独立行政法人情報処理推進機構 委託

2013 年度ソフトウェア工学分野の先導的研究支援事業
「形式仕様とテスト生成の部分的・段階的な活用
～探索を通したコード中心インクリメンタル型開発の支援」
成果報告書

平成 27 年 2 月

情報・システム研究機構　国立情報学研究所

本報告書は独立行政法人情報処理推進機構　技術本部　ソフトウェア高信頼化センターが実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の公募による採択を受けて情報・システム研究機構　国立情報学研究所　GRACE センター（研究責任者　石川冬樹）が実施した研究の成果をとりまとめたものである.

目次

研究成果概要

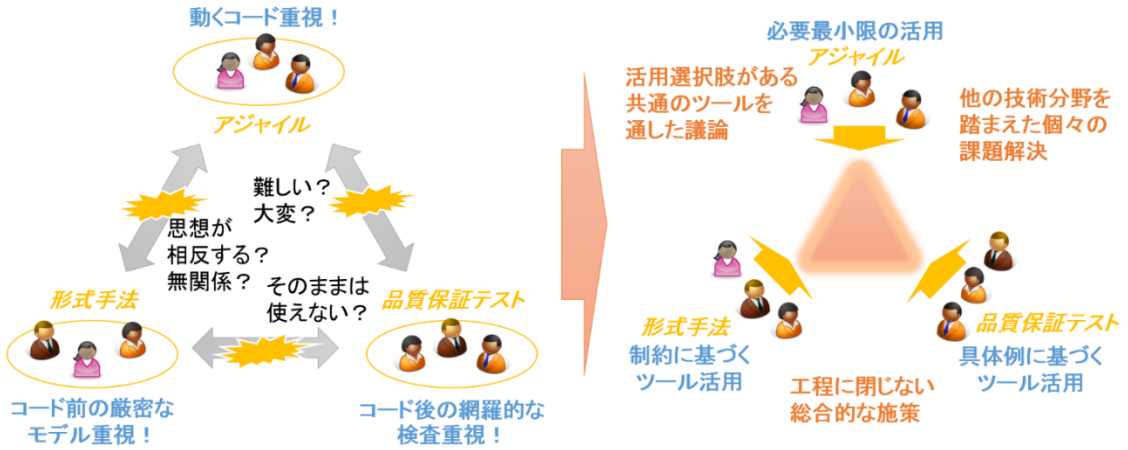
【背景・動機】

産業界において，アジャイル開発，形式手法，品質保証テストの３つの技術分野は，近年特に高い注目を集めている．一方で，それぞれの技術分野においては，産業界が実際に導入をするにあたって，それぞれの特徴に起因する課題が存在する．加えて，それぞれの技術分野に閉じた議論がなされがちで，相互補完や総合的な施策につながりにくいという課題もある．

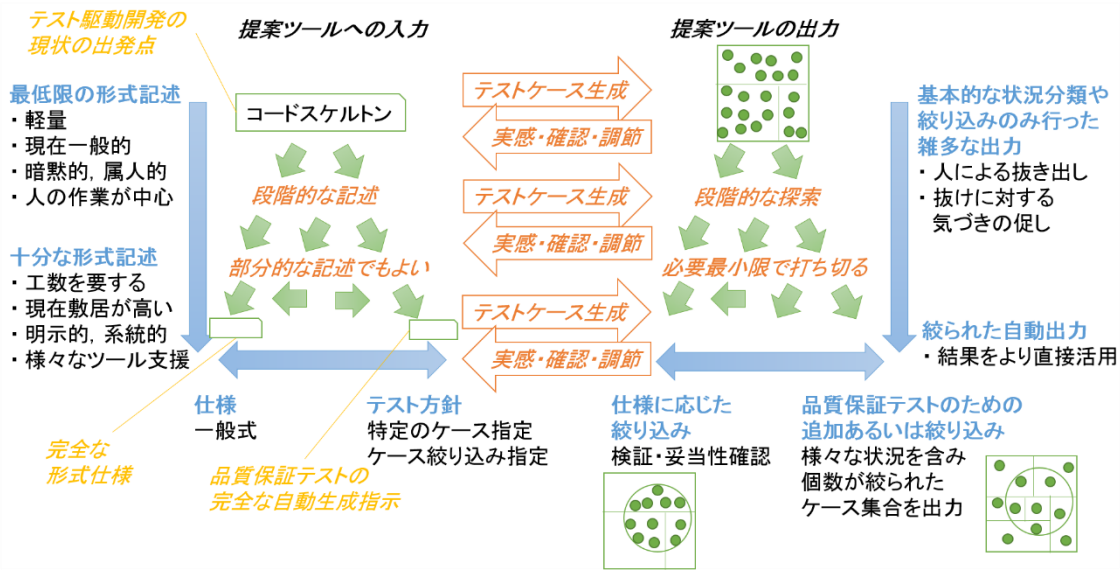
技術的な観点に関し具体的には第一に，形式仕様記述などの形式手法，モデルベーステストなど品質保証テストにおけるテスト自動生成において，仕様やテスト設計の厳密な記述を与えることが難しいという課題がある．これは工数に関する費用対効果の懸念もあれば，実感がもちにくい宣言的，抽象的な記述において意図に合った記述を行うことが論理的に難しいという側面もある．第二にその一方で，強い確信を持ちやすいテストケース（具体例）を開発の拠り所として用いるテスト駆動開発においては，妥当なテストスイート（テストケースの集合）を定めることの難しさがある．これについては，論拠となるテスト設計からテストケースを導出したり，テストケースをテスト設計と照らして確認したり，論拠となるテスト設計について議論をしたりするための支援がないということである．

【研究目標】

本研究ではこれらの課題に対応し，コードスケルトン（変数定義やメソッドシグネチャ定義）上に書き加えた断片的な仕様やテスト設計を基に，テストケースを探索，提示するツールを構築する．その際に与える記述のために，仕様，テスト設計，テストケースを混在させた形で与えることができる言語を提案する．このような言語で与えた記述に基づき，提案ツールにおいては，テスト駆動開発，形式仕様記述，テスト自動生成の原則・考え方を総合的に活用することができるようになっている．この分野横断的，融合的なアプローチについて下図に示す．



次に技術的な観点からの提案言語による記述と提案ツールの活用の考え方について下図に示す.



整数や文字列などの基礎的な知識や絞り込みパターンをツールに埋め込むことにより，コードスケルトンだけでもテストケースを探索，提示することがある程度できる（図の上部に相当）．ただし，スケルトンの情報に合致するというだけでは，雑多なテストケースが生成されてしまう.雑多であることは,抜けに対する気づきを促す点では有用でもあるが，その結果を直接利用することは難しい．このため必要に応じ仕様やテスト設計を加え，不正なケースの排除やケースの絞り込みを行う．

この結果を踏まえることにより，テスト駆動開発の拠り所となるテストを，より確信を持って設定することができる（テスト駆動開発の課題に対応する活用方法）．ここで，仕様あるいはテスト設計の記述を，段階的にさらに加えていくことも考えられる（図左下あるいは右下に進む方向）．これにより，分析や検証，テスト生成の自動化や，提示結果のより直接的な利用に十分な情報量を持つ記述に近づけていく．これにより，段階的に提案ツールの提示する結果の有用性や，既存ツール利用の可能性を高めていき，必要な結果が得られれば形式記述を打ち切ることもできる（形式仕様記述の課題やテスト自動生成の課題に対応する活用方法）．

研究期間内では,取り組み範囲を個々のデータ構造やメソッドに対するテストケース(単体テスト）に絞ることとする．個々のデータ構造やメソッドにおいて重要な場合分けを整理できれば，複数メソッドの重要な組み合わせ（シナリオ，ストーリー）も多くの場合それに起因し，導出可能であると考えられる．またテスト駆動開発などはインクリメンタル開発を想定し，複雑なシステムを多数の小さな部品に分割し，段階的に対象範囲を広げていく．本研究の取り組み範囲により，インクリメンタル開発における構築単位（増分）が十分に扱えると考えられる．なお，本研究の取り組み範囲によって基礎技術が確立されれば，メソッドの組み合わせ（シナリオ，ストーリー）に関する指示に対応するようツール

を拡張することも容易と考えられる．また，対象プログラミング言語としては，一般性が高いと思われる Java を扱う．

提案ツールを実際に利用するシナリオとしては，次の 3 つを想定する．一つの対象部品に対し，以下のうち一つの利用を行ってもよいし，複数の利用を行ってもよい．またいずれにおいても，記述を段階的に追加しながら，随時確認をすることとなる．

- アジャイル開発において，開発者が拠り所となるテスト設定を行う際に，提案ツールを用いてテストケースを生成させることにより，その設定の妥当性確認や，想定以外の例示による漏れの探索を行う．
- 従来日本語コメントなどで記述していた，あるいは暗黙であった制約記述を提案言語で与える．そして提案ツールを用いることにより，与えた制約の妥当性を確認するとともに，十分性についても把握することができる（明記した制約だけでは厳密には弱いが，明記していない点は他の開発者が自明に予測できるであろうことの確認など）．ここで，この記述と確認を段階的に繰り返すことにより，ドキュメントとして，あるいは形式検証等に十分な制約記述を得ることもできる．
- 従来，人手でテスト技法を適用し，多くの情報を暗黙としていた表計算ソフトウェアなどを用いて結果のみを記述していた品質保証テスト設計工程において，テスト設計を提案言語で与える．そして提案ツールを用いることにより，テスト設計の結果に流用できる部分的な情報を生成させることができる．加えて，様々なテストケースを生成させることにより，テスト設計の妥当性を確認するとともに，テスト設計の内容自体や記述ドキュメントにおける漏れの気づきも促される．ここで，この記述と確認を段階的に繰り返すことにより，生成されたテストケースをそのままテスト設計とする（十分なテスト設計記述を与え，テストケースは自動生成するようにする）こともできる．

加えて，提案ツールは，コミュニティにまたがる議論の促進，総合的なスキルの教育，スキル傾向などの実態調査にも有効であると考えられる．よって，提案ツールそのものの研究開発に加え，提案ツールを用いたセミナーの教材構築と実施，セミナー結果を受けてのツールの評価，および開発者の実態調査も本研究の範囲に含める．

【提案言語・ツールの概要】

まず提案言語による記述例，および提案ツールの実行例を示す．

本研究期間の範囲では，提案言語は Java プログラムに対して追加の情報を記述するものとなる．このため，Java におけるコメント内に記述を追加するスタイルをとる．これにより，標準の Java 言語としても扱うことができる（例えば Java コンパイラーに構文エラーを起こさない）記述を行うことができる．

以下に示す Java クラスでは，簡単なメソッドとして，ある人の年齢と性別（それぞれ引数 `age`，`isMale`）を受け取り，婚姻可能かどうかを判別する `canMarry` が定義されている．この Java クラスにおいて，提案言語による様々な種類のアノテーションが与えられている．

```
public class CityOffice{

  //@ ¥pre { age >= 0 && age <= 150 }

  //@ ¥post { age >= 18 || (age >= 16 && !isMale) <==> ¥result }

  //@ ¥partition:canMarry {
  //@     ¥"男可" isMale && age >= 18,
  //@     ¥"男不可" isMale && age < 18,
  //@     ¥"女可" !isMale && age >= 16,
  //@     ¥"女不可" !isMale && age < 16
  //@ }

  //@ ¥case:canMarry { ¥"例１" age == 30 && isMale && ¥result }
  //@ ¥case:canMarry { ¥"例２" age == 16 && isMale == false && ¥result }

  public boolean canMarry(int age, boolean isMale);

  //@ ¥run:canMarry "main" { }

}
```

//@ から始まる多くの行において，提案言語による追加記述（アノテーション）を行っている．条件式の記述などは Java 構文に準拠しているが，独自のキーワードに関してはバックスラッシュを付けて名前の衝突が起きないようにしている．アノテーションのうち一部のものは，フィールドにもメソッドにも付加することできる．そのようなものには，コロン記号に続いてメソッド名を明記するようになっている．メソッドのみに付加されるものは，該当メソッドの直前（他のフィールドやメソッド定義をはさまず）に記述する．

図の例におけるアノテーションそれぞれについて，その意味を以下で概観する．

- 事前条件： pre から始まるアノテーションはメソッドが正しく実行できるための前提となる事前条件を表す．図の例の場合，年齢が 0 以上 150 以下であることを示している．
- 事後条件： post から始まるアノテーションはメソッドの正しい実行結果を定める事後条件を表す．図の例の場合，戻り値（キーワード ¥result ）の真偽は，「年齢が 18 歳以上または，年齢が 16 歳以上で女性である」ことと合致することを示している．
- テスト設計： partition から始まるアノテーションはテスト設計の一種として，生成結果に含まれるべきテストケースの条件（場合分け）を表す．図の例においては，男性 18 歳以上，男性 17 歳以下，女性 16 歳以上，女性 15 歳以下という 4 つの場合分けを示している．この例では，一つのアノテーションの中に，コンマ区切りで複数の

条件を与えているほか，各条件においてバックスラッシュに続く文字列としてタグ文字列を与えている．この複数条件の記述，およびタグ文字列の記述は，他のアノテーションにおいても可能である．

- テストケース（具体例）：case から始まるアノテーションは，テストケース（具体例）を表す．図の例においては，「入力は 30 歳男性，結果は婚姻可」，「入力は 18 歳女性，結果は婚姻可」という 2 つのテストケースをタグ文字列付きで指定している．
- コマンド： run から始まるアノテーションは，提案ツールへの実行指示内容となるコマンドを表す．図の例においては，main というタグ（名前）の付けられたコマンド 1 つが定義されている．中括弧内には実行オプションを与えることができるが，今回は特に与えていない．

以上の記述に対し，提案ツールを実行した際に生成されるテストケースの一例を示す．

age	isMale	¥result	Tag
30	true	true	［男可，例１］
18	false	true	［女可，例２］
8	true	false	［男不可］
0[LowerB]	false	false	［女不可］
121	true	true	［男可］
10	false	False	［女不可］

図の例で与えた記述においては，仕様（事前条件および事後条件）は十分なものであるため，不適切な入力や出力は生成されていない．与えた場合分け，およびテストケースはすべて含まれるように生成され，それらには識別のためのタグ文字列が付加される．明示的に与えたタグのほかに，制約の形式から明確に判別できる境界値条件についてもタグが付与されている（LowerB）．

以上で用いていない構文やツール機能としては，以下のようなものが挙げられる．

- クラス内外への変数へのアクセスの宣言（読む，読み書きする）を行うことができる．無数に存在しうるクラス内外の変数については，条件に含まれる過去の宣言に含まれるかにより，当該メソッドに関連すると宣言されたときのみテストケースに含まれるようになる．
- 与えた条件を満たさない具体例（反例）の生成を行うことができる．これにより，与えた条件式の意味についてより確信を深めることができる．
- テストケースを自動生成せず，与えたテストケースがテスト設計に対して十分であるかどうかのチェックを行うことができる．

なお，上記の例はほぼ完成された記述である．本研究の特徴としては，このように完成された記述にいたる過程においてもツールを実行し，記述の意味を確認できることがある．すなわち，形式化や自動化の部分的な活用をする，あるいは十分な活用をする場合でも段階的にそこに至るようにできるようにするということである．

【提案言語・ツールの特徴】

提案言語およびツールは以下の特徴を持つ.

