修 士 論 文

モデルベースによる
体系的テスト駆動開発環境の研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

北 篤

2010 年 3 月

修 士 論 文

モデルベースによる
体系的テスト駆動開発環境の研究

指導教員　Defago Xavier 准教授

審査委員主査　Defago Xavier 准教授
審査委員　青木利晃 准教授
審査委員　岸知二 客員教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0810017 北 篤

提出年月: 2010 年 2 月

概 要

テスト駆動開発は，開発促進のテストを行う特徴や，コード主体の開発法であるという特徴がある．これから，テストケースがアドホックになりがちであるという課題や，要求の変更が生じたときにどのテストケースを変更するか分かりづらいという課題が生じる．その課題に対して，本研究では多角的な視点に基づくテスト設計を考慮するに加え，テスト駆動開発で扱う情報をモデルベースで体系的に整理し，そのモデル上で変更波及解析を行うことができる環境を提案した．

目 次

第1章　はじめに

1.1　背景

近年、ソフトウェアの大規模化、高機能化により品質の確保が焦点となっている．その中で、ソフトウェアを開発するプロセス全体の大部分を占めるテスト工程の重要さが認識されている．しかし、この工程は最後に位置づけられることが多く、上流工程によるしわよせによって十分に品質を確保できないことが起こっている．その解決策の１つとして、上流工程からテスト工程を検討するという考えがあり，その中でも、テストコードをソースコードよりも先に書き，品質を作り込むという特徴をもつテスト駆動開発が注目されている．

1.2　課題

テスト駆動開発はコード主体の開発方法であり，開発者のテストを主体としてソースコードを記述していく手順をとる．この開発者のテストは，実装したコードを自身が意図したとおりに動作するかを確認するためや，テストで早期に得られたフィードバックを開発に反映させるために用いられる．しかし，あくまで開発するためのテストであり，従来のソフトウェアテストの技術を用いないため，テスト設計は重視されていない．よって，テストケースをアドホックに作成してしまいがちである．さらに，コード主体という特徴から，要求が変更された際にどのテストケースを修正すべきかを特定しづらいという課題があり，これらに対して対策が必要である．

1.3　目的

以上述べた課題を解決するためには，テストケース作成についてテスト設計の概念を取り入れること，そして，開発中に扱う情報を体系的に管理し，変更要求に伴う修正作業を支援することが重要だと考えられる．そこで本研究では，多角的な視点に基づくテスト設計を考慮するに加え，テスト駆動開発で扱う情報をモデルベースで管理する開発環境を提案することを目的とする．

1.4　論文の構成

論文の構成について述べる．第 2 章は研究の対象であるテスト駆動開発について詳細に説明し，第 3 章でその課題について述べる．第 4 章では提案する概要について述べる．そして，第 5 章では多角的な視点に基づくテスト設計を支援するテスト観点テンプレートについて説明し，第 6 章はテスト駆動開発で扱う情報を整理するためのモデルを提案する．さらに，第 7 章は提案したモデル上での変更波及解析手法について述べる．そして，第 8 章では提案する環境の実装について述べ，第 9 章では適用例を用いて，実装した環境の動作を確認し，その結果を考察した．最後に，本研究のまとめを 10 章で述べる．

第2章　テスト駆動開発

本章では，本研究で対象とするテスト駆動開発で用いるテスト手法とその有用性について説明する．

2.1　テスト駆動開発とは

テスト駆動開発は，XP(eXtreme Programing) という開発手法で提案されているプラクティスの一つで，本来は要求の変更が起こりやすい web システム開発などに適用されていた．そして，近年では組み込みソフトウェアに適用した例 [1] や通信ソフトウェアに適用する研究 [2] が出てきており，XP と切り離して議論されることが多くなっている．

テスト駆動開発は，テストを実装の中心においたコード主体の開発方法であり，実装の対象であるソースコードを書く前に，テストコードを先に書くという流れで開発を進めていく．この手順はレッド・グリーン・リファクタリングという手順を繰り返して実施される [3]．これは，テスト駆動開発を提唱した K.Beck や E.Gamma によって開発された xUnit と呼ばれる単体テスティングフレームワークの挙動を表している．この xUnit はテストを自動化するために使われ，その挙動は，テストがパスした場合はグリーンバーが現れ，パスしない場合はレッドバーが現れる．これらを踏まえた上でのテスト駆動開発の手順を以下に示す．

- レッド：機能に対するテストコードを書く
  （ソースコードが存在しないためテストはパスしない）
- グリーン：テストコードをパスするまでソースコードを書く
- リファクタリング：ソースコードの内部構造を書きかえる

また，これらの手順を解釈すると以下の図 1.1 として表現できる．

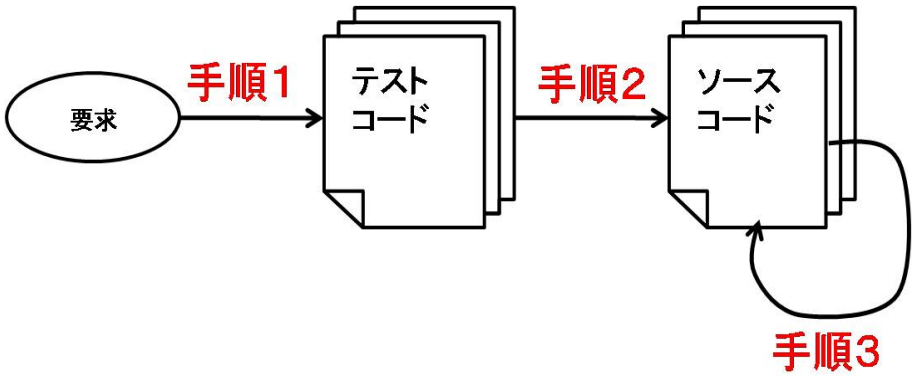

図 1.1:テスト駆動開発の全体像

手順 1： システムに実装したい機能に対するテストコードを書く

手順 2： それをパスするようにソースコードを記述する

手順 3： パスしたソースコードをリファクタリングする

手順 1 は，テスト駆動開発で得られる要求から，どのような機能が必要かを考え，開発対象がどのような振る舞いをしたら機能が実装できるかを考えてテストコードを書く．また，テスト駆動開発で得られる要求は，ユーザーストーリーやユースケースなどで，ユーザが実際にシステムを使うシナリオを定義していることが多い．

手順 2 は，手順 1 で書いたテストコードをパスするまで，ソースコードを記述する．あくまで実行できるコードを優先としてソースコードを記述する．

手順 3 では，リファクタリングを行う．リファクタリングとは，パスしたソースコードの外部的な動作を変えずに内部構造を書き換えていくことを示す．この作業により，動くだけのソースコードでなく，重複などないきれいなソースコードを実装できる．この手順を終えたら，次に実装する要求を決め，手順 1 に戻る．

2.2　テストの立場と手法

本項ではテスト駆動開発で扱われるテストの立場や，そのテスト手法について説明する．テスト駆動開発でのテストの定義やテスト手法に関して，注意をおきたい点は，テストの立場とテストのブレークダウンの 2 点である．

- テストの立場
  テスト駆動開発のテストは開発者の立場として行われる．これは，開発促進を目的として行われるテストである．具体的には，開発者が実装したコードが確実に機能を実装していることを保証するためや，早期に得られるテストのフィードバックを用いて，テストの改善をするために実行される．

- テストのブレークダウン
  これはテスト駆動開発のテスト作成におけるテクニックのひとつであり，文献 [3] で下位のテストとして紹介されている．2.1 項の手順 2 において，テストコードが複数の機能やシナリオを表しているため，複雑であり，それをパスするソースコードを書くことが容易ではない場合がある．このとき，そのテストコードから失敗する部分を抜き出して，適切な大きさのテストコードにする．適切な大きさとは，テストコードまたは，それをパスするソースコードを容易に書ける程度であり，開発者によって異なる．たとえば，電卓を考えると,「１＋１＝を入力したら正しい答え２を表示する」を表すテストコードを書いた場合，それをパスするためのソースコードを一度に記述するよりも，足し算機能がある，入力機能がある，表示機能があるといった３つの機能に分割してソースコードを記述した方が容易であると考える．

2.3 テスト駆動開発の有用性

テスト駆動開発の特徴から，以下のような有用性がある．

- デグレードの早期検出
  これは，開発が進むにつれて蓄積するテストケースを自動で実行するよう xUnit というテスティングフレームワークを用いているためである．これにより，開発中のシステムが常に開発者の意図どおりに動作することを保証させることができる．

- 開発工程を細かく改善できる
  これは開発工程でテスト結果によるフィードバックが多く得られるためである．
  例えば，ソースコードがテストをパスしなかった際に，その結果から原因を特定し，ソースコードを改善することや，テスト作成の際に，新たなテストすべき点に気付くことが挙げられる．

第3章　課題

テスト駆動開発には以上に述べたように様々な特徴があるが，以下のような課題がある．

- テストケースがアドホックになりがちである
  これは，2.2 項で述べたテストの立場から生じる課題である．この開発者のテストは，実装したコードを自身が意図したとおりに動作するかを確認するためや，テストで早期に得られたフィードバックを開発に反映させるために用いられる．しかし，あくまで開発するためのテストであり，従来のソフトウェアテストの技術を用いないため，テスト設計は重視されていない．よって，テストケースをアドホックに作成してしまいがちである．テストケースがアドホックに作成されてしまうと，テストケースの情報が整理されない場合が多くなり，再度テストケースを確認した際に，何をテストしているのかを把握しづらい．また，気まぐれさがテストの質を左右してしまうことや，偏向的な目的のテストケース作成をしてしまう可能性がある．これらに対して，テスト設計の概念を取り入れ，テストケースの管理をする必要がある．

- 要求の変更が生じた際に、どのテストケースを変更するか分かりづらい
  これはテスト駆動開発のコード主体という特徴が原因である．コード主体で開発を進めると、成果物として残るのはソースコードとテストコードである．この場合，各テストコードがどの要求のために書かれているのかを理解することが困難だと考えられる．これに対しては，コードで扱われてきたテスト駆動開発の情報を体系的かつ見通し良く整理し，要求とテストコードの割り当てを明示する必要がある．さらに，それらの整理された情報を用いて要求変更の修正作業支援をする必要がある．

第4章　提案

本章では，提案する開発環境に求められる機能やそれを実現するためのアプローチの概要を説明する．

4.1　アプローチ

前章で述べたように，アドホックなテストケースになりがちという課題を解決するためには，テスト設計の概念を取り入れ，テストケースを管理する必要がある．また，変更された要求に対応するテストコードが特定しづらいという課題を解決するためには，テスト駆動開発で扱う情報を体系的に管理し，各情報に関連する情報を追跡できる必要がある．これらを踏まえ本研究では，

機能1： 多角的な視点に基づくテスト設計ができる

機能2： モデルベースで体系的に情報を管理した上で，変更波及解析を行うことができる

といった機能をモデル変換技術によって実現した環境を提案する．

さらに，これらの機能を実現するために複数のアプローチをとった．まず，機能1を満たすために，テスト観点という概念を用いたテンプレートの作成を行う．そして，機能2を満たすために，テスト駆動開発で扱う情報を体系的に管理できるようモデルを提案する．ここでテスト駆動開発の要求とテストケースは要求モデルに，テストケースの詳細はテスト詳細モデルに分ける．そして，そのモデル上で各モデル要素のトレーサビリティを確保し，変更波及解析を行う手法の検討をする．最後に，以上に挙げたアプローチをモデル変換技術によって1つの開発環境として実現する．これらのアプローチは，以下のように整理できる．

機能1に対するアプローチ： テスト観点テンプレートの作成

機能2に対するアプローチ： モデル整備（要求モデル，テスト詳細モデル）
変更波及解析の支援

全体としてのアプローチ： モデル変換技術による提案環境の実現

4.2　全体像

以上に述べた機能 1 や機能 2 と，そのアプローチの対応をテスト駆動開発の全体像に位置付けると図 4.1 のようになる．

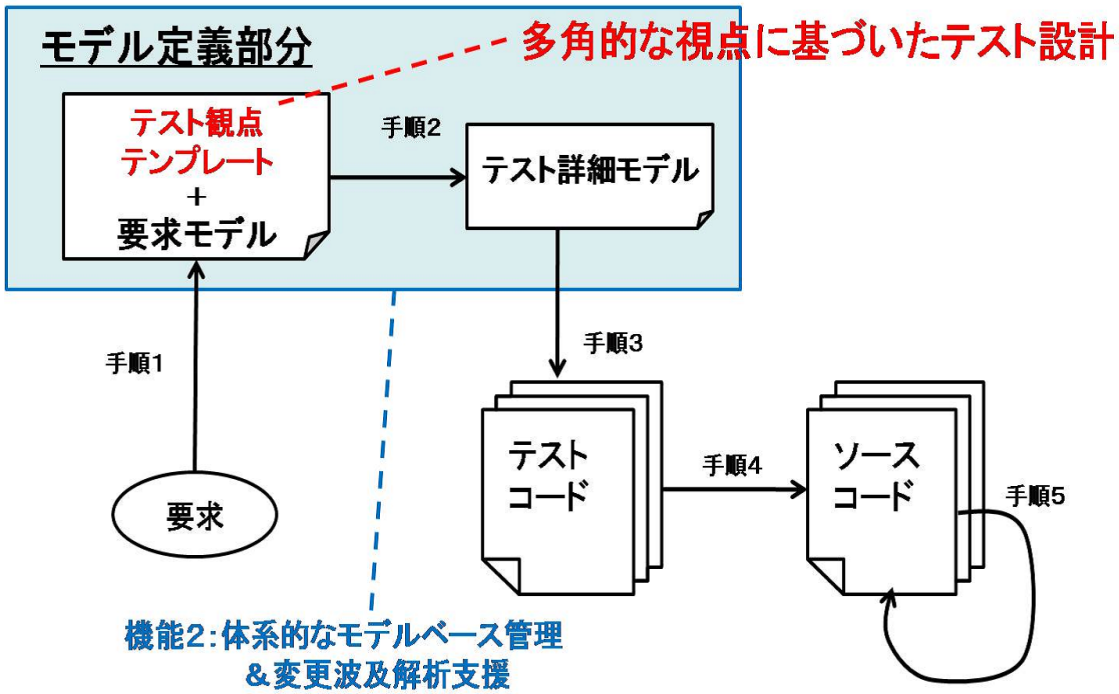

図 4.1:提案する環境の全体像とそのアプローチの対応

この図 4.1 での要求，テストコードそしてソースコードについては，2.1 項で示した全体像と同じものを指す．従来のテスト駆動開発との大きな違いは水色部分で囲ったモデル定義部分である．このモデル定義部分ではテスト駆動開発で扱う情報をモデル化し整理する手順をとる．そして，その各開発手順は以下のようにして行われる．

手順 1： 要求とテストの整理，およびそれらの関連付け

要求やそれに対するテストを定義し，それらの関連を要求モデルで表現する．要求を定義する際は，あらかじめ提供されているテスト観点テンプレートの情報を参考にして，どの観点のテストが必要かを選択する．

手順 2： 要求モデル中のテストの詳細化

手順 1 で要求に割り当てたテストの具体的な振る舞いや構造をモデル化する．ここで，どのクラス，モジュールがテスト対象であるか，またそのテスト対象と協調するクラス，モジュールはどれかを考え，モデルに記述する．

手順 3： テスト詳細モデルからのテストコード半自動生成

テスト詳細モデルで定義したテストのモデルより，空のテストコードを自動生成させる．テストコードの内容は，手動でユーザが記述する．

手順4： テストコードをパスするようなソースコード記述

手順3で記述したテストコードをパスするように，ソースコードを記述する．

手順5： テストコードをパスするまで，ソースコードを記述，パスしたら手順1に戻る

4.3 環境の実現

以上に説明した3つのアプローチを1つのモデルベース環境として実現するためにモデル変換技術を用いた．モデル変換技術とは，あるモデル要素から別のモデル要素を生成する働きをもつ．各々のアプローチに対してのモデル変換技術の使われ方を示す．

まず，テスト観点テンプレートの作成に対しては，あらかじめ定義しておいたテンプレートの情報をユーザが記述する要求モデルやテスト詳細モデルに取り込むために用いた．これにより，ユーザはテスト観点テンプレートをシームレスに扱うことができる．

そして，要求モデルやテスト詳細モデルに対しては，記述支援のために用いた．これは，要求モデルやテスト詳細モデル間で同じ意味や，派生した意味をもつモデル要素は，自動で生成させることを示している．これにより，モデル要素を定義するうえでの人為的ミスを減らすことができる．

最後に，変更波及解析においては，要求モデルやテスト詳細モデルに定義された各モデル要素のトレーサビリティを確保するために用いた．具体的には，波及解析に必要な各要素の対応の情報を要求モデルとテスト詳細モデルから自動で生成させた．

第5章　テスト観点テンプレート

5.1　概要

テスト駆動開発のテストにテスト設計の概念を取り入れることにおいて，本研究ではテスト観点テンプレートを提案する．これを用いることによって，開発対象システムのテストすべき側面が見通し良く整理できるに加え，多角的な視点に基づくテストケースの作成ができると考える．

5.2　テスト観点

テスト観点とは，西が提案する概念であり，テスト対象の持つテストすべき側面，テスト対象が達成すべき性質と定義されている [4]．このテスト観点のモデリングを行うことで，テストにおける様々な観点を見通し良く整理でき，テスト漏れによる欠陥を少なくすることができる．よって，テスト観点はテスト設計をするうえで有用な概念である．また，テスト観点の例をあげると，機能の観点，扱うデータの観点，動作環境の観点などが挙げられる．

5.3　テンプレート

テスト観点テンプレートとは，一般的なシステム開発において，テストすべき観点やその観点でテストすべき情報をまとめたものである．これを用いることにより，多角的な視点に基づくテスト設計を支援できると考える．このテンプレートはユーザによるカスタマイズを許し，テストから得られるフィードバックからテンプレート自身を改善していくことができるテスト駆動開発と相性が良い．また，このテンプレートからユーザが手動で記述をしていき，テスト観点図を作成することを想定している．

本研究では開発対象のソフトウェアの種類を限定せず，さらにテスト駆動開発に適するようにテンプレートを作成した．なお，テスト観点は文献 [5] から抽出してまとめた．さらに，テスト設計をより支援するために，各テスト観点において起こりやすい不具合をまとめた．テンプレート作成における流れは文献 [6] を参考にして，図 5.1 のように行った．

図 5.1:テンプレート作成の流れ

手順 1

一般的なソフトウェアに起こる不具合集から，テスト駆動開発で行うテストによって取り除けると判断できる不具合を抽出する．テスト駆動開発では GUI や並列処理，データベースにかかわる処理には適用できないため，それらにかかわる不具合は対象とせずに抽出する．

