タイトル	ユーザインターフェース仕様の検証機能のデジタルモックアップ用プラットフォームへの統合
著者	菊地，慶仁
引用	北海学園大学工学部研究報告，33：143-152
発行日	2006-02-20

ユーザインターフェース仕様の検証機能のデジタルモックアップ用プラットフォームへの統合

菊 地 慶 仁*

Integration of Validation Function for User Interface Specification to Digital Mockup Platform

Yoshihito KIKUCHI*

Abstract

This paper describes an integration of our validation method to BehaveView software platform. The BehaveView platform is based on STEP (ISO10303) architecture and has a schema description of state transition machine and database of state transition specification. Our validation algorithm is integrated to the schema description as constraint rules of the schema to guarantee state transition system's valid data.

1. はじめに

本研究では，デジタルカメラに代表される小型電子機器のユーザインターフェース設計初期段階を対象としている．製品ライフサイクルが短いこれらの機器で，開発期間やコストを短縮させるためには，予め設計初期段階でユーザビリティや仕様一貫性などの検証を行うことで開発途中での仕様変更を減らし，また下流のソフトウェア開発工程で仕様情報を再利用する必要がある．本研究では，このような利用のための，製品仕様の記述方法と検証を目的としている．

これまで，小型電子機器のユーザインターフェース仕様を状態遷移機械として代数仕様の形で形式的に表現し，予め規定した公理系を用いて仕様の妥当性を検証する方式について提案した．合わせてMathematicaによる試作システムの開発と，Visual Wireless Communicatorの仕様を対象として検証を行った結果について報告した[1)]．

しかしながら実用的なシステムへの実装は，その後の課題として残っており，本報告では，

＊北海学園大学工学部電子情報工学科
＊Department of Information and Electronics Engineering, Faculty of Engineering, Hokkai-Gakuen University

その実現方法について報告する．第２章では，以前に提案した仕様の記述及び検証方式の概要と，ソフトウェアプラットフォームのBehaveViewについて述べ，実装における課題を整理する．第３章では，具体的な検証アルゴリズムの具体的な実装について報告し，考察及び結論とする．

2．関連技術と本報告における課題

2.1 公理系を用いた検証システム

2.1.1 検証システムの概要

図１は，モデル記述と検証におけるデータの流れを示している．図はIDEF0（Integration Definition for Function Modeling-0）形式で表現されている[2]．四角い箱はアクティビティを示し，箱に左から入る矢印がアクティビティへの入力を，右側に出る矢印が出力を示している．他に，上から入る矢印が参照用のデータ，下から入る矢印がアクティビティで使用するメカニズムを示す．

アクティビティA0は，状態遷移機械としてユーザインターフェースの仕様を生成する．プロジェクト中の他の研究テーマで，本報告はその出力を利用する立場で進めている．

本報告では仕様の記述と検証に，製品仕様の代数的な記述表現と公理に基づく数学的な検証方法を用いた．これは，C.A.R Hoareによって提唱され計算機言語のPASCALの意味的な仕様をエミュレートする際に用いられた方法である[3]．Margrenは，この方法論を２次元グラフィックディスプレイ上の２つの描画画像が等価であるかどうかの証明に用いた[4]．

この方法での検証は２つのステージで行われる．始めにアクティビティA1で代数的に表現された状態遷移機械の仕様から「表明（Assertion）」が生成される．表明は，内部変数と状態遷移に関する事実の集まりである．表明は代数式による仕様宣言の集成とも言える．

