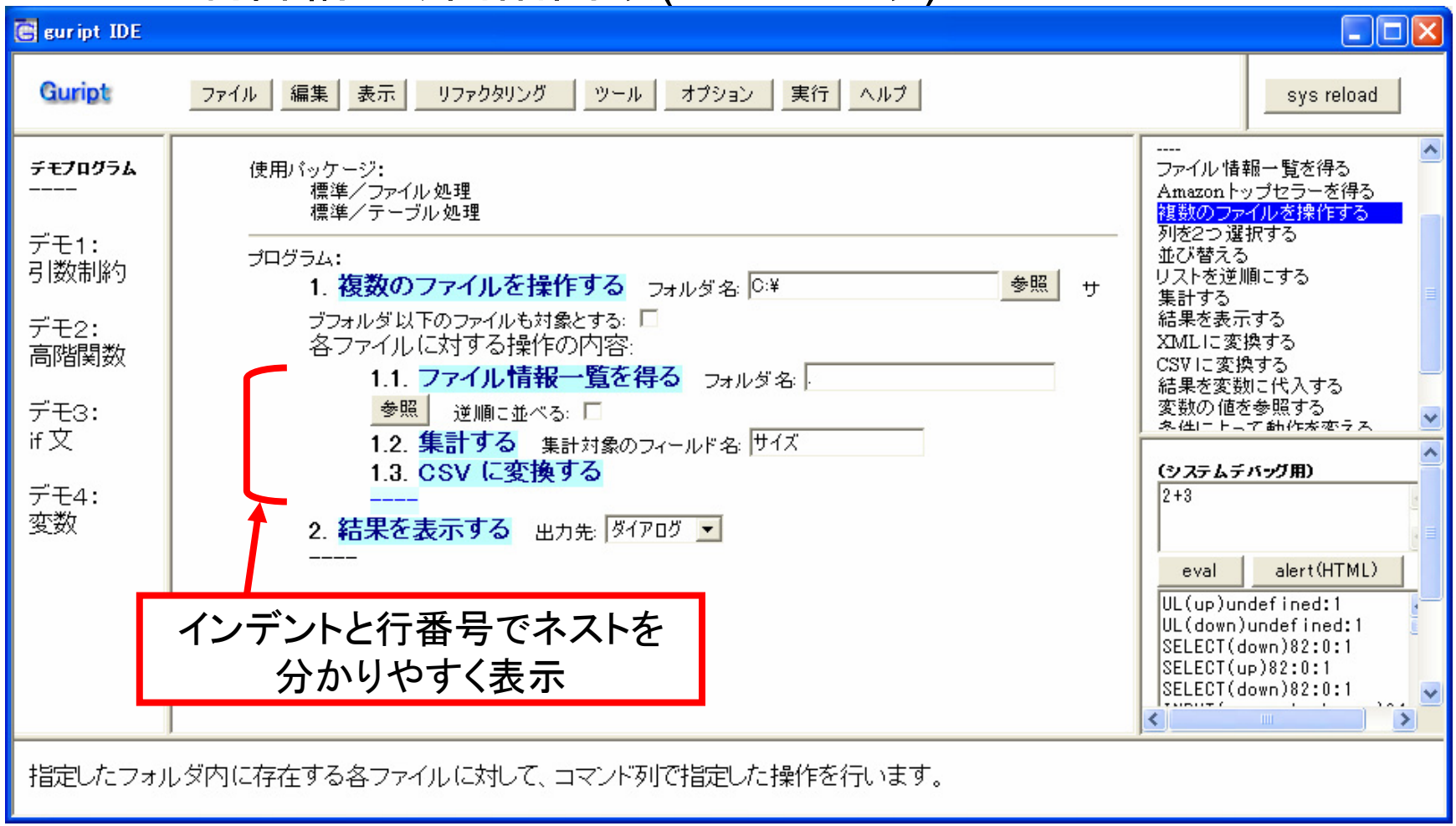
guript は静的型付言語

- 編集中に逐次型チェック、表示オプションで型情報を表示
- 型情報は、パラメタの入力支援にも使われる:**「引数制約」**