手順 2

抽出した不具合からどんなときに不具合が起こったのかなどのメカニズムを分析し，その不具合を予防するためのテストケースを促すキーワードを考える．また，類似したメカニズムを持つ不具合を分類して抽象化し，テスト観点を抽出する．

5.4 全体像

テスト観点テンプレートの全体像を以下に示す．また詳細なモデル化は次章で述べる．

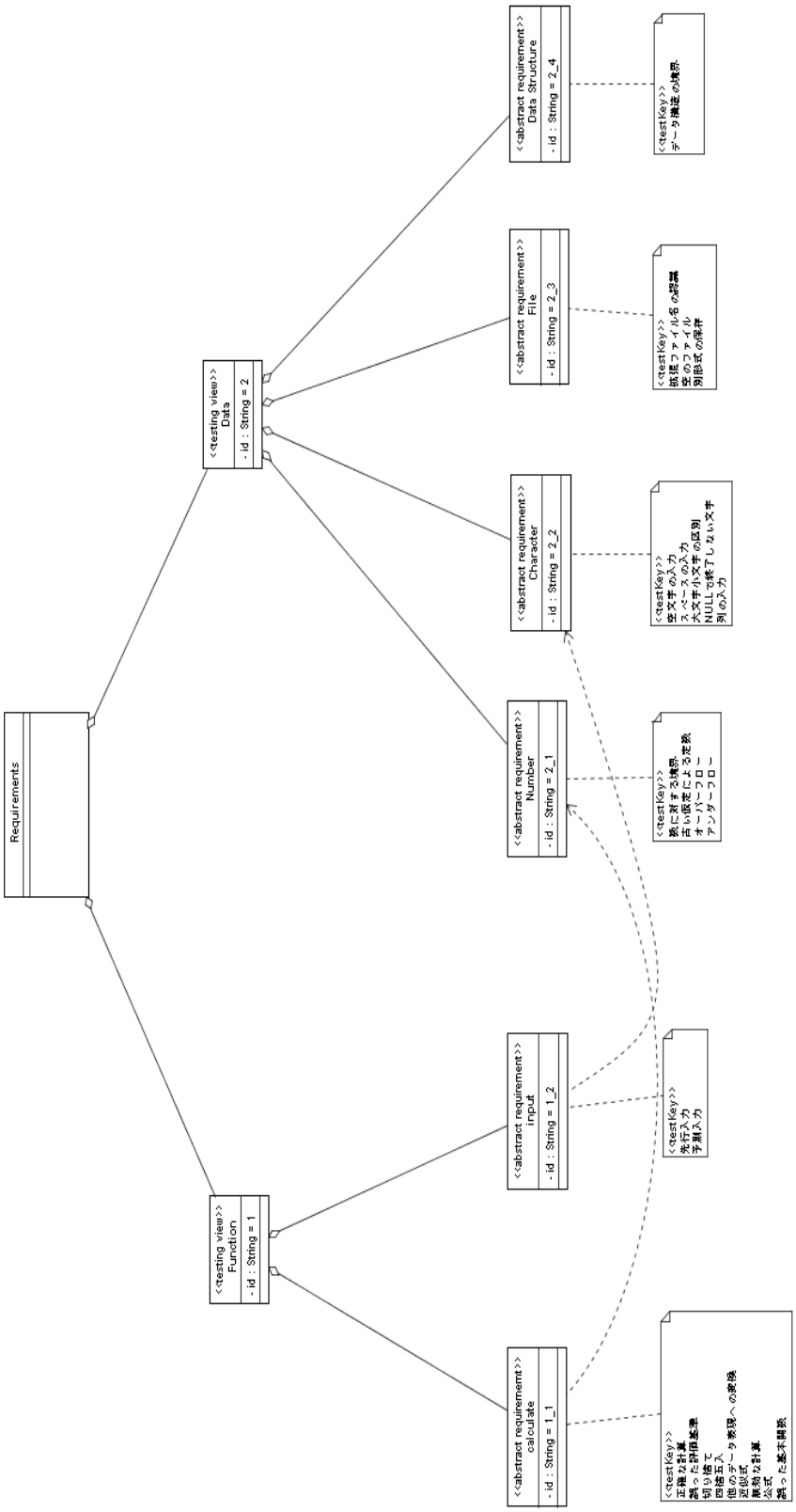

図 5.2:テスト観点テンプレートの機能，データ部分

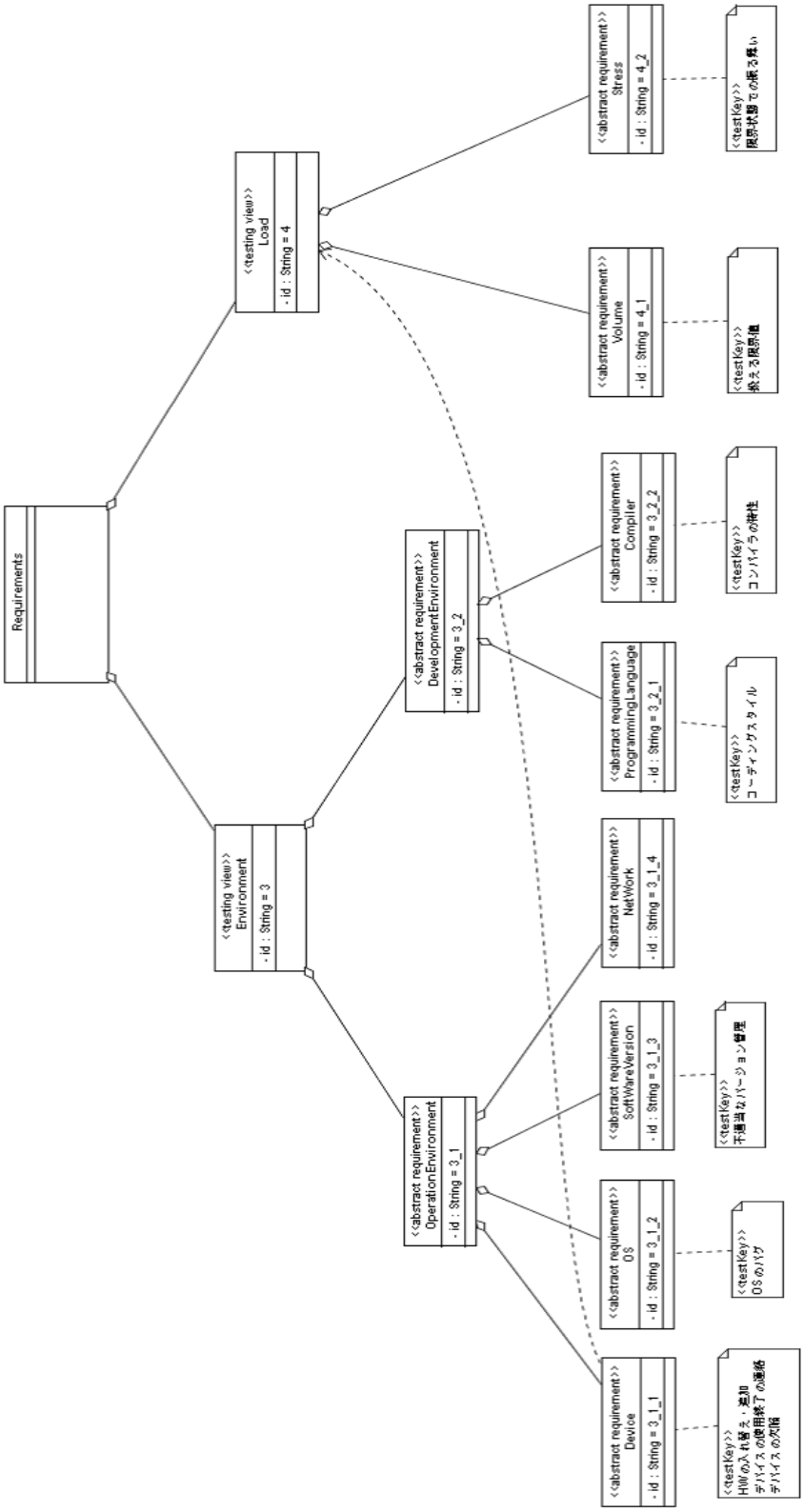

図 5.3:テスト観点テンプレートの環境，負荷部分

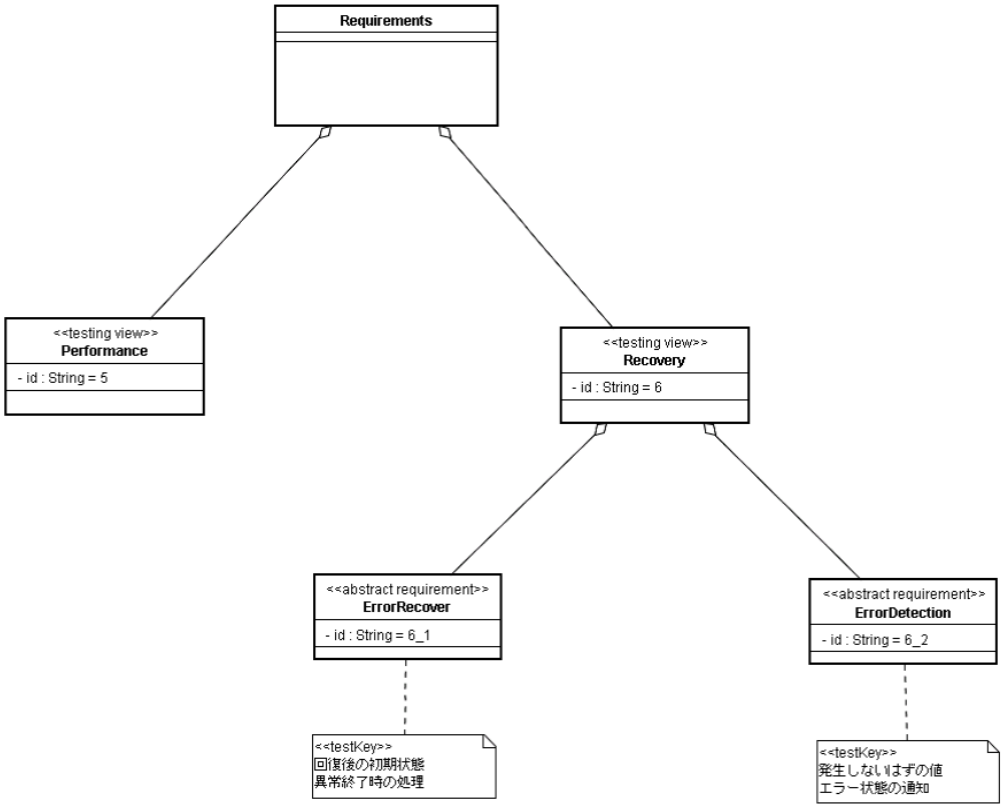

図 5.4:テスト観点テンプレートの性能，回復部分

第6章　モデルの提案

本章では，テスト駆動開発で扱う情報を体系的に管理するために．モデル化する情報の整理，要求モデルやテスト詳細モデルで定義した図表の説明をし，その記述例を示す．

6.1　概要

テスト駆動開発において，まず要求を元に実装する機能を決定し，テストコードを書き，それからソースコードを記述していく．この手順より，テスト駆動で扱う要求，テストケース，テストケースの詳細，そしてそれらの関連を見通し良く整理することが重要となってくる．そこで本研究では，要求モデル，テスト詳細モデルを定義し，テスト駆動開発の情報を整理する方法をとる．この２つのモデルは，UML のプロファイルである SysML[7] や UTP(UML Testing Profile)[8] を参考にして新たに定義したメタモデルに従い，さらに複数の図表から成る．以下に各モデルが含む図表を示す．

- 要求モデル
  RTM(Requirement and Testing Management) 図
- テスト詳細モデル
  テストコンテキスト図
  テスト構造図
  テスト振る舞い図
  ユニットテストテーブル
- テスト観点テンプレート
- テスト観点図

RTM 図は SysML の要求図の記法を参考にした図であり，テスト駆動開発で扱う要求やテスト，そしてその関係を見通し良く整理できる図である．

テストコンテキスト図は，RTM 図で定義した要求に対して，どのようなテストケースがあるかをまとめる図である．

テスト構造図はテストケースを実施するときの詳細な構造を示す図である．そして，テスト振る舞い図はテストケース実施時に各クラス，モジュールがどうやりとりするかを示

す図である．この２つの図に関しては，どういったモデル要素がどう表現されるかまでは触れないが，今後の拡張を見込んで定義した．

最後にユニットテストテーブルは，テストケースからブレークダウンしたテストケースの情報をまとめる表である．テスト観点テンプレートは第５章で説明したとおりである．テスト観点図はテスト観点テンプレートを元にユーザが手を加えた図である．
以上これらの図表を用いてテスト駆動開発全体で扱う情報をモデル化する．

6.2　全体像

以下にモデルの提案における全体像を示す．また，これは図 4.1 のモデル定義部分をフォーカスした全体像でもある．

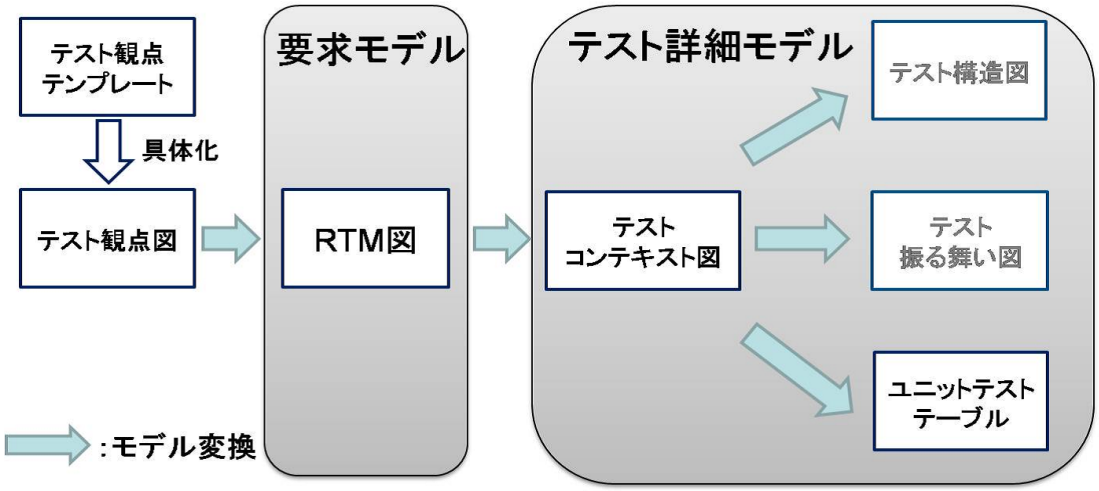

図 6.1:モデルの提案における全体像

全体像のテスト観点テンプレートから矢印をたどるように，各図を定義していく．水色の矢印はモデル変換を示している．これは各モデル要素の記述支援のために行われるが，モデル要素を機械的に定義する部分だけ支援することとする．つまり，各図で同じ意味をもつモデル要素や派生して作られるモデル要素は自動生成するという形をとる．そのため，各図のモデル要素を自動生成した後，足りない情報はユーザーが手動で加える手順をとる．詳細なモデル要素の変換は後で説明する．

また，テスト観点テンプレートからテスト観点図への具体化の矢印については，テスト観点テンプレートを元にユーザが要素を加えるなどのカスタマイズをする形でテスト観点図を作成するということを示している．

6.3　テスト観点テンプレート

テスト観点テンプレートとは，第５章で説明したとおり，一般的なシステム開発において，テストすべき観点やその観点でテストすべき情報をまとめたものである．また，これ

を元にしてユーザがテスト観点図を作成することを想定している．ここでは，テスト観点テンプレートのモデル化のため，メタモデルの定義について説明する．

6.3.1 メタモデル

テスト観点テンプレートを構成するメタモデルを以下のように定義した．

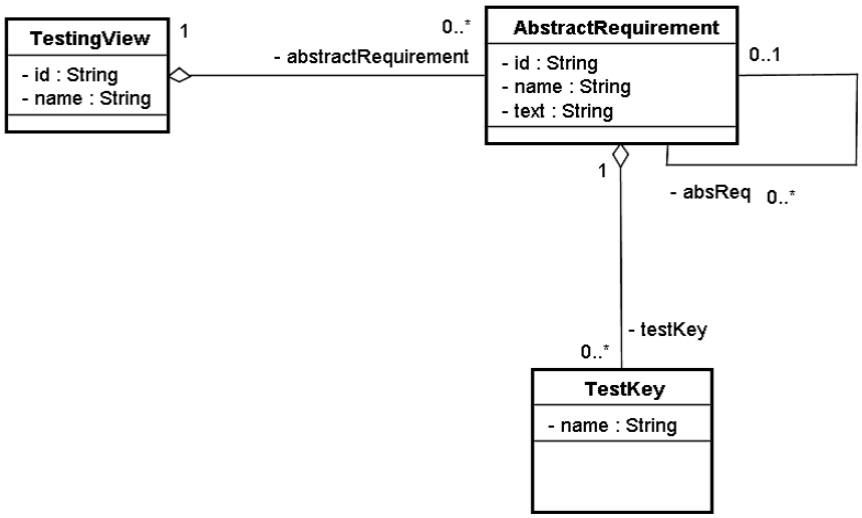

図 6.2:テスト観点テンプレートのメタモデル

テスト観点テンプレートで扱う各メタクラスについて説明する．

- TestingView
  テスト観点であり，開発対象をテストするうえで考慮すべき観点を示す．例としては，機能の観点，データの観点，動作環境の観点などが挙げられる．

- AbstractRequirement
  TestingView で分類され，一般的なソフトウェアでの要求を抽象的に表現した要素である．たとえば機能の観点では，計算機能や記憶機能，そして認証機能などである．また，AbstractRequirement どうしで関連を引くことができる．これはテスト観点を組み合わせてテストケースを作る場合があると考えたためである．たとえば，AbstractRequirement の計算は，数を扱うため，別の AbstractRequirement の数に関連を引くといったことが挙げられる．これによって，計算においてのテストすべき観点と，数においてのテストすべき観点を組み合わせたテストケースが作成できる．

- TestKey
  AbstractRequirement に分類され，開発対象が起こしやすい不具合を示唆する情報である．たとえば，AbstractRequirement の計算機能を考えた場合，TestKey とし

て「正確な計算」,「切り捨て」が考えられる．これは，正確な計算ができるか，切り捨てを考慮しての計算はどうか，などのテストケース作成を促す情報となる．

6.4 RTM図

RTM図とはテスト駆動開発で扱う要求やテスト，そしてその関係を示す図である． 観点ごとにRTM図を定義していく．

6.4.1 記述例

以下にRTM図の記述例を示す.