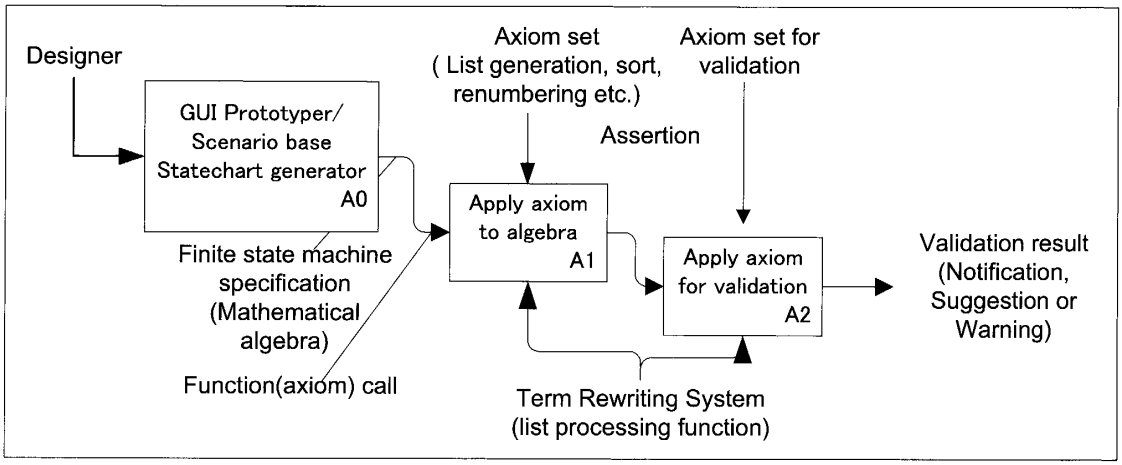

図１ 代数表現した状態遷移仕様に対する公理を用いた検証
Fig. 1 Validation of algebraic specification for state transition based on axiom set

「公理（Axiom)」は論理学的な取り扱いでは「自明の，もしくはあまねく認識されている真実」である．代数仕様と表明の間には1対1の対応関係がある．公理は，代数仕様から表明の変換を定義するルールであり，代数仕様が表明中に残す効果を表現している．全ての代数的仕様を公理によって変換すると，最終的な表明が残される．この作業によって状態遷移機械の数学的な表現を導出したことになる．

アクティビティA2では，検証項目毎に用意した公理が表明に適用されて仕様の検証が行われ，発見された事実の告知や，代替案の示唆や警告などが出力される．

図2に，前報告で行ったVisual Wireless Communicatorの代数的仕様表現の例を示す．Start() は仕様の開始を示し，以下状態名（State)，イベント（Event)，アクション（Action)，状態の平行階層構造（HierarchalState）及び状態遷移（StateTransition）を示している．

2.1.2 検証項目

また前報告では，次に上げる項目で公理系とプログラムを用意して状態遷移機械の検証を行った．

(a) 状態遷移機械としての基本的な性質

1）完全性：全ての状態から，少なくとも一つの他の状態への遷移が存在するかどうかの検証

2）強連結性：任意の状態u及びvの間での状態遷移と，その逆の遷移が可能であるかどうかの検証

3）状態数の最小性：同じイベントとアクションの組み合わせで並行した状態遷移が定義されているかどうかの検証．該当する場合は，状態遷移の設計が不完全で，後述する非決定性問題を起こす可能性がある．

(b) 階層構造の検証

```
Start();
State({"Power_Off","Power_On","Waiting"
       ...});
Event({"PowerButtonON","PowerButtonOFF"
       ,"Failure"...});
Action({"curr=y","curr=x","prev=none","
       prev=curr"...});
HierarchalState(
       {"Power_On",
       {"Waiting","Trying_to_Unit_x","Tr
       ying_to_Unit_x","Connect_to_Unit_
       curr","Connect_to_Unit_self"},
       {"use_start_state","Waiting"}});
         :
StateTransition({"Power_Off","Power_On"
       ,"PowerButtonON"})
```

図2　Visual Wireless Communicatorユーザインターフェースの代数的仕様表現例
Fig. 2　Example of algebraic description of Visual Wireless Communicator user interface specification

この項目では，検証はさらに細分化された詳細な項目に関して行われる．

1）状態の親子（サブ／スーパー）関係（階層構造）の定義が一貫したツリー構造となっているか．

2）サブ状態からスーパー状態へ戻る遷移が少なくとも一つ定義されているかどうか．

3）スーパー状態への遷移が定義されているときに，その中で初期状態が宣言されているか．

4）終了状態への遷移が定義されているときに，終了状態に対応する外部の状態が宣言されているかどうか．

5）初期状態／終了状態への直接の遷移が定義されていないかどうか．

6）サブ状態からの遷移が一定の方式に従っているかどうか．これはGUIの表現などで操作の一貫性の評価に繋がる．

(c) 非決定性仕様

階層構造の中で２つ以上の遷移が，同じイベント／アクションの組み合わせで発生する場合，その仕様は非決定性問題（矛盾した仕様）を含んでいる可能性がある．例えば，同じイベントによって階層構造の外側への遷移と，階層構造の内側での遷移が定義されているような場合である．

2.2 BehaveViewプラットフォーム

BehaveViewは，３次元CADからの製品形状データ，機器から出力される動作に関する情報，状態遷移仕様を統合してデータベース管理することで，レポート出力や，設計対象のWeb 3D出力によるユーザビリティ評価を目的として本研究の別プロジェクトとして開発された．図３に全体的な構成を示す．以下に，その技術的な特徴について述べる．

