社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－SE－144 (7)
2004／3／18

プロトタイプモデリング環境のための記述言語及び処理系の設計

斎藤 貴　　上田 賀一

茨城大学大学院 理工学研究科 情報工学専攻
〒316-8511 茨城県日立市中成沢町 4-12-1

本研究では，メタ階層に基づいたプロトタイピング環境においてモデルの意味記述に用いるERF言語の仕様とその処理系を設計した．まずERF言語の基本パラダイムをオブジェクトベース概念とし，言語設計のための方針を掲げ，それに基づいて基本文法や制御構文の扱いを決定した．言語処理系の設計では，実装すべきオブジェクトの種類，属性およびメソッドをまとめた．また，言語処理の流れをまとめ，処理に必要なデータ構造の設計を行った．さらに，ERF言語の記述性や有用性を確認するためにモデル作成例を示した．その結果，ERF言語は開発の各工程の担当者に必要な専門性の区別という目標を達成していると考えられた．

Design of modeling language and its interpreter for prototype modeling environment

Takashi SAITO,　　Yoshikazu UEDA

Ibaraki University
4-12-1 Naka-Narusawa, Hitachi, Ibaraki, 316-8511 Japan

This study presents the design specification of ERF modeling language used for semantics description of models in prototyping environment based on meta hierarchy and its interpreter. First, authors adopt object-base concept to its basic paradigm, made a design plan, and decided its grammer and control flow statements. Secondly, objects, their attributes and methods for the interpreter are collected. Besides, summarized processes of interpretation, and designed data structure. In addition, model examples are shown to confirm the capability of description and usefulness of ERF language. As a result, the language achieved the goal of distinction between specialities each developer needs.

1　はじめに

ソフトウェアの開発手法の一つであるプロトタイピング手法において，ユーザと開発者の意思疎通を容易にするために，システムの構造・振舞いに図式表現を使用することは有効である．しかし，この図式表現がソフトウェア開発者にとって理解しやすいものでも，必ずしもユーザが要求を表現するために適切であるとはいえない．ここに，開発対象分野（ドメイン）に特化した図式表現の使用を可能とするプロトタイピング支援環境の必要性がある．

本研究の支援環境によるプロトタイピングでは，ドメイン固有のメタモデルを定義するドメインモデル開発 (DM) フェーズと実行可能なモデルを作成するアプリケーションモデル開発 (AM) フェー

ズに工程分割される．また，モデル記述にERF言語を開発し，DMフェーズにおけるメタモデルの意味記述，AMフェーズにおけるベースモデルから作成したアプリケーションモデルの意味記述という2つの場面で使用する．本研究ではERF言語の仕様を策定し，プロトタイピング支援環境に組み込まれる言語処理系を設計した．

本論文では，2章でERFモデルとメタ階層アーキテクチャについて説明する．3章で本研究で扱うプロトタイピングを定義し，その流れを示す．4章でERFモデル記述言語について説明する．5章でERFモデル記述言語の処理系について述べる．6章で関連研究を紹介する．7章で本研究のまとめ及び今後の課題について述べる．

2 ERFモデルとメタ階層アーキテクチャ

2.1 ERFモデル

ERFモデルとは，対象世界を実体（Entity）と関連（Relationship）で表現するERモデルをベースに，要素の集約を表す場（Field）の概念を取り入れ，それぞれの要素をオブジェクトとして扱うことにより，属性と振舞いを持たせることを可能としたモデルである．よって，ERFモデルの要素は以下の3つである．

エンティティ　モデルの実体を表すオブジェクト

リレーションシップ　エンティティ間の関連を表すオブジェクト

フィールド　要素の集約を表すオブジェクト

以下にERFモデルの特徴を示す．

- 属性や振舞いを持つことにより，関連の多重度や制約条件のチェックが可能となる．
- フィールドを用いることで，階層関係を表現できるようになる．
- 3項関連以上は複数の2項関連で表現できることから，制約の簡略化のために，リレーションシップは2項関連のみをサポートする．

2.2 メタ階層アーキテクチャ

本研究で扱うプロトタイピング手法[1]では，ユーザのドメインで使用される図式表現を用いて，プロトタイプの設計を行うことが可能である．これを実現するために，プロトタイピング支援環境はドメインに特化した図式表現を統合的に扱う機能を持つ必要がある．このため，図式表現の構造・振舞いを定義するメタモデルを支援環境上で作成する．ここで，メタモデルを「モデルを記述・解釈するモデル」と定義する．