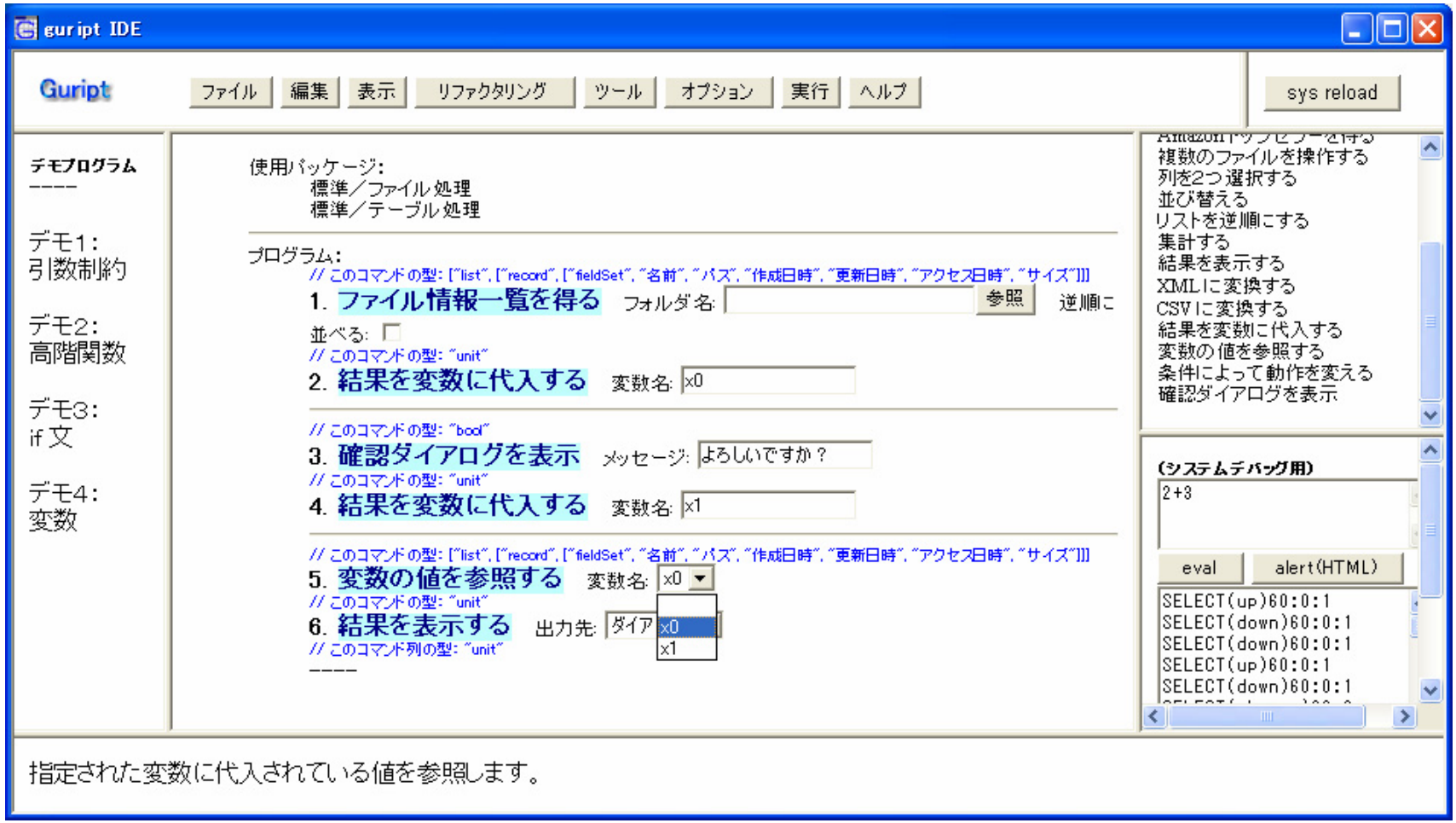
コマンドのネスト

- コマンドの種類によっては、ネストが可能
  - 制御構造、高階関数(block 引数)