性質による記述と例示による記述の融合：従来形式手法，形式仕様記述においては，性質(Property)，すなわち宣言的な命題により仕様記述を行ってきた．これはテスト設計(テストスイートに対する仕様）においても同様である．一方でテスト駆動開発からの発展として，例示による仕様（Specification by Example）というキーワードも用いられている．提案言語はこれらの双方を使い分けることを可能とする．さらに，双方を両方与え互いの確認に用いることも可能になっている．

仕様，テスト設計，テストケース（具体例）の混在した記述：上記と同様のことであるが，より具体的な実現として，仕様，テスト設計，テストケースを混在させ，互いに互いの確認に用いたり，相補的に用いたりすることができるようになっている．

早いフィードバックを与えるツール：与えた制約式に対し，その式の意味を具体例にて示す．これにより，何か記述を行った際に，様々な記述をそこから積み上げる前に，すぐに行った記述の意味を確認することが可能である．すなわち，逐次意味を確認しながら，段階的に形式的な記述を構築していくことができる．また，ある一通りの式を与えないとフィードバックを得られないということはないということでもある．これはテスト駆動開発において重要な原則である一方，仕様あるいはテスト設計について部分的にのみ形式的な記述を与えた場合でもツールを活用できることを意味する．

仕様を得るタスクおよびテストケースを得るタスク双方への活用：提案言語およびツールは，テストケース生成のためのものであるが，とらえ方によっては「仕様とテストを行き来する」ためのものであるとも言える．すなわち，必ずしも生成するテストケースが成果物であるとは限らず，その生成を通して仕様記述やテスト設計記述の妥当性確認を行っているとも言える．このように提案言語およびツールは，仕様記述やテスト設計を成果物とするタスクにおいても，テストケースを成果物とするタスクにおいても利用することができる．

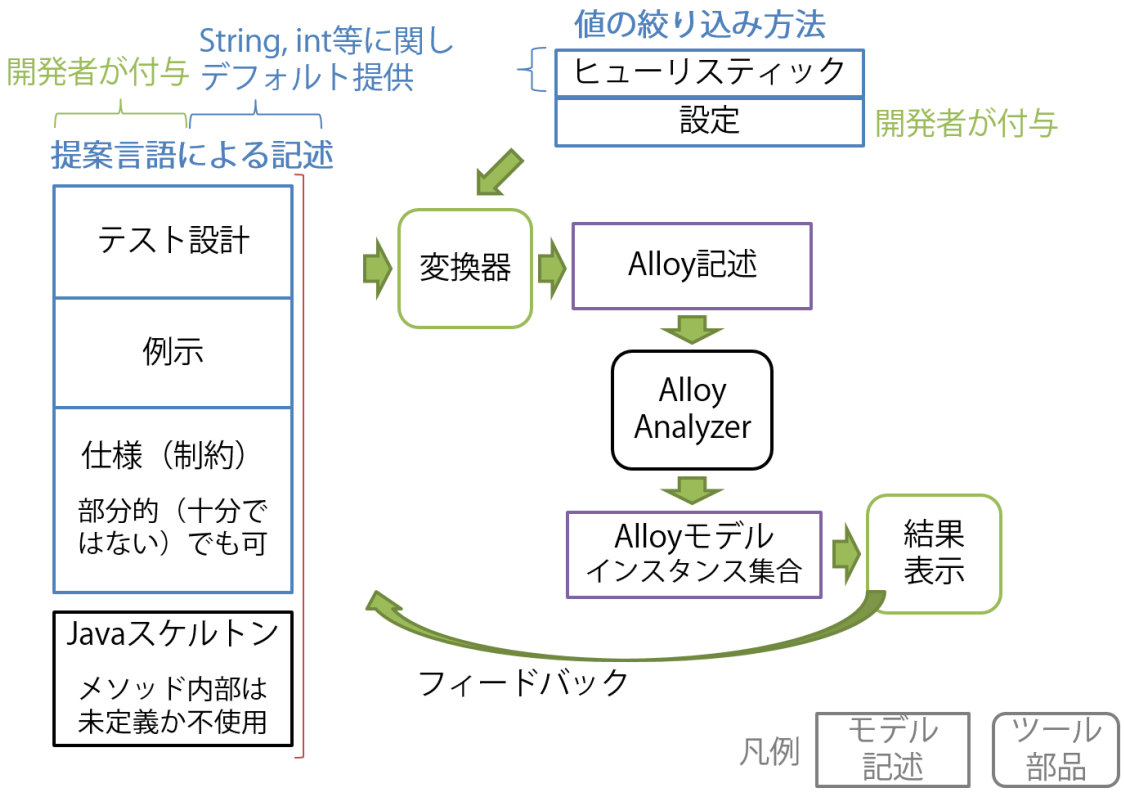
例と反例双方の確認：与えた式を満たす場合だけでなく，満たさない場合も確認することができるようにする．特に部分的な条件（弱い条件）を与えた場合には，条件を満たす例は意図に合わないものが多くなる．その場合でも条件を満たさない反例を確認することで，どういう可能性をまずは排除したのかを確認することができる．これに限らず，条件が強すぎて意図しない場合が排除されているときにそれを反例の中から見つける可能性があるなど，条件を満たさない反例の確認には意義があると考えられる．

命題に対するテストケース・テスト駆動開発：テスト駆動開発では，まだ記述していない機能や観点に対してテストケース（具体例）を与えたとき，失敗する（Red/Fail）ことが開発サイクルの出発点となる．これはテスト駆動開発・テストファーストであるかに限

らないが，プログラムと異なり仕様記述の場合，まだ記述していないことについては「どういう出力でも受け入れる」ととらえることができる．すなわち，非決定的な記述が可能である．これに対して，既存の仕様記述では不十分であることを明示するためには，あるテストケースの値が受け入れられるというだけではなく，その入力に対して唯一の受け入れられる出力を定めているということをアサーションとしたい．提案言語はテストケースが決定的であることを宣言することができ，この要求に対応している．

【提案言語・ツールにおける技術アプローチ】

技術的な取り組みによる成果物１（言語）および成果物２（ツール）に関し，構成の概要を以下に示す．



図のうち青枠の四角形は成果物１の提案言語による記述（モデル），緑枠の角丸四角形は成果物２のツール部品である．紫枠の四角形は既存言語による記述（モデル），黒枠の角丸四角形は既存のツールである．図の左に示すように，提案言語は仕様，テスト設計，例示のすべてを扱う記述能力を持ち，Java のスケルトンコード（フィールドおよびインターフェース記述）に対して付与される．与えた記述はモデル発見器への入力（Alloy）に変換し，テストケースの生成を行う．その際には，少ない記述でもより意味のある生成ができるようにヒューリスティックを加える．生成したテストケース（Alloy インスタンス）は GUI 上に表示し，それに対する記録や修正などのフィードバックを可能とする．

上述のように，提案ツールの探索部分については，SAT（satisfiability problem）ソル

バーを内部的に用いる既存ツール Alloy Analyzer を活用する［12］．このアプローチにより，抽象度の高い Alloy 言語を用いるとともに，複数の SAT ソルバーへの対応などは Alloy Analyzer に委ね，提案ツールの開発および保守の効率を高める．一方で，問題に応じた複数ソルバーの使い分け(SMT Solver の活用など)については本研究期間では範囲外とした．

現在，Xtext［18]などの DSL（Domain Specific Language）作成支援ツールが広く利用可能である．これらのツールでは，言語の文法定義や他言語への変換定義を与えれば，構文解析器や GUI エディター，言語変換器などを自動生成することができる．このため本研究でも提案ツールの効率的な構築のために活用した．

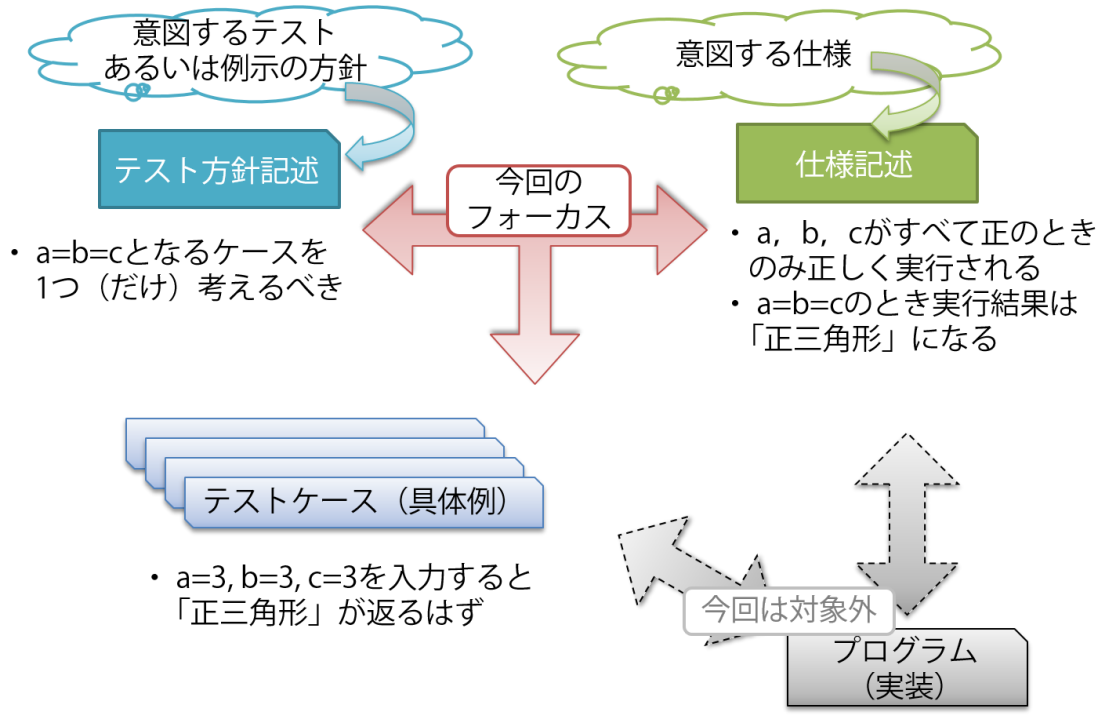
この技術アプローチにおいて想定される難しさは主に 2 点ある．第一に，必要最低限の入力を基に，テストケースを生成，提示しようとするものであるため，Alloy Analyzer により発見されるテストケースが，冗長で類似なものを多数含むなど，膨大となることが考えられる．第二に，問題によっては状態爆発により Alloy Analyzer そのものでは実行時間がかかるあるいは実行できないことがありうる．これらに本質的に対応するためには，高度で斬新な技術への取り組みが必要である場合，実用化・具体化を目指す本研究の範囲外とする．代わりに，研究セミナー内で用いるシナリオに特化することにより対応した．

【セミナーの実施】

本研究の評価のため，産業界の技術者を対象としたセミナーを開催した．研究責任者も参加する，産業界向けに様々なセミナーなどを提供している NPO 法人「トップエスイー教育センター」［39]において，セミナーの実施について提案を行い，セミナーを開催した．（本報告書執筆の段階では http://topse.or.jp/2014/11/2243 にて公開されている.）1 日のセミナーを同一の内容で 2 日実施した．開催場所の都合および，演習時のツール利用へのサポートの難しさなどから定員を設けている．メーリングリストおよび twitter/Facebook での参加募集により，両日ともに定員に達し募集を打ち切った．

本研究はテスト駆動開発，形式手法，および品質保証テストの 3 つの分野に横断した技術や議論を通し，それぞれの課題を解決しようとするものである．またそれらの分野が互いに強く関連していることを認識し，総合的な課題解決の施策に取り組んでいく姿勢を得ることを促していくことも目指している．このことからセミナーの内容は，まず上記 3 分野それぞれの導入を行い，提案言語およびツールを用いた演習を行うものとした．

さらに，それら分野の相関や観点の違いについて端的に整理されており，それに基づき気づきや議論を促進するように心がけた．具体的には，仕様，テスト設計（テスト方針），テストケース（具体例）に対し，どれに重きをおいているか，何をインプット・アウトプットと考えているかなどの観点から，各分野の位置づけや比較，相関の議論を行った．各分野の概要を述べる際，あるいは例題や演習問題などで提案ツールを用いる際に，どの記述項目を与え，どの記述項目と照らし合わせているのかなど，同じ構造の図の上で随時示すことにより，用語のぶれなどなく一つの枠組みでの対比や議論が行えるようにした．このための図を以下に示す．



終了後には以下のような内容を含むアンケートを採った．

1. 一般的な背景：業務内容
2. 本研究が論じている 3 分野それぞれに関するとらえ方：自身にとっての興味と経験の有無と程度，セミナーを受けた上での自身にとっての評価
3. 提案言語およびツールの評価：有用であると評価する特徴，期待できる活用方法，改善すべき点
4. セミナーの評価：有用であると評価する特徴，改善すべき点

このうち 3 に関する結果を以下に抜粋する．

【提案言語・ツールの利点】

提案言語およびツールについて，そのよい点をどう考えたかを調査した．これは複数選択可能な形式とした．

A. テスト駆動・形式手法・テスト技法の考え方を 1 つのツールで扱える点	22
B. 何か部分的にでも書くとすぐにツールを動かせる点	22
C. 形式手法やソルバーツールとしては取っつきやすい点	17
D. 様々なアプローチを織り込み視点を広げる・学ぶよい機会になる点	16
E. 入力内容を変え様々なタスクをある程度こなせる汎用的なツールである点	14
F. 「テスト一部自動生成」など軽い導入がしやすい点	14
G. その他	1

基本的には多くの特徴について一定数の合意を得られている．人によって受け取り方，他のツールに対して考えている問題などが異なるため，受け取り方がばらけることは当然であると考えられるが，特に重要でないという観点はなさそうである．回答項目 A，B は，本研究の特徴である「分野横断的な取り組みを支援する点」，「とにかく部分的な記述でもツールを動かし早いフィードバックが得られる点」という点であるが，特に多くの参加者に合意を得られている．D は実作業の支援というよりも意識向上・教育観点からの評価であり，ツールに関する評価としては本来二次的なものではあるものの，本研究の狙いの一つである．この観点からの有用性に合意している参加者が一定数いた．G の「その他」については，「テストケースの妥当性を確認できる，それらしいデータを出せる」ということが挙がっていた．

【提案言語・ツールの課題】

提案言語およびツールについて，どのような点を改善あるいは強化すべきかを調査した．これも複数選択可能な形式とした．

A. 実行速度の改善	18
B. UI の洗練(記述の自動補完・情報の表示など)	14
C. UML・Excel など広く使われている表記との相互変換	8
D. その他	8
E. 直接扱える問題の拡張(操作列や部品の統合など)	3
F. 特定の利用法に絞って機能・UI の特化	3
G. 出力結果の数や多様性などのチューニング	2

A では，性能が最も改善を要する点だということが指摘された．この点については今回の実施しなかった様々なソルバーの使い分け，より踏み込んだ実装の工夫などが必要になる．一方で関連した懸念であった G についてはそれほどの指摘はなかった．これはセミナーで暑かった遠州が比較的簡単なものであったからと考えられる．B，C は一般的にツールの完成度として求められるもので，これは工数をかければ達成できることであると考えられる．E は，本研究の期間内で扱う対象が狭すぎないかという観点であったが，特に大きな問題とは考えられなかったようである．F は本研究の特徴に関するトレードオフとして，「いろいろできるが，個別のタスクでやりたいことに踏み込んでできない」ということが懸念としてあったが，それほど問題とは考えられなかったようである．D の「その他」については，ドキュメントの充実や，利用環境の充実などがあった．多くは B，C と同様に，工数をかけてしっかり取り組むというものであった．ただし，ソフトウェアモデル検査やランダムテストとの融合，Web での提供など，さらなる高度なツール化のアイディアなどもあったため，参考にしていきたい．

【提案言語・ツールの利用法】

提案言語およびツールについて，どのような目的・タスクでの利用が効果的であるかに

ついて調査を行った．これも複数選択可能な形式とした．

A. 中堅者の視点を広げる教育・概念整理・意識向上	15
B. テスト駆動開発におけるテストケースの整理や議論	14
C. 初心者の基礎的な考え方・技術への入門・教育	12
D. ドメイン分析や仕様化における制約の洗い出しや確認	11
E. 品質保証におけるテスト設計の補助	10
F. 特定なタスクによらず個人でのロジック整理・確認	9
G. プログラムに対する（自動処理できる）定型コメント	7
H. 既存の形式手法を使う準備としての記述の厳密化や確認	7
I. 論理的にややこしい問題への手軽なソルバー適用	4
J. その他	1