本研究のメタ階層アーキテクチャ[1][2]は，上記のモデル・メタモデルに加えて，メタモデルを記述・解釈するためのメタメタモデルを定義する．支援環境ではメタメタモデルにERFモデルを採用している．メタモデルの作成においては，まずモデルの構成要素を最少限網羅したベースモデルを用意し，ベースモデルの要素を，エンティティ・リレーションシップ・フィールドといったERFモデルの要素を用いてモデリングを行う．ここで，ベースモデルとメタモデルを合わせてドメインモデルと呼ぶ．

3 プロトタイピング手法

本研究のプロトタイピング手法は，生成されたプロトタイプは要求定義のために試用するものであり，製品版は別に作成する方式を想定している．この場合，プロトタイプと製品版を別に作成するため開発コストは比較的多くかかる．そのかわり，プロトタイプの作成において，ユーザのドメインに特化した図式表現を用いてプロトタイプを表現し，動作を確認できるという特徴を持つ．これにより，ユーザが不慣れな図式表現を用いる場合より多くの要求を引き出すことが可能となる．

本研究のプロトタイピングは，ドメイン固有のメタモデルを定義するドメインモデル開発(DM)フェーズと，実行可能なモデルを作成するアプリケーションモデル開発(AM)フェーズに工程分割され，それぞれの工程を別の担当者が受け持つことが可能である．図1にプロトタイピングのおおまかな流れを示す．

DMフェーズではドメインモデル開発者がベースモデルの作成からメタモデルの定義までを担当し，AMフェーズではアプリケーションモデル開

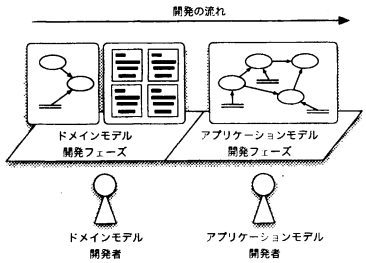

図 1: 本プロトタイピング手法と開発フェーズ

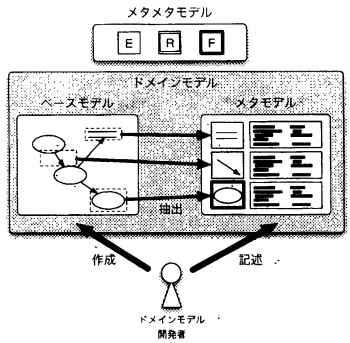

図 2: ドメインモデル作成のイメージ

発者がアプリケーションモデルの作成とモデルのシミュレーションを行う．以下でそれぞれの工程について説明する．

3.1 DM フェーズ

ドメインモデル開発者は DM フェーズにおいて，ドメインモデルの作成を行う．ドメインモデルとはベースモデルとメタモデルの組である．

ベースモデルは，ユーザが属するドメインで使用される図式表現の要素を最少限網羅したモデルである．メタモデルはベースモデルの要素について，各要素の持つ属性やメソッドを定義し，制約条件を定めたモデルである．ドメインモデル開発者はベースモデルの各要素をメタメタモデルの定義に従って抽出・記述する．また，これらの要素が備えるべき属性・メソッドを ERF モデル記述言語を用いて記述する．

このようにドメインモデル開発者は，対象ドメインに関する深い知識が必要であり，メタ階層アーキテクチャや ERF モデル記述言語についても高い専門性が求められる．図 2 にドメインモデル作成のイメージを示す．

3.2 AM フェーズ

AM フェーズはアプリケーションモデル開発者が担当する開発工程である．アプリケーションモデル開発者は，ドメインモデル開発者が作成したドメインモデルををコピー・編集することにより，実行可能なアプリケーションモデルを作成する．ドメインモデルにはアプリケーションモデルの要素として備えるべき，属性や制約条件といった情報を全て保持しているため，アプリケーションモデル開発者はドメインやメタ階層アーキテクチャに関する深い知識は必要ない．ここに，両開発者に必要とされる専門性を差別化することができる．図 3 にアプリケーションモデル作成のイメージを示す．