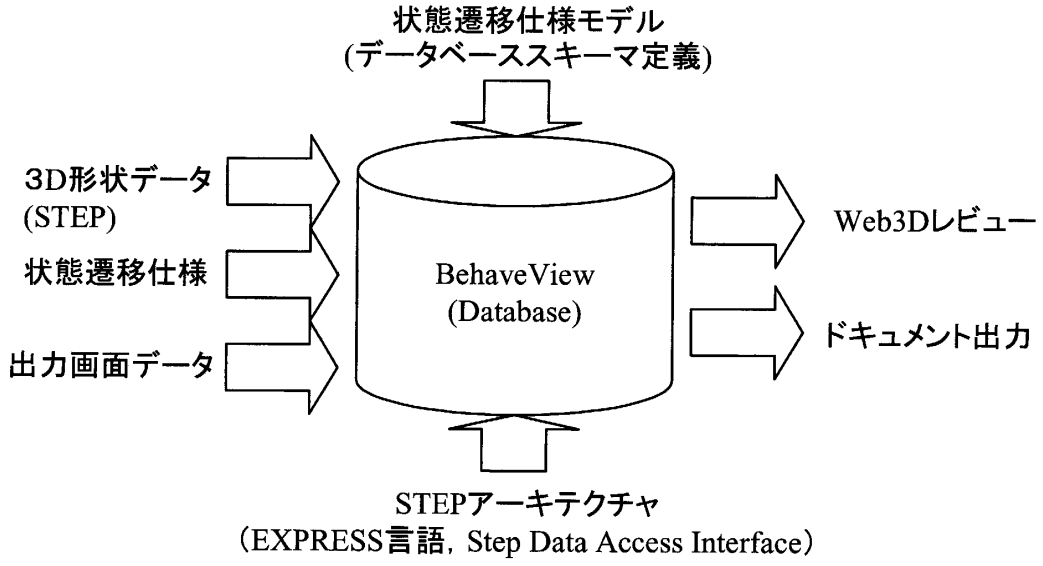

図３ BehaveViewの機能
Fig. 3 Functionality of BehaveView

2.2.1　独自の状態遷移モデルとスキーマ記述

BehaveViewではWeb3Dによるユーザビリティ評価を一つの目的としている．このため，設計対象となる機器の状態遷移に関する仕様情報とWeb3D上での筐体動作やLCDモニタからの出力に対応する画像イメージなどを連動させる必要がある．そのためStateChartsに代表される状態遷移機械に筐体動作や画像などの外部出力情報を付加した独自の状態遷移モデルを用いている．またXMI形式で表現されたStateChartsに準拠した状態遷移モデルをインポートすることも可能である．

2.2.2　STEP技術に基づいたアーキテクチャ

システムの特徴として，BehaveViewは状態遷移仕様データのメタモデル記述と関連するデータの入出力にISO10303（STEP：Standard for Exchange of Product Model Data）[5]に基づく方式を用いている．

ここでの状態遷移仕様とは，機器がある状態にある場合にスイッチ押し下げなどのイベントが発生すると，どのような別の状態に遷移し，どのようなアクションによって外部出力が行われるかといった情報である．この情報をシステムで取り扱い可能とするために，システムでの操作対象となる“状態”，“イベント”，“アクション”などの概念とそれらがどのようなタイプの属性値を持った上で相互に関連しているかを定義する必要があり，STEPの世界ではこれらを製品モデル呼んでいる．

状態遷移機械に関する製品モデルは，ISO10303で開発された形式仕様記述言語EXPRESS[6]で記述される．EXPRESSでは，どのような属性を用いてデータ構造が作られるかを定義することが可能であり，また個別の要素やモデル全体での一貫性を維持するための制約条件を定義することができる．このような形式的な言語を用いることで，人間に対しての曖昧性の無い理解，計算機による自動処理，図式表記などを可能とする．

図4に状態遷移モデルの図式表記を示す．この図はEXPRESS言語の図式標記用規格のEXPRESS-Gに基づいている．図の太線は上位型／下位型での継承関係であり，細い実線は属性としての参照関係である．図の黄色の四角が独立して存在できるエンティティであり，緑の四角はエンティティから参照される属性値である．

図の上部が，状態遷移に関するモデル記述部であり，アクションに関するモデルを下部で行っている．具体的には，アクションに関する抽象的なエンティティを定義し，実際に画像の出力や筐体動作などを行うアクションを，その下位型として宣言し，必要な画像などを属性として参照できるようにしている．