変数

- 変数宣言不要、型宣言不要

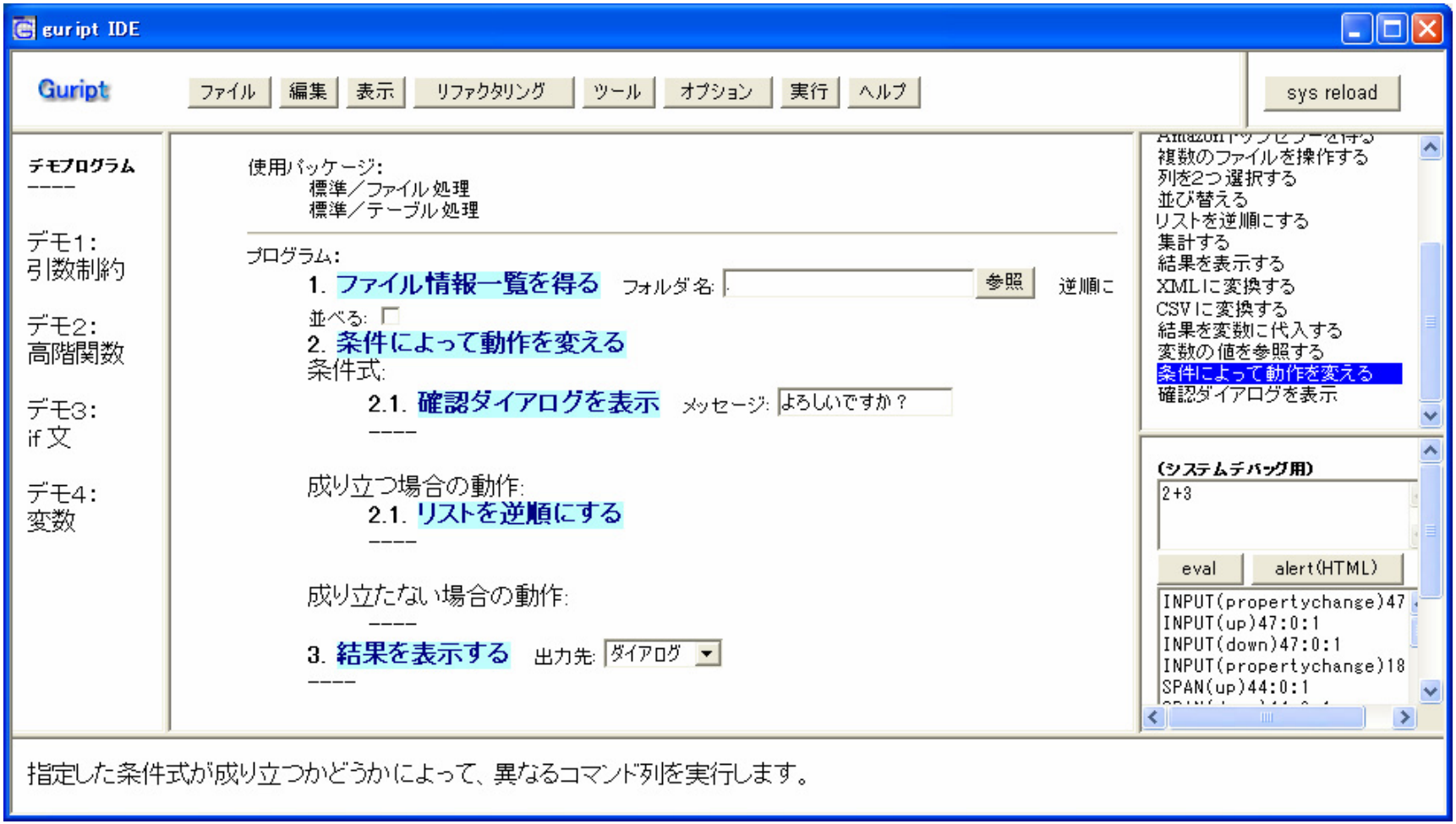


guript の変数の特徴

- 変数のスコープは、1つのサブルーチン全体
- 代入文では、自動的にデフォルトの変数名 x1, x2, ... が付けられる
- 変数の参照では、変数名をセレクタで選ぶ
- 変数の型は代入の直前のコマンドの型で決まる
- 同じ変数に2度以上代入するときは、型が同じでなければ型エラー
- 代入されない変数を参照するとエラー
  - (Java でいう「確実な代入」)