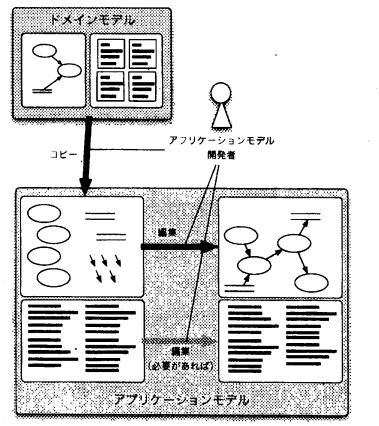

図 3: アプリケーションモデル作成のイメージ

アプリケーションモデルの完成後，アプリケーションモデル開発者はシミュレーションを行い，ユーザとともにアプリケーションモデルの評価を行う．図 4 にアプリケーションモデルのシミュレーションのイメージを示す．

本来プロトタイピング手法は，プロトタイプの完成を明示することが困難な手法である．本手法ではシミュレーション終了後も，ユーザからの更なる要求を反映してアプリケーションモデルの修正を行うことが可能である．また，シミュレーショ

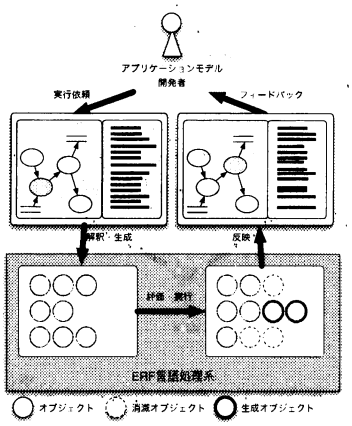

図 4: シミュレーションのイメージ

ン実行中も任意の時点で実行を中断し，オブジェクトの修正を行い，実行を再開する機能を目指す．

3.3 支援環境

本研究室では，前節まで述べたプロトタイピング手法を用いた開発を支援するための環境の構築を行っている．支援環境は以下の機能を持つ．

- GUI による図式表現の作成

  本支援環境はオープンソースソフトウェアである Argo/UML がベースである．Argo/UML はカリフォルニア大学アーバイン校にて開発された CASE ツールで，GUI を利用して UML のモデル作成が可能である．本支援環境は Argo/UML の GUI モデル作成機能を拡張し，UML 以外の図形の編集を可能にした．この機能を利用することで，ドメインモデル・アプリケーションモデルの図形部分を作成することができる．

- ERF モデルを概念としてサポート

  メタモデルとアプリケーションモデルの意味記述には ERF モデル記述言語を使用する．本支援環境は意味記述のためのエディタを用意するとともに，記述された ERF 言語の解釈機能を有する．

- 作成されたドメインモデル・アプリケーションモデルの保存と読み出し

  一度作成されたモデルは特定のフォーマットにより保存が可能である．このため，同じドメインで異なるプロトタイプを作成したい場合，以前作成したドメインモデルを読み出すことによって，アプリケーションモデルの構築のみでプロトタイプの作成を行うことができる．

- アプリケーションモデルのシミュレーション

  アプリケーションモデル作成後，支援環境に実行指示を出すことで，アプリケーションモデルのシミュレーションを行う．シミュレーション実行中は変化したオブジェクトの状態に応じて随時図形部分の更新が行われる．

なお，記述された ERF 言語の解釈・実行は ERF 言語処理系が行う．ERF 言語については次章で，言語処理系については次々章で詳しく説明する．

4 ERF モデル記述言語

本章では，メタモデル及びアプリケーションモデルの意味記述に用いられる ERF モデル記述言語について説明する．

4.1 言語の性質

属性・メソッドを持つことから，全ての要素をオブジェクトとして扱う．また，プロトタイプベースのオブジェクト指向の概念を持ち，オブジェクトの生成はコピーで行う．従ってクラスやインスタンスの概念はない．

なお，以前本研究室で行われた研究の一つに，同じプロトタイプベース言語である Bramble 言語 [3][4] の開発がある．本研究では ERF 言語の仕様を決める際に，Bramble 言語の特性を参考にした．