上記の状態遷移モデルは，SDAI（Step Data Access Interface）[7]に基づくデータベースに読み込まれる．SDAIは，EXPRESS言語で表現されたデータを読み込んで処理を行うためのデータベースアクセス処理手順を各種計算機言語用のAPIとして規定している．この仕様に基づいて

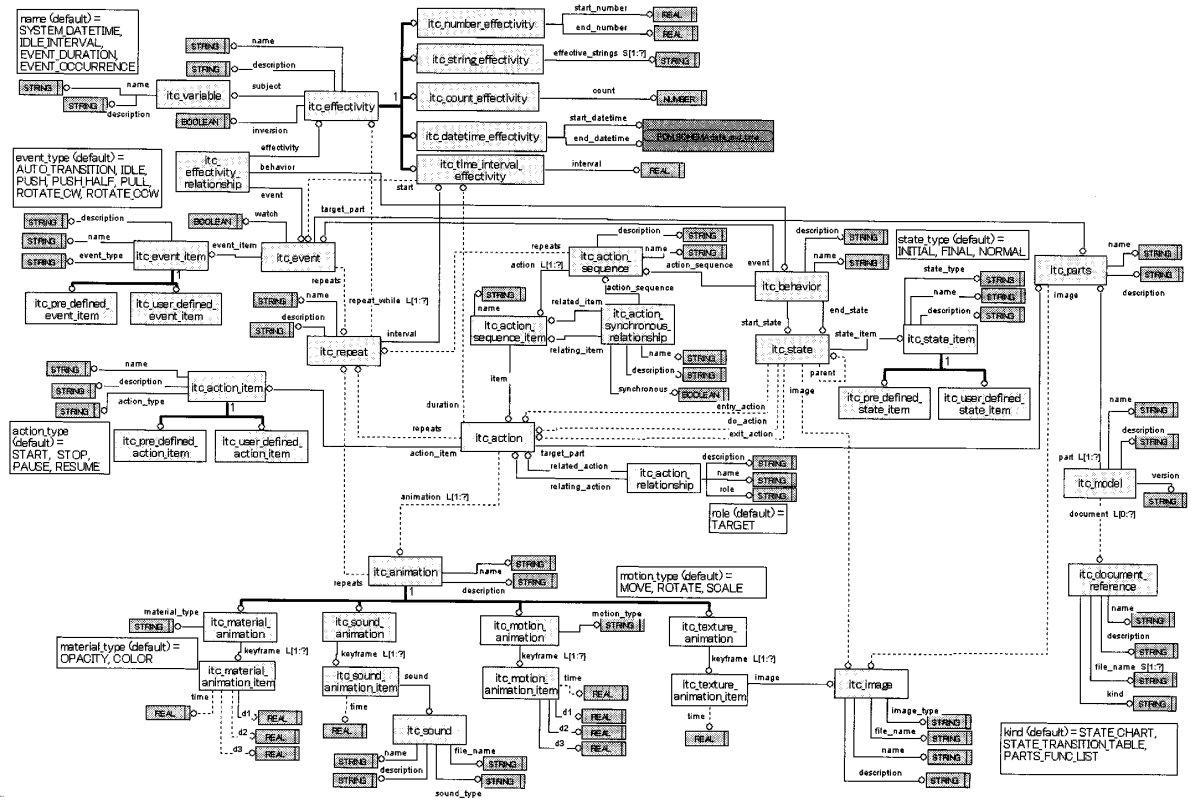

図 4　BehaveViewシステムのデータベースに格納される状態遷移モデルの構造（EXPRESS-G標記）
Fig. 4　Structure of state transition model stored in BehaveView system database (EXPRESS-G Diagram)

実装されたデータベースでは，EXPRESS言語で表現された製品モデルをスキーマ定義として読み込み，対象機器の形状データ，状態遷移データ，機器のLCDモニタからの出力画像データを自由に入出力し，Web3D出力などを行うことが可能となる．