A，C は意識向上・教育観点からの評価であり，ツールに関する評価としては本来二次的なものではあるものの本研究の狙いの一つである．初心者向けか中堅者向けかで意見が分かれているものの，いずれにしても非常に高い評価が得られていると考えられる．

B，D，E，H は特定のタスクに関する支援の有用性である．形式手法を使うという前提の H は数が少ないが，それぞれの利用法について一定の評価が得られている．特に，今回のセミナー参加者の多くがアジャイル開発・テスト駆動開発を実践しているが，それらの参加者が B を選んでいる点はよい評価であると見なせる．F，G，I は特定のタスクというよりも，電卓や Excel のように汎用的な小道具として用いるような方向性である．この方向でも一定の評価が得られている．J の「その他」はシンボリック実行によるテストとの組み合わせを挙げており，高度な技術である．

以上のように，参加者のそれぞれが，自分なりの使い方をイメージすることができる，柔軟性が高いツールであるという解釈ができる．

【まとめと今後の取り組み】

本研究の根幹は，本質的には表裏一体である仕様やテスト設計に関する一般規則・性質・命題と，テストケース・シナリオ・具体例とを照らし合わせることの支援を実現したことにある．これにより 1.3 にて述べたように，前者に主眼をおく分野やタスク，後者に主眼をおく分野やタスクの支援を行う意義が得られると期待される．アンケート結果（3.5.2）にも見られるように，提案言語・ツールの活用方法は，開発者の背景や想いに応じて様々なものを挙げることができ，要求仕様の洗練，テスト駆動開発におけるテスト設計の補助，単体テスト工程での品質保証テスト設計補助など，組織・プロジェクト・課題意識に応じた活用ができるようになっている．

また本研究は同時に，一見異なる方向性として捉えられがちな分野が本質的につながっていることを示すものである．このため，意識向上や教育にも有効性が高いと考えられる．このこともアンケート結果（3.5.2）に示されている．

本研究では技術の研究開発だけでなく，産業界の技術者を対象としたセミナーを構築し，

実施した．これにより，技術の実証評価を行うとともに，意識向上や教育の観点から今すぐ産業界に展開できる成果を得ることができた．

ここまで述べたように，本研究では以下の原則・アプローチを実現する言語およびツールを構築した．

- 性質による記述と例示による記述の融合
- 仕様，テスト設計，テストケース（具体例）の混在した記述
- 早いフィードバックを与えるツール
- 仕様を得るタスクおよびテストケースを得るタスク双方への活用
- 例と反例双方の確認
- 命題に対するテストケース・テスト駆動開発

またセミナーという形で産業界の技術者に試用してもらい，その有用性に関する評価を得た．このことには非常に大きな意義があると考えている．

今後は，課題となった性能の向上や，ツールの完成度の向上，セミナーの継続的な開催および拡張などに引き続き取り組んでいく．

1 研究の背景および目的

1.1 背景

現在，ソフトウェアは社会における様々な側面に対し，価値を生みだす基盤としてますます重要となっている．これに伴い，複雑化・大規模化するソフトウェアの高品質・高信頼性の確保や，ツールを活用した開発・保守プロセスの生産性向上等，ソフトウェア工学に求められる役割も大きくなってきている．

産業界においては，(ソフトウェア開発の現場にとって) 新しいソフトウェア工学手法を学び，整備し，採り入れようとする試みが盛んである．アジャイル開発，形式手法，品質保証テストの 3 つの技術分野は，近年特に高い注目を集めている．一方で，それぞれの技術分野においては，産業界が実際に導入をするにあたって，それぞれの特徴に起因する課題が存在する．加えて，それぞれの分野に関するコミュニティに閉じた議論がなされがちで，相互補完や総合的な施策につながりにくいという課題もある．以下に各技術分野の概要とその分野における課題について述べる．

【テスト駆動開発やその発展系に関する背景】

アジャイル開発は，迅速性，適応性を重視した開発アプローチの総称である [1]．アジャイル開発には顧客との対話など様々な側面が含まれるが，開発の技術的側面としては，テスト駆動開発やその発展系（ビヘイビア駆動開発，受け入れテスト駆動開発など）が用いられることが多い．これらの開発手法では，テストケースをまず定義し，それらを開発の目標を示す拠り所として見なし定め開発を進めていく小さなサイクルを繰り返す [2] [3] [4]．これらの開発手法の利点は第一に，小さなサイクルで素早くフィードバックを得ながら進めていくことにある．言い換えると，検証や妥当性確認をせずに仕様の記述あるいはコーディングを繰り返し，うまくいかない原因の追及が難しくなったり，大きな手戻りが発生してしまったりすることを避けるようにする．第二に，“Tests as Documents”や“Specification by Example”といったキャッチフレーズに示されるように，テストケースは，開発の目標あるいは仕様を端的に説明する具体例となる．様々な状況を一括して述べるような条件分岐や命題など一般的な言い回しでは，誤解や思い込みなどがありうるが，具体例により，関連するステークホルダー間の議論を誘起し，確信を深めていくことにつながる.現在テスト駆動開発やその発展系に対する支援としては,テストケースを記述し，実行，管理するフレームワークが中心となっている．Java に対する JUnit など，XUnit と総称される一般的なテストフレームワークのほか，より文書としての記述方式を意識したフレームワークが注目されている (Cucumber [5] など).

しかし，テスト駆動開発の利点は，拠り所となるテストスイートの設計を妥当に行うことが前提となっている．アジャイル開発の原則に則れば，その設計は探索的に，インクリメンタルに行ってもかまわず，そのための議論も行われていることが期待される．しかし多くのプロジェクトにおいては，最終的には技術的な原則や指針を踏まえ，テストケースに対してある意味での網羅性を担保すテスト設計を明確にし，その設計を満たすテストスイートが得られているようにする必要があると考えられる．この点に対して現状では手法やツールによる支援が十分に考えられていない.

【形式手法に関する背景】

形式手法は，数理論理学に基づいた記述や検証の手法を活用する開発アプローチの総称である．特に，上流の仕様や設計の記述を形式的に行うことにより，曖昧さを排除するとともに，様々なツールによる分析や検証を可能とする．国内では，インタプリタ実行を通し仕様の検証を行うVDMがよく知られている［6］．その他より強力な検証手段として証明や段階的詳細化を用いるBメソッド［7］やEvent-B［8］，プログラムコードに対し仕様記述を付加する言語やそれらの記述を基に検証を行うツール（JML［9］やFrama-C［10］）など，様々な手法やツールがある［11］．共通して重要な点としては，仕様，言い換えると検証の対象となる性質を，実現詳細を捨象し宣言的に，命題の形式で記述することである．

しかし，上流工程にて厳密な仕様を記述しツールによる検証を行うことに対する工数の懸念がある．これは手戻りを防ぐことによる費用対効果が大きい場合もあると考えられるが，あらゆるプロジェクトや，プロジェクト内のあらゆるコンポーネントに対してそうなるわけではない．加えて，宣言的に，十分な仕様記述を行うことは難しい．暗黙の了解や，厳密な記述の意味と意図とのずれなどにより，命題の記述はしばしば誤ったもの，特に制約が弱い（条件十分でない）ものになりがちである．これらに関しては，現在の開発者や組織が十分に経験を積んでいないという側面もある．Alloyによるモデル発見は，仕様記述の妥当性確認や段階的な記述に適しているが，集合と関係に関する非常にプリミティブな語彙を用いる必要があり，一般の開発者には扱いが難しい［12］［13］．確信を持って記述を行ったり，より手軽に活用をしたりすることの支援が必要である．

【品質保証テストの設計支援・自動生成に関する背景】

品質保証テストは，製品として出荷されるソフトウェアに対する品質保証のために行われるテストの総称である．このうち，単体テストなど単純だが系統的・網羅的であるべきものについては，効率化，品質の安定化のため自動生成（設計支援）手法も議論されている（モデルベーステストなど）．例えば起きえない入力の組み合わせ（禁則）など，仕様の中から関連する重要な情報を抽出し，それに基づいて自動的にテスト設計を生成することができる．これにより，人手でテストケースを設計し，Excelやテストプログラムコードを書き込むような工数を削減するとともに，誤りなく，ある基準で系統的・網羅的なテストケースを得ることができるようにする．現在利用可能なツールとしては，原因結果グラフに基づくテスト，状態遷移テスト，組み合わせテストなど，特定の手法を実現するツールが多い［14］［15］［16］［17］．

しかし，こういったツールの活用のためには，仕様およびテスト設計の十分な記述（機械処理できる何かしらの形式言語によるもの）が必要となる．このことに対する工数への懸念や難しさを解決する必要がある．特に，実際には，特定の手法に寄らず，問題に応じて様々な手法の考え方や，その問題ならではのヒューリスティックを交えて柔軟に適切なテスト設計を扱える必要がある．費用対効果が大きい部分に絞っての自動化の活用や，テスト設計に対する確信を高めるための支援が必要である．

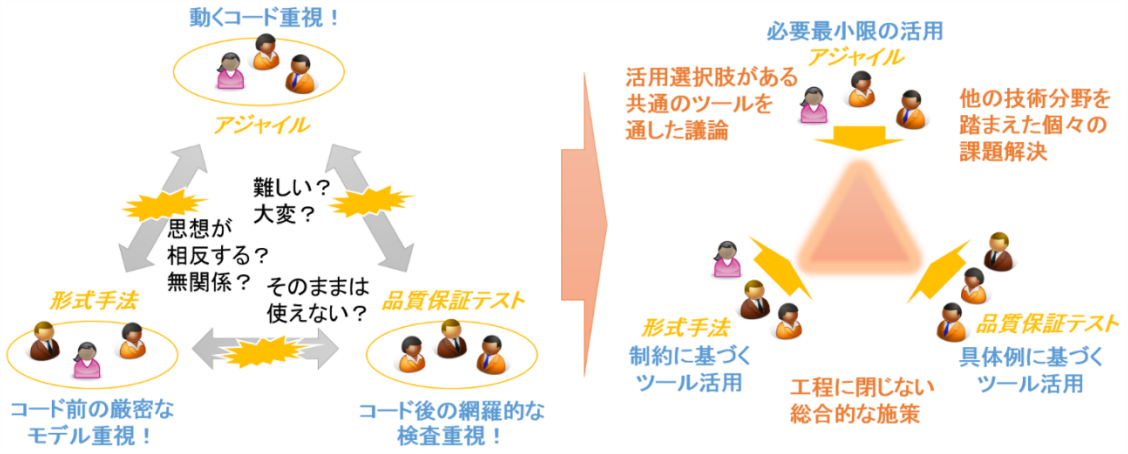


図 1-1 取り組みアプローチ

【上記分野にまたがる背景】

以上で述べた課題は各技術分野の特徴に起因するものであるため，他の技術分野のアプローチを反映して解決できる可能性もある．しかし，個々の技術分野におけるコミュニティでは，それぞれの典型的な手法，ツールを取り上げ，それぞれに閉じた議論がなされることが多い．このため，共通して用いることができる手法やツールを軸とするような，横断的な解決策の議論は活発的にはされていない．

1.2 研究課題

(1) で挙げた各技術分野の課題に対して，他の技術分野のアプローチも踏まえ，取り組むべき研究課題を示す．

【課題－１】

アジャイル技術分野においては，拠り所となるテスト設定を妥当に行うことの難しさを解決する必要がある．これに対し，形式手法技術分野と品質保証テスト技術分野を踏まえると，制約や場合分けの系統的な検討や，確認や気づきのための自動探索の活用が望ましい．ただし，アジャイル技術分野では，コード構築前のモデル記述に時間をかける形式手法や，コード構築後の品質保証に時間をかけるテスト技術そのものは，動くコードを第一とする方針に反するとも見なされ，直接活用することはできない．このため，よりアジャイル技術分野に合致するよう，必要最低限の利用ができることが必要である．

【課題－２】

形式手法技術分野においては，厳密な仕様を十分に記述することに対する工数の懸念や難しさを解決する必要がある．これに対し，アジャイル技術分野と品質保証テスト技術分野を踏まえると，必要最低限の活用や，テストケースのように一般式ではなく具体例に基づく活用ができることが望ましい．テストやコードに関する検討のために必要最小限の制約記述を都度与えたり，場合分け観点や具体例などより弱い記述を与えたりした場合でも，

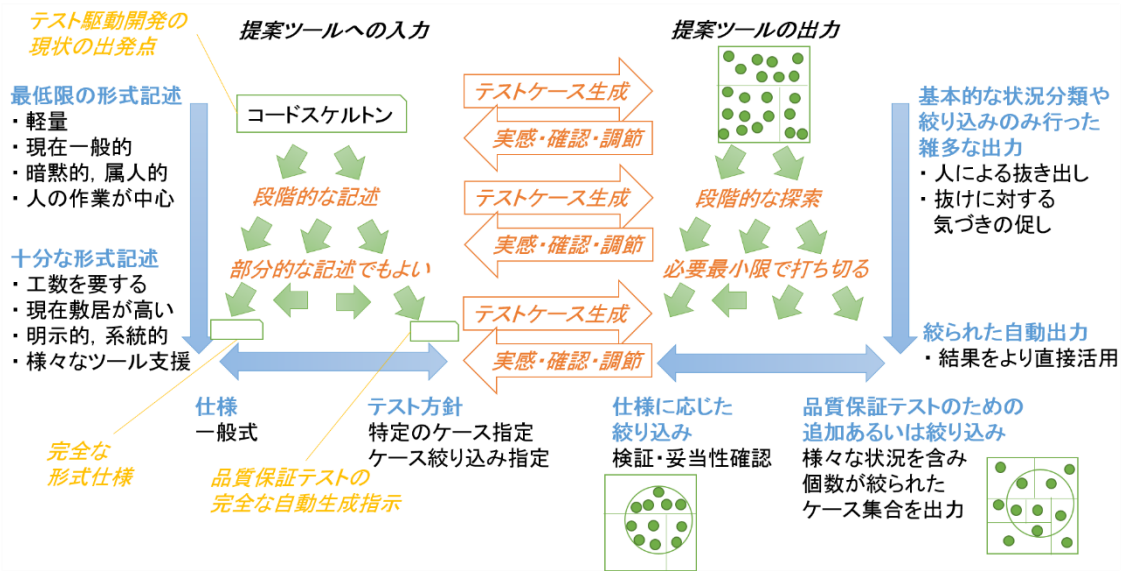


図 1-2 提案ツールの概要

ツールを用いた活用を可能とすることが必要である．

【課題－３】

品質保証テスト技術分野では，様々なテスト技法が盛んに学習，議論されている．しかし，自動化のために十分な指定を与えることの工数への懸念や難しさを解決する必要がある．これに対し，アジャイル技術分野と形式手法技術分野を踏まえると，必要最低限の活用ができたり，ツールに基づいた支援ができたりすることが望ましい．ただし，品質保証テスト技術分野では，既成のテスト生成ツールが注目されており，現状の表形式等での整理に加えた必要最低限の活用や，厳密な制約記述に基づくツールの活用は，議論されておらず，実際にツールもない．このため，必要最小限のテスト設計を都度与えたり，制約記述に基づく探索を活用したりして，ツールを用いた活用を可能とすることが必要である．

【課題―総合】

以上の課題に挙げたように，各技術分野における課題を他の技術分野のアプローチを反映して解決する必要がある．以上の個々技術分野の課題の解決に寄与するような共通ツールを提供することにより，広い視点からの問題解決やそのための議論を促進する必要がある．以上に示した課題への取り組みアプローチを図 1-1 に示す．