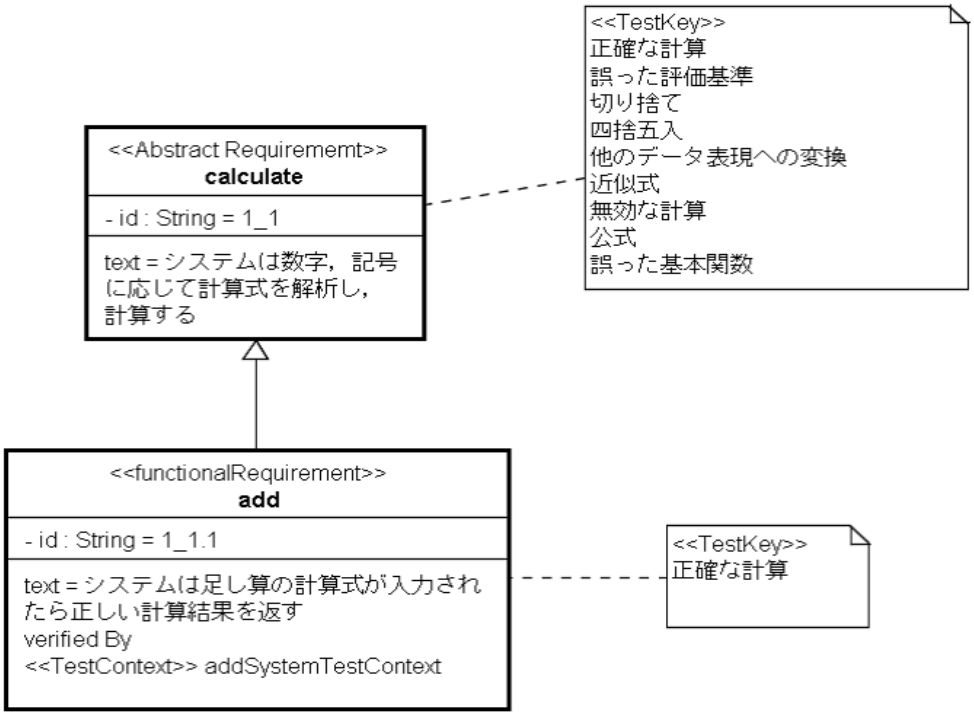

図6.3:RTM図の記述例（機能の観点）

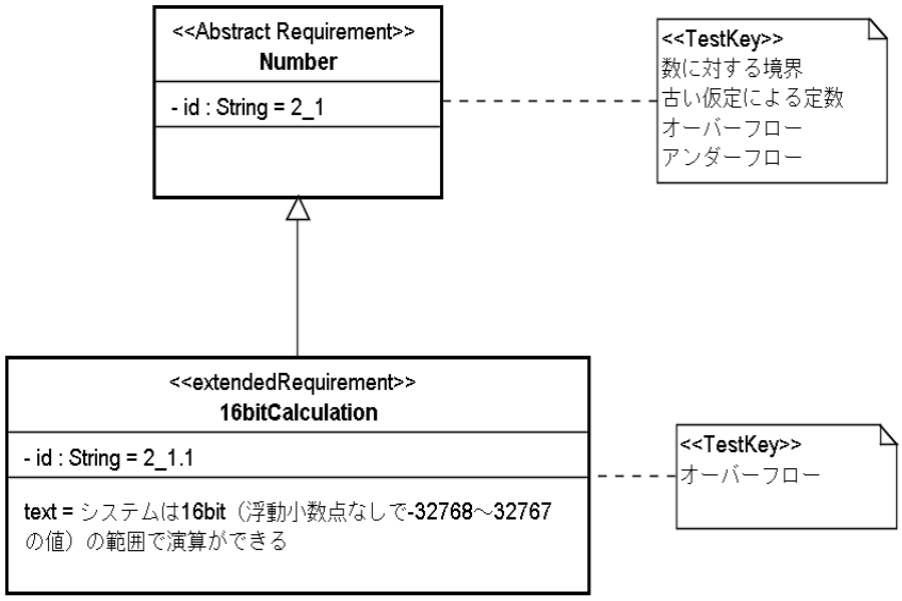

図 6.4:RTM 図の記述例（データの観点）

まず，機能観点の RTM 図は自動で生成される Abstract Requirement の calculate を具体化して，機能要求を示す functional Requirement の add を定義している．この add は属性 id と text からなる．id は図のとおり，親の Abstract Requirement の id を派生させて決める．そして AbstractRequirement がもつ TestKey から定義する要求に起こりそうな不具合を選択する．さらに，定義した functional Requirement にどういった TestContext で検証されるのかという情報を verified by 下に記述している．この例では add は addSystemTestContext に検証されるということを定義している．

データ観点では，機能観点の RTM 図と記述は同様だが，functional Requirement を定義するのではなく，extended Requirement を定義する．ここでは，functional Requirement と違い，どんな TestContext で検証されるという情報を記述しない．この場合，16bitCalculation の親である Number とテスト観点図で関連を持つ AbstractRequirement のモデル要素にて検証される．例えば図 6.3 と図 6.4 の AbstractRequirement が関連を持つ場合，extended Requirement の 16bitCalculation は，addSystemTestContext で検証されることになる．

6.4.2 メタモデル

以下の図 6.5 に RTM 図のメタモデルを示す．

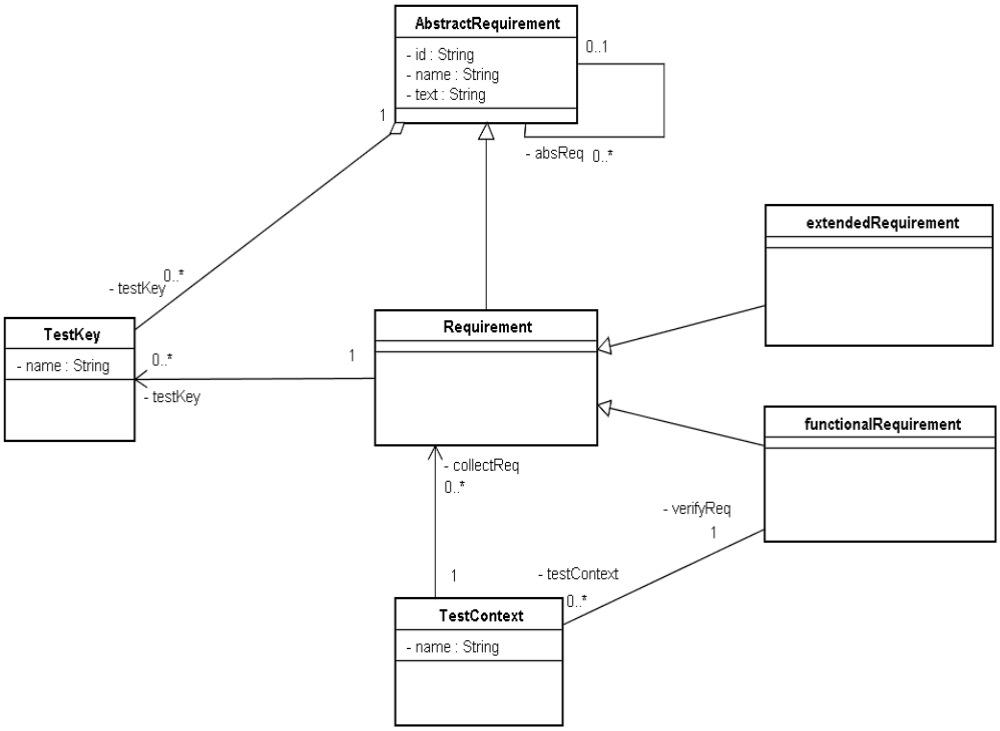

図 6.5:RTM 図のメタモデル

RTM 図で用いる各メタクラスについて説明する．

- AbstractRequirement
  テスト観点テンプレートでの AbstractRequirement と同様で，複数の TestKey をもつ．

- TestKey
  テスト観点テンプレートでの TestKey と同様である．

- Requirement
  テスト駆動開発で扱われる要求であり，適した AbstractRequirement を具体化して定義する．属性 text は，ユーザがソフトウェアを利用するシナリオである．

- functionalRequirement
  機能の観点に分類される要求である．この要素は後述する TestContext によって検証される．

- extendedRequirement
  機能以外の観点に分類される要求である．この要素は functionalRequirement とは違い TestContext とは関連を持たない．これは functionalRequirement と関連をもつ TestContext でまとめて検証されるからである．

- TestContext
  TestCase の集合である．

6.5 テストコンテキスト図

テストコンテキスト図は RTM 図で要求に割り当てられた TestContext を詳細化する図である．RTM 図で定義した TestContext ごとに対してテストコンテキスト図を定義していく．また，UTP(UML Testing Profile) の表記法を参考にした．

6.5.1 記述例

以下の図 6.6 に記述例を示す．

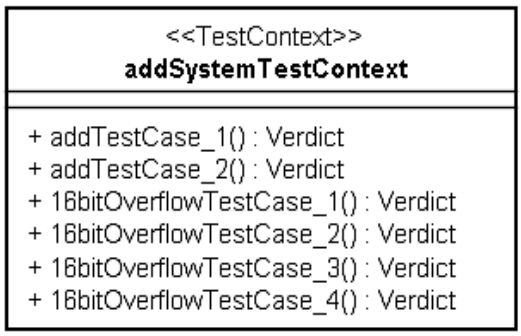

図 6.6:テストコンテキスト図の記述例

この図 6.6 では，TestContext として addSystemTestContext を定義し，そのテストケースとして addTestCase_1()，addTestCase_2() などを定義している．また，このテストケースは，図上には表れていないが，id やどの要求を検証しているのかの対応関係を定義している．また，テストコンテキスト図は，見かけ上 TestContext の要素しか現れないが，UTP では，他のモデル要素を登場させてテストを詳細に表現しているため，今後の拡張を見込んで，テストコンテキスト図でを定義した．

6.5.2 メタモデル

図 6.7 にテストコンテキスト図のメタモデルを示す．

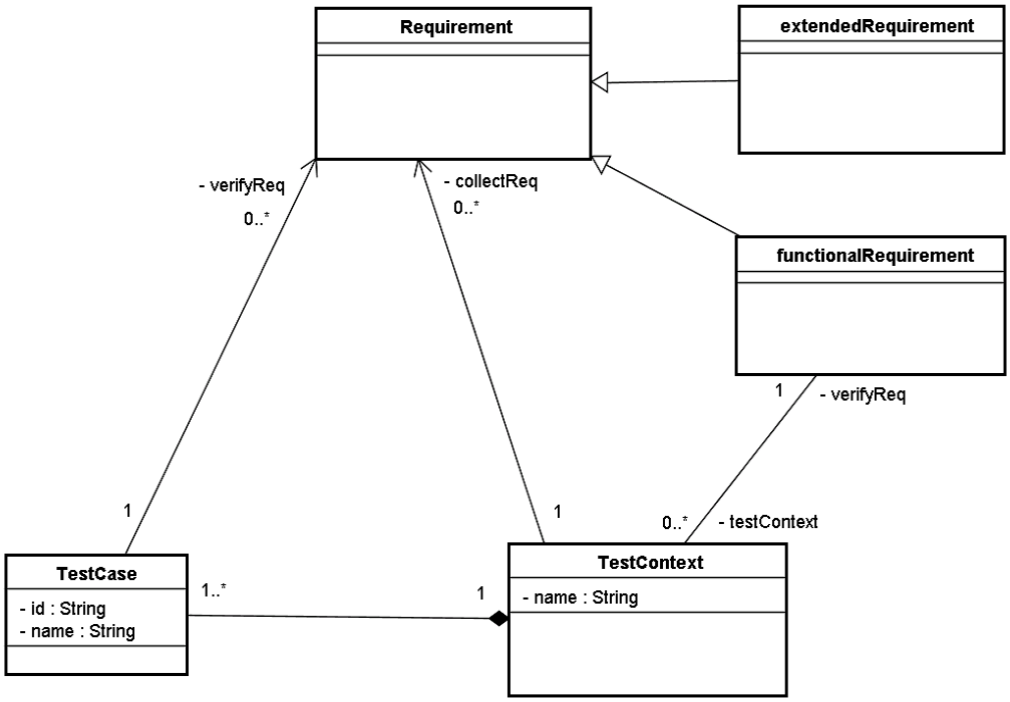

図 6.7:テストコンテキスト図のメタモデル

テストコンテキスト図で用いる各メタクラスについて説明する．

- Requirement
  RTM 図で用いられる Requirement と同様である．TestContext や TestCase によって参照される．

- functionalRequirement
  RTM 図で用いられる機能の観点に分類される要求である．

- extendedRequirement
  RTM 図で用いられる機能以外の観点に分類される要求である．

- TestContext
  TestCase の集合である．テストコンテキスト図で各 functionalRequirement と extendedRequirement に対して TestCase を定義する．また，参照 collectReq は，TestContext がどの要求に対して TestCase を作らなければならないかを示す．

- TestCase
  要求を満たすためのテストケースを示す．各テストケースに id や検証する要求を指す verifyReq という参照がある．

6.6　ユニットテストテーブル

ユニットテストテーブルとはブレークダウンしたテストケースの情報を記述する表である．ブレークダウンしたテストケースは，テストコンテキスト図で定義したテストケースとは粒度が異なる．このテストは頻繁にかつ多く作成されるため，モデル化せずに表形式とした．

6.6.1　記述例

表 6.1：ユニットテストテーブルの記述例

id	テスト名	対象クラス	対象メソッド	入力	期待される出力
1.1.1_1_1	testCheckCalculate_1	Calculate	calculate	文字列型リスト[1, +, 3, =]	int型4
1.1.1_1_2	testAdd_1	Calculate	add	値1,3	int型4

このユニットテストテーブルでは，2 つのテストケースを定義している．testCheckCalculate_1 に関して，対象クラスは Calculate，対象メソッドは calculate と記述されている．これは，testCheckCalculate_1 は開発対象物のクラス Calculate のメソッド calculate をテストしていることを示している．そして，入力，期待される出力には，文字列型リスト [1, + , 3, =]，int 型 4 と記述されている．これは，testCheckCalculate_1 が入力として文字列型のリストを用いて，int 型の 4 がテストの結果として期待されていることを示している．

6.6.2　メタモデル

図 6.8 にユニットテストテーブルのメタモデルを示す．

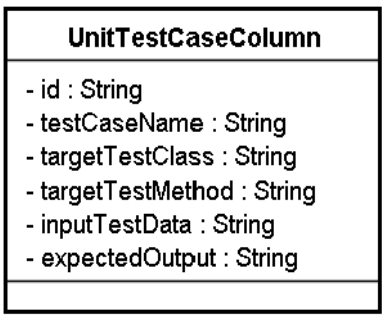

図 6.8:ユニットテストテーブルのメタモデル

ユニットテストテーブルで用いる各メタクラスについて説明する．

- UnitTestCaseColumn
  ユニットテストテーブルでの行である．id はテストケースの通し番号を示す．testCaseName はテスト名を示す．targetTestClass は開発対象のクラス名，targetTestMethod は開発対象のメソッド名を表す．そして，inputTestData はテスト対象に入力するデータを，expectedOutput は入力に対しての期待される出力を示す．

6.7 図表間のモデル変換

各モデル変換の詳細は以下のとおりである．

- テスト観点テンプレートから RTM 図のモデル変換
  TestingView の属性 name が付いた空の RTM 図を生成
  AbstractRequirement の全属性からそのまま AbstractRequirement を生成
  Testkey の全属性からそのまま TestKey を生成

- RTM 図からテストコンテキスト図のモデル変換
  Requirement の属性 id を派生した id が付いた TestCase を生成
  ex) Requirement の属性 id が 1 _1.1 とすると，属性 id が 1 _1.1 _が付与された TestCase が生成される．
  TestContext の属性 name が付いた空のテストコンテキスト図を生成
  TestContext の全属性からそのまま TestContext を生成

- テストコンテキスト図からテスト構造図，テスト振る舞い図，ユニットテストテーブルのモデル変換
  TestCase の属性 name が付いた空のテスト構造図を生成
  TestCase の属性 name が付いた空のテスト振る舞い図を生成
  TestCase の属性 name が付いた空のユニットテストテーブルを生成

第7章　変更波及解析

この章では，変更波及解析を実現するためのモデル要素のトレーサビリティ確保の手法について説明する．また，変更波及解析において考慮すべき変更要求のパターンの類別や，変更波及する情報に対する修正順序についても説明する．

7.1　概要

ソフトウェアの開発において，変更要求はしばしば起こる．そして，テスト駆動開発も例外ではない．さらに，テスト駆動開発はコード主体の開発法なため，変更要求が起こるとどの情報を修正するかがわかりづらいため，その修正作業に労力や時間がかかってしまう．そこで本研究では，この修正作業を支援するためにテスト駆動開発で扱う情報のトレーサビリティを確保し，自動で変更箇所を特定できる仕組みを提案する．

まず，テスト駆動開発で扱う情報のトレーサビリティを確保するために，我妻らが提案するトレーサビリティメタモデル [9] を用いた．そのトレーサビリティメタモデルに従うモデルをトレーサビリティと定義する．そして，トレーサビリティモデルと要求モデル，テスト詳細モデルの３つのモデルを利用して，変更波及解析できる方式を検討した．

また要求の変更は様々な種類があるため，それを分類し整理した．また，変更波及を特定した情報に関しては修正順序を付与させ，さらなる修正作業の支援を行った．

7.2　トレーサビリティモデル

まず，トレーサビリティメタモデルを図 7.1，それに従うモデルは以下の図 7.2 のようになる

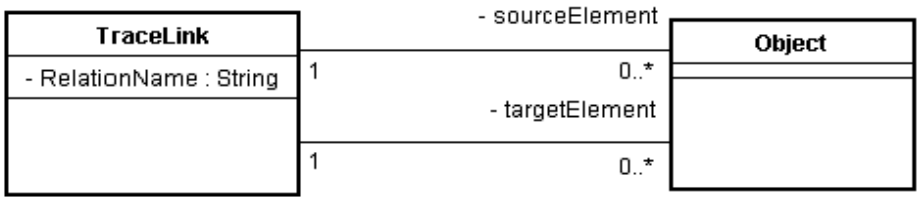

図 7.1:トレーサビリティメタモデル

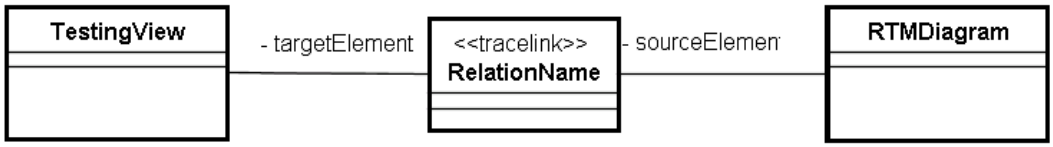

図 7.2:トレーサビリティモデルの例

図 7.2 のトレーサビリティモデルはモデル要素の TestingView と RTMDiagram のトレーサビリティの情報を示している．つまり，異なる図の間における関連を示す．