条件分岐

- 条件式の値によって then か else に値が流れる
- 型 : 'a->(unit->bool)->('a->b)->('a->'b)->'b



構造エディタの利点

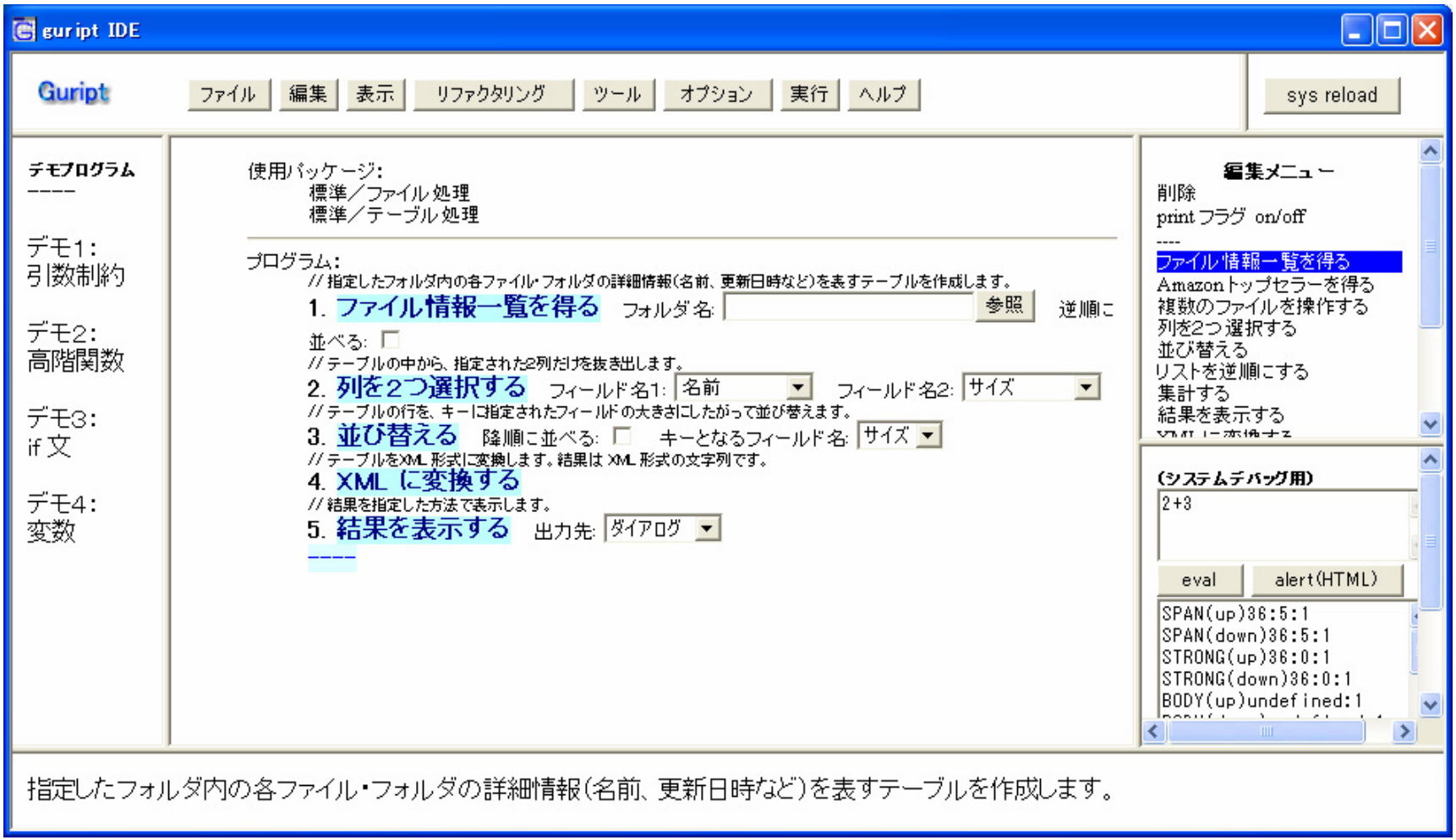
- シンタックスエラーが起きない
- コマンド名、パラメタ名の日本語化（国際化）
- 引数制約などによる入力支援
- さまざまな付加情報の提示
- 文字列リテラル内のエスケープが不要