【研究内容】

本研究ではこれらの課題に対応し，コードスケルトン（変数定義やメソッドシグネチャ定義）上に書き加えた断片的な仕様やテスト設計を基に，テストケースを探索，提示するツールを構築する．このうち，課題－総合に対応するために，「技術分野横断的に，各技術分野の技術を活用し，各技術分野に応用できるような共通ツールを提供することが，他技術分野への興味や気づきを促す」ことを仮説として期待する．提案ツールは，各技術分野の技術を活用し，各技術分野に応用できる共通ツールとなるようにするとともに，評価の

過程で仮説の検証も行う．

提案ツールの概要を図 1-2 に示す．整数や文字列などの基礎的な知識や絞り込みパターンをツールに埋め込むことにより，コードスケルトンだけでもテストケースを探索，提示することがある程度できる（図の上部に相当）．ただし，スケルトンの情報に合致するというだけでは，雑多なテストケースが生成されてしまう．雑多であることは，抜けに対する気づきを促す点では有用でもあるが，その結果を直接利用することは難しい．このため必要に応じ仕様やテスト設計を加え，不正なケースの排除やケースの絞り込みを行う．

この結果を踏まえることにより，テスト駆動開発の拠り所となるテストを，より確信を持って設定することができる（課題－１に対応する活用方法）．ここで，仕様あるいはテスト設計の記述を，段階的にさらに加えていくことも考えられる（図左下あるいは右下に進む方向）．これにより，分析や検証，テスト生成の自動化や，提示結果のより直接的な利用に十分な情報量を持つ記述に近づけていく．これにより，段階的に提案ツールの提示する結果の有用性や，既存ツール利用の可能性を高めていき，必要な結果が得られれば形式記述を打ち切ることもできる（課題－２，課題－３に対応する活用方法）．提案ツールは，アジャイル，形式手法，品質保証テストの技術分野でのアプローチを横断的，融合的に活用するものである（課題－総合）．

研究期間内では,取り組み範囲を個々のデータ構造やメソッドに対するテストケース(単体テスト）に絞ることとする．個々のデータ構造やメソッドにおいて重要な場合分けを整理できれば，複数メソッドの重要な組み合わせ（シナリオ，ストーリー）も多くの場合それに起因し，導出可能であると考えられる．またテスト駆動開発などはインクリメンタル開発を想定し，複雑なシステムを多数の小さな部品に分割し，段階的に対象範囲を広げていく．本研究の取り組み範囲により，インクリメンタル開発における構築単位（増分）が十分に扱えると考えられる．なお，本研究の取り組み範囲によって基礎技術が確立されれば，メソッドの組み合わせ（シナリオ，ストーリー）に関する指示に対応するようツールを拡張することも容易と考えられる．また，対象プログラミング言語としては，一般性が高いと思われる Java を扱う．

提案ツールを実際に利用するシナリオとしては，次の 3 つを想定する．一つの対象部品に対し，以下のうち一つの利用を行ってもよいし，複数の利用を行ってもよい．またいずれにおいても，記述を段階的に追加しながら，随時確認をすることとなる．

- アジャイル開発において，開発者が拠り所となるテスト設定を行う際に，提案ツールを用いてテストケースを生成させることにより，その設定の妥当性確認や，想定以外の例示による漏れの探索を行う．
- 従来日本語コメントなどで記述していた，あるいは暗黙であった制約記述を提案言語で与える．そして提案ツールを用いることにより，与えた制約の妥当性を確認するとともに，十分性についても把握することができる（明記した制約だけでは厳密には弱いが，明記していない点は他の開発者が自明に予測できるであろうことの確認など）．ここで，この記述と確認を段階的に繰り返すことにより，ドキュメントとして，あるいは形式検証等に十分な制約記述を得ることもできる．
- 従来，人手でテスト技法を適用し，多くの情報を暗黙としていた表計算ソフトウェアなどを用いて結果のみを記述していた品質保証テスト設計工程において，テスト設計

を提案言語で与える．そして提案ツールを用いることにより，テスト設計の結果に流用できる部分的な情報を生成させることができる．加えて，様々なテストケースを生成させることにより，テスト設計の妥当性を確認するとともに，テスト設計の内容自体や記述ドキュメントにおける漏れの気づきも促される．ここで，この記述と確認を段階的に繰り返すことにより，生成されたテストケースをそのままテスト設計とする（十分なテスト設計記述を与え，テストケースは自動生成するようにする）こともできる．

加えて，提案ツールは，コミュニティにまたがる議論の促進，総合的なスキルの教育，スキル傾向などの実態調査にも有効であると考えられる．よって，提案ツールそのものの研究開発に加え，提案ツールを用いたセミナーの教材構築と実施，セミナー結果を受けてのツールの評価，および開発者の実態調査も本研究の範囲に含める．

1.3 研究の意義

アジャイル，形式手法，品質保証テストはいずれも近年産業界からの注目が高く，1.1で挙げた各技術分野の課題（課題－１・２・３）を解決することは非常に重要と考えられる．一方で，各技術分野における特徴への印象が強い故に，その限界や排他性を短絡的，二元的に判断してしまうようなことも多い（課題－総合）．このような思い込みは，実際に求められる様々な基礎・原則の学習機会や，それらを踏まえた総合的な施策打ち出しを阻害してしまう．このため，こういった思い込みを排除し，技術分野にとらわれない問題解決を促していくことは重要である．その際，「とりあえず動かしてみて実感し，随時追記すればよい」というフィードバックの早いツールを提供することも重要である．

本研究は様々な技術に関連する．総括すると，形式記述や自動化の取り組みは，十分，完全な記述から，いかに高度な分析ができるかを追求している．しかし実際には，十分，完全な形式記述を与えることは工数の懸念や難しさがあり，また分析が高度になるほど精度や例外の問題が生じる．このため，人手での作業や判断を支援すること，より手軽に導入できるようにすることが重要である．本研究は，アジャイルの思想も踏まえて，仕様記述とテスト設計記述を部分的でも活用できるようにすること，それにより現状の人手での作業に対し形式記述と自動化の活用程度を選べるようにすることが，特に独創的な点である．

本研究の成果により，アジャイル，形式手法，品質保証テストの技術分野において，1.1にて挙げた課題の解決を行うことができる．

アジャイル技術分野におけるテストとコードを中心とした開発手法（テスト駆動開発など）において，拠り所となるテスト設定を探索し，より系統的に，より高い確信を持って進めることができるようになる．この際にはあくまで，動くコードとそれを導くテストという目的に対して必要最低限な範囲で，追加の記述を行えばよい．より大きな視点では，テストの自動実行ツールのみが多数提供され，「優秀な開発者（多能工）が柔軟に進める」とされがちなアジャイルに対し，テスト設定時点での工学的な支援を提供する．これにより，より広い開発者による，組織的なアジャイルの適用促進の一助となる．

形式手法技術分野において，現状のプロセスに対し大きな作業追加を伴うことなく，よ

り手軽に活用する選択肢が提供される．形式手法に対する国内産業界からのニーズの大半は，厳密，明示的，系統的な仕様記述による品質一定化・属人性排除であると考えられる．この観点では，VDM や JML などの記述よりもさらに手軽に，重要な部分に絞って形式記述を活用することができるようになる．一方，形式検証という観点はコード本体を扱わない本研究の対象外である．しかし，検証において最重要となる検証項目（仕様）の記述と妥当性確認を，実感を通して段階的に行うことができるようになる．

品質保証テスト技術分野において，結果としてのテストケースを書き記すのではなく，テスト設計（仕様）を抽象的に明記し，それに従い系統的にテストケースを導出することが促進される．特に単純なテスト技法については，自動化・ツール支援を促進するとともに，結果ではなく方針（仕様）を明記し，変更対応に備えるべきと考えられる．加えて，単に自動化するだけでなく，例示により品質保証テストに関する試行錯誤や妥当性確認が行いやすくなるほか，テストケースの一部情報のみを自動生成するような手軽な利用も可能となる．

これらの課題解決は，下記 2 つの方向性に対し，柔軟にバランスをとった形で実践することができる．

- 仕様などコード以前の成果物，および高度な形式記述や自動化・ツール支援の重視
- 動くコードとそのためのテスト，および現状からの移行コスト低減や，人による妥当性確認を交えた作業の重視

加えて，上記のアジャイル、形式手法、品質保証テストの技術分野に対し横断的，融合的なアプローチを具体的に示すことにより，技術分野に閉じない視点の重要性を強く促す．すなわち，各技術分野において代表的で特徴が強い（強すぎる）手法そのものを見て「関係ない」と見なすことなく，一見異なる技術分野でも，原則となるアイディアは必要，有用，手軽に活用であることへの気づきと探求を促す．

特に，具体的なセミナー教材を提供することにより，演習を通して異なる技術分野の原則や，その重要性を学ぶことができるようになる．これにより，技術分野の名称など表面的な観点から閉じることなく，より総合的なスキルを身につけることができるようになる．このセミナーでは，同一の教材に基づき，上流工程担当者，プログラマー，品質保証担当者（テスター）など，様々な背景・業務を持つ開発者に対し用いることができる．

加えてこのように，同一の教材を様々な開発者に取り組ませるため，探索能力，形式化能力，一般式記述能力，テスト観点指摘能力など，様々なスキルに関する実態調査も行えるようになる．また本研究の期間内においても，そのような実態調査を行い，現状のソフトウェア業界における教育課題や，背景・業務に応じ適用すべき教育などに関する示唆が得られることが期待される．

2 実施内容

2.1 研究アプローチ

2.1.1 研究の全体像

研究課題 1.2 で述べた課題を踏まえ，本研究で取り組む成果物 3 つと，それらへの取り組み全体像および方針を述べる．

【成果物１】　仕様とテスト設計の断片付加言語

Java によるコードスケルトン（変数・メソッドシグネチャ定義）に対して，仕様やテスト設計の断片を付与する記述言語を定義する．仕様については，一階述語論理を用いた不変条件，事前・事後条件の記述能力に対し，場合分けの存在のみ示唆したり，一部の場合のみ言及したりするなど断片的な記述を行えるよう文法を改変，拡張する．テスト設計については，入力や出力に関する同値クラスの定義，テストに用いるべき値，テストの絞り込み方針など，一般的な方針を部分的に指定する記述能力を持たせるようにする．

【成果物２】　テストケース探索・提示ツール

成果物１の言語による記述とコードスケルトンを入力し，考えられるテストケースを探索，提示するツールを構築する．ここでのテストケースとは，データ値の構築（コンストラクタ）を含めて，メソッドに対する単体テストを扱う．事後条件が与えられていない場合は，入力となる引数値や状態値の組により構成される．事後条件が与えられている場合は，期待される結果（戻り値と状態値）も含まれる．また，Java の基本型・基本ライブラリについて,成果物１の言語によるデフォルト記述や,結果の絞り込みパターンを定義し，コードスケルトンだけでも提案ツールを動作させることができ，単純な場合はそれだけで活用ができるようにする．

【成果物３】　セミナーを通した評価

成果物１の言語，成果物２のツールに関するセミナー教材を構築し，背景・業務の異なる開発者を対象として実施する．この際，テスト駆動開発，形式手法，品質保証テストの 3 つの観点に基づく内容を含める．このセミナーを通し，提案ツールの有効性と有用性を評価する．加えて，背景・業務の異なる開発者のスキル傾向などの実態調査にも活用できることを評価し，またセミナー参加者を対象とした調査も行う．

【取り組みの全体像と方針】

技術的な取り組みによる成果物１（言語）および成果物２（ツール）に関し，構成の概要を図 2-1 に示す．図のうち青枠の四角形は成果物１の提案言語による記述（モデル），緑枠の角丸四角形は成果物２のツール部品である.紫枠の四角形は既存言語による記述(モデル)，黒枠の角丸四角形は既存のツールである．図の左に示すように，提案言語は仕様，テスト設計，例示のすべてを扱う記述能力を持ち，Java のスケルトンコード（フィールドおよびインターフェース記述）に対して付与される．与えた記述はモデル発見器への入力

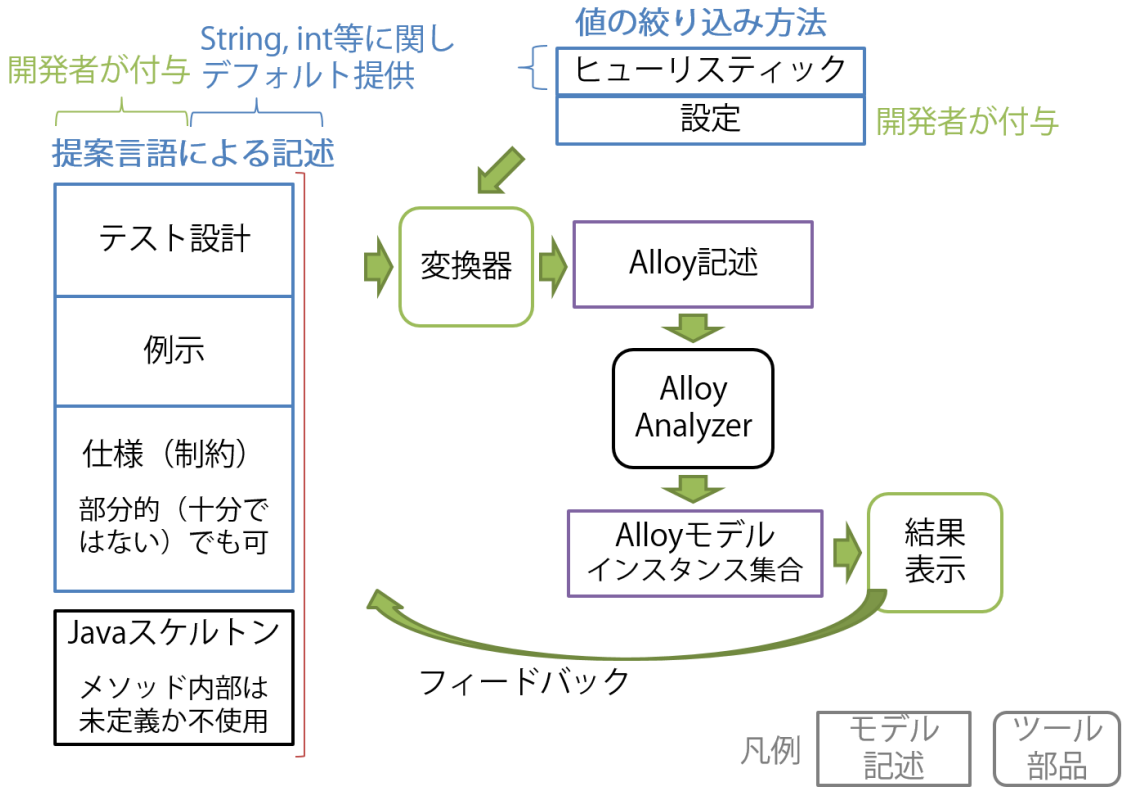


図 2-1 取り組み全体の構成

（Alloy）に変換し，テストケースの生成を行う．その際には，少ない記述でもより意味のある生成ができるようにヒューリスティックを加える．生成したテストケース（Alloy インスタンス）は GUI 上に表示し，それに対する記録や修正などのフィードバックを可能とする．

上述のように，提案ツールの探索部分については，SAT（satisfiability problem）ソルバーを内部的に用いる既存ツール Alloy Analyzer を活用する［12］．このアプローチにより，抽象度の高い Alloy 言語を用いるとともに，複数の SAT ソルバーへの対応などは Alloy Analyzer に委ね，提案ツールの開発および保守の効率を高める．一方で，問題に応じた複数ソルバーの使い分け(SMT Solver の活用など)については本研究期間では範囲外とする．

現在，Xtext［18]などの DSL（Domain Specific Language）作成支援ツールが広く利用可能である．これらのツールでは，言語の文法定義や他言語への変換定義を与えれば，構文解析器や GUI エディター，言語変換器などを自動生成することができる．このため本研究でも提案ツールの効率的な構築のために活用する．