また，トレーサビリティモデルには，関連の種類を示す RelationName という属性が割り当てられている．RelationName とはモデル要素間の関連名であり，以下のような種別がある．またここで，ソースを生成元，ターゲットを生成先とする．

- ［AnyAttribute］Identity：ターゲットの情報とソースの情報が同一となる関連
  例）Name Identity, Id Identity など
- ［AnyAttribute］Derived：ターゲットの情報から派生してソースの情報とする関連
  例）Name Derived, Id Derived など

たとえば，TestingView と RTMDiagram のトレーサビリティモデルの RelationName が NameIdentity の場合，TestingView の name 属性と RTMDiagram の name 属性は同一であることを示す．

7.3 トレーサビリティモデルの生成法

トレーサビリティモデルは，各図のモデル変換によって変換されるときに，同時に生成される．たとえば，RTM 図の TestContext からテストコンテキスト図の TestContext にモデル変換したとき，RTM 図の TestContext とテストコンテキスト図の TestContext のトレース情報を示すトレーサビリティモデルが同時に生成される．以下の図 7.3 は，そのイメージである．

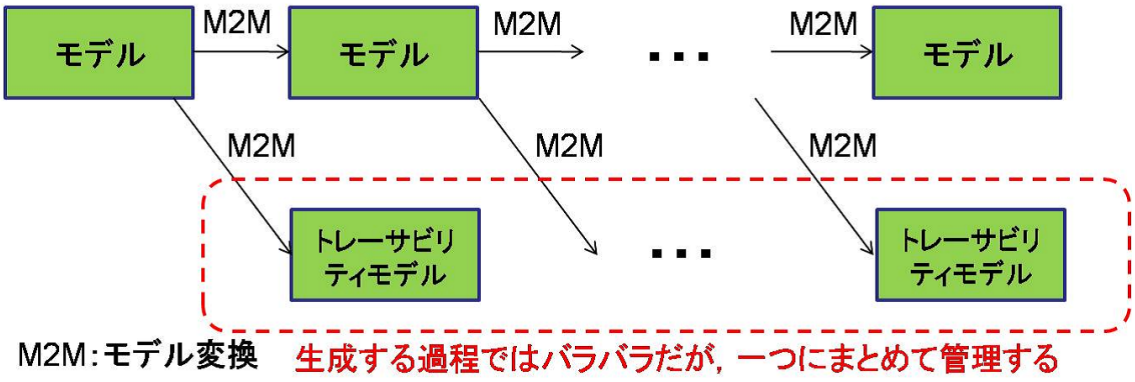

図 7.3:トレーサビリティモデルの生成

7.4 変更要求の分類

提案するモデルでは，要求がどう変わるかによって波及解析の仕方が変わってくるため，どう要求が変更されるかを分類する必要がある．そこで，本項では提案する環境での変更要求をいくつかの前提を置いて分類した．

7.4.1 前提

変更要求について分類する前に，以下のような前提を置いた．

前提 1： TestKey は修正作業中に変更されない

前提 2： 変更前に関連を持っていたモデル要素には修正作業をしない

まず，前提 1 について説明をする．TestKey はテスト観点テンプレートで提供されているモデル要素で，各観点の起こりやすい不具合を示す．この TestKey は，RTM 図で Requirement を定義する際に同時に定義するモデル要素である．しかし，あらかじめテスト観点テンプレートで提供される TestKey に適する TestKey がない場合，ユーザが新たに追加する作業が考えられる．本研究では，この場合は考えないこととする．

次に前提 2 について説明する．たとえば RTM 図中では AbstractRequirement を具体化する形で Requirement が定義されるが，Requirement の内容が変更された際に，AbstractRequirement の関連を変更する修正作業が考えられる．この場合，変更前の Requirement に関連していた AbstractRequirement は何らかの修正作業がなされるが，考えないものとする．

7.4.2 分類

本研究では，変更要求を以下のように分類した．また，要求を表すモデル要素である Requirement には，functionalRequirement と extendedRequirement の 2 種類ある．その種類ごとにも変更の仕方が変わってくる点にも注意して分類をした．

- 要求の追加パターン
  要求が追加された場合を示す．電卓の例で考えると，functionalRequirement の追加では，統計計算の機能を追加する場合などが考えられる．また，extendedRequirement では，新たに負の値を扱えるようにする場合が考えられる．これらの要求が追加された場合，追加した要求を検証する TestContext の特定をする必要がある．

- 要求の削除パターン
  任意の要求が削除された場合を示す．削除された要求を検証していた TestContext や TestCase などのモデル要素の変更または削除といった修正作業が考えられるため，それらのモデル要素の特定が必要である．

- 要求の内容変更パターン
  要求の内容が変更された場合を示す．この内容の変更にはひとつ注意すべき点がある．それは，変更する要求を分類している AbstractRequirement の変更である．つまり，Requirement は AbstractRequirement を具体化する形で定義するが，その具体化する元となる AbstractRequirement を変更することを示す．例えば以下の図のようなパターンが考えられる．なお，以下の図は説明のため，複数の図にまたがるモデル要素を一つの図として書いている．

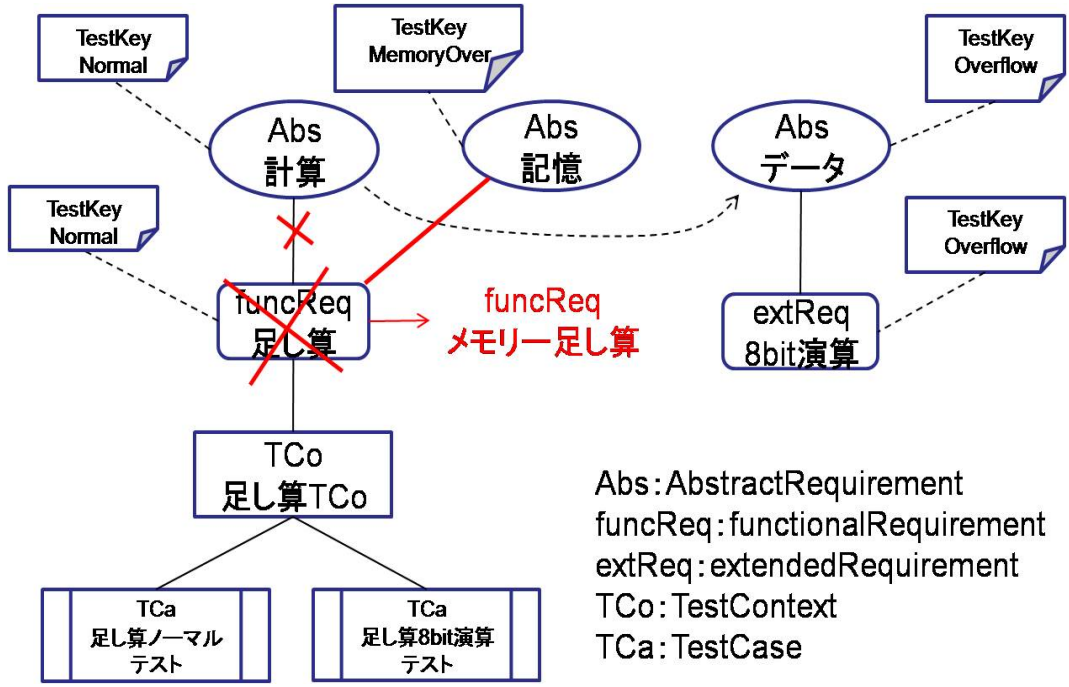

図 7.4:AbstractRequirement の変更

この図 7.4 は，足し算という functionalRequirement がメモリー足し算に内容が変更され，AbstractRequirement の関連が変更された場合を示している．これは，足し算は AbstractRequirement の計算を具体化して定義されていた，つまり計算に分類される形で定義されていたが，内容を変更した後では，AbstractRequirement である記憶の方が適していると判断され，AbstractRequirement への関連を変更するという修正作業がなされた．この変更の重要な点として，変更前後の AbstractRequirement がもつ TestKey の変化にある．また，変更前後で TestKey に変化がなければ，変更された要求を参照していたモデル要素には波及はしないが，図 7.4 での TestKey の変化をみると，Normal から MemoryOver に変わり，Normal が失われているため，それを参照しているモデル要素は波及する可能性がある．また，これは extendedRequirement の内容が変更された場合も同様のことが言える．

7.4.3　修正順序

変更要求に伴い，修正する可能性があるモデル要素を取り出せたとしても，どのモデル要素から見るべきなのかは分からない．そこでそれらのモデル要素の修正順序を考え，修正作業のさらなる支援を行う．なお，修正順序はモデル要素の開発手順に従ってトップダウンに付与されていく．たとえば，波及するモデル要素に TestContext と TestCase が存在した場合，TestContext のほうが先に開発されるため，TestContext に TestCase よりも早い修正順序が付与される．

以下の図 7.5 が要求の内容変更が生じたときに考えられる修正順序付与の一例である．この図 7.5 は，要求 8bit 演算が 16bit 演算に変更されたときの特定したモデル要素，そして修正順序を示している．モデル要素の傍にある数字が修正順序となっている．この図 7.5 は説明のため，本来複数の図表にまたがるモデル要素をまとめて書いている点や 1 部のモデル要素しか示していない点に注意してほしい．

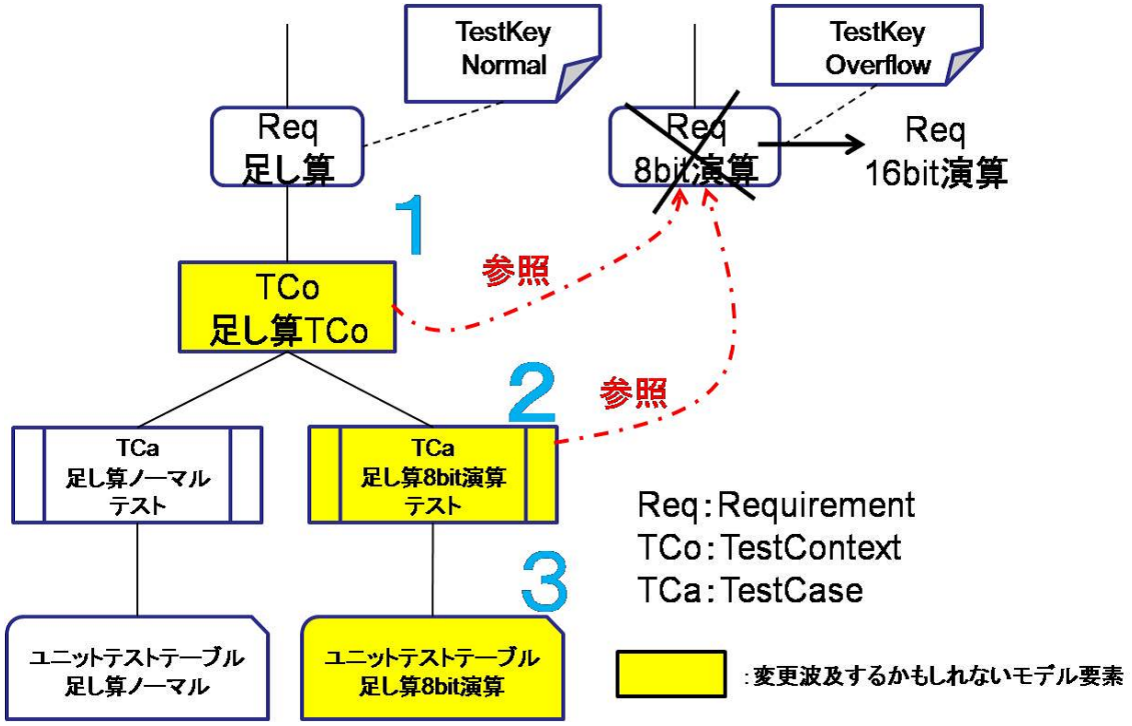

図 7.5:修正順序付与のイメージ

第8章　提案する環境の実装

8.1　概要

提案手法を例題に適用するために実装を行った．以下に実装した個所を示す.

- 要求モデル，テスト詳細モデルならびにテスト観点テンプレートのメタモデルの実装
- テスト観点テンプレートから RTM 図への変換
- RTM 図からテストコンテキスト図への変換
- テストコンテキスト図からテスト構造図への変換
- テストコンテキスト図からテスト振る舞い図への変換
- テストコンテキスト図からユニットテストテーブルへの変換
- 各図からトレーサビリティメタモデルへの変換
- 各図のモデル要素から全要素収集モデルへの変換
- モデルを用いた変更波及解析プログラム

8.2　全体像

以下に実装を行った環境の全体像を示す.

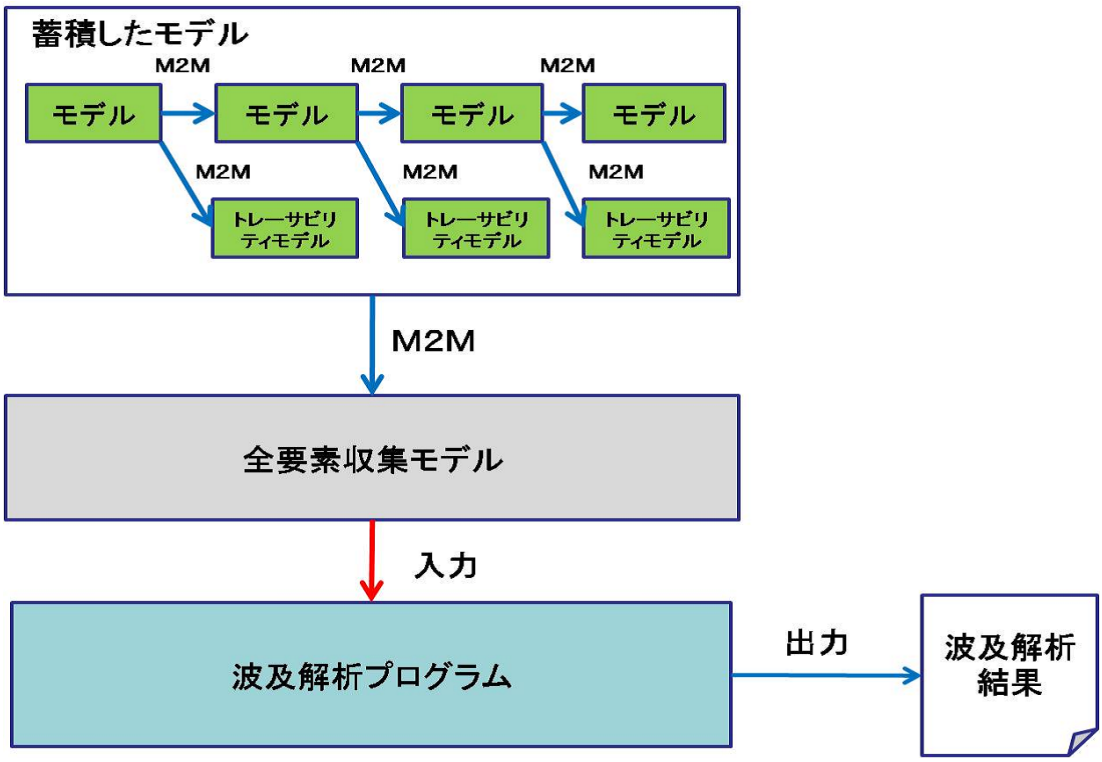

図 8.1:実装した環境の全体像

まず，提案したモデル間でモデル変換を行い，トレーサビリティモデルの生成をする．次に，それらのモデルを収集して，全要素収集モデルをモデル変換にて自動生成する．そして，全要素収集モデルを入力とした変更波及解析プログラムにより，波及解析結果が出力される流れとした．ここで，全要素収集モデルとは，モデル要素を全て収集したモデルであり，要求モデル，テスト詳細モデル，そしてトレーサビリティモデルの全情報を含む．

8.3 モデル

本システムで用いるモデルやメタモデルは，統合開発環境である Eclipse のモデリングフレームワークである EMF(Eclipse Modeling Framework) を用いた．EMF とは MDA(Model-Driven Architecture) 開発に的を絞って実装されているモデリングフレームワークである．その Ecore モデルは OMG が提唱する MOF(Meta Object Facility) を実装したモデルであり，それに従うようにしてモデルを記述することができる．さらに，そのモデルを利用してモデル変換やコード生成などもできるため，モデル開発において EMF は有用なフレームワークであるといえる．

本システムではこの EMF を用いて，提案するモデルのメタモデルの定義を行った．

8.4　モデル変換

本システムでは，重要な位置づけとなるモデル変換を ATL(ATLAS Transformation Language) を用いて行った．ATL はモデル変換における変換ルールを OCL(Object Constraints Language) を利用して記述されるモデル変換言語であり，それを用いたコンポーネントが，Eclipse のプラグインとして配布されている．以下の図 8.2 が ATL を用いたモデル変換のイメージである．

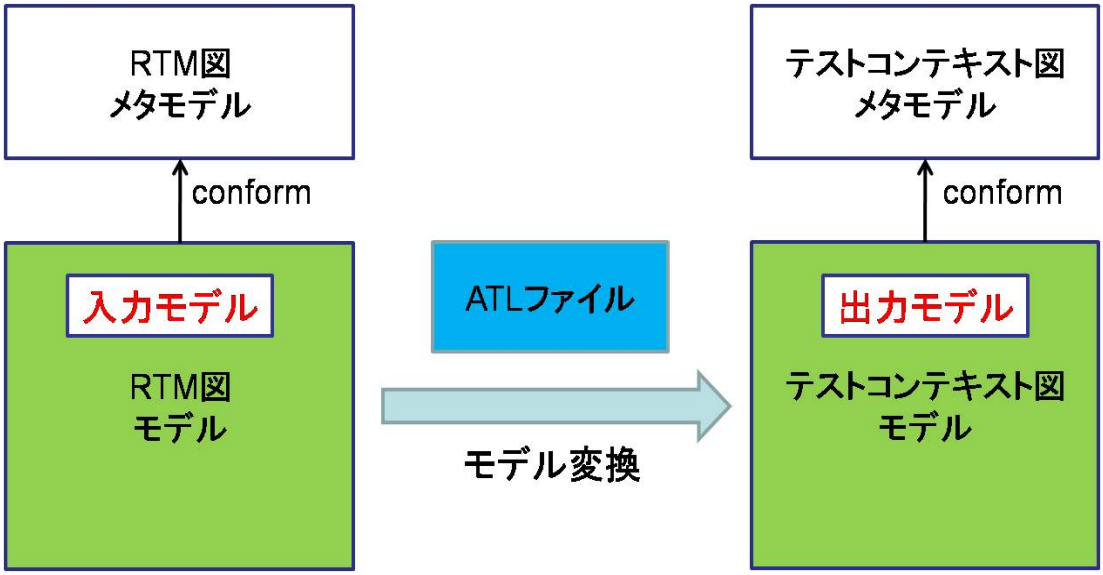

図 8.2:ATL を用いたモデル変換のイメージ

8.5　開発画面

以下に実際の開発画面を示す.