→ 非プログラマの人には必須の機能

（プログラマにもありがたい）

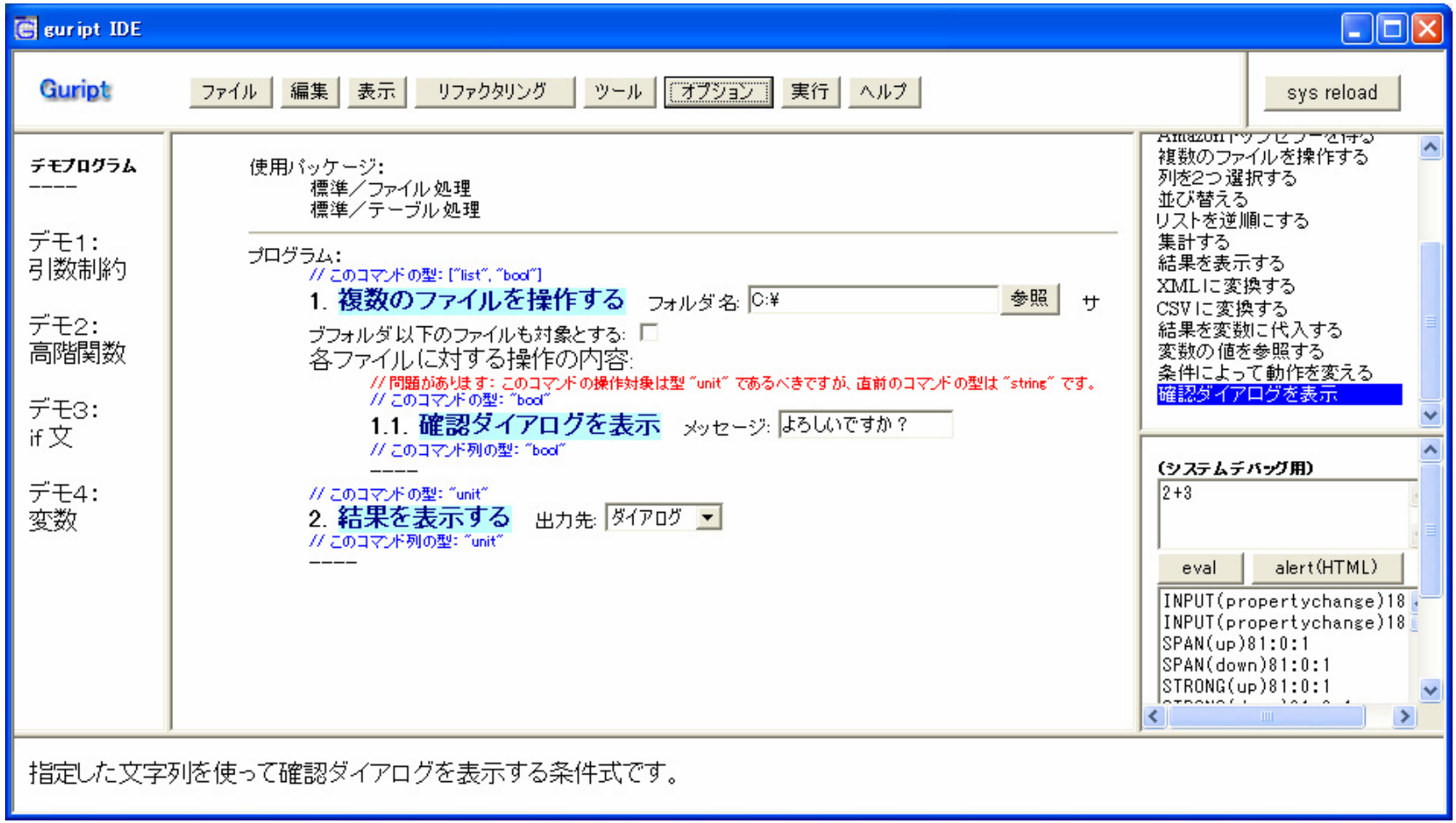
表示オプション：説明を表示

- 各コマンドの上に簡潔な説明を表示
- マニュアルがなくても、印刷したソースコードが読める



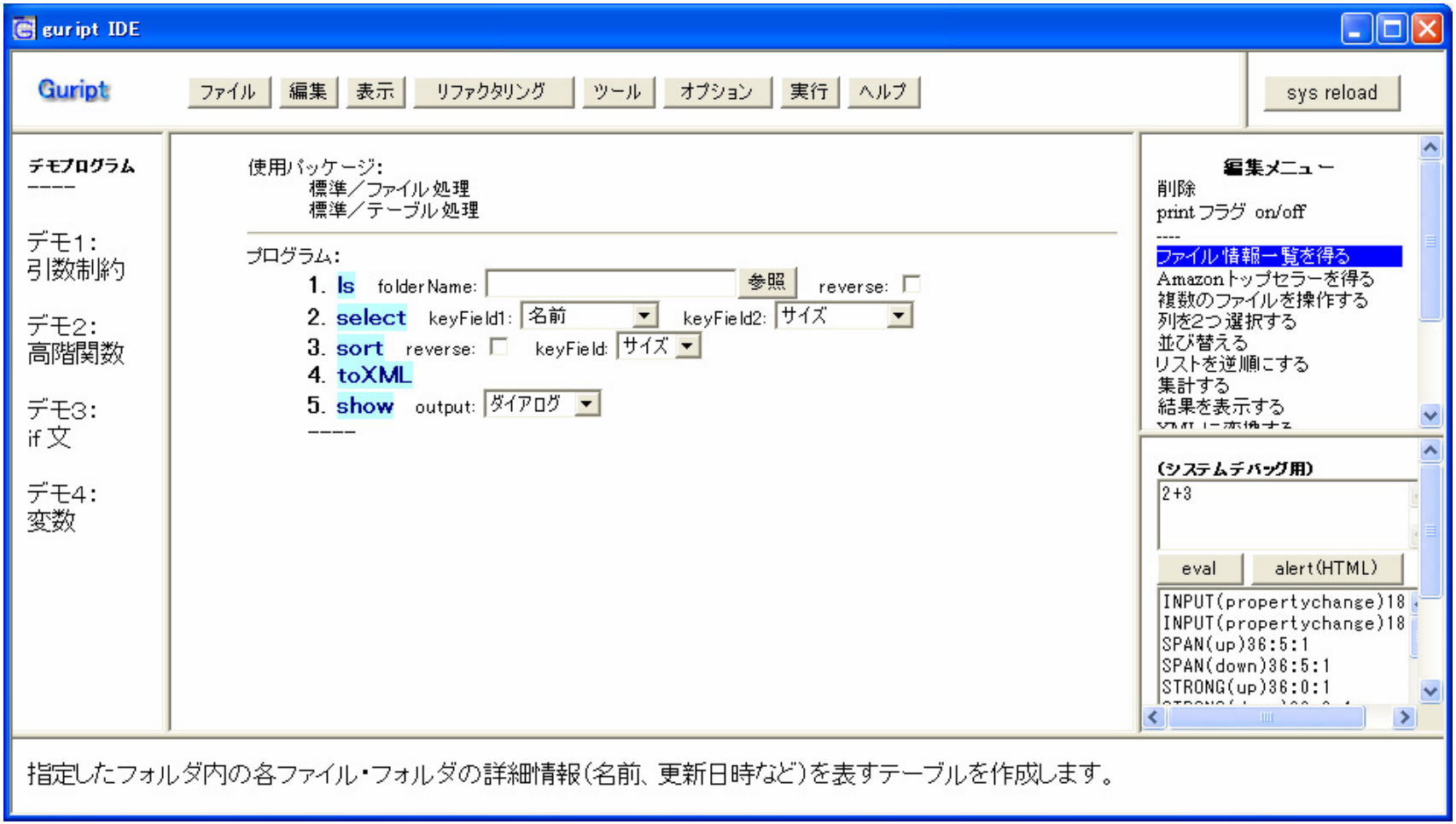
表示オプション：型エラーメッセージ

- 種々のエラーを編集中に随時表示
- **エラーがあっても実行可能**



表示オプション：簡潔表示モード

- 上級者向け
- **慣れればキーボードで直接コマンドを打てる**（未実装）



実装

- 現在のところ IE6 上の JavaScript で記述（約1800行）
  - LiveConnect で Java と連携可能
    - Webサービス、データベースなどへのアクセス
- Windows HTA(HTML application) として起動
  - ActiveX が利用可能
    - Excel, IE, Outlook などの自動制御
- guript コマンドの追加は容易