本研究において想定される難しさは主に 2 点ある．第一に，必要最低限の入力を基に，テストケースを生成，提示しようとするものであるため，Alloy Analyzer により発見されるテストケースが，冗長で類似なものを多数含むなど，膨大となることが考えられる．第二に，問題によっては状態爆発により Alloy Analyzer そのものでは実行時間がかかるあるいは実行できないことがあると考えられる．これらに本質的に対応するためには，高度で斬新な技術への取り組みが必要である場合，実用化・具体化を目指す本研究の範囲外とす

る．代わりに，研究セミナー内で用いるシナリオに特化することにより対応する．

本公募が原則 1 年の取り組みであることを踏まえ，1 年目で根幹となる基礎技術は実現するようにする．2 年目は，セミナーを通した評価に必要な教材の準備，セミナー実施，および研究評価を中心に行う．1 年目の取り組みの最初には，本質的な表現能力や探索能力の決定に集中するため，Alloy 言語および Alloy Analyzer を直接用いて検討を行う．成果物２の基本型・基本ライブラリの範囲に絞って，提案ツールの機能と利用シナリオに関する検討と，Alloy 上での実現確認を行う．その後，それを開発者が実際に使うための，言語およびツールの構築や，多様なシナリオに対応するための言語拡張を検討する．

2.1.2 関連するこれまでの研究について

本研究は以下で述べるように様々な技術に関連する．総括すると，形式記述や自動化の取り組みは，十分，完全な記述から，いかに高度な分析ができるかを追求していることが多い．しかし実際には，十分，完全な形式記述を与えることは工数の懸念や難しさがあり，また分析が高度になるほど精度や例外の問題が生じる．このため，実際の開発現場に対しては，人手での作業や判断を支援すること，より手軽に導入できるようにすることが重要である．本研究においては，アジャイル開発の思想も踏まえて，仕様記述とテスト設計記述を部分的でも活用できるようにしたこと，それにより現状の人手での作業に対し形式記述と自動化の活用程度を選べるようにしたことが，特に独創的な点である．

以上の取り組みに対し，セミナーを通して産業界の開発者・研究者からの評価を得たことも重要である．アジャイル，形式手法，品質保証テストについては，それぞれ非常に盛んにセミナーなどが開催されているが，それらを横断的に議論したり，技術分野で区切ることなく実感をもって視点を広げたりするセミナーなどは，提案者らの知る限り提供されていない．

以下では技術的な側面についてより詳細に論じる．

【関連研究－アジャイル開発】

アジャイル開発に関してはその宣言など，思想や人による柔軟な取り組みの進め方はよく知られている［1］．しかし具体的な支援としては，テスト駆動開発，ビヘイビア駆動開発，受け入れテスト駆動開発といったパラダイムに基づき，拠り所となるテストをどういう形式で記述し，自動実行するかに関する取り組みがほとんどである（JUnit，Cucumber 等のフレームワーク）［2］［3］［4］［5］．アジャイル開発の思想を踏まえつつ，すなわち従来の品質保証テストとは異なる側面も踏まえつつ，鍵となるテストをどのように設計するかに関する支援は，手法あるいはツールとして確立していない．

より工学的な手法として，形式手法やテスト自動生成は役に立つはずだ，技術ではなくコミュニティ分断が問題だ，という議論もなされている［19］．この議論では，テスト自動生成や，Alloy によるモデル発見の有用性も論じられているほか，様々な側面におけるバランス，トレードオフの取り方についても疑問が投げかけられている．

しかし具体的な取り組みとしては，一般的なモデル検査をインクリメンタルに行う［20］など，あくまで従来の形式手法やテスト生成手法の技術の使い方を想定しているものが多い．その場合，仕様やテスト設計のモデルを十分に構築することにより開発のサイクルを

早める考え方になるが，開発者のスキルなどの現状とのギャップや，「動くコードが一番の成果物」といったアジャイル開発の原則から，受け入れられない状況があると考えられる．

【関連研究－形式手法】

形式手法，特にホーア論理に基づく関数や操作に対する仕様記述に関しては，VDM [6] や B メソッド [7]，Event-B [8]などの仕様記述言語，あるいは JML [9]や Frama-C [10] などプログラミング言語上での仕様記述言語が，国内産業界からも広く注目を集めている [11]．定理証明やテストなど方法は多岐にわたるが，ほとんどの取り組みは，検証対象となる仕様（制約）を与えることにより検証を行う．この際には，十分，完全な形式記述を事前に与えることの難しさと工数への懸念がある．

これらに対しては，例示と試行錯誤を通した段階的な記述が有効である．これは Alloy 言語とそのツール Alloy Analyzer においてすでに示されているものである [12][13]．またアジャイル開発への適用可能性についてもすでに論じられている [19]．以下に Alloy に基づく関連研究について述べる．

TestEra [21]では，Alloy により記述された事前条件を用いてテスト入力を生成するとともに，テストの実行結果に対して Alloy により記述された事後条件の検証も行う．しかし関係ですべてを表現する Alloy など，ソルバーやモデル発見器へのプリミティブな入力を多くの開発者が直接用いることは難しい．例えば本研究で扱う関数・操作であれば，様々なデータ型の関係による表現や，事前条件や事後条件の記述だけでなく，フレーム条件の考慮なども行う必要がある．これに対し，Alloy Analyzer やソルバーはあくまでバックエンドとして用い，開発者は問題を直接表現する入出力のみを扱えるようにした取り組みは多い．そのうち JML による仕様記述から Alloy への変換を扱った研究 [22]は本研究に近い．以上のいずれにおいても，仕様の記述のみではモデル発見器からの出力は，任意の並び順で提示され，特定の場合分けが含まれないなど確信度を高めることが難しい．

Allunimum [23]では，Alloy を拡張し，含まれる集合や関係が最小となるインスタンスを表示し，段階的に大きなインスタンス（本研究でのテストケースやその構成要素）を生成することができるようになっている．これは本研究で扱う，機能上の意味を踏まえて人が採る段階的なプロセスとは合致しないものであるとともに，予期しないインスタンスを得ることができない．初期段階に本研究のアプローチと補完的に活用することは考えられるが，本研究期間内には扱わない．

以上のように，Alloy に関しても，開発者にとって馴染みやすい語彙でテスト設計や例示も与え，開発者の確信度を高めつつ進める点や，アジャイル開発のコミュニティとの相互関係を追求する点において，本研究と同様の取り組みはこれまで議論されていない．

【関連研究－品質保証テスト】

品質保証テストに関しては，「テスト技法」などと呼ばれるテストケース設計手法に対し，モデルベーステストなど，形式記述からの生成のものが盛んに議論されている [14] [15] [16] [17]．しかし属人性排除と効率化のための自動化支援については，単純なものに決めてしまう実装や，非常に高度だが精度が限られるランダムテストなどの学術研究（例えば [24]など）に限られる．このため，部分的な記述と人手での探索などの柔軟な支援や，例

示と試行錯誤を通した段階的な実現確認は，提案者らの知る限り議論されていない．

ただし，テストにて議論されているシンボリック化 [25]や生成値のプール [24]など，スケーラビリティーのアプローチについては参考にある部分も多い．本研究の期間内においては，セミナーに必要な場合のみそれらの技術も適用することを考えている．

【研究責任者のこれまでの研究】

研究責任者が取り組んできた研究のうち代表的なものについて述べる．

（１）サービス指向コンピューティングにおける合意に基づいた協調的な移動性

既存のビジネスプロセス記述をモバイルエージェント化して実現するための設計・実装記述言語を初めとして，エージェント間の実行時合意形成・遵守機構，実行環境に応じた横断的変更反映，それらの形式化と検証（Ambient Calculus，Event Calculus を利用）などに取り組んだ [26]．その結果は産学連携プロジェクトにおいて研究開発されたエージェント開発フレームワーク Smartive に取り入れられ，特に自律型コンテンツ流通・提供の実現手段として用いられた．

この取り組みは，独自制約言語の定義や形式検証の活用という点では本研究と関連するが，斬新なコンピューティングパラダイムを追求しており，ソフトウェア開発現場を対象とした本研究とは異なる．

（２）サービス合成において整合性ある契約管理を実現するフレームワーク

（2007 年度 科学研究費補助金 若手スタートアップ，2008～2010 年度 科学研究費補助金 若手研究Ｂ）

複数サービスを組み合わせて消費者向けのサービスを合成する場合に，合成サービスの利用契約と,各部品サービスの利用契約との整合性を管理するための枠組みに取り組んだ．契約としては，状況に応じた権利や義務などを記したものを扱い，Event Calculus と SAT ソルバーベースの探索ツールを用いたシミュレーションや検証を行った [27]．

この取り組みは，探索と提示に基づいた支援という点では本研究と関連するが，義務論理やイベントに応じた変化を扱っており，関数・操作に関わる一般的な機能仕様を扱う本研究とは異なる．

（３）要求工学の応用による，法とその解釈のモデル化・分析

（2012～2013 年度 科学研究費補助金 挑戦的萌芽研究）

ゴール指向要求分析を模倣して，法に現れる抽象概念の解釈をモデル化し，データや処理がその概念に該当するかどうかを判断するための枠組みに取り組んでいる [28]．法解釈としては，論理的な必要条件の提示だけでなく，総合的に判断すべきという弱い条件や，具体例の提示も扱っている [29]．

この取り組みは，部分的な情報であっても厳密に表現し，可能な限り活用する点で本研究と関連するが，法解釈というドメイン特化であり自然言語上の用語を扱う取り組みであり，関数・操作に関わる一般的な機能仕様を扱う本研究とは異なる．

（４）スマートシティにおける市民の影響力を拡張する Cloud of Things 基盤技術

（2013～2015 年度 情報通信研究機構新世代ネットワークの実現に向けた欧州との連携による共同研究開発 モノのネットワークとクラウドを融合するネットワークサービス基盤の研究開発）

スマートシティの実現のために，クラウドコンピューティングとモノのネットワーク（IoT：Internet of Things）との融合技術に取り組んでいる［30］．その中で特に，以前取り組んでいたスマート空間において人に対し物理的な影響を及ぼすアクチュエーターの制御仕様記述に対するモデル化と検証技術の拡張，適用に取り組んでいる［31］．

この取り組みは，馴染みやすい言語からの変換を通した形式検証の活用という点では本研究と関連するが，斬新な応用アプリケーションを追求しイベント駆動型のモデルを扱っており，ソフトウェア開発者の支援を中心とし関数・操作に関わる一般的な機能仕様を扱う本研究とは異なる．

（５）段階的詳細化における複雑さの分散と整合性の保証に関する研究

（2014～2016 年度 科学研究費補助金 若手研究Ａ）

Event-B など自由度の高い段階的詳細化を扱う形式手法において，異なる段階的詳細化の方法を比較する指標,手法を構築し,それを基に適切な段階的詳細化を行うための支援を行う手法に取り組んでいる［32］．またその結果に基づき，形式手法に限らずゴールモデルなどにおいても，詳細化の良し悪しを追求している．

この取り組みは，形式手法の活用支援という点では本研究と関連するが，定理証明および段階的詳細化を用いた高信頼性の確保を主眼においており，より幅広い，手軽な活用を指向する本研究とは異なる．

（６）形式手法やテスト生成に関する産業界への教育・適用研究

研究責任者の所属機関にて 2005 年より開かれている，産業界の開発者・研究者向けの教育コース「トップエスイー」や，日本科学技術連盟ソフトウェア品質管理研究会（SQiP）にて，産業界への教育・適用研究に取り組んでいる［33］［34］［35］［36］．特に，形式手法やテスト自動生成手法の活用に関する適用研究を多数行ってきた．その中には， Web アプリケーションにおける準形式仕様からのテスト自動生成や，Alloy による提示を通したテストケース生成ロジックの妥当性確認など,本研究に関連するものも多く含まれている．

これらの取り組みは各者の状況に特化したものであり，一般的，統合的な言語およびツールの構築を目指す本研究とは異なる．

以上の全体に関し，特に仕様記述とテスト設計記述を部分的でも活用できるようにすることは，これまでの技術的な経験を活かしつつ，異なる実用を目指すものである．

2.1.3 研究目標

以下に本研究の根幹となる研究目標５つを述べる．基礎技術として研究目標１・２・３は初年度に，セミナーを通した評価に関する研究目標４・５は２年目に達成する．まず 2.1.1 で挙げた成果物のうち，成果物２のツールの根幹を構築するための基礎技術に取り

組み，実現可能性や困難さを明らかにし，取り組みへの確信度を高めるようにする．この方針に従い定めた研究目標を以下に示す.

- 研究目標１　Alloy 探索モデル構築
- 研究目標２　言語・言語変換の基礎構築
- 研究目標３　言語一般化・詳細化
- 研究目標４　セミナーの教材構築・準備
- 研究目標５　セミナー実施・実態調査・研究評価

研究目標１では,まず Alloy のみを用いて探索能力の確立に取り組む.これを踏まえて，研究目標２では，実際に開発者が利用する言語を構築し，その Alloy への変換を定義する（2.1.1 において定めた成果物１と成果物２のプロトタイピングに該当する）．以上により根幹となる基礎技術のアプローチを固めた上で，研究目標３として，その一般化と詳細化に取り組む（成果物１と成果物２の洗練）．これら初年度の成果に基づき，セミナーを通した評価の準備に研究目標４として取り組み，実際のセミナー実施，実態調査，研究全体の評価を研究目標５として取り組む（2.1.1 における成果物３に該当する）．

2.2 研究の活動実績・経緯

以下ではまず研究全体のスケジュールと，各研究目標に対する作業項目について概観する（詳細については 3 にて改めて述べる）．その後活動に関し特筆すべき点として，外部専門家との打ち合わせと外注について述べる．

【研究全体の実施スケジュール】

2.1.3 で挙げた研究目標のうち，研究目標１から３までは，技術の構築であり，核となる本質的な部分の検討から始め，段階的により実用的な全体像を組み立てるようになっている．このためこれらの研究目標については，この順に 1 年目に実施を行った．具体的な実施時期は下記の通りである．ただし，一部作業項目を後回しにするなどの詳細な調整は含めていない．

- 研究目標１：2013 年 6 月～9 月上旬
- 研究目標２：2013 年 9 月中旬～12 月
- 研究目標３：2014 年 1 月～3 月

次に研究目標４は実施するセミナーの準備，研究目標５はセミナーの実施とその評価である．ここで研究目標４においては，1 年目の技術構築において残った困難さ，特に本研究の範囲を超えるが必要性の高いものや，セミナーを円滑に実施するための実用化の取り組みなどにも取り組むことを想定していた．実際にはこの部分に想定よりも多くの時間をかけ，実施時期は以下のようになった．

- 研究目標４：2014 年 4 月～11 月
- 研究目標５：2014 年 12 月～1 月

【研究目標１　Alloy 探索モデル構築】

成果物２の根幹をなす探索技術を，Java の基本型や基本ライブラリに関する基礎的な内容に絞り，直に Alloy 記述および Alloy Analyzer を用いて実現する．図 2-1 において，Alloy 記述から Alloy Analyzer を経て Alloy モデルを得る部分に該当する．この部分に集中して様々な試行を行うことにより本研究の根幹となる探索能力を確立するとともに，結果表示に関する要件を整理する．

この研究目標においては，以下の作業項目に取り組む．

- 【作業項目１ａ】 簡易な場合の試行
  技術の実現方針に関する評価，議論のため，非常に簡単な数個のシナリオに関して，本研究で実現する探索，提示を Alloy 上にて試行する．

- 【作業項目１ｂ】 デフォルト内容構築
  到達目標２に含まれるデフォルト内容に相当するものとして，Java の基本型や基本ライブラリに関する基礎的な内容について，Alloy モデルとして実現するとともに，様々な探索を試行する．