4.2 設計方針

ERF 言語は以下の機能を実現する必要がある．

- メタモデルコンポーネントの宣言

  ベースモデルのコンポーネント，コンポーネント間の関連，コンポーネントの集約をそれぞれ ENTITY, RELATIONSHIP, FIELD

として宣言する．そのための宣言構文を用意する．

- メタモデルコンポーネントの属性・メソッド定義

  メタモデルコンポーネントの状態・振舞い・制約条件を表現するための属性・メソッドを定義する．

- アプリケーションモデルコンポーネントのコピー・編集

  アプリケーションモデル作成は，ベースモデルのコンポーネントや他のアプリケーションモデルコンポーネントをコピー・編集することで行う．そのためのコピー・編集用の構文が必要である．

- アプリケーションモデルオブジェクトの評価順

  アプリケーションモデルのシミュレーションを行うとき，どのオブジェクトの評価から始めればいいのかを示す順番を定義する．

4.3 言語仕様

ERF 言語で書かれた記述文の基本単位はメッセージ送信である．メッセージを受け取ったオブジェクトは，メッセージと同名のメソッドを起動して処理を行う．メソッドに含まれる最後のメッセージ文の評価オブジェクトが，メッセージの返り値となる．

4.3.1 メッセージ文

メッセージ文の書式を図5に示す．

```
ERFobj "<-" msgName parName:parObj ...

ERFobj   メッセージを受け取るオブジェクト
msgName  メッセージ名を表す文字列
parName  メッセージのパラメータ名
parObj   メッセージのパラメータオブジェクト
```

図5: メッセージ文の書式

メッセージによってはパラメータを複数指定することもある．パラメータは名前・オブジェクトがセットになったタグとして扱われ，パラメータを記述する順番に規定はない．ただし，パラメータ数やパラメータの種類が異なるメッセージは別メッセージとして扱われる．

メッセージで実現している構文は以下のようなものがある．

- メタモデルコンポーネントの宣言文

  それぞれ Entity, Relationship, Field というキーワードに続けてコンポーネントの名前を記述する．なお，コンポーネントの属性やメソッドを定義するためには，対象となるコンポーネントの宣言が属性・メソッド定義より前に行われていなければならない．

- 属性・メソッドの定義文

  属性・メソッドの定義はメッセージ文で行う．宣言後のコンポーネントに対して slot メッセージを送信することで定義を行う．メッセージの引数には属性・メソッドのオブジェクトタイプを記述する．オブジェクトタイプの種類を以下の表1に示す．

- 制御構文

  制御構文はメッセージ文で表現する．実装される制御構文は if 文，while 文，switch 文，foreach 文の4つがあり，真偽値オブジェクト（true, false），数値オブジェクト，文字列オブジェクト，リストオブジェクトのメソッドとして実現される．C 言語や Java 言語に登場する break 文や continue 文は実装しない．これは，将来メタモデルの妥当性を検証しようとした時に，jump 機能を持った制御文を入れることで，妥当性検証のためのツールが，処理を追っていくことが困難になると考えたためである．

5 ERF 言語処理系

ERF 言語処理系は，ドメインモデル開発者によって記述されたメタモデルの意味記述の解析と，アプリケーションモデル開発者が作成したアプリケーションモデルの意味記述の解析・評価を行う．処

表 1: オブジェクトタイプ

属性系	説明
string	文字列型
integer	整数型
real	実数型
boolean	真偽値型
	true もしくは false を指す
list	線形リスト型
array	連想配列型
entity	エンティティ型
relationship	リレーションシップ型
field	フィールド型
メソッド系	説明
method	メッセージ駆動型の処理
pre-action	メッセージ評価前に一度だけ行う処理
in-action	メッセージ受信に関係なく一定の条件で行う処理
post-action	メッセージ評価後に一度だけ行う処理

理系の動作およびデータの流れを図 6 に示す．この章では，これらの動作・データ構造の詳細について述べる．

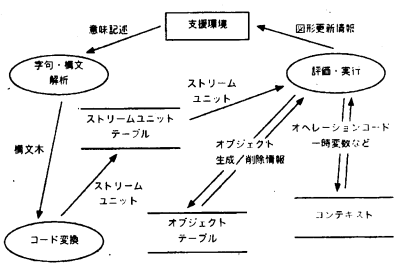

図 6: ERF 言語処理系

5.1 字句・構文解析

言語処理系は始めに ERF 言語を構文解析し，構文木を作成する．ここで，構文解析器を自動生成するツールとして JavaCC[5] を用いる．JavaCC(Java Compiler-Compiler) は Sun Microsystems が開発したフリーソフトウェアで，拡張 BNF で記述された文法規則に従って，字句・構文解析を行うパーサクラスを Java 言語のソースコードとして生成するツールである．