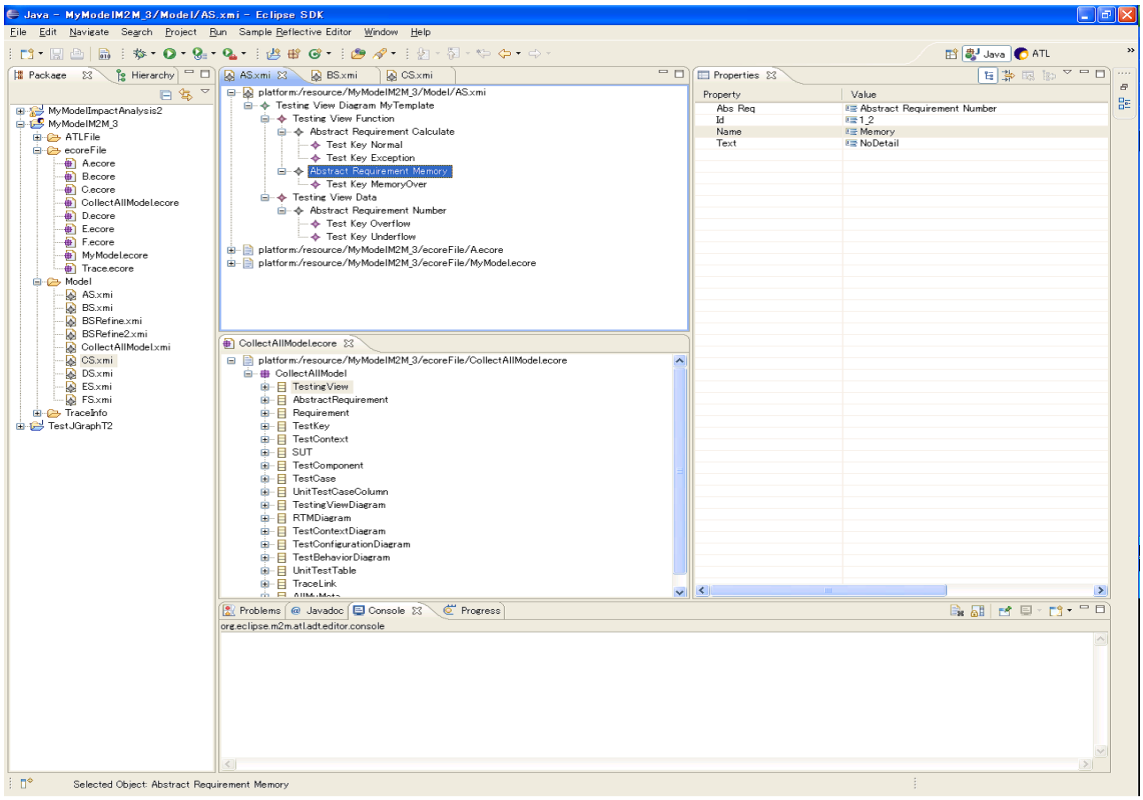

図 8.3:モデルの操作画面

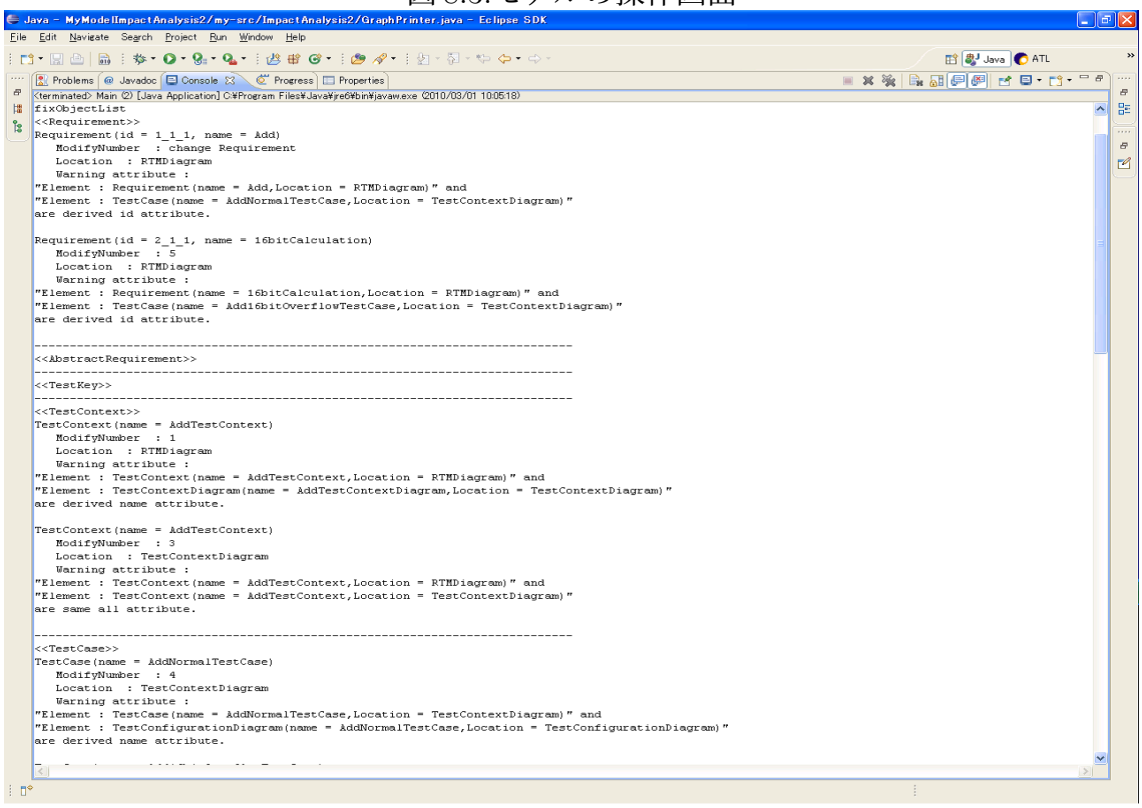

図 8.4:変更波及解析結果画面

第9章　適用例

本研究で提案した開発環境上で簡易電卓の開発を行い，テスト観点を考慮したテスト設計をする．そして，モデルベース上で体系的に整理された情報を用いて，変更波及分析プログラムの動作を確認する．

9.1　簡易電卓

本研究の適用例題として簡易電卓の開発を行う．この電卓は JavaVM 上で動作するアプリケーションで，足し算ができる，16bit 演算ができるという 2 つの機能をもつ．

また，テスト観点テンプレートを元にして図 9.1 に示すようなテスト観点図作成し，それを用いることとした．そして，テスト駆動開発で難しいとされている並列処理，データベースにかかわる処理は，開発の対象としないこととする．また，今回の適用例では，GUI はすでに完成しているものとして話を進める．図 9.2 に電卓の GUI を示す．

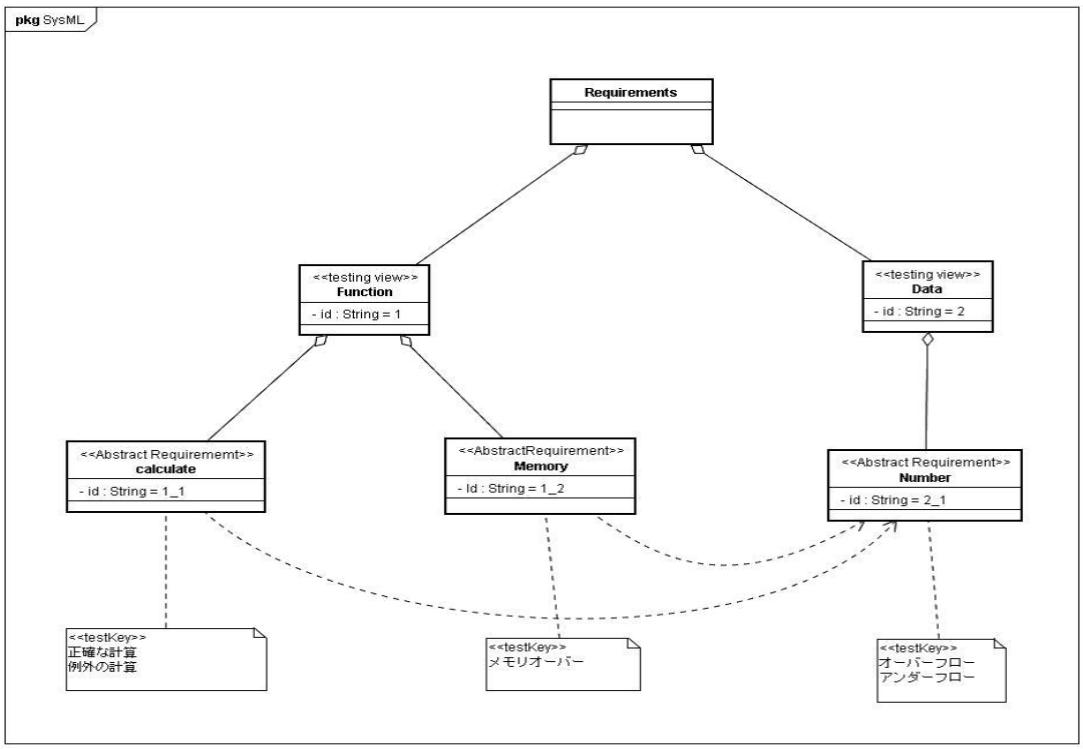

図 9.1:適用例で用いるテスト観点図

図 9.2:電卓の GUI

この電卓は，ボタンによって計算式を入力し，テキストボックスにその計算式の答えを表示する設計となっている．つまり，１，＋，１，＝，と順にボタンを押すとテキストボックスに２という表示がなされる．

9.1.1 開発の手順

電卓の例題を以下のような手順で開発を行った．

0. テスト観点テンプレートの情報を取り込んだ RTM 図を自動生成する
1. 0 で生成した RTM 図へ要求および，それを検証するテストコンテキストを割り当てる
2. RTM 図からテストコンテキスト図を自動生成する
3. 2 で生成したテストコンテキスト図に，TestCase を記述する
4. テストコンテキスト図から，TestCase ごとに空のテスト構造図，テスト振る舞い図，システムテストコード，そしてユニットテストテーブルを自動生成する
5. 空のテスト構造図を記述する
6. 空のテスト振る舞い図を記述する
7. 空のシステムテストコードを 5,6 で記述したモデルを元に記述する
8. ユニットテストテーブルに，ブレークダウンしたテストを記述する
9. ユニットテストテーブルに記述しているブレークダウンしたテストを手動でテストコードにする
10. 9 で記述したテストコードをパスするようにソースコードを記述する

9.2　テンプレートを用いたテスト設計

本項では，テスト観点図がどう用いられ，どのようにして多角的な視点に基づくテスト設計がなされるのかを示す．9.1.1 で示した手順では，0〜3 の手順が対応するため，その流れに沿って，開発画面を交えながら説明する．

- 0. テスト観点テンプレートの情報を取り込んだ RTM 図を自動生成する
  9.1 項で説明したテスト観点図は，ツール上では図 9.3 のようになる．さらに，このテンプレートから RTM 図を自動生成した結果を図 9.4 に示す．

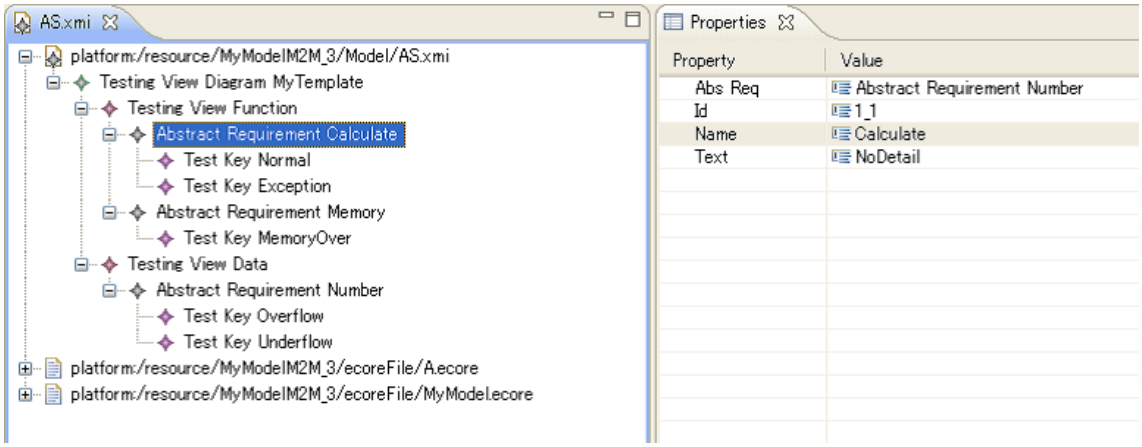

図 9.3:ツール上の縮小版テスト観点テンプレート

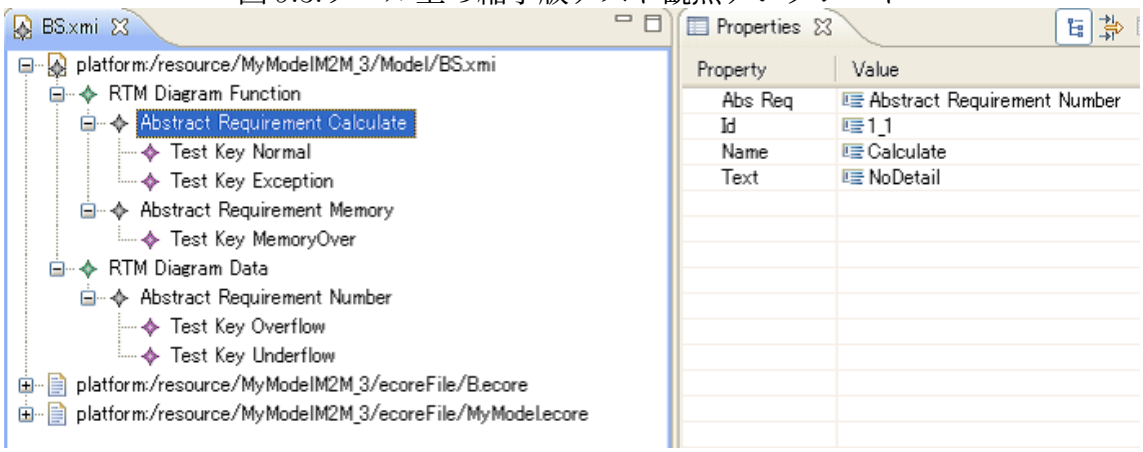

図 9.4:縮小版テスト観点テンプレートから自動で生成された RTM 図

- 1.RTM 図へ要求および，それを検証する TestContext を割り当てる
  自動生成された RTM 図に対して，Requirement とそれに対する TestContext を割り当てる．

  – Requirement の定義
    AbstractRequirement の Calculate に分類されるよう Add という Requirement を定義，そして AbstractRequirement の Number に分類されるよう 16bit-Operation という Requirement を定義した．

さらに，これらの Requirement の TestKey を，定義した Requirement の親である AbstractRequirement がもつ TestKey から選択し，割り当てる．Add を定義する場合，その要求の親である Calculate がもつ TestKey の Normal と Exception から適した TestKey を選択する．ここでは Normal を割り当てた．この TestKey を選択することで，定義する要求を満たすソフトウェアが起こしやすい不具合を明示的にすることができる．また，16bitOperation に対しては同様にして，TestKey に Overflow を割り当てた．

– TestContext の定義
要求 Add に対して AddTestContext を割り当てた．

これらの記述の結果を図 9.5 に示す．

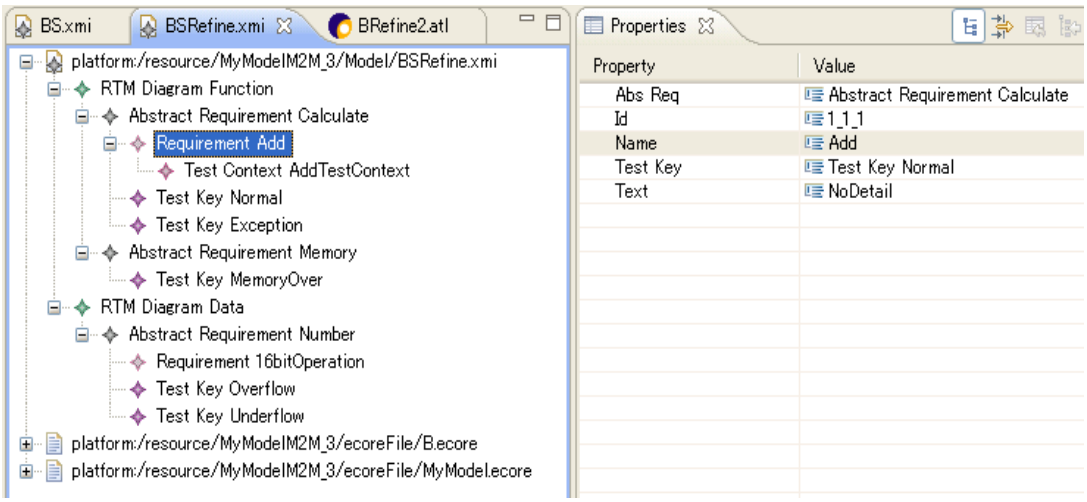

図 9.5:要求を定義し，TestContext を割り当てた RTM 図

- 2.RTM 図で記述した情報を用いて，テストコンテキスト図を自動生成する
図 9.5 での RTM 図から，テストコンテキスト図を自動生成する．その結果を以下の図 9.6 に示す．