- 【作業項目１ｃ】 整理・リファクタリング

以降の研究活動における拡張が行いやすいよう，構築したAlloyモデルを整理，リファクタリングする．

- 【作業項目１ｄ】 結果表示機能の仕様構築
  Alloy Analyzerによる探索の結果出力を，表示するツール機能の要件を検討し，仕様書を構築する．

【研究目標２　言語・言語変換の基礎構築】

研究目標１の結果を基に，相当する成果物１の言語，および成果物２の中心となる言語変換を定義する．図 2-1 において，提案言語から変換器を経てAlloy記述を得る部分に該当する．この部分について基礎的な場合に絞って通して実現することにより，実現の方針や課題を明確にする．これにより，Xtext などの DSL 作成支援ツールを用いて，言語変換器などを生成，実行できるようにするとともに，提案ツール全体の設計を得る．

この研究目標においては，以下の作業項目に取り組む．

- 【作業項目２ａ】 方針検討
  言語表現の方針および，その処理系の実装手段について検討する．

- 【作業項目２ｂ】 簡易な場合の試行
  言語の簡単な一部分に関し，言語定義，言語変換定義を行い，変換器を用いた Alloy モデルの生成，およびAlloy Analyzerを用いた探索まで一通りのツール動作が実現できることを確認する．

- 【作業項目２ｃ】 言語定義・言語変換定義
  研究目標１で得た結果から，中心となる構文要素を定め，言語定義および言語変換定義を与える．

- 【作業項目２ｄ】 統合機能の仕様構築
  作業項目１ｂにて動作確認した変換器，Alloy Analyzerおよび作業項目１ｄにて構築（外注）した結果表示機能を統合するものとして，総合的なツールの要件を検討し，仕様書を構築する．

【研究目標３　言語一般化・詳細化】

研究目標１の範囲に限らず様々なシナリオを検討することにより，研究目標２で定義した言語，言語変換を一般化および詳細化する．

この研究目標においては，以下の作業項目に取り組む．

- 【作業項目３ａ】 多様なシナリオの検討
  様々なプロジェクトや開発手法を想定し，提案ツールの様々な活用シナリオを検討，列挙する．

- 【作業項目３ｂ】 定義拡張
  作業項目３ａの結果を踏まえて，それに対応する言語定義，言語変換定義およびツール機能を拡張（一般化・詳細化）する．

- 【作業項目３ｃ】 技術全体の確認・検証
  これまでに構築したツールを用いて，例題の構築と確認を行い，様々な観点からの確認・検証を行う．

【研究目標４　セミナーの教材構築・準備】

研究目標３で挙げたシナリオも踏まえ，セミナー教材を構築する．またこの過程で，提案ツールのタスク管理機能に関する外注を始められるようにする．これらにより，セミナーを実施できるようにする.このセミナーには下記の３つのシナリオを含めるようにする．

- テスト駆動シナリオ: 各コード部品スケルトンだけを基に可能なテストケース入力を洗い出し提示する．必要に応じ，簡単な場合分け記述，データ制約や入力制約も活用し，絞っていきつつ確認する
- 品質保証テストシナリオ: 品質保証のためのユニットテスト設計に必要な情報を一通り入力し，境界値分析，ペアワイズ法などの技法を適用させ，テストケースの入力部分を自動生成する
- 形式仕様シナリオ： 事後条件を与え，仕様記述の妥当性を確認しながら段階的に記述をしたり，期待結果も含めてテストケースを生成したりする

この研究目標においては，以下の作業項目に取り組む．

- 【作業項目４ａ】 セミナー向けシナリオ検討
  作業項目３ａにて挙げたシナリオから，セミナーの実施時間を踏まえ適切なシナリオを選択，調整する．またその実現可能性について，これまでに構築したツール（作業項目２ｄで仕様を定め統合したもの）を用いて検証する．

- 【作業項目４ｂ】 タスク管理機能の仕様構築
  作業項目４ａにて検討したセミナー内容も踏まえ，セミナーにおけるツール演習を滞りなく行うためのタスク管理機能の要件を検討し，仕様書を構築する．拡張が容易であり普及している IDE として，Eclipse の拡張として機能の構築を検討する．

- 【作業項目４ｃ】 教材構築・試行
  セミナーの教材資料，および演習に必要なモデル等を作成し，試行を行うとともに，そのフィードバックを受け改善をする.この際,ここまでに構築したツールを用いる．本作業項目の前半では，初年度に構築した，作業項目２ｄで仕様を定め統合したものを用いる．最終的には作業項目４ｂにて仕様を定めた部分も含めて用いる．

- 【作業項目４ｄ】 実態調査内容検討・構築

セミナー実施時に行う調査内容について検討し，アンケートおよび演習結果評価を通した調査を準備する.

【研究目標５　セミナー実施・実態調査・研究評価】

研究目標４で構築したセミナーを実施することにより，提案ツールの実証評価を行うとともに，可能であれば背景・業務の異なる開発者のスキル傾向などの実態も調査する．セミナーは実施期間にて，累計 60 名程度の参加者数を目処に複数回開催する.

この研究目標においては，以下の作業項目に取り組む.

- 【作業項目５ａ】 企画・提案
  セミナーの実施日時など最終的な企画と募集文面を定め，セミナー開催の企画提案を行う.

- 【作業項目５ｂ】 実施
  これまでに構築されたツール全体を用い，セミナーを実施するとともに，評価・調査も行う.

- 【作業項目５ｃ】 調査結果まとめ
  セミナー開催時に行った評価・調査の結果を分析し，まとめる.

- 【作業項目５ｄ】 研究全体評価
  セミナー開催時に行った評価・調査の結果も踏まえ，研究成果に対する評価を様々な観点から行う.

【研究チーム内での作業・議論の進め方】

研究チーム内においては，研究責任者が研究全体や言語・ツール実装を担当し，そのアプローチや活用方法について個々のメンバーが各自の専門知識を取り込んでいく体制をとった.このため,研究責任者と各メンバーとの打ち合わせを必要に応じて個別に行いつつ,定期的に月例の打ち合わせにおいて全体の進捗状況や課題を共有し，方向性などの議論を行った.

【外部専門家との打ち合わせ】

研究開始後の早い時期（2013 年 7 月）にイギリス Newcastle University における形式手法の専門家 Alexander Romanovsky 教授のグループを訪問し，本研究の方向性と進め方に関する議論を行った.

その後も,2014 年 3 月にアルゼンチン University of Buenos Aires の Sebastian Uchitel 教授，2015 年 1 月にイタリア Politecnico di Milano の Carlo Ghezzi 教授と世界有数のソフトウェア工学専門家と議論を行った．いずれにおいても方向性については適切であるという評価を受け，例えば複雑な問題へのスケーラビリティーや，活用可能性について考

えられる限界や対応策について様々なフィードバックを得ることができた.

【外注】

本研究で取り組む技術を実際に活用するためにはツールの支援を実装することが必要になる．そのうち研究内容の根幹となるテストケース探索・提示機能や，そのための Alloy 記述への変換機能については，それぞれ既存のツール機能（Alloy Analyzer），および研究内容（文法定義と変換定義）から自動生成したツールを用いることができる．これらに加えて，セミナーにおける演習実施や，産学における試用に十分となるための機能を，外注により実現する．具体的には以下の GUI を外注対象とした.

- 結果表示機能： Alloy Analyzer からの出力結果をツール利用者に対して表示し，またフィードバックを受け取る機能（2014 年 1 月中旬～3 月下旬に外注実施）
- 統合機能・タスク管理機能： Alloy Analyzer，生成された変換器，および結果表示を統合する機能，およびツール利用者が行いタスクに応じて機能の起動やその結果の確認を行うための機能（2014 年 9 月中旬～10 月中旬に外注実施）

いずれについても，2 年目後半に行うこととしていた（実際は 2014 年 12 月開催）セミナーでの円滑な演習実施のために用いるものである．またそのための準備においても，活用（試用）を行う．このスケジュールに間に合わせるだけでなく，手続きの負担を分散して進められるように，上記のように必要な準備が異なる部分に分割した上で，準備ができ次第の発注を行っている．その実施時期は上述の通りであった.

2.3 研究実施体制

【主な研究メンバー】

実施機関においては，部署横断的なメンバーによりソフトウェア工学の研究・教育・実践に取り組む「先端ソフトウェア工学・国際研究センター（GRACE センター）」を設けている［37］．本研究はこのセンターのメンバーを中心として進める．以下に中心となるメンバーについて示す．本研究は複数の技術分野にまたがるものであるが，それぞれの専門に応じた役割をメンバーが担っている．

氏名	役職	本研究における役割
石川 冬樹 （研究責任者）	准教授	全体総括，形式仕様・テスト側面でのシナリオ・技術の構築，セミナー実施
田辺 良則	特任教授	形式手法側面でのシナリオ・技術の構築
吉岡 信和	准教授	アジャイル側面でのシナリオ・技術の構築
坂本 一憲 （2014 年 4 月～）	助教	品質保証テスト側面でのシナリオ・技術の構築

これらのメンバーは，産業界向けソフトウェア工学教育コース「トップエスイー」における講師と運営担当も務めており，教育および実践研究については多くの経験を持っている［33］．また産業界との共同研究も各自，あるいは共同で多数行っている．

個々の専門と本研究における役割について以下に述べる．

石川（研究責任者）は，仕様や契約，ポリシーなどのモデルを活用し，ソフトウェア工学やサービス指向などの次世代ソフトウェアシステムに関する先端研究に取り組んできた．また産業界の開発者や研究者に対し，形式手法や仕様記述に関する講義や個別指導を盛んに行っている．研究責任者としての全体の総括のほか，技術の構築全体，セミナー実施を担当する．

田辺はこれまで，特にソースコードを対象としたモデル検査や解析技術に関する先端研究に取り組んできたため，形式手法側面でのシナリオ・技術の構築を担当する．

吉岡はこれまで，セキュリティやプライバシーも含め，要求分析やパターン，アジャイル開発に関する先端研究に取り組んできたため，アジャイル側面でのシナリオ・技術の構築を担当する．

坂本はこれまで，テスト技術やソースコードの分析に関する先端研究に取り組んできたため，品質保証テスト側面でのシナリオ・技術の構築を担当する．

【補助メンバー】

上記のメンバーに加え，以下の補助メンバーが短期間加わっている．

- リサーチ・アシスタント　小林 努　（2013 年 6 月 15 日～2014 年 1 月 31 日）
  東京大学大学院の博士課程学生（情報理工学系研究科 コンピュータ科学専攻 本位田研究室所属）である．形式仕様記述を専門としており，先端研究だけでなく開発者と

の議論など様々な経験を持つため，形式仕様の構築など，高度な作業の補助を担当した．

- 研究員　方 菱　（2014 年 6 月 16 日～7 月 31 日）
  形式検証の活用に関する様々な経験をもっており，セミナーの準備に関して例題の構築や議論などを担当した．

そのほか，実施機関にて受け入れている連携大学院生や，海外大学からのインターン生などが，本研究の実装や，例題の構築などに関する補助作業を担当している．

【外注】

2.2 で述べたように，研究内容自体ではないが，セミナーにおける演習実施や，産学における試用に十分となるための GUI については,複数社の比較検討の上外注を行っている．

3 研究成果

3.1 研究目標１「Alloy 探索モデル構築」

3.1.1 当初の想定

(1) 研究内容

研究目標１においては，成果物２（ツール）の根幹をなす探索技術を，Java の基本型や基本ライブラリに関する基礎的な内容に絞り，直に Alloy 記述および Alloy Analyzer を用いて実現する．すなわち，実際に開発者にとって使いやすい構文やツールインターフェースなどは考えず，Alloy を使いこなせる研究チームが技術アプローチを直接 Alloy 上において検討する．

以下作業項目ごとに研究内容について述べる．

【作業項目１ａ】 簡易な場合の試行

技術の実現方針に関する評価，議論のため，非常に簡単な数個のシナリオに関して，本研究で実現する探索，提示を Alloy 上にて試行する．簡単なクラス構造やその中のメソッドに対し，不変条件，事前条件，事後条件，およびテスト設計やテストケースに相当する情報を Alloy で記述する．これにより想定している記述の基本構造を確立するとともに，結果を確認する上で有用な出力制御など Alloy Analyzer の設定についても確認する．

【作業項目１ｂ】 デフォルト内容構築

作業項目１ａで確認した方法に基づき，基礎的な内容について Alloy モデルとして実現するとともに，様々な探索を試行する．ここでの基礎的な内容とは，到達目標２に含まれるデフォルト内容に相当するものとして，整数や文字列，ブール値，列挙型，オブジェクト型，集合，リスト，写像など，Java において基本型あるいは標準ライブラリに含まれる型の Alloy 上での表現を定義する．また Alloy 上ではシンボリックな表現を用い，別途プログラム処理により実際の値を生成するなど，提示する値を生成する方針についても検討する．その上で，整数に対する大小比較や四則演算，リストに対する包含判定や要素取得など，Java における基本的な演算子あるいは標準ライブラリにおけるメソッドに相当する Alloy の表現を整備する．この際には，元々Alloy 自身の構文や標準ライブラリに含まれているものの確認に加え，本研究で必要と考えられるものの追加や再定義も行う．

【作業項目１ｃ】 整理・リファクタリング

以降の研究活動における拡張が行いやすいよう，作業項目１ｂにおいて構築した Alloy モデルを整理，リファクタリングする．研究目標１で構築したような Alloy モデルは，最終的には開発者が入力する言語から変換により生成されるものである．このため，これまで人手で構築してきた Alloy モデルを，規則的な記法にて統一された形式に整理すれば，以降で構築する変換器からの出力サンプルと見なすことができる（変換器のテストにおいて，期待する結果として用いることができる）．その他，一般的な整理やリファクタリング

も行う.

【作業項目１ｄ】 結果表示機能の仕様構築

作業項目１ｃにて整理し確定したAlloy Analyzerによる探索の結果出力の形式を踏まえ,それを表示するツール機能の要件を検討する．このために，あらゆる要素が集合・関係として含まれるAlloy Analyzerによる出力形式に対し，必要な出力のみを整理する．その出力から，Alloy記述の意図を踏まえた結果表示方法を定義する．2.2にて述べたように，結果表示のGUIは外注するため，この定義に基づき仕様書を構築する．

(2) 当初の到達目標と期待される効果

研究目標１は本研究の準備段階である．Alloy 上で直接本研究が想定する探索（テストケースの生成）を試行することにより，以下の項目について，その様々な可能性を検討し，以降の取り組みに対して実現方針を定めることが目的となる．

- 行いたい生成に関する要求
- 関連するAlloy Analyzerの挙動を踏まえ，生成を適切に制御するためのAlloy記述やAlloy Analyzerの設定
- 基本的な仕様，テスト設計およびテストケースに関するAlloy上での表現方法
- 基本的な型やその演算のAlloy上での表現方法
- Alloy Analyzerからの出力のうち確認すべき部分の抽出方法

これらの検討を通し実現方針を定めることにより，確信を持って以降の研究目標を進めることが可能となる．

また上記の最後の項目について検討することにより，結果表示を行うGUI について仕様を定め，外注を開始することができるようになる．

3.1.2 研究プロセスと成果

(1) 研究プロセス

研究目標１に対しては，以下のプロセスで取り組んだ．

① 初期の試行を簡単な例題数個に対して行った（作業項目１ａ）．
② 基本的な型や演算の語彙を意識した例題を構築し,様々なテストケース生成を試行し，整理・リファクタリングを行うサイクルを繰り返した（作業項目１ｂ，１ｃ）．
③ 結果表示GUIの仕様について議論し，仕様書を構築した（作業項目１ｄ）．

(2) 具体的な研究成果（結果）

【準備：Alloyの導入】

研究成果の説明のため，簡単にAlloyに関する導入を行う．図 3-1に簡単なAlloyモデル記述の例を示す．Alloy においては，シグネチャーとして，モデル上に表現され，種類が区別されるべき要素の集合を定義する（型，あるいはオブジェクト指向におけるクラスとも見なせる）．図においてはキーワード sig に続き以下のようなシグネチャーを定義し

```
sig 学生 {
  履修する : set 講義
}

sig 講義 {
  開講される : 時間
}

abstract sig 時間 { }
one sig 月1, 水1, 金1 extends 時間 { }

run { some 学生 } for 3
```

図 3-1 簡単な Alloy モデルの例

ている.