本研究では，JavaCC を利用して，拡張 BNF で記述した ERF 言語の文法規則から，ERF 言語用の構文解析器を自動生成する手順をとる．なお，JavaCC の記述ファイルは Java 言語のソースファイルにステートメントを追加したのみの構成であり，専用のプリプロセッサを通すことで Java のパーサクラスを生成する．

5.2 コード変換

言語処理系は構文解析器によって生成された構文木をストリームユニットの列にコード変換する．ストリームユニットとは，オペレーションコードとその引数をまとめたものであり，オペレーションコードが処理系が言語を評価する際の単位となる．ストリームユニットの列はコード変換の際にメソッド単位でまとめられる．図 7 にストリームユニットのイメージを示す．

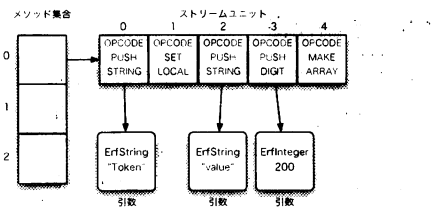

図 7: ストリームユニット

5.3 評価・実行

オペレーションコードへの変換が終わると，言語処理系はユーザのシミュレーション指示とともにオペレーションコードの解釈を開始する．

処理系はストリームユニットテーブルからストリームユニットを取り出しコンテキストに格納する．コンテキストからオペレーションコードを取り出し，コードの解釈に伴って登場した一時変数等をコンテキストに格納する．コードの解釈によりオブジェクトの生成・削除が発生した場合はそ

の情報をオブジェクトテーブルに格納する．処理系は一定のタイミングでオブジェクトの更新情報を支援環境に送信し，支援環境はその情報をもとに図形表現の更新を行う．

5.3.1 コンテキスト

コンテキストはスコープごとに異なるデータを管理するデータ構造である．スコープはERF言語で記述されたプログラムの入れ子の深さを表す概念である．コンテキストが管理するデータは，実行中のメソッドのストリームユニット，メソッドで使用するローカル変数，メソッドの引き渡された引数，オペレーションコードの解釈時にデータを一時的に格納するスタックなどである．

5.3.2 オブジェクトテーブル

オブジェクトテーブルはオペレーションコードの解釈によって生成・削除されるオブジェクトの情報を管理する．オブジェクトテーブルはドメインモデル用とアプリケーションモデル用の2つから成り立つ．

ドメインモデル用はメタモデルの意味記述中で宣言されたオブジェクトを管理する．また，slotメッセージで定義された属性やメソッドの情報をオブジェクトに関連づけて保持する．アプリケーションモデル用はアプリケーションモデルの意味記述中のcopy. removeメッセージで生成・削除されるオブジェクトを管理する．また，slotメッセージでオーバライドされた属性・メソッドの情報をアプリケーションモデルのオブジェクトと関連づけて管理する．

6 関連研究

本研究のERF言語と同様に，フロトタイピングにおける仕様記述のための言語の一つとしてSpecC言語 [8][10] がある．

SpecC言語は1997年にカリフォルニア大学アーバイン校 (UCI)[9] で開発された，ハードウェアとソフトウェアを含むデジタル組込みシステムの仕様・設計を形式的に記述する言語である．SpecC言語はANSI-C言語の最上位層に位置し，組み込みシステムに必要な動作，並列性，通信，同期，例外処理等の記述をサポートする．

SpecC言語を仕様記述に用いるツールにInter-Design社のVisualSpec[10] がある．VisualSpecは組込みシステムのための要求分析以降の仕様設計をサポートするツールである．VisualSpecの想定する開発の流れとSpecC言語の位置づけを図8に示す．

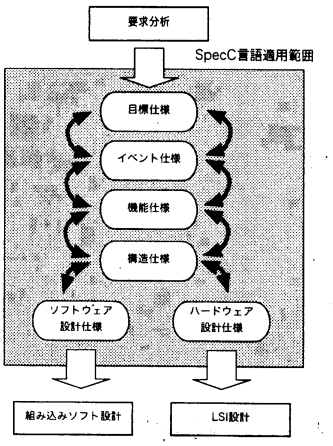

図 8: VisualSpecにおけるSpecC言語の位置づけ

各仕様設計ステップで，開発者は設計データをSpecC言語を用いて記述する．VisualSpecはSpecCコンパイラを保持しており，設計データをC++プログラムに変換できる．C++プログラムを他のC/C++コンパイラに引き渡すか，あるいは専用のSpecCシミュレータを用いることで，各工程で仕様モデルのシミュレーションを行うことができる．仕様モデルはプロトタイピング手法により，シミュレーションの結果に応じて随時仕様に変更を加えていく．