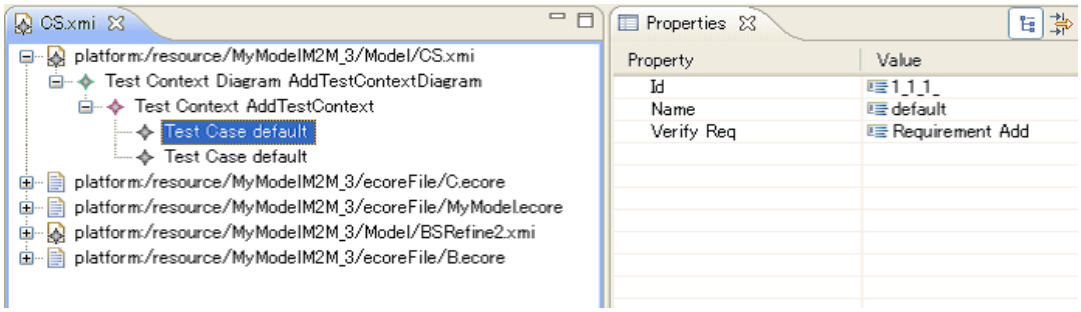

図 9.6:RTM 図から自動生成したテストコンテキスト図

- 3. 生成したテストコンテキスト図に，TestCase を記述する
各要求に対する空の TestCase に対して，ユーザが記述を行う．生成されたテスト

コンテキスト図をみると，2 つの TestCase が自動で生成されているが，これは要求 Add と 16bitCalculation を満たすための空の TestCase である．これに対して，AddNormalTestCase と Add16bitCalculationTestCase を定義した．これ以降はこの TestCase の振る舞いや構造などを作成していくことになる．その際に，TestCase が何の要求を満たすのかを表すパラメータ verityReq を参照し，その Requirement の TestKey を参考にしながら，それらの作成を行う．こうすることで，テスト観点テンプレートで扱う TestingView や TestKey を意識してテストケース作成を行うことができる．

9.3 変更要求ごとの変更波及解析

以上開発したモデルとトレーサビリティモデルを用いて，変更要求ごとに変更波及解析を行う．まず，適用例でのモデル情報とトレーサビリティモデルを図 9.7 に示す．

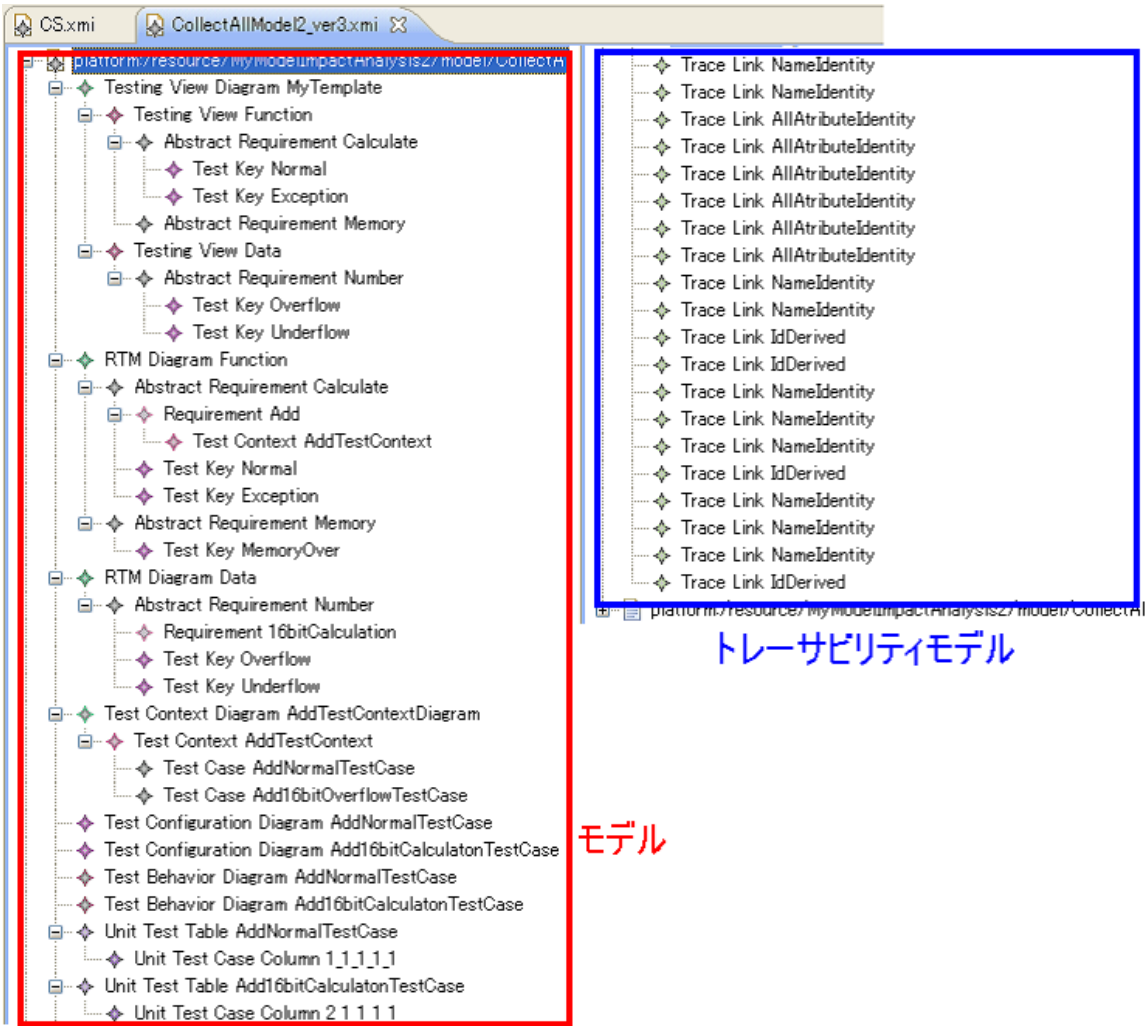

図 9.7:適用例でのモデルとトレーサビリティモデル

このモデル情報とトレーサビリティモデルを用いて，以下の変更要求が起こったと想定して波及解析をする．

- 要求の削除パターン：
  -要求 Add が削除
  -要求 16bitCalculate が削除

- 要求の追加パターン：
  -AbstractRequirement の Calculate に要求 Sub を追加
  -AbstractRequirement の Number に要求 Minus を追加

- 要求の内容変更パターン：
  -要求 Add を RepeatAdd に変更
  -要求 16bitCalculation を 8bitCalculation に変更
  -要求 Add を MemoryAdd に変更

9.3.1 波及解析結果の見かた

各要求変更の波及解析結果を示す前に，ツール上での波及解析結果の見かたについて説明する．各波及解析をした結果は以下の図 9.8 のようになる．

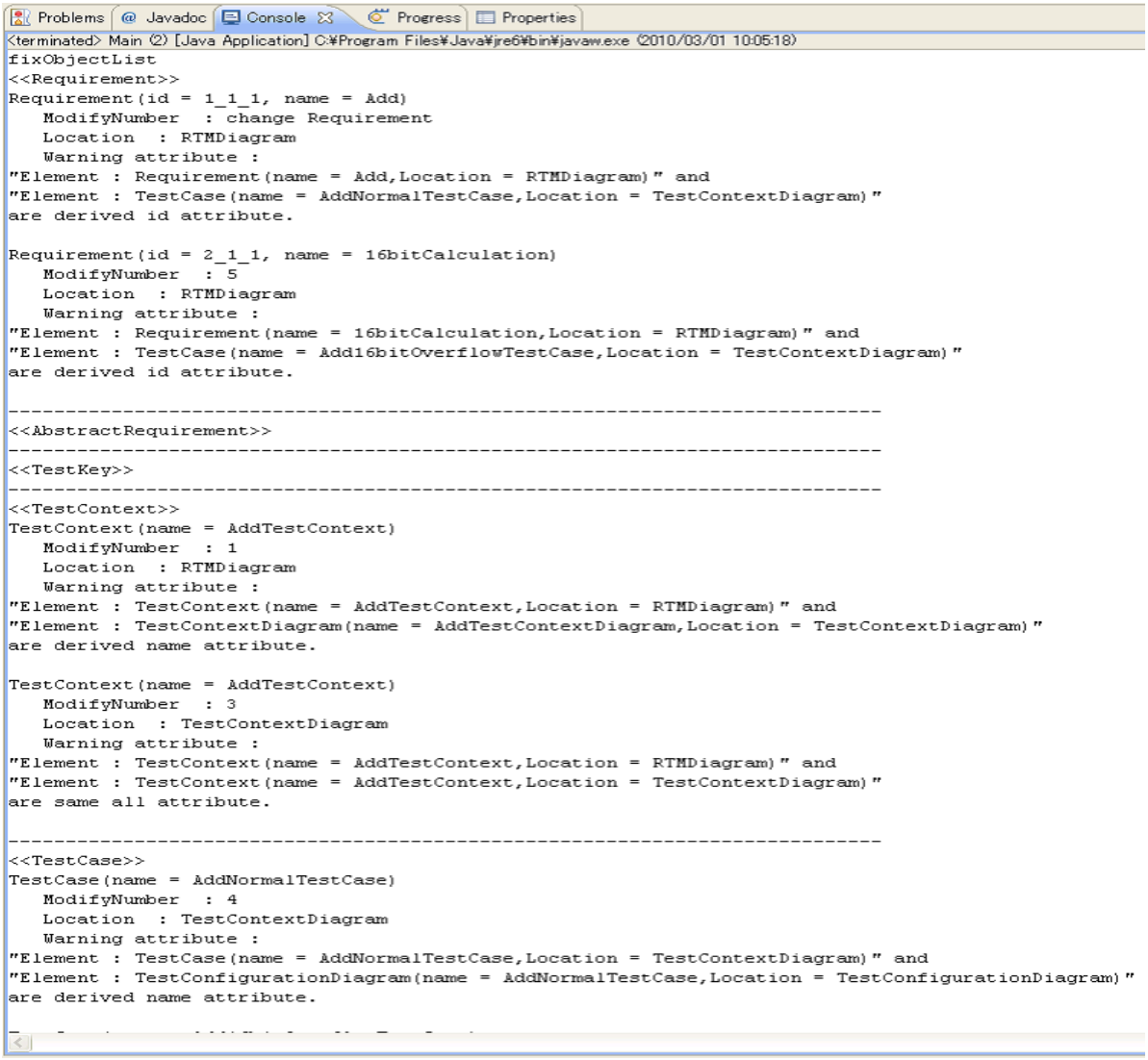

図 9.8:波及解析結果画面

本稿では，波及解析結果の全てを載せるのではなく，要点をまとめた表を示し，その結果を考察するにとどめる．全ての解析結果は付録 B に示す．また，波及解析結果はモデル要素ごとに表示される．そして，各要素には以下の 3 つの情報が結果として出る．

1. ModifyNumber：修正順序を示す
   モデル要素に波及しているか確認する順となっている．

2. Location：モデル要素が属する図を示す

3. Warning attribute：注意すべき要素の属性を示す
   Warning attribute はモデル変換にて自動生成された属性を示し，その属性が変更された場合はこのメッセージを参考にして整合性をとる．たとえば，上図 9.8 の Requirement の 16bitCalculation では，TestCase である Add16bitOverflowTestCase と

id 属性が派生関係にあり，整合性に注意をしなければならないということを示す.

9.3.2 要求の削除パターンに対する変更波及解析結果

- 要求 Add が削除された場合
  Add が削除された場合の波及解析結果を以下に示す.

表 9.1: Add が削除された場合の波及解析結果リスト

重要度	モデル要素	場所
1	TestContext(AddTestContext)	RTM 図
2	TestContextDiagram（AddTestContextDiagram）	テストコンテキスト図
3	TestContext(AddTestContext)	テストコンテキスト図
4	TestCase（AddNormalTestCase）	テストコンテキスト図
4	TestCase（Add16bitOverflowTestCase）	テストコンテキスト図
5	TestConfigurationDiagram(Add16bitOverflowTestCase)	テスト構造図
5	TestBehaviorDiagram(AddNormalTestCase)	テスト振る舞い図
5	UnitTestTable(AddNormalTestCase)	ユニットテストテーブル
5	TestConfigurationDiagram(Add16bitOverflowTestCase)	テスト構造図
5	TestBehaviorDiagram(Add16bitOverflowTestCase)	テスト振る舞い図
5	UnitTestTable(Add16bitOverflowTestCase)	ユニットテストテーブル
5	Requirement(16bitCalculate)	RTM 図
6	UnitTestCaseColumn(testCheckAnalaysis_3)	ユニットテストテーブル
6	UnitTestCaseColumn(testCheckCalculate_1)	ユニットテストテーブル

この結果より，要求 Add が削除された場合，Add のために作った TestCase だけでなく，16bitCalculation のために作った TestCase までも変更が生じる可能性があると分かる．そして，最終的な修正すべき TestCase は testCheckAnalysis_3 と testCheckCalculate_1 となった．また，表に示していないが，実際の波及解析結果にはどのテストクラス，どのテストメソッドなのかという情報が記述されているため，テストコードまで波及する箇所を突き止められる.

- 要求 16bitCalculation が削除された場合
  要求 16bitCalculation が削除された場合を想定したときの波及解析結果は以下のとおりである.

表 9.2: 16bitCalculate が削除された場合の波及解析結果リスト

重要度	モデル要素	場所
1	TestContext(AddTestContext)	RTM 図
2	TestContextDiagram（AddTestContextDiagram）	テストコンテキスト図
3	TestContext(AddTestContext)	テストコンテキスト図
4	TestCase（Add16bitOverflowTestCase）	テストコンテキスト図
5	TestConfigurationDiagram(Add16bitOverflowTestCase)	テスト構造図
5	TestBehaviorDiagram(Add16bitOverflowTestCase)	テスト振る舞い図
5	UnitTestTable(Add16bitOverflowTestCase)	ユニットテストテーブル
6	UnitTestCaseColumn(testCheckAnalaysis_3)	ユニットテストテーブル

この結果より，AddTestContext の 16bitOverflow にかかわるモデル要素が特定できていることが分かる．

9.3.3 要求の追加パターンに対する変更波及解析結果

- AbstractRequirement の Calculate に要求 Sub を追加した場合
  新たに引き算の機能を開発するという状況を考える．そのために AbstractRequirement の Calculate に要求 Sub を追加した．この変更に対して波及解析をした結果，修正の可能性があるモデル要素は存在しなかった．これは，新たに追加した要求に対して開発を進めて良いことを示す．

- AbstractRequirement の Number に要求 Minus を追加した場合
  新たに負の値を用いることができるよう，要求 Minus を AbstractRequirement の Number の子として追加するときの波及解析結果は以下のとおりである．

表 9.3: Minus が追加された場合の波及解析結果リスト

重要度	モデル要素	場所
1	TestContext(AddTestContext)	RTM 図
2	TestContextDiagram（AddTestContextDiagram）	テストコンテキスト図
3	TestContext(AddTestContext)	テストコンテキスト図

この結果より，足し算の TestContext に対して，新たに負の値を用いた TestCase を作成していくことが分かる．

9.3.4 要求の内容変更パターンに対する変更波及解析結果

- 要求 Add を RepeatAdd に変更した場合
  足し算の要求をリピート足し算ができるよう要求を拡張する場合を考える．この変更に対しての波及解析結果は以下のようになった．

表 9.4: Add が RepeatAdd に変更された場合の波及解析結果リスト

重要度	モデル要素	場所
1	TestContext(AddTestContext)	RTM 図
2	TestContextDiagram（AddTestContextDiagram）	テストコンテキスト図
3	TestContext(AddTestContext)	テストコンテキスト図
4	TestCase（AddNormalTestCase）	テストコンテキスト図
4	TestCase（Add16bitOverflowTestCase）	テストコンテキスト図
5	TestConfigurationDiagram(Add16bitOverflowTestCase)	テスト構造図
5	TestBehaviorDiagram(AddNormalTestCase)	テスト振る舞い図
5	UnitTestTable(AddNormalTestCase)	ユニットテストテーブル
5	TestConfigurationDiagram(Add16bitOverflowTestCase)	テスト構造図
5	TestBehaviorDiagram(Add16bitOverflowTestCase)	テスト振る舞い図
5	UnitTestTable(Add16bitOverflowTestCase)	ユニットテストテーブル
5	Requirement(16bitCalculate)	RTM 図
6	UnitTestCaseColumn(testCheckAnalaysis_3)	ユニットテストテーブル
6	UnitTestCaseColumn(testCheckCalculate_1)	ユニットテストテーブル

この結果より，Requirement の Add のために作った TestCase だけでなく，16bitCalculation のために作った TestCase まで影響するかもしれないということが分かる．

- 要求 16bitCalculation を 8bitCalculation に変更した場合
  16bit 演算から 8bit 演算に内容の変更が行われた場合を考える．この変更によって影響を受ける可能性のあるモデル要素を表 9.5 に示す．

表 9.5: 16bitCalculation が 8bitCalculation に変更された場合の波及解析結果リスト

重要度	モデル要素	場所
1	TestContext(AddTestContext)	RTM 図
2	TestContextDiagram（AddTestContextDiagram）	テストコンテキスト図
3	TestContext(AddTestContext)	テストコンテキスト図
4	TestCase（Add16bitOverflowTestCase）	テストコンテキスト図
5	TestConfigurationDiagram(Add16bitOverflowTestCase)	テスト構造図
5	TestBehaviorDiagram(Add16bitOverflowlTestCase)	テスト振る舞い図
5	UnitTestTable(Add16bitOverflowlTestCase)	ユニットテストテーブル
6	UnitTestCaseColumn(testCheckAnalaysis_3)	ユニットテストテーブル

この結果より，16bitCalcuration のために作った TestCase に波及する可能性があることが分かる．

- 要求 Add を MemoryAdd に変更した場合
  足し算の機能をメモリー足し算に拡張する．この機能は，計算結果を電卓に覚え込ませながら足し算ができる機能である．そして，変更した要求の AbstractRequirement の分類を Calculation から Memory に再分類したときを考える．このときの波及解析結果を以下に示す．

表 9.6: Add が MemoryAdd に変更された場合の波及解析結果リスト

重要度	モデル要素	場所
1	TestContext(AddTestContext)	RTM 図
2	TestContextDiagram（AddTestContextDiagram）	テストコンテキスト図
3	TestContext(AddTestContext)	テストコンテキスト図
4	TestCase（AddNormalTestCase）	テストコンテキスト図
5	TestConfigurationDiagram(AddNormalTestCase)	テスト構造図
5	TestBehaviorDiagram(AddNormalTestCase)	テスト振る舞い図
5	UnitTestTable(AddNormalTestCase)	ユニットテストテーブル
6	UnitTestCaseColumn(testCheckCalculate_1)	ユニットテストテーブル

この結果より，TestKey の Normal にかかわる要素を取り出していることが分かる．これは，AbstractRequirement の再分類をしたことにより，参照できる TestKey が変化したためである．この場合，TestKey の Normal が参照できなくなったため，それに関わるモデル要素は修正する可能性がある．

第10章　まとめ

10.1　まとめ

本研究は，多角的な視点に基づくテスト設計ができ，テスト駆動開発で扱う情報をモデルベースで管理した上で変更波及解析を行う環境を提案し，その振る舞いを電卓の適用例を用いて確認した．このことから，テスト駆動開発に必要な情報を体系的に管理できること，影響波及を特定できることが確認できた．

10.2　課題

本研究には以下の課題が残った．

- AbstractRequirement および Requirement の多段定義を考慮した波及解析方法
  本研究では AbstractRequirement と Requirement に対して，多段に定義することもありうる．たとえば，電卓の機能で足し算機能と割り算機能が備わっている中で，平均計算機能を加えるとき，平均計算機能は足し算機能や割り算機能に依存する．この場合，これらに対して関連を引いたほうが要求をより整理できる．それに対する波及解析方法を考える必要がある．

- テスト観点テンプレートの拡張
  本研究でまとめたテスト観点テンプレートは，フィードバックと共に改善することができるが，本研究であらかじめ提案したテンプレートは要素数が少ないため，改善の余地がある．

- テスト駆動開発の適用の限界
  テスト駆動開発は，並列処理やデータベース，そして GUI にかかわる機能を実装することに対して適用することが難しい．並列処理の場合は，タイミングの問題など様々な事象が考えられテストしきれないためである．そして，データベースの場合はテストをするデータベースの状況を準備することに手間がかかるからである．最後に GUI では，頻繁に変更要求が起こるため，テストを書いてコードを書くという手順を踏むことが徐々に足かせになってくるためである．本研究はこれを踏まえてテスト観点テンプレートの抽出方法や適用例に制約を設けて議論をした．

10.3 関連研究

テスト駆動開発の関連研究として，[10] がある．この研究では，テスト駆動開発とモデリングについて研究をされ，モデルによるテスト駆動型開発（TDDM）を提唱している．これは，テスト駆動開発のプロセスをモデリングに適用した研究であり，テスト駆動開発で扱う情報を体系的に管理している．本研究ではモデリングに焦点を当てず，ソースコード開発を対象としたテスト駆動開発にしぼり，テスト設計の概念を取り入れ，さらに情報を体系的に管理できる環境を提案した．

10.4 謝辞

本研究を進めるにあたり，遠隔地にも関わらず，いつも熱心にご指導してくださった岸知二客員教授に深く御礼を申し上げます．そして，合同ゼミや合同合宿にて貴重なコメントをくださった Defago Xavier 准教授に感謝いたします．また，身近な存在でもあり，忙しい時でも，いつも親身になって相談に乗ってくださった岸研究室，Defago 研究室の先輩方ありがとうございました．最後に研究活動を共に励んできた同期の方々に感謝をし，謝辞といたします．

参考文献

[1] 日経 BP 社, 組み込みソフトウェア開発のための最新技法と基礎知識 設計からトラブルシューティングまで, 2007.