2.3　統合の方式についての課題

1）プラットフォームの違いの統合．これまで開発した公理系に基づく検証アルゴリズムを，BehaveViewとを統合するためには，（1）検証アルゴリズムの実行は，試作システムを開発したMathematicaプラットフォーム上で行う方式と，（2）BehaveViewが実装されているSTEPアーキテクチャ上に移植し，その上で検証アルゴリズムを実行させる2つの方式が考えられる．検証システム試作はMathematicaで行ったが，Mathematica特有の数式処理機能などは用いずに，主にリスト処理機能しか用いていない．このため，実行上の違いは特に無いと考えられ，本報告ではSTEPアーキテクチャに移植する方式をとった．これは，モデルを形式的に表現する際には，そのモデルが一貫していて正しいと判断することができる制約条件を課すことが必ず必要となるが，BehaveViewの状態遷移モデルでは，モデル全体としての一貫性を定義するための制約が完全には整備されていない．このために，不完全な状態遷移モデルでも受け入れてしまうことになるので，状態遷移機械に関する仕

様表現として完全な形にする意味で，実行プラットフォームを分けないで統合する方法を取った．

2）公理系に基づいたアルゴリズムの実装．前報告では検証アルゴリズムは，Mathematicaをプラットフォームとして，代数仕様形式から表明（Assertion）に変換してから検証を行う形で実装した．これをBehaveViewのデータモデルに対して適用する場合，表現の形式が異なっているための対応が課題となる．この課題については，3.3で述べる．

3．BehaveViewへの検証アルゴリズムの統合

3.1　統合の方式

前報で報告した検証アルゴリズムは，以下の構成をとってEXPRESS言語によって書き直す．検証の種類によって，どのような仕様となるかを以下に示す．

1）状態遷移機械のモデルが一貫していることの保証

完全性，強連結性，非決定性仕様など，状態遷移機械が常に満たしていなくてはならない機能はEXPRESS言語でのRULE節中で定義する．RULE節は，特定のエンティティに対して，及び複数のエンティティの組み合わせ，すなわちモデル全体に対して規定することができる．このように設定したRULE節は，データベース中へのデータの読み込みや何らかの操作が行われる度にデータベース管理システムによって自動的に起動される．RULE節は，そのモデルが正常な場合は，常に真の値を返し，モデルの一貫性が失われて正常ではなくなった場合は真以外の値を返す．

2）状態遷移機械モデルの一貫性についての警告

RULE節は，データベースに格納されるデータの全体が常に満たしていなくてはならない制約条件であり，仕様に対して非常に強い拘束力を持っている．一方，必ずしも満たしていなくても問題は無いが，仕様の明晰性や仕様変更などの際に構造が変わってしまう恐れがある仕様には，警告を出すためのレベルの検証がある．このようなレベルの検証は手続きPROCEDUREとして，データベース操作の際に呼び出すこととする．

3）検証用関数群

上記の2つの検証機能は，基本的には同じ公理系に基づいた検証であり，呼び出すアルゴリズムは共通のものとなる．したがって，このように共通に呼び出す機能は関数として定義する．

3.2　代数的表記の取り扱い

前報告では，代数仕様形式から表明（Assertion）に変換してから検証を行う形で実装した．これは，代数仕様形式が，プログラム言語としては一つ一つの事実の宣言に対応するために，公理として処理可能なリスト構造に置き換える必要があったためである．これをBehaveView

のデータモデルに対して適用する場合は，個別の宣言はEXPRESS言語に従った状態遷移機械仕様となるが，システムへの入力としてはSTEP規格のPart21[8]となる．Part21はSTEP規格に従ってデータモデルをシーケンシャルファイルの形式として入出力するための規格である．しかしながら，BehaveViewではSDAI規格に従ってPart21ファイルを読み込むことが可能となっており，データベース内部では，EXPRESS表現されたデータとして格納されている．