- 学生を表すシグネチャー．この種類の要素には，履修する講義が 0 個以上（set）関連付けられている.
- 講義を表すシグネチャー．この種類の要素には，開講される時間がちょうど 1 個関連付けられている.
- 時間（講義の開講時間，コマ）を表すシグネチャー．このシグネチャーは抽象的である（abstract）と定義されている．これは Java などでの抽象クラスと同様であり，このシグネチャーを型とするインスタンスは考えない（これを継承したシグネチャーのインスタンスを考える）ことを意味する.
- 具体的な時間を表すシグネチャー3 つ．これらは型というよりもインスタンスであるが，Alloy においては型もインスタンスも同じく集合あるいは関係であり，要素が 1 個しかないシグネチャーとしてインスタンスを表現することができる（キーワード one）．キーワード extends により，これら 3 つのシグネチャーは，時間シグネチャーに属し，互いに素である（すなわち別の型あるいはインスタンスを表す）ことを示している.

最後の run から始まる行は実行コマンドを表している．この場合は，学生が 1 つ以上含まれるようなインスタンス例を生成することを指示している（中括弧にて囲まれた部分）．最後の for は探索範囲に関する指示であり，今回はすべての種類のシグネチャーについて，高々3 個まで生成してそれらの組み合わせにより考えられるインスタンスを生成する.

図の Alloy 記述を，Alloy Analyzer に与えてこのコマンドを実行すると，例えば図 3-2 と図 3-3 Alloy Analyzer の実行結果例（1）

に示すような結果が表示される．Alloy Analyzer では，与えられたモデル記述から考えられるインスタンスを順次表示する（GUI では「次を表示する」ためのボタンがある）．ここではではそのうち 2 つだけ示しており，これらのインスタンスにおいては，要素の数は同様であるが，履修関係・開講関係が異なる．このように，与えられた探索範囲内で，考

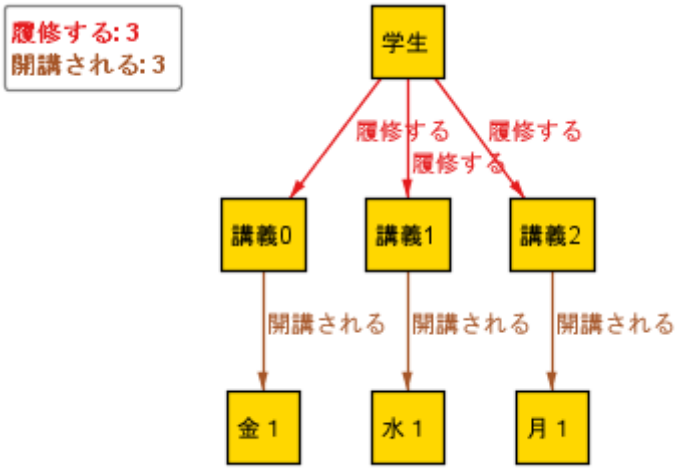


図 3-3 Alloy Analyzer の実行結果例（１）

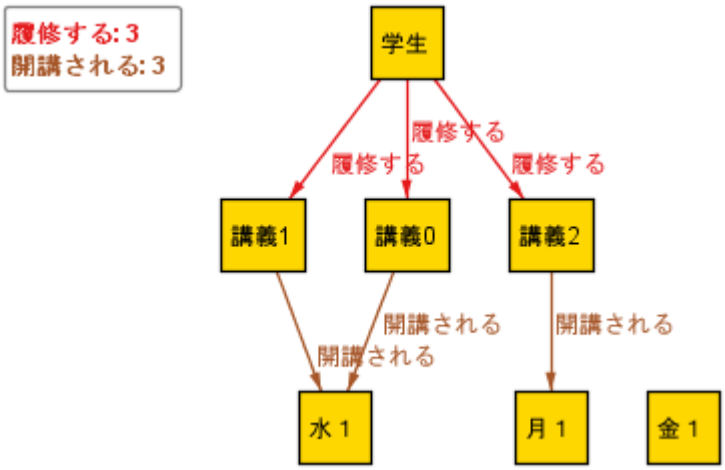


図 3-2 Alloy Analyzer の実行結果例（２）

えられる様々なインスタンスを基本的にはすべて表示していくことができる．ただし，「講義 0」と「講義 1」を入れ替えただけなど，対称性があり実質同じインスタンスは何度も表示されない．図では，Alloy Analyzer が持つ図形式での表示結果を示しているが，この結果は XML にて出力することもでき，本研究ではそちらを用いることとなる．

なお，今回の例では図 3-3 Alloy Analyzer の実行結果例（１）

において，1 人の学生が同じ時間に開催される講義 2 つを履修している．これは一般的には不適切な状況である．以下のように制約を加えると，そのようなインスタンスは表示されなくなる．

```
all s : 学生 |
    all disj c1, c2 : s.履修する |
        c1.開講される != c2.開講される
```

この制約では，あらゆる学生について，その学生が履修するどの異なる講義 2 つに対しても，開講される時間が一致することはないことを示している．

```
class TestClass{

  int x;

  public boolean judge(int a);

}
```

```
sig TestCase {
  x : int,
  a : int,
  result : Bool
}

sig TestSuite {
  some TestCase
}
```

図 3-4 本研究で用いる Alloy モデル構造（簡易版）

以上のように Alloy では，定義された集合および関係に対し，制約を満たすようなインスタンスを列挙することができる．

【本研究の対象に関するモデル化】

1.2 で述べたように，本研究では取り組み範囲を個々のデータ構造やメソッドに対するテストケース（単体テスト）に絞っている．上述の例は，データ構造に対するテストケース（具体例）の生成と見なすことができる．ただし，データ構造に対するテストは通常行うものではないため，具体例の生成と呼ぶことが適切である（コンストラクタのテストケースであると見なすこともできるが)．本研究ではテストケースという語を統一して用いる．

メソッドに対するテストケースを生成する場合，入力と出力の値の組を構成するシグネチャーを定義し，そのインスタンスを生成させればよい．図 3-4 に簡単な Java クラスと，そのテストケースを生成するための Alloy モデルを示す．図上部に示された Java クラス TestClass は，整数値のフィールド x を持つ．メソッド judge においては，引数 a の値とこの x の値を比べ，後者の方が大きいかどうかを真偽値で返す．

図下部にはこのメソッドに対応する Alloy 記述が示されている．まず，このメソッドに対するテストケースを表すシグネチャーである TestCase が定義されている．このシグネチャーは，テストケースを構成する入力値および出力値を含む．一般には入力値は，引数の値，および読み取り対象となるフィールドの実行前の値からなる．出力値は，戻り値の値，および書き込み対象となるフィールドの実行後の値からなる．図下部の Alloy 記述には，テストスイート（テストケースの集合）を表すシグネチャーである TestSuite が定義されている．これは単にテストケースの集合である．

今回の Alloy 記述には何の制約も含めていないため，型情報から考えられるあらゆるテストケースが生成される可能性がある．例えば，「x の値が 3 であり，a の値が 5 であるときに戻り値が true」となるといった組み合わせが無数に出てくることになる．本研究では，Java に対して以下のようなアノテーション記述を加え，それに応じて生成されるテストケースを逐次確認しながら変えていくことを考えている．以下 Alloy ではなく日本語により，与える記述の種類について述べる．形式定義については，最終的な構文定義と関連して，3.3.2（研究目標３に関する成果の説明）において示す．

【仕様の Alloy での表現】

まず，仕様を与えた場合，生成されるテストケースはすべて，その仕様を満たすものとなる．上記の例であれば，フィールド値の取り得る値が正に限られるという不変条件や入力値が正に限るという事前条件があるかもしれない．あるいは，上述したメソッドの意味を事後条件として与えることも考えられる．これらの条件は TestCase シグネチャーに対する制約として，あるいは TestSuite が保持するテストケース集合に対する制約として与えればよい（どちらかよいかは下記で述べる）．

それらの条件は弱い，あるいは十分ではない可能性もある．例えば弱い事後条件として，「x が a より大きいならば，戻り値は true でなければならない」という条件が挙げられる．「x が a より大きい」場合以外については，戻り値に関する制約が示されていないため，意図としては正しくない実行結果を，正しいと判断する可能性があるということである．本研究では，こういった部分的な記述であっても，有用な状況がある，あるいは，より十分な記述を与えるための途中過程であるとして考えている．

ここで，事前条件を満たさない不適切な入力（異常系）のテストケースを生成したり，事後条件を満たさない誤った実行のテストケース（具体例と呼ぶ方が適切である）を確認したりすることも考えられる．このように様々なテストスイートを生成することが考えられるため，TestCase シグネチャーには制約を与えておかず，生成指示に応じて TestSuite の制約を設定する方が対応しやすいと考えられる．

【テスト設計およびテストケースの Alloy での表現】

次にテスト設計について考える．上述の例であれば，「x が a より大きい場合・等しい場合・小さい場合」といった同値分割および境界値分析が考えられる．そのほか一般には，異常値の種類ごとのテストケース，特定の組み合わせを網羅するテストケースなども考えられる．など，基本的にはテストスイートに特定の条件を満たす（特定の場合分けに属する）テストケースが含まれていることが必要となる．こういった制約は，TestSuite が保持するテストケース集合に対する制約として与えればよい．

具体的なテストケースを与え，そのテストケースがテストスイートに含まれるようにすることも考えられる（手動でのテスト設定と自動生成との融合）．この場合，条件が特定のテストケースを一意に定めるだけであり，Alloy 上の表現は上記のテスト設計と同じでかまわない．

研究目標１の範囲では，上記のような実現方針を確認，確立することが目的であった．他にも多くの発見や検討が行われたが，それらの詳細については以降の研究目標にて対応

方針を検討，決定した部分もあるため，ここでは以下の 3.1.3 において特筆すべき点のみ挙げる．

【結果表示とフィードバック】

前述のように Alloy にはグラフ形式にて要素とそれらの関係を図示する機能がある．本研究においても同様の図示によりオブジェクト間の関係を表すことが考えられる.例えば，UML オブジェクト図を用いることも考えられる．一方で，様々なテストケースが含まれるテストスイート全体を俯瞰するには表形式がよく，開発現場でも表計算ソフトなどが用いられる．このため，本研究の期間内では表形式を用いた結果表示 GUI を構築することとした．ここでオブジェクトのフィールド値がオブジェクトである場合など，値の入れ子構造については Eclipse のデバッグ画面などと同様クリックで展開するインターフェースを用いることとした.

結果表示において重要であると考えられる点は，どのテストケースがどういう記述の意図に対応して生成されているのかを判別可能にすることである．例えば境界値を生成させた場合，あるいは指示がなくても境界値がたまたま生成された場合，その値が境界値であるという情報を結果表示に含めるようにする．本研究ではこのように，テストケース内の値，あるいはテストケースに付与される情報をタグと呼ぶ.

タグの意義として，開発者が想定しないテストケースやその分類に容易に気づけるようにすることがある．例えば場合分けの条件に抜けがあり，意図ではその場合分けに属さないと考えているテストケースが,その場合分けに属するようになってしまっているとする．この場合，テストケースの値に対して意図しないタグが付いていることですぐに気づく．逆にタグが付いていないことの気づきへは，より開発者の意識的な確認が必要になるが，いずれにしても確認を行うための情報を提示することが必要である.

研究目標１，研究目標２に対する作業を進める過程で，表示された結果を基に与えた記述を修正する（テストケース生成結果の分析を踏まえ，記述にフィードバックする）ことが重要であるという認識が得られた．このため当初想定していなかったフィードバックの種類について検討を行った．例えば以下のようなフィードバックが考えられる.

- 生成されたテストケースを採用し，次回の生成以降で常に含めるようにする．その際値を適宜変更する可能性もある.
- 場合分けが重複したテストケース，現在の検討範囲外のテストケース，禁則に該当するテストケースなど，生成されたテストケースは以後用いないようにする.
- 付与されるべきであるタグが付いていない（あるいは付与されるべきではないタグが付いている）場合に,そのタグが次回の生成で付くようになる（付かないようになる）ことを確認するためのアサーションを設定する.

これらのフィードバックの実現方法について詳細な検討を必要とするが，フィードバックを与えたいという要求，そのためのインターフェースについては，この時点での定義をすることができた．このため結果表示 GUI にフィードバックのインターフェースも含めることとなった.

以上の検討を踏まえて構築された結果表示 GUI のスクリーンショットを図 3-5 に示す．この画面ではテストケースに対するタグが右から 2 番目の列に表示されている．一番右の

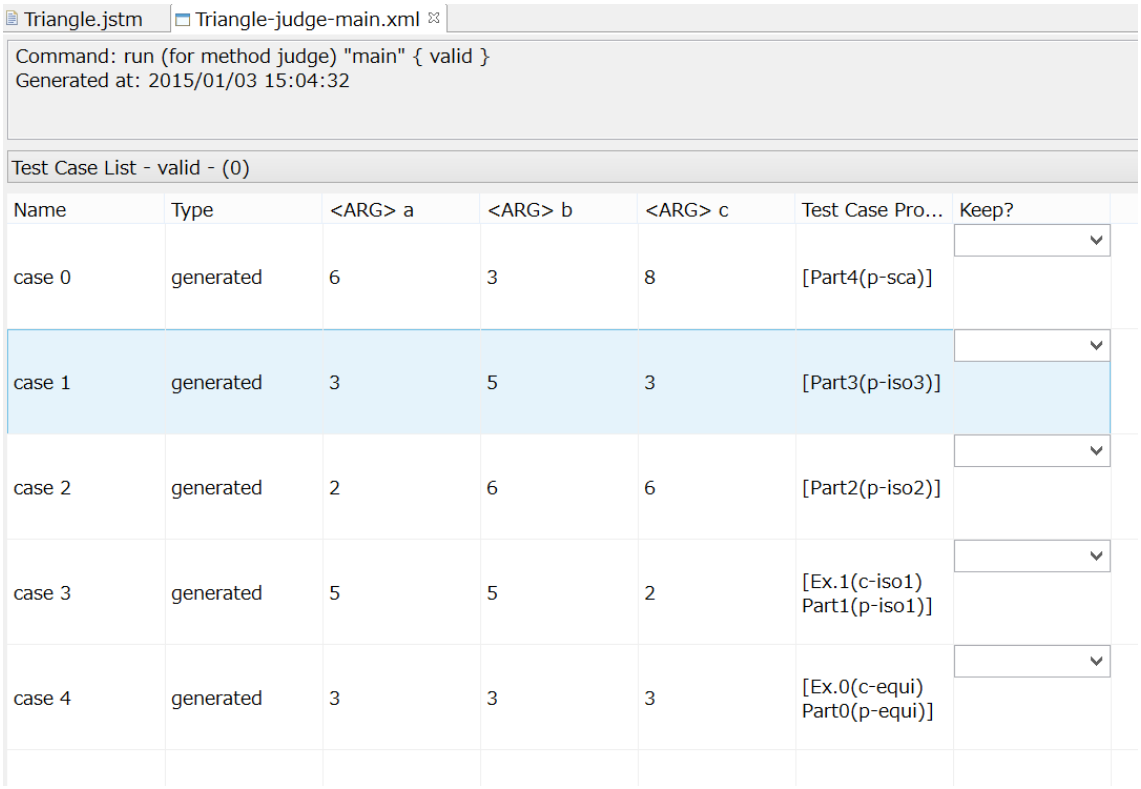


図 3-5 結果表示 GUI

列は上記のフィードバックのうち最初の 2 つ（次回以降の生成におけるテストケース採用の是非）をプルダウンメニューから行うものである．その他，テストケース内の値に対するタグの表示，該当要素のダブルクリックによる値の編集やタグに対するフィードバックの入力なども可能になっている．