SpecC言語と本研究のERF言語は以下の点に違いがある．

- ドメイン適用範囲の違い

  SpecC言語は組込みシステムの仕様記述に特化した言語である．そのためソフトウェアの仕様はハードウェアとの連携を意識して記述する必要がある．本研究のERF言語はメタ階層アーキテクチャの導入により，ある程度汎用的なドメインに適用可能である．ただし，プロトタイプのシミュレーションはJava

仮想機械上での動作を前提としており，ハードウェアへの干渉は想定していない．

- プロトタイプの利用方法

SpecC 言語で作成した仕様記述は専用のコンパイラを利用することで C++ ソースコードへの変換が可能である．従って，プロトタイプの仕様記述を発展させて製品版を作り出すことができる．本研究のプロトタイピング手法はプロトタイプを発展させていくことを目的としていないため，ERF 言語を他のプログラミング言語に変換する機能は現在のところ考案されていない．しかし，依頼者のドメインに存在するダイアグラムを用いた仕様記述が可能であるという点で，プロトタイピングの目的である依頼者の要求獲得において本研究で使用する言語の方が優れていると考えられる．

7 まとめ及び課題

本研究では，ユーザのドメイン特有の図式表現を用いたプロトタイピングを支援する環境において，モデルの意味記述を行うための ERF モデル記述言語の仕様を策定し，同言語の処理系の設計を行った．

ERF 言語はメタ階層アーキテクチャに基づいたモデリングに使用することを設計方針として掲げ，そのために必要なオブジェクト，属性，メソッドの整備を行った．

言語処理系は支援環境に組み込むことを考え，支援環境と同じ Java 言語で実装することを念頭に設計した．そのため，Java の構文解析器を出力できるツール JavaCC を利用した．また，ドメインモデル，アプリケーションモデルの作成時に順次生成／削除されるオブジェクトを管理するテーブルを用意し，言語処理の流れを整理した．

さらに，本研究では ERF 言語の記述性や有用性を確認するために各フェーズのモデル作成例を示した．DM フェーズではドメインやメタ階層に関する深い知識が要求されるが，AM フェーズではモデルに必要なオブジェクトが列挙できれば問題ないということが考えられた．これにより両者に必要とされる専門性に区別を付けることができ，ソフトウェア開発に関わる両者の負担を減らすことができると考える．

今後の課題としては，言語処理系の実装と支援環境への組み込み方法の考案や，より柔軟な記述が可能となるための ERF 言語文法の改良が挙げられる．

参考文献

[1] 大塚 聖也, 小飼 敬, 上田 賀一："メタ階層を用いたプロトタイピング手法の提案", 情報処理学会 第 140 回 ソフトウェア工学研究会 (2003)

[2] 庄司 龍一："メタ階層アーキテクチャに基づくフロトタイプモデル生成・解釈機構の実現", ソフトウェア工学の基礎 VI, pp.212-219, 近代科学社 (1999)

[3] "BrambleIV Tutorial", 上田研究室 (1999)

[4] 佐々木 貴生："モデル記述言語 Bramble 処理系の再設計", 平成 11 年度茨城大学情報工学科卒業論文 (2000)

[5] Howard Katz : "JavaCC, 構文解析ツリー, および XQuery の文法", URL:http://www-6.ibm.com/jp/developerworks/xml/030221/j_x-javacc1.html

[6] 内田 智史, 秋本 勝, 北川 雅巳, 大津 崇："C 言語によるプログラミング スーパーリファレンス編", オーム社

[7] Joseph O'Neil, 武藤 健志："独習 Java", 翔泳社

[8] "SpecC Technology Open Consortium", http://www.specc.gr.jp/eng/index.htm

[9] "SpecC System", http://www.ics.uci.edu/ specc/

[10] InterDesign 社："VisualSpec", http://www.interdesigntech.co.jp/producttop.htm