応用の可能性

- 一般のパソコンユーザの作業の自動化
  - ファイル整理、メール自動処理、フォーム自動入力、Webサイトの自動更新
  - Webサービス
- エンドユーザコンピューティング
  - CMS、グループウエア、ワークフローシステムなどの動作設定
- 情報家電
  - 膨大な録画データからの情報検索
  - エアコン、ロボット掃除機などの動作設定
- 教育用

今後の研究課題

- 実用的ライブラリの充実
  - Webサービス、表処理、XML処理、・・・
- 引数制約のフレームワーク確立
  - 型推論、mediator pattern などを検討
  - 入力の快適さと部品の再利用性とのトレードオフ
- サードパーティによるコマンド作成の支援
  - 型チェックルーチンの定義支援等
- 複雑な式も快適に編集可能な構造エディタ
  - パラメタに定数リテラル以外の式を書けるように

関連研究1:SqueakToys (eToys)

- Alan Kay
- 教育用言語
  - ゲームなどが簡単に作れる
  - 構造エディタ
  - 日本語化されている

関連研究2:
Intentional Programming

- **Charles Simonyi** (Microsoft Research)
  - "The WYSIWYG"
- representation と notation の分離
- 構造エディタ
- 問題領域に適した表示
- 公開されていないため、使い勝手は不明

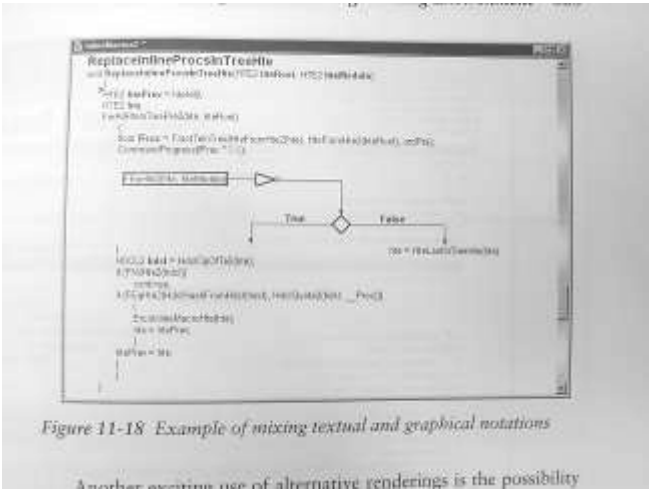
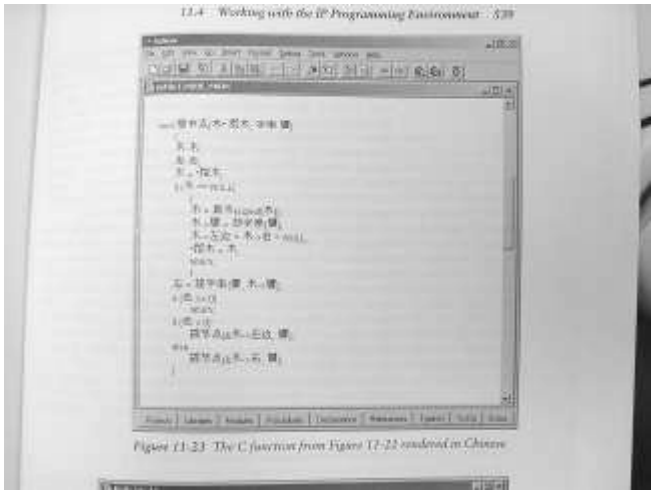