[2] 郷祐一, 水野修, 白神彰則, 安部剛, 田中良明, テスト駆動開発の通信ソフトウェアへの適用モデルと評価, 2005.

[3] ケント・ベック, 長瀬嘉秀（監訳）, テスト駆動開発入門, （株）ピアソン・エデュケーション, 2003.

[4] 西康晴, テスト観点に着目したソフトウェアテスト設計プロセス, SEA 関西プロセス分科会＆てふかん共同開催,2006.

[5] Cem Kaner,Hung Quoc Nguyen,Jack Falk, テスト技術者交流会 (訳), 基本から学ぶソフトウェアテスト―テストの「プロ」を目指す人のために, 2001.

[6] 河野哲也, 西康晴, バグのパターンを用いたテストの提案－バグのナレッジマネジメント－, 2006.

[7] OMG, OMG Systems Modeling Language(OMG SysML), V1.0, OMG, 2007.

[8] OMG, UML 2.0 Testing Profile Specification # :ptc, 03-07-01.

[9] 我妻智之, 神谷慎吾, 大平直宏, 松下誠, 楠本真二，井上克郎, メタモデルに基づくトレーサビリティ技術の提案, 2005.

[10] 潘沂冰, 林晋, UML モデリングのためのテスト駆動開発, 2003.

[11] Sanford Friedenthal, Alan Moore,Rick Steiner,Morgan Kaufmann, A Practical Guide to SysML -The Systems Modeling Language, Revised 版,2009.

[12] Paul Baker, Zhen Ru Dai, Jens Grabowski, Oystein Haugen, Ina Schieferdecker, Clay Williams, Model-driven Testing -Using the UML Testing Profile, Springer, 2007.

[13] 小谷正行, 落水浩一郎, UML 記述の変更波及解析に利用可能な依存関係の自動生成, 2008.

[14] スコット・W・アンブラー, 株式会社オージス総研 訳, アジャイルモデリング—XPと統一プロセスを補完するプラクティス, 2003.

[15] 和田卓人, デベロッパーテスティング〜ソフトウェア開発者の基礎体力, オブジェクト倶楽部 2006 夏イベント, 2006.

付録A　テスト観点テンプレートの全体像（表形式）

View	AbsReq	TestKey	なぜ起こったか	備考
機能	計算	正確な計算	テスト自体が間違っていた	計算ができるかのチェック、通常の計算
		誤った評価基準	式の表記法を考慮していなかった。	式の表現チェック(例：階乗とかけ算の表記)
		切り捨て	切り捨ての場合があることを考慮していなかった。	
		四捨五入	四捨五入の場合があることを考慮していなかった。	
		他のデータ表現への変換	データ表現の変換が正しく行われていないことを考慮していなかった。	char型から数字の変換(-48)などをチェック
		近似式	近似処理を行う関数の特性を考慮していなかった。	
		無効な計算	数学的に無効な計算を考慮していなかった。	負数に対して平方根を求めようとした、0除算など
		公式	公式自体にバグがあることを考慮していなかった。	公式が正しく用いているかどうかのチェック
		誤った基本関数	利用する基本関数自体にバグがあるということを考慮していなかった。	サイン、コサイン、双曲関数の近似を考慮したチェック

View	AbsReq	TestKey	なぜ起こったか	備考
データ	数	数に対する境界	値に対して境界値のテストをしていなかった。	
		古い仮定による定数	昔に決めた定数(コンピュータに接続できる端末の台数の最大値など)を更新していなかった。	
		オーバーフロー	オーバーフローする場合を考えていなかった。	
		アンダーフロー	アンダーフローする場合を考えていなかった。	
	文字	大文字小文字の区別	大文字と小文字の入力によって 処理が変わる可能性を考えていなかった。	
		空文字の入力	空文字の入力を考えていなかった。	
		NULLで終了しない文字列の入力	NULLで終了しない文字列が入力されることを考えていなかった。	
	ファイル	拡張ファイル名の認識	ファイル名の長さによって認識できないシステムがあることを考えていなかった。 (昔のシステムではスペックの都合上ファイル名や変数の名称の長さは6～8文字までしか認識できないらしい)	
		空のファイル	空のファイルが入力される場合を考えていなかった。	
		別形式の保存	さまざまな種類のファイルを出力できることを考慮していなかった。	
	データ構造	データ構造の境界	データ構造の境界を考えていなかった。	

View	AbsReq	TestKey	なぜ起こったか	備考
回復	エラー回復	初期状態が正しいか	メモリ領域が0から始まらない場合を考慮していなかった 初期化処理のチェックを行っていなかった	メモリ領域の値が0で開始するとき、プログラムは必ずOが設定済みと思うのではなく、 その他のデータが設定されている可能性を考えること
		異常終了時の処理	異常終了時の処理を考慮していなかった	異常終了時に原因を記録したかどうか、ユーザがプログラムを止められるか、 プログラム自身が自動で処理を止められるか
	エラー検出	発生しないはずの値	あり得ない値を与えられた時の対処を考慮していなかった	たとえば、2月31日のような日付の受付など
		エラー状態の通知	エラー状態であると通知する機能を実装していなかった	エラー状態であると、ユーザーに知らせるかどうかのチェック

View	abs req	abs req	TestKey	なぜ起こったか	備考
環境	動作環境	デバイス	HWの入れ替え・追加	以前に定めたソフトウェアの制約が 新しいハードウェアを制御できない	購入当初とは別のHWが使えるかどうかのチェック
			デバイスの使用終了の連絡		デバイスが使用状態から解放されたときのメッセージのシグナルがあるかどうかのチェック
			デバイスの欠陥		
		OS	OSのバグ		OSのバグは把握しているか
		SWのバージョン	不適当なバージョン管理の対処	新しいバージョンのファイルで構成されていなかった。	新しいバージョンのファイルで構成されているかどうかの機能があるかチェック
		ネットワーク			
	開発環境	開発言語	コーディングスタイルによるバグ		自動変数の自動的な初期化をしない、文字の大文字・小文字が区別されるなど
		コンパイラ	コンパイラの特性		自動的に初期値が0にならないコンパイラでのチェックP406

View	abs req	abs req	TestKey	なぜ起こったか	備考
負荷	大容量	ディスク、メモリなどデバイス対象	扱える限界値		
	ストレス	ディスク、メモリなどデバイス対象	限界状態でのテスト		

View	absReq	abs req	TestKey	なぜ起こったか	備考
性能	プログラムのスピード	タイムアウト時間	タイムアウト時間	タイムアウト時間がうまく設定されていなかった。	
			タイムアウト時の処理	タイムアウト時にコマンド入力されている 状況があることを考えていなかった。	
		長い処理	処理通知機能	チェック漏れ	
			処理キャンセル機能	チェック漏れ	
			処理経過表示	チェック漏れ	

付 録B　変更波及解析

- 要求の削除パターンに対する変更波及解析結果

解析結果 B.1: 要求 Add が削除された場合

```
1  fixObjectList
2  <<Requirement>>
3  Requirement(id = 1_1_1, name = Add)
4      ModifyNumber   : change Requirement
5      Location   : RTMDiagram
6      warning attribute :
7  "Element : Requirement(name = Add, Location = RTMDiagram)" and
8  "Element : TestCase(name = AddNormalTestCase,
9  Location = TestContextDiagram)"
10 are derived id attribute.
11
12 Requirement(id = 2_1_1, name = 16bitCalculation)
13     ModifyNumber   : 5
14     Location   : RTMDiagram
15     warning attribute :
16 "Element : Requirement(name = 16bitCalculation, Location = RTMDiagram)"
17 and
18 "Element : TestCase(name = Add16bitOverflowTestCase,
19 Location = TestContextDiagram)"
20 are derived id attribute.
21
22 ------------------------------------------------------------------------
23 <<AbstractRequirement>>
24 ------------------------------------------------------------------------
25 <<TestKey>>
26 ------------------------------------------------------------------------
27 <<TestContext>>
28 TestContext(name = AddTestContext)
29     ModifyNumber   : 1
30     Location   : RTMDiagram
31     warning attribute :
32 "Element : TestContext(name = AddTestContext, Location = RTMDiagram)"
33 and
34 "Element : TestContextDiagram(name = AddTestContextDiagram,
35 Location = TestContextDiagram)"
36 are derived name attribute.
37
38 TestContext(name = AddTestContext)
39     ModifyNumber   : 3
```

```
40      Location   : TestContextDiagram
41      warning attribute :
42  "Element : TestContext(name = AddTestContext,
43  Location = RTMDiagram)" and
44  "Element : TestContext(name = AddTestContext,
45  Location = TestContextDiagram)"
46  are same all attribute.
47
48  ------------------------------------------------------------------------
49  <<TestCase>>
50  TestCase(name = AddNormalTestCase)
51      ModifyNumber   : 4
52      Location   : TestContextDiagram
53      warning attribute :
54  "Element : TestCase(name = AddNormalTestCase,
55  Location = TestContextDiagram)" and
56  "Element : UnitTestTable(name = AddNormalTestCase,
57  Location = UnitTestTable)"
58  are derived name attribute.
59
60  TestCase(name = Add16bitOverflowTestCase)
61      ModifyNumber   : 4
62      Location   : TestContextDiagram
63      warning attribute :
64  "Element : TestCase(name = Add16bitOverflowTestCase,
65  Location = TestContextDiagram)" and
66  "Element : UnitTestTable(name = Add16bitCalculatonTestCase,
67  Location = UnitTestTable)"
68  are derived name attribute.
69
70  ------------------------------------------------------------------------
71  <<UnitTestCase>>
72  UnitTestCaseColumn(name = testCheckAnalysis)
73  TargetClass   : Analysis
74  TargetMethod : analysis
75      ModifyNumber   : 6
76      Location   : UnitTestTable
77      warning attribute :
78  "Element : TestCase(name = Add16bitOverflowTestCase,
79  Location = TestContextDiagram)" and
80  "Element : UnitTestCaseColumn(name = testCheckAnalysis,
81  Location = UnitTestTable)"
82  are derived id attribute.
83
84  UnitTestCaseColumn(name = testCheckCalculate)
85  TargetClass   : Calculate
86  TargetMethod : calculate
87      ModifyNumber   : 6
88      Location   : UnitTestTable
89      warning attribute :
90  "Element : TestCase(name = AddNormalTestCase,
91  Location = TestContextDiagram)" and
92  "Element : UnitTestCaseColumn(name = testCheckCalculate,
```

```
93  Location = UnitTestTable)"
94  are derived id attribute.
95
96  ------------------------------------------------------------------------
97  <<Other Model Element>>
98  TestContextDiagram(name = AddTestContextDiagram)
99     ModifyNumber  : 2
100    Location  : TestContextDiagram
101    warning attribute :
102 "Element : TestContext(name = AddTestContext,
103 Location = RTMDiagram)" and
104 "Element : TestContextDiagram(name = AddTestContextDiagram,
105 Location = TestContextDiagram)"
106 are derived name attribute.
107
108 TestBehaviorDiagram(name = AddNormalTestCase)
109    ModifyNumber  : 5
110    Location  : TestBehaviorDiagram
111    warning attribute :
112 "Element : TestCase(name = AddNormalTestCase,
113 Location = TestContextDiagram)" and
114 "Element : TestBehaviorDiagram(name = AddNormalTestCase,
115 Location = TestBehaviorDiagram)"
116 are derived name attribute.
117
118 TestConfigurationDiagram(name = Add16bitCalculatonTestCase)
119    ModifyNumber  : 5
120    Location  : TestConfigurationDiagram
121    warning attribute :
122 "Element : TestCase(name = Add16bitOverflowTestCase,
123 Location = TestContextDiagram)" and
124 "Element : TestConfigurationDiagram(name = Add16bitCalculatonTestCase,
125 Location = TestConfigurationDiagram)"
126 are derived name attribute.
127
128 TestBehaviorDiagram(name = Add16bitCalculatonTestCase)
129    ModifyNumber  : 5
130    Location  : TestBehaviorDiagram
131    warning attribute :
132 "Element : TestCase(name = Add16bitOverflowTestCase,
133 Location = TestContextDiagram)" and
134 "Element : TestBehaviorDiagram(name = Add16bitCalculatonTestCase,
135 Location = TestBehaviorDiagram)"
136 are derived name attribute.
137
138 UnitTestCaseTable(name = Add16bitCalculatonTestCase)
139    ModifyNumber  : 5
140    Location  : UnitTestTable
141    warning attribute :
142 "Element : TestCase(name = Add16bitOverflowTestCase,
143 Location = TestContextDiagram)" and
144 "Element : UnitTestTable(name = Add16bitCalculatonTestCase,
145 Location = UnitTestTable)"
```

```
146 are derived name attribute.
147
148 TestConfigurationDiagram(name = AddNormalTestCase)
149     ModifyNumber   : 5
150     Location   : TestConfigurationDiagram
151     warning attribute :
152 "Element : TestCase(name = AddNormalTestCase,
153 Location = TestContextDiagram)" and
154 "Element : TestConfigurationDiagram(name = AddNormalTestCase,
155 Location = TestConfigurationDiagram)"
156 are derived name attribute.
157
158 UnitTestCaseTable(name = AddNormalTestCase)
159     ModifyNumber   : 5
160     Location   : UnitTestTable
161     warning attribute :
162 "Element : TestCase(name = AddNormalTestCase,
163 Location = TestContextDiagram)" and
164 "Element : UnitTestTable(name = AddNormalTestCase,
165 Location = UnitTestTable)"
166 are derived name attribute.
167
168 total Element = 15
```

解析結果 B.2: 要求 16bitCalculation が削除された場合

```
 1 fixObjectList
 2 <<Requirement>>
 3 Requirement(id = 2_1_1, name = 16bitCalculation)
 4     ModifyNumber   : change Requirement
 5     Location   : RTMDiagram
 6     warning attribute :
 7 "Element : Requirement(name = 16bitCalculation,
 8 Location = RTMDiagram)" and
 9 "Element : TestCase(name = Add16bitOverflowTestCase,
10 Location = TestContextDiagram)"
11 are derived id attribute.
12
13 ------------------------------------------------------------------------
14 <<AbstractRequirement>>
15 ------------------------------------------------------------------------
16 <<TestKey>>
17 ------------------------------------------------------------------------
18 <<TestContext>>
19 TestContext(name = AddTestContext)
20     ModifyNumber   : 3
21     Location   : TestContextDiagram
22     warning attribute :
23 "Element : TestContext(name = AddTestContext,
24 Location = RTMDiagram)" and
25 "Element : TestContext(name = AddTestContext,
26 Location = TestContextDiagram)"
```

```
27 are same all attribute.
28
29 TestContext(name = AddTestContext)
30    ModifyNumber  : 1
31    Location  : RTMDiagram
32    warning attribute :
33 "Element : TestContext(name = AddTestContext,
34 Location = RTMDiagram)" and
35 "Element : TestContextDiagram(name = AddTestContextDiagram,
36 Location = TestContextDiagram)"
37 are derived name attribute.
38
39 ---------------------------------------------------------------------
40 <<TestCase>>
41 TestCase(name = Add16bitOverflowTestCase)
42    ModifyNumber  : 4
43    Location  : TestContextDiagram
44    warning attribute :
45 "Element : Requirement(name = 16bitCalculation,
46 Location = RTMDiagram)" and
47 "Element : TestCase(name = Add16bitOverflowTestCase,
48 Location = TestContextDiagram)"
49 are derived id attribute.
50
51 ---------------------------------------------------------------------
52 <<UnitTestCase>>
53 UnitTestCaseColumn(name = testCheckAnalysis)
54 TargetClass  : Analysis
55 TargetMethod : analysis
56    ModifyNumber  : 6
57    Location  : UnitTestTable
58    warning attribute :
59 "Element : TestCase(name = Add16bitOverflowTestCase,
60 Location = TestContextDiagram)" and
61 "Element : UnitTestCaseColumn(name = testCheckAnalysis,
62 Location = UnitTestTable)"
63 are derived id attribute.
64
65 ---------------------------------------------------------------------
66 <<Other Model Element>>
67 TestContextDiagram(name = AddTestContextDiagram)
68    ModifyNumber  : 2
69    Location  : TestContextDiagram
70    warning attribute :
71 "Element : TestContext(name = AddTestContext,
72 Location = RTMDiagram)" and
73 "Element : TestContextDiagram(name = AddTestContextDiagram,
74 Location = TestContextDiagram)"
75 are derived name attribute.
76
77 TestConfigurationDiagram(name = Add16bitCalculatonTestCase)
78    ModifyNumber  : 5
79    Location  : TestConfigurationDiagram
```

```
80      warning attribute :
81  "Element : TestCase(name = Add16bitOverflowTestCase,
82  Location = TestContextDiagram)" and
83  "Element : TestConfigurationDiagram(name = Add16bitCalculatonTestCase,
84  Location = TestConfigurationDiagram)"
85  are derived name attribute.
86
87  UnitTestCaseTable(name = Add16bitCalculatonTestCase)
88      ModifyNumber  : 5
89      Location  : UnitTestTable
90      warning attribute :
91  "Element : TestCase(name = Add16bitOverflowTestCase,
92  Location = TestContextDiagram)" and
93  "Element : UnitTestTable(name = Add16bitCalculatonTestCase,
94  Location = UnitTestTable)"
95  are derived name attribute.
96
97  TestBehaviorDiagram(name = Add16bitCalculatonTestCase)
98      ModifyNumber  : 5
99      Location  : TestBehaviorDiagram
100     warning attribute :
101 "Element : TestCase(name = Add16bitOverflowTestCase,
102 Location = TestContextDiagram)" and
103 "Element : TestBehaviorDiagram(name = Add16bitCalculatonTestCase,
104 Location = TestBehaviorDiagram)"
105 are derived name attribute.
106
107 total Element = 9
```