従って，本報告ではデータベースに格納されたEXPRESS表現のモデルを表明（Assertion）として扱い，このデータを順に辿る形をとる形で検証を行う．

3.3 RULE節での制約の記述

図5にRULE節として記述した検証アルゴリズムの一部を示す．始めにitc_behaviourエンティティとitc_stateエンティティを示している．このitc_behaviourエンティティが，開始状態，終了状態と，その状態遷移を起動するイベントを関連付けている．次に，この記述を基にして本

```
ENTITY itc_behavior;
        end_state       : itc_state;
        start_state     : itc_state;
        event           : itc_event;
        name            : STRING;
        description     : OPTIONAL STRING;
        action_sequence : OPTIONAL itc_action_sequence;
END_ENTITY;

ENTITY itc_state;
        state_item      : itc_state_item;
        image           : OPTIONAL itc_image;
        parent          : OPTIONAL itc_state;
        entry_action    : OPTIONAL itc_action;
        do_action       : OPTIONAL itc_action;
        exit_action     : OPTIONAL itc_action;
  END_ENTITY;

RULE itc_completness_validation FOR (itc_state);
LOCAL
   behaviours : SET OF itc_behavour :=[];      --空の集合として宣言
   states : SET OF itc_state :=[];             --空の集合として宣言
   start_state_set : SET OF itc_state :=[];    --空の集合として宣言
   end_state_set : SET OF itc_state :=[];      --空の集合として宣言
END_LOCAL;
   behaviours := QUERY( tmp <* itc_behavior | TRUE);   --全ての behaviour のインスタンスを含む集合を生成
   states := QUERY( tmp <* itc_state | TRUE);          --全ての state インスタンスを含む集合を生成
   REPEAT i:= 1 TO SIZEOF(behaviours);
      start_state_set := start_state_set + behaviours[i].start_state ;     --behaviour から全ての sutart_state を集める
      end_state_set := end_state_set + behaviours[i].end_state ;           --behaviour から全ての sutart_state を集める
   END_REPEAT;
WHERE
   wr1 : states == start_state_set ; -- 最終的な制約の判断，この式の評価値が FALSE になると一貫性が満たされて
   wr2 : states == end_state_set ; -- いない
END_RULE;
```

図5 開始状態，終了状態及びトリガとなるイベントを関連付けるbehaviourの宣言と，それを基にした状態遷移の完全性に関する検証RULE

Fig. 5 behaviour description to relate start state, end state and event that trigger state transition, and validation RULE for completeness of state transition

報告中で定義したRULE itc_completness_validationを示す．

状態遷移機械における完全性は，ある状態への遷移と，同じ状態から他への遷移の２つが全ての状態に関して定義されていることについての制約なので，この制約を満たしている状態遷移機械の開始／終了状態の集合は，全てのstateの集合と等しくなくてはならない．

このRULEでは，始めに局所変数としてbehaviourエンティティから開始／終了状態として参照されているstateエンティティの集合を作り出し，WHERE節中で，この集合とstateエンティティを全て集めた集合を比較している．WR１，WR２は検証する項目のラベルであり，もし制約が満たされていない場合には，RULE itc_completness_validation中の項目ラベルで異常が通告されることになる．

４．まとめ

4.1　報告内容

本報告では，以下の項目に関して報告を行った．

１）BehaveViewプラットフォームへの代数的仕様表現に基づく検証アルゴリズムの統合について提案した．

２）統合における課題を検討し，検証事項を制約としてモデル化する方法を提案した．

報告を執筆している時点では，最終的な動作確認までは至れなかったため今後の課題となる．また，今回設けた制約は，複数の状態遷移機械としての一貫性に関するものであり，個別のエンティティについての一貫性については依然として不十分な点があるので，この点も対応する必要がある．

4.2　本研究の発展の方向

現在，本プロジェクト中ではユーザビリティテストに必要な要件情報と，テスト結果に関するデータモデルの定義を進めつつある．この２つのモデルが実現されると，データベース上にテスト対象データ，テスト方法，テスト結果の情報を蓄積することが可能となり，テスト結果からの知識獲得や設計データ変更への指針としていくことが可能となると考えられる．

謝辞

本報告は，文部科学省「知的クラスター創生事業」“札幌ITカロッツェリア”中の次世代工業デザイン手法研究開発プロジェクトの成果であります．ここに関係緒機関及び関係者各位に謝意を表します．

参考文献

１）菊地，金井，岸浪：UI挙動仕様の代数的表現と検証；ヒューマンインターフェースシンポジウム（2004）

2) Draft Federal Information Processing Standards Publication 183, Announcing the Standard for INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF 0) 5, (1993).

3) C. A. R. Hoare, N Wirth, : An axiomatic definition of the programming language Pascal ; Acta Information2, pp. 335−355, (1973)

4) Margren, W. : Formal Specification of Graphic Data Types ; ACM Transactions on Programming Languages and systems, vol. 4, No. 4, pp. 687−710, (1982).

5) ISO10303 "Industrial automation systems and integration − Product data Representation and exchange" Part 1 : Overview and fundamental principles, (1994).

6) ISO10303 "Industrial automation systems and integration − Product data Representation and exchange"−11 ; Description methods : The EXPRESS language reference manual (1994).

7) ISO10303 "Industrial automation systems and integration − Product data Representation and exchange" Part22 : Implementation methods : Standard data access interface (1998).

8) ISO10303 "Industrial automation systems and integration − Product data Representation and exchange" Part21 : Implementation methods : Clear text encoding of the exchange structure (1994).