Figure 11-18 Example of mixing textual and graphical notations

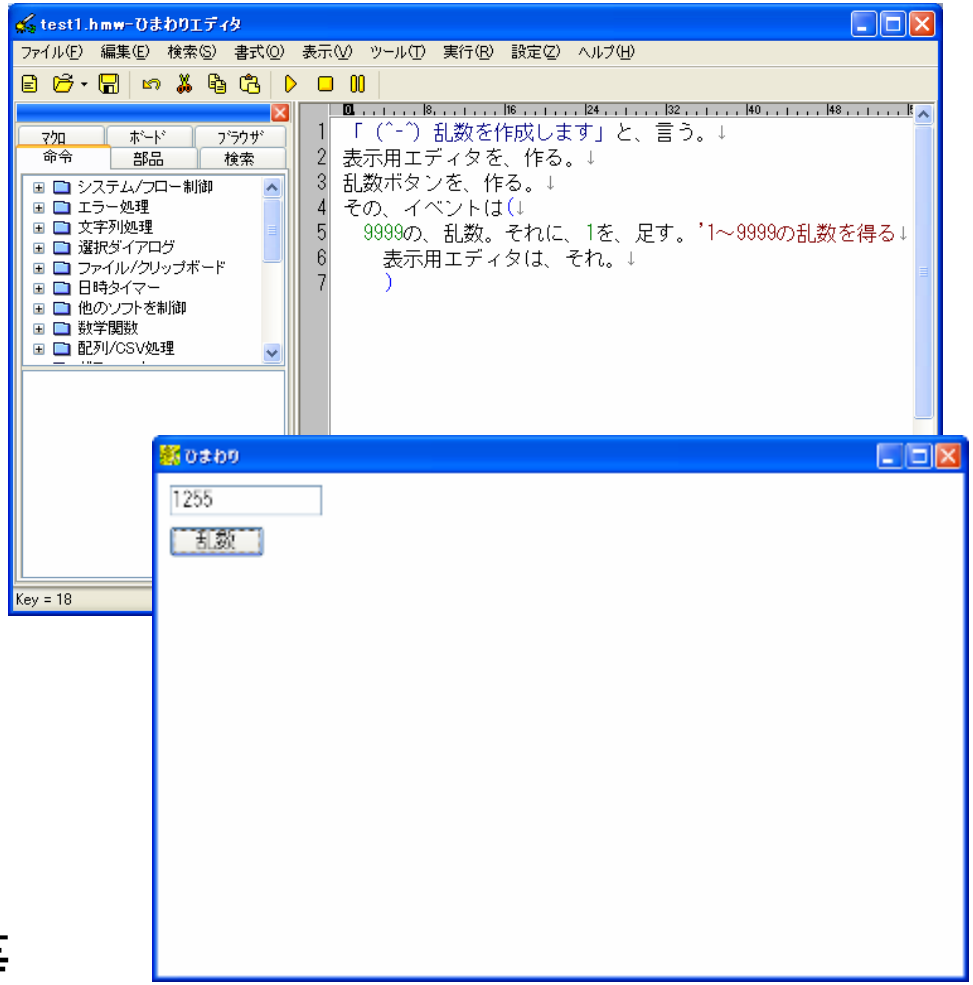
Another exciting use of alternative renderings is the possibility



Generative Programming,
Krysztof Czarnecki, Ulrich W. Eisenecker, Addison Wesley, p.535, p.539

関連研究3：日本語プログラミング

- ひまわり、TTSneo
  - フリーソフト
  - とっつきやすい
  - わかりやすいマニュアル
  - すぐれた開発環境
  - 実用的なライブラリ
- 他に、Mind 、言霊、旧AppleScript など
- 入力に多少難あり
  - 語順、送り仮名、句読点等に関する自明でない規則



ひまわりエディタと
プログラム実行例

構造エディタの可能性

- テキストベース言語の限界
  - block 引数など、構文解析に苦労(ruby, groovy)
  - HTML、XML リテラルは
    テキストエディタで扱いづらい

```
html {
 head {
  title("XML encoding with Groovy")
 }
 body {
  h1("XML encoding with Groovy")
  p("this format can be used as an alternative markup to XML")
  a(href:'http://groovy.codehaus.org') ["Groovy"]
  p [
    "This is some", b("mixed"),
    "text. For more see the",
    a(href:'http://groovy.codehaus.org') ["Groovy"],
    "project"
  ]
 p("some text")
 }
}
```

groovy による HTML の記法

- すでに汎用エディタより IDE が主流に
  - コード補完、リファクタリング、・・・
- Microsoft と Intentional Software の動向

かつての構造エディタの問題点

- 1970s～1980sの構造エディタはなぜ使いにくかったのか？
- 単に作りこみが甘い
  - キー割り当てや編集機能に、emacs/vi ほどのこだわり・工夫が見られる構造エディタがなかった
- 既存のテキストベース言語を扱っていた
  - テキストエディタに最適化された文法は、構造エディタで扱いにくいのは当然
- 四則演算など式の入力・編集に困難
  - 最もチャレンジングな問題、現在取り組み中