- 要求の追加パターンに対する変更波及解析結果

解析結果 B.3: AbstractRequirement の Calculate に要求 Sub を追加した場合

```
1  no impact element.
```

解析結果 B.4: AbstractRequirement の Number に要求 Minus を追加した場合

```
1  fixObjectList
2  <<Requirement>>
3  ------------------------------------------------------------------------
4  <<AbstractRequirement>>
5  ------------------------------------------------------------------------
6  <<TestKey>>
7  ------------------------------------------------------------------------
8  <<TestContext>>
9  TestContext(name = AddTestContext)
10     ModifyNumber  : 1
11     Location  : RTMDiagram
12     warning attribute :
13 "Element : TestContext(name = AddTestContext,Location = RTMDiagram)"
14 and
15 "Element : TestContextDiagram(name = AddTestContextDiagram,
```

```
Location = TestContextDiagram)"
are derived name attribute.

TestContext(name = AddTestContext)
    ModifyNumber   : 3
    Location   : TestContextDiagram
    warning attribute :
"Element : TestContext(name = AddTestContext,
Location = RTMDiagram)" and
"Element : TestContext(name = AddTestContext,
Location = TestContextDiagram)"
are same all attribute.

--------------------------------------------------------------------------------
<<TestCase>>
--------------------------------------------------------------------------------
<<UnitTestCase>>
--------------------------------------------------------------------------------
<<Other Model Element>>
TestContextDiagram(name = AddTestContextDiagram)
    ModifyNumber   : 2
    Location   : TestContextDiagram
    warning attribute :
"Element : TestContext(name = AddTestContext,
Location = RTMDiagram)" and
"Element : TestContextDiagram(name = AddTestContextDiagram,
Location = TestContextDiagram)"
are derived name attribute.

total Element = 3
```

- 要求の内容変更パターンに対する変更波及解析結果

解析結果 B.5: 要求 Add を RepeatAdd に変更した場合

```
fixObjectList
<<Requirement>>
Requirement(id = 1_1_1, name = Add)
    ModifyNumber   : change Requirement
    Location   : RTMDiagram
    warning attribute :
"Element : Requirement(name = Add, Location = RTMDiagram)" and
"Element : TestCase(name = AddNormalTestCase,
Location = TestContextDiagram)"
are derived id attribute.

Requirement(id = 2_1_1, name = 16bitCalculation)
    ModifyNumber   : 5
    Location   : RTMDiagram
    warning attribute :
"Element : Requirement(name = 16bitCalculation,
Location = RTMDiagram)" and
```

```
"Element : TestCase(name = Add16bitOverflowTestCase,
Location = TestContextDiagram)"
are derived id attribute.

-------------------------------------------------------------------------------
<<AbstractRequirement>>
-------------------------------------------------------------------------------
<<TestKey>>
-------------------------------------------------------------------------------
<<TestContext>>
TestContext(name = AddTestContext)
    ModifyNumber  : 1
    Location  : RTMDiagram
    warning attribute :
"Element : TestContext(name = AddTestContext,
Location = RTMDiagram)" and
"Element : TestContextDiagram(name = AddTestContextDiagram,
Location = TestContextDiagram)"
are derived name attribute.

TestContext(name = AddTestContext)
    ModifyNumber  : 3
    Location  : TestContextDiagram
    warning attribute :
"Element : TestContext(name = AddTestContext,
Location = RTMDiagram)" and
"Element : TestContext(name = AddTestContext,
Location = TestContextDiagram)"
are same all attribute.

-------------------------------------------------------------------------------
<<TestCase>>
TestCase(name = AddNormalTestCase)
    ModifyNumber  : 4
    Location  : TestContextDiagram
    warning attribute :
"Element : TestCase(name = AddNormalTestCase,
Location = TestContextDiagram)" and
"Element : UnitTestTable(name = AddNormalTestCase,
Location = UnitTestTable)"
are derived name attribute.

TestCase(name = Add16bitOverflowTestCase)
    ModifyNumber  : 4
    Location  : TestContextDiagram
    warning attribute :
"Element : TestCase(name = Add16bitOverflowTestCase,
Location = TestContextDiagram)" and
"Element : TestConfigurationDiagram(name = Add16bitCalculatonTestCase,
Location = TestConfigurationDiagram)"
are derived name attribute.

-------------------------------------------------------------------------------
```

```
71  <<UnitTestCase>>
72  UnitTestCaseColumn(name = testCheckAnalysis)
73  TargetClass   : Analysis
74  TargetMethod : analysis
75     ModifyNumber   : 6
76     Location   : UnitTestTable
77     warning attribute :
78  "Element : TestCase(name = Add16bitOverflowTestCase,
79  Location = TestContextDiagram)" and
80  "Element : UnitTestCaseColumn(name = testCheckAnalysis,
81  Location = UnitTestTable)"
82  are derived id attribute.
83
84  UnitTestCaseColumn(name = testCheckCalculate)
85  TargetClass   : Calculate
86  TargetMethod : calculate
87     ModifyNumber   : 6
88     Location   : UnitTestTable
89     warning attribute :
90  "Element : TestCase(name = AddNormalTestCase,
91  Location = TestContextDiagram)" and
92  "Element : UnitTestCaseColumn(name = testCheckCalculate,
93  Location = UnitTestTable)"
94  are derived id attribute.
95
96  ------------------------------------------------------------------------
97  <<Other Model Element>>
98  TestContextDiagram(name = AddTestContextDiagram)
99     ModifyNumber   : 2
100    Location   : TestContextDiagram
101    warning attribute :
102 "Element : TestContext(name = AddTestContext,
103 Location = RTMDiagram)" and
104 "Element : TestContextDiagram(name = AddTestContextDiagram,
105 Location = TestContextDiagram)"
106 are derived name attribute.
107
108 TestBehaviorDiagram(name = AddNormalTestCase)
109    ModifyNumber   : 5
110    Location   : TestBehaviorDiagram
111    warning attribute :
112 "Element : TestCase(name = AddNormalTestCase,
113 Location = TestContextDiagram)" and
114 "Element : TestBehaviorDiagram(name = AddNormalTestCase,
115 Location = TestBehaviorDiagram)"
116 are derived name attribute.
117
118 TestBehaviorDiagram(name = Add16bitCalculatonTestCase)
119    ModifyNumber   : 5
120    Location   : TestBehaviorDiagram
121    warning attribute :
122 "Element : TestCase(name = Add16bitOverflowTestCase,
123 Location = TestContextDiagram)" and
```

```
124 "Element : TestBehaviorDiagram(name = Add16bitCalculatonTestCase,
125 Location = TestBehaviorDiagram)"
126 are derived name attribute.
127
128 UnitTestCaseTable(name = Add16bitCalculatonTestCase)
129    ModifyNumber  : 5
130    Location  : UnitTestTable
131    warning attribute :
132 "Element : TestCase(name = Add16bitOverflowTestCase,
133 Location = TestContextDiagram)" and
134 "Element : UnitTestTable(name = Add16bitCalculatonTestCase,
135 Location = UnitTestTable)"
136 are derived name attribute.
137
138 TestConfigurationDiagram(name = Add16bitCalculatonTestCase)
139    ModifyNumber  : 5
140    Location  : TestConfigurationDiagram
141    warning attribute :
142 "Element : TestCase(name = Add16bitOverflowTestCase,
143 Location = TestContextDiagram)" and
144 "Element : TestConfigurationDiagram(name = Add16bitCalculatonTestCase,
145 Location = TestConfigurationDiagram)"
146 are derived name attribute.
147
148 TestConfigurationDiagram(name = AddNormalTestCase)
149    ModifyNumber  : 5
150    Location  : TestConfigurationDiagram
151    warning attribute :
152 "Element : TestCase(name = AddNormalTestCase,
153 Location = TestContextDiagram)" and
154 "Element : TestConfigurationDiagram(name = AddNormalTestCase,
155 Location = TestConfigurationDiagram)"
156 are derived name attribute.
157
158 UnitTestCaseTable(name = AddNormalTestCase)
159    ModifyNumber  : 5
160    Location  : UnitTestTable
161    warning attribute :
162 "Element : TestCase(name = AddNormalTestCase,
163 Location = TestContextDiagram)" and
164 "Element : UnitTestTable(name = AddNormalTestCase,
165 Location = UnitTestTable)"
166 are derived name attribute.
167
168 total Element = 15
```

解析結果 B.6: 要求 16bitCalculation を 8bitCalculation に変更した場合

```
1 fixObjectList
2 <<Requirement>>
3 Requirement(id = 2_1_1, name = 16bitCalculation)
4    ModifyNumber  : change Requirement
```

```
5     Location   : RTMDiagram
6     warning attribute :
7  "Element : Requirement(name = 16bitCalculation,Location = RTMDiagram)"
8   and
9  "Element : TestCase(name = Add16bitOverflowTestCase,
10 Location = TestContextDiagram)"
11 are derived id attribute.
12
13 ---------------------------------------------------------------------------
14 <<AbstractRequirement>>
15 ---------------------------------------------------------------------------
16 <<TestKey>>
17 ---------------------------------------------------------------------------
18 <<TestContext>>
19 TestContext(name = AddTestContext)
20    ModifyNumber   : 3
21    Location   : TestContextDiagram
22    warning attribute :
23 "Element : TestContext(name = AddTestContext,
24 Location = RTMDiagram)" and
25 "Element : TestContext(name = AddTestContext,
26 Location = TestContextDiagram)"
27 are same all attribute.
28
29 TestContext(name = AddTestContext)
30    ModifyNumber   : 1
31    Location   : RTMDiagram
32    warning attribute :
33 "Element : TestContext(name = AddTestContext,
34 Location = RTMDiagram)" and
35 "Element : TestContextDiagram(name = AddTestContextDiagram,
36 Location = TestContextDiagram)"
37 are derived name attribute.
38
39 ---------------------------------------------------------------------------
40 <<TestCase>>
41 TestCase(name = Add16bitOverflowTestCase)
42    ModifyNumber   : 4
43    Location   : TestContextDiagram
44    warning attribute :
45 "Element : TestCase(name = Add16bitOverflowTestCase,
46 Location = TestContextDiagram)" and
47 "Element : TestConfigurationDiagram(name = Add16bitCalculatonTestCase,
48 Location = TestConfigurationDiagram)"
49 are derived name attribute.
50
51 ---------------------------------------------------------------------------
52 <<UnitTestCase>>
53 UnitTestCaseColumn(name = testCheckAnalysis)
54 TargetClass   : Analysis
55 TargetMethod : analysis
56    ModifyNumber   : 6
57    Location   : UnitTestTable
```

```
58      warning attribute :
59  "Element : TestCase(name = Add16bitOverflowTestCase,
60  Location = TestContextDiagram)" and
61  "Element : UnitTestCaseColumn(name = testCheckAnalysis,
62  Location = UnitTestTable)"
63  are derived id attribute.
64
65  ----------------------------------------------------------------------
66  <<Other Model Element>>
67  TestContextDiagram(name = AddTestContextDiagram)
68      ModifyNumber  : 2
69      Location  : TestContextDiagram
70      warning attribute :
71  "Element : TestContext(name = AddTestContext,
72  Location = RTMDiagram)" and
73  "Element : TestContextDiagram(name = AddTestContextDiagram,
74  Location = TestContextDiagram)"
75  are derived name attribute.
76
77  UnitTestCaseTable(name = Add16bitCalculatonTestCase)
78      ModifyNumber  : 5
79      Location  : UnitTestTable
80      warning attribute :
81  "Element : TestCase(name = Add16bitOverflowTestCase,
82  Location = TestContextDiagram)" and
83  "Element : UnitTestTable(name = Add16bitCalculatonTestCase,
84  Location = UnitTestTable)"
85  are derived name attribute.
86
87  TestBehaviorDiagram(name = Add16bitCalculatonTestCase)
88      ModifyNumber  : 5
89      Location  : TestBehaviorDiagram
90      warning attribute :
91  "Element : TestCase(name = Add16bitOverflowTestCase,
92  Location = TestContextDiagram)" and
93  "Element : TestBehaviorDiagram(name = Add16bitCalculatonTestCase,
94  Location = TestBehaviorDiagram)"
95  are derived name attribute.
96
97  TestConfigurationDiagram(name = Add16bitCalculatonTestCase)
98      ModifyNumber  : 5
99      Location  : TestConfigurationDiagram
100     warning attribute :
101 "Element : TestCase(name = Add16bitOverflowTestCase,
102 Location = TestContextDiagram)" and
103 "Element : TestConfigurationDiagram(name = Add16bitCalculatonTestCase,
104 Location = TestConfigurationDiagram)"
105 are derived name attribute.
106
107 total Element = 9
```

解析結果 B.7: 要求 Add を MemoryAdd に変更した場合

```
1  fixObjectList
2  <<Requirement>>
3  Requirement(id = 1_1_1, name = Add)
4      ModifyNumber   : change Requirement
5      Location   : RTMDiagram
6      warning attribute :
7  "Element : Requirement(name = Add
8  ,Location = RTMDiagram)" and
9  "Element : TestCase(name = AddNormalTestCase
10 ,Location = TestContextDiagram)"
11 are derived id attribute.
12
13 ------------------------------------------------------------------------
14 <<AbstractRequirement>>
15 ------------------------------------------------------------------------
16 <<TestKey>>
17 ------------------------------------------------------------------------
18 <<TestContext>>
19 TestContext(name = AddTestContext)
20     ModifyNumber   : 1
21     Location   : RTMDiagram
22     warning attribute :
23 "Element : TestContext(name = AddTestContext
24 ,Location = RTMDiagram)"
25 and
26 "Element : TestContextDiagram(name = AddTestContextDiagram,
27 Location = TestContextDiagram)"
28 are derived name attribute.
29
30 TestContext(name = AddTestContext)
31     ModifyNumber   : 3
32     Location   : TestContextDiagram
33     warning attribute :
34 "Element : TestContext(name = AddTestContext,
35 Location = RTMDiagram)" and
36 "Element : TestContext(name = AddTestContext,
37 Location = TestContextDiagram)"
38 are same all attribute.
39
40 ------------------------------------------------------------------------
41 <<TestCase>>
42 TestCase(name = AddNormalTestCase)
43     ModifyNumber   : 4
44     Location   : TestContextDiagram
45     warning attribute :
46 "Element : TestCase(name = AddNormalTestCase
47
48 ,Location = TestContextDiagram)" and
49 "Element : UnitTestTable(name = AddNormalTestCase
50
51 ,Location = UnitTestTable)"
```

```
52 are derived name attribute.
53
54 ---------------------------------------------------------------------------
55 <<UnitTestCase>>
56 UnitTestCaseColumn(name = testCheckCalculate)
57 TargetClass   : Calculate
58 TargetMethod : calculate
59    ModifyNumber   : 6
60    Location   : UnitTestTable
61    warning attribute :
62 "Element : TestCase(name = AddNormalTestCase
63
64 ,Location = TestContextDiagram)" and
65 "Element : UnitTestCaseColumn(name = testCheckCalculate
66
67 ,Location = UnitTestTable)"
68 are derived id attribute.
69
70 ---------------------------------------------------------------------------
71 <<Other Model Element>>
72 TestContextDiagram(name = AddTestContextDiagram)
73    ModifyNumber   : 2
74    Location   : TestContextDiagram
75    warning attribute :
76 "Element : TestContext(name = AddTestContext
77
78 ,Location = RTMDiagram)" and
79 "Element : TestContextDiagram(name = AddTestContextDiagram
80 ,Location = TestContextDiagram)"
81 are derived name attribute.
82
83 TestBehaviorDiagram(name = AddNormalTestCase)
84    ModifyNumber   : 5
85    Location   : TestBehaviorDiagram
86    warning attribute :
87 "Element : TestCase(name = AddNormalTestCase
88 ,Location = TestContextDiagram)" and
89 "Element : TestBehaviorDiagram(name = AddNormalTestCase
90 ,Location = TestBehaviorDiagram)"
91 are derived name attribute.
92
93 TestConfigurationDiagram(name = AddNormalTestCase)
94    ModifyNumber   : 5
95    Location   : TestConfigurationDiagram
96    warning attribute :
97 "Element : TestCase(name = AddNormalTestCase
98 ,Location = TestContextDiagram)" and
99 "Element : TestConfigurationDiagram(name = AddNormalTestCase
100 ,Location = TestConfigurationDiagram)"
101 are derived name attribute.
102
103 UnitTestCaseTable(name = AddNormalTestCase)
104    ModifyNumber   : 5
```

```
105     Location   : UnitTestTable
106     warning attribute :
107 "Element : TestCase(name = AddNormalTestCase
108 ,Location = TestContextDiagram)" and
109 "Element : UnitTestTable(name = AddNormalTestCase
110 ,Location = UnitTestTable)"
111 are derived name attribute.
112
113 total Element = 9
114 ------------------------------------------------------------------------
```