3.1.3 課題とその対応

主に Alloy および Alloy Analyzer の特徴（設定項目や制限）と，結果の正しさや性能（状態爆発）などその影響に関連した留意事項について述べる．

【Alloy Analyzer 実行の単位】

Alloy 記述に対して実行させるコマンドとしては，テストスイートを生成するというコマンドを 1 回実行しても，個々のテストケースを生成するというコマンドを複数回実行してもよい．さらに両者の中間としてテストスイートを適切な大きさに分割して生成を行うことも考えられる．このことに意義がある可能性としては，大きなテストスイートを生成する際の状態爆発を避けつつ，個々のテストケースを細かく生成した際にソルバー起動等のオーバーヘッドを避けることができる場合である．

ここで，テスト設計の中には，境界値分析など特定のテストケースを他のテストケースとは別に作るべき技法もあり，そういった場合個々にテストケースを作っても問題ない．その一方，直交表やペアワイズ法などの組み合わせテスト技法では，個々のテストケースが複数の条件を満たすようになっている（例えば，あるテストケースが複数のペアをカバ

ーしている).これによりテストスイート全体として少ない数のテストケースで条件を満たすようになっている．もしも，ペアごとにそれを含めるようなテストケースを生成した場合，テストケースの数が膨大となってしまう．組み合わせテストに限らず，特定の値をどれか一つのテストケースで使う，本研究固有のものとしては指定された例や場合分けを含めるというテスト設計においても同様のことが言えるであろう．

以上のように，Alloy Analyzer の実行単位としては，テストケースごとの生成，テストスイート（の一部分）のまとめての生成を適宜使い分けることが必要となる．本研究期間内ではこの性能に関する追求や詳細な実験は行わず，2.1.1 で述べたように，セミナーに必要な範囲での検討，実装にとどめる．

【シンボリック表現】

Alloy Analyzer は SAT ソルバーを内部的に用いている．SAT ソルバーは真偽値の列挙に基づく形式で制約を満たすインスタンスを発見することになるため，数値や文字列など無数に値が存在する型に関しては状態爆発による性能悪化，あるいは制約を満たす値の生成失敗を起こしやすい．

この問題に対してはまず，実際の値を Alloy 上で表現するのではなく，シンボリック表現を用いることが考えられる．すなわち，「任意の整数」「0 以上 10 以下の整数」など，1 つの値にて実際の複数の値を表現する．本研究とは異なるが，テスト実行ではよく知られたアプローチである（シンボリック実行やその発展系，例えば [25]）．ただしこれにより実際にはとることのない値が生成されるなどがないように適切に行う必要がある．本研究期間内では，このシンボリック表現に関する包括的な実現や，現在の技術で自明でない高度な実現は必須とはせず，2.1.1 で述べたように，セミナーに必要な範囲での検討，実装にとどめる．

【様々な値の生成手法の使い分け】

上述の状態爆発に関する問題に対しては，数値計算などの定理を扱う SMT ソルバーの利用や，制約の形によってはプログラム側での値の生成など，様々な技術アプローチを複合して対処することが考えられる．ただしそれぞれの技術の適用箇所，適用方法や組み合わせは自明ではない．本研究では上記のシンボリック表現，およびそこからの値の生成（ランダムあるいは正規分布など）以外のアプローチ，特に複数ソルバーの使い分けについては扱わない．

3.2 研究目標 2「言語・言語変換の基礎構築」

3.2.1 当初の想定

(1) 研究内容

研究目標１の結果を基に,相当する成果物１(提案言語),および成果物２(提案ツール)の中心となる言語定義および Alloy への変換について，基礎的な内容に絞って定義する．すなわち,提案言語による記述を受け取ってテストケースを生成するまで一通りを構築し,

実現の方針や課題を明確にする．研究目標２の範囲においては，中心となる構文要素全体を俯瞰しそれぞれの実現方法（言語定義，言語変換定義）について確認する．構文糖衣や構文のバリエーションなどの一般化，詳細化は研究目標３（3.3）にて取り組む．

以下作業項目ごとに研究内容について述べる．

【作業項目２ａ】 方針検討

言語表現の方針および，その処理系の実装手段について検討する．具体的には，採用する構文の形式（記述箇所や，いくつかの類似言語を参考にした構文スタイルの決定），文法定義から構文解析器や言語変換器を構築するツール・フレームワークの調査と選択などを行う．

【作業項目２ｂ】 簡易な場合の試行

作業項目２ａで定めた方針に従い，言語の簡単な一部分に関して，言語定義および言語変換定義を行う．これにより，バックエンドの Alloy ではなく実際に開発者が与える記述から，変換器を用いた Alloy モデルの生成，および Alloy Analyzer を用いた探索，テストケース生成まで一通りの動作が実現できることを確認する．

【作業項目２ｃ】 言語定義・言語変換定義

研究目標１で得た結果から，中心となる構文要素を定め，言語定義および言語変換定義を与える．この定義は作業項目２ａで定め作業項目２ｂで試行した実現方式（ツール・フレームワーク）で行う．

【作業項目２ｄ】 統合機能の仕様構築

作業項目１ｂにて動作確認した変換器,Alloy Analyzer および作業項目１ｄにて構築(外注）した結果表示機能を統合し提案ツール全体の制御を実現するツールの要件を検討し，定義する．2.2 にて述べたように，この統合機能は外注するため，この定義に基づき仕様書を構築する．

⑵ 当初の到達目標と期待される効果

研究目標２は本研究の核となる技術の構築にあたる．この部分に関して実装を行うことにより，以降の研究目標３および研究目標４における以下のような検討，評価，拡張を繰り返していくための枠組みが整備される．

- 構文糖衣の追加やライブラリの拡充など提案言語の拡張
- 実行オプションの拡充やヒューリスティックの埋め込みなどツールの拡張
- 様々な例題による評価とそのフィードバックを受けた言語やツールの拡張

またこの研究目標２にて得る言語変換器や，研究目標１にて整理した Alloy Analyzer の利用方法などツールの構成要素の役割，機能，インターフェースが整備される．これらを統合する全体アーキテクチャーが確立されるため，その外注を開始できるようになる．

3.2.2 研究プロセスと成果

(1) 研究プロセス

研究目標２に対しては，以下のプロセスで取り組んだ.

① DSL（Domain Specific Language）構築ツールや言語変換ツールに関する調査や試行を行った（作業項目２ａ）.

② 初期の試行を簡単な例題数個に対して行った（作業項目２ｂ）.

③ テスト駆動開発同様のプロセスの繰り返しを行った．繰り返すサイクルとして，研究目標１の成果を基に変換により得られるべき Alloy 記述をテストケースとして設定し，それに対応する文法および Alloy への変換の定義を与える．このサイクルを，これまでの議論から本研究の中心となるとした構文要素に対して行った．（作業項目２ｃ）.

④ 統合機能の仕様について議論し，仕様書の概要を構築した（作業項目２ｄ）．実際には，この仕様書は作業項目４ｂで構築するタスク管理機能の仕様書と統合して外注を行うことになった.

(2) 具体的な研究成果（結果）

【実装方針】

提案言語の構文解析器や Alloy 記述への言語変換器は，Xtext [18]を用いて構築を行った．Xtext は Eclipse 上に構築された，プログラミング言語やドメイン固有言語（DSL）の開発のためのフレームワークである．Xtext においては，BNF の一種による文法定義を与えると，構文解析器や，該当する記述構造を保持する Java クラス，Eclipse 上でのエディターなどを生成することができる．なお，Java 言語の構文定義は用意されており，本研究のようにそれを拡張することも容易となっている．加えて，Xtend と呼ばれるプログラミング言語により言語変換の記述を行うこともできる．Xtend においては，文字列の中に条件分岐や繰り返しなどを埋め込む記法も可能であり，変換結果の記述が行いやすい．その他，Java に対し，型情報の省略などの拡張を行った先端プログラミング言語になっている．研究責任者は以前 Xtext を用いた研究開発の経験があり，そのときよりも機能が豊富になっていることから，本研究においても採用した.

【構文スタイル】

1.2 で述べたように，本研究期間の範囲では，提案言語は Java プログラムに対して追加の情報を記述するものとなる．このため，Java におけるコメント内に記述を追加するスタイルをとる．これにより，標準の Java 言語としても扱うことができる（例えば Java コンパイラーに構文エラーを起こさない）記述を行うことができる.

エラー！参照元が見つかりません。に提案言語による記述の例を示す．この Java クラスでは，簡単なメソッドとしッドにも付加することできる．そのようなものには，コロン記号に続いてメソッド名を明記するようになっている．メソッドのみに付加されるものは，該当メソッドの直前（他のフィールドやメソッド定義をはさまず）に記述する．て，ある人の年齢と性別（それぞれ引数 age，isMale）を受け取り，婚姻可能かどうかを判別する canMarry が定義されている．それに対し， //@ から始まる多くの行において，提案

```
public class CityOffice{

  //@ ¥pre { age >= 0 && age <= 150 }

  //@ ¥post { age >= 18 || (age >= 16 && !isMale) <==> ¥result }

  //@ ¥partition:canMarry {
  //@     ¥"男可" isMale && age >= 18,
  //@     ¥"男不可" isMale && age < 18,
  //@     ¥"女可" !isMale && age >= 16,
  //@     ¥"女不可" !isMale && age < 16
  //@ }

  //@ ¥case:canMarry { ¥"例１" age == 30 && isMale && ¥result }
  //@ ¥case:canMarry { ¥"例２" age == 16 && isMale == false && ¥result }

  public boolean canMarry(int age, boolean isMale);

  //@ ¥run:canMarry "main" { }

}
```

図 3-6 提案言語による記述例

言語による追加記述（アノテーション）を行っている．条件式の記述などは Java 構文に準拠しているが，独自のキーワードに関してはバックスラッシュを付けて名前の衝突が起きないようにしている．アノテーションのうち一部のものは，フィールドにもメソッドにも付加することできる．そのようなものには，コロン記号に続いてメソッド名を明記するようになっている．メソッドのみに付加されるものは，該当メソッドの直前（他のフィールドやメソッド定義をはさまず）に記述する．

【中心となる構文】

図の例におけるアノテーションそれぞれについて，その意味を以下で概観する．なお記述の意味については 3.1.2 で示したとおりである．

- 事前条件： pre から始まるアノテーションは事前条件を表す．図の例の場合，年齢が 0 以上 150 以下であることを示している．
- 事後条件： post から始まるアノテーションは事後条件を表す．図の例の場合，戻り値（キーワード ¥result ）の真偽は，「年齢が 18 歳以上または，年齢が 16 歳以上で女性である」ことと合致することを示している．
- テスト設計： partition から始まるアノテーションはテスト設計の一種として，生

成結果に含まれるべきテストケースの条件（場合分け）を表す．図の例においては，男性 18 歳以上，男性 17 歳以下，女性 16 歳以上，女性 15 歳以下という 4 つの場合分けを示している．この例では，一つのアノテーションの中に，コンマ区切りで複数の条件を与えているほか，各条件においてバックスラッシュに続く文字列としてタグ文字列を与えている．この複数条件の記述，およびタグ文字列の記述は，他のアノテーションにおいても可能である．

- テストケース（具体例）：case から始まるアノテーションは，テストケース（具体例）を表す．図の例においては，「入力は 30 歳男性，結果は婚姻可」，「入力は 18 歳女性，結果は婚姻可」という 2 つのテストケースをタグ文字列付きで指定している．
- コマンド： run から始まるアノテーションは，提案ツールへの実行指示内容となるコマンドを表す．図の例においては，main というタグ（名前）の付けられたコマンド 1 つが定義されている．中括弧内には実行オプションを与えることができるが，今回は特に与えていない．

以上の記述に対し，提案ツールを実行した際に生成されるテストケースの一例を表 3-1 に示す．

表 3-1 提案ツールの実行結果の例

age	isMale	¥result	Tag
30	true	true	[男可，例１]
18	false	true	[女可，例２]
8	true	false	[男不可]
0[LowerB]	false	false	[女不可]
121	true	true	[男可]
10	false	False	[女不可]

図 3-6 の例で与えた記述においては，仕様（事前条件および事後条件）は十分なものであるため，不適切な入力や出力は生成されていない．与えた場合分け，およびテストケースはすべて含まれるように生成され，それらには識別のためのタグ文字列が付加される．明示的に与えたタグのほかに，制約の形式から明確に判別できる境界値条件についてもタグが付与されている（LowerB）．

以上で用いていない構文としては，クラス内外への変数へのアクセスの宣言（読む，読み書きする）が挙げられる．無数に存在しうるクラス内外の変数については，当該メソッドに関連すると宣言されたときのみテストケースに含まれるようになる．

なお，上記の例はほぼ完成された記述である．本研究の特徴としては，このように完成された記述にいたる過程においてもツールを実行し，記述の意味を確認できることがある．すなわち，形式化や自動化の部分的な活用をする，あるいは十分な活用をする場合でも段階的にそこに至るようにできるようにするということである．

例えば，上記の例では事後条件を十分に強く与えている．そこに至る過程として，例えば「18 歳以上なら（性別問わず）結婚できる」という記述から始めるかもしれない．その

際に生成される例は，18 歳以上ならば結果は true となっているようになる．一方，17 歳以下については true の場合と false の場合が任意に含まれる．このように具体例を見て与えた記述の意味を実感することができる．ただし，限られた数の生成では，17 歳以下については true の場合と false の場合が任意に含まれることはわかりにくいかもしれない．与えた条件を満たさない反例を生成すると，すべて 18 歳以上なのに結果が false になっているような例になっている．このように，制約条件（何かの境界）で区切られた領域双方の例を見ることで，より記述の意味についての核心を高めることができる．

様々な言語構文，ツール機能についての全容については，一般化・詳細化に取り組む研究目標３の成果としてまとめる（3.3.2）．

【統合機能】

当初の計画においては，研究チームが実装するのは提案言語から Alloy 記述への変換部分と考えていた．その後の Alloy Analyzer，および作業項目１ｅにて外注した結果表示 GUI の起動も併せた全体の制御（統合機能）については，研究固有の要素もなく外注をする予定であった．しかし，3.1.3 で述べたシンボリック表現の導入のため，研究チームによる実装部分が増えたほか，その試行錯誤を効率化するためにも提案ツール全体の実装が必要となった．このため研究目標２においては，上記のように中心となる構文について，提案言語による記述から Alloy への変換，Alloy Analyzer によるテストケースの生成までを行うよう提案ツールの核となる部分を実装した．これにより，様々な構文，ツール機能の詳細について，反復的に試行錯誤や追加・変更を行うことができるようになった．以上により統合機能の外注については想定のうち一部分となることになり，外注手続きの効率化のためにも後に予定しているタスク管理機能の外注（3.4.2）と一括して行うこととなった．

3.2.3 課題とその対応

研究目標２は様々な課題に取り組んでいくための仕組みを構築するものであるため，これ自体に生じた課題は特にない．本研究期間では扱わない長期的な課題として，技術のさらなる活用に適した実装方針を検討することが挙げられる．例えば，テストケースの記述は Excel，JUnit，Cucumber など広く用いられる形式と互換性があることが望ましい．また，本研究の内容は，本質的には Java に特化するものではないため，他のプログラミング言語や形式仕様記述言語にも適用できるような，汎用的なツール構成も考えられる．例えば言語共通の構造モデルを定義し，それに対して本研究の仕組みを構築するアプローチが考えられる．ただし，このような実装方針については研究期間外の取り組みとし，本研究期間内では Java 上のアノテーションによる記述形式に関し完成度を高めることに集中する．

3.3 研究目標３「言語一般化・詳細化」

3.3.1 当初の想定

(1) 研究内容

研究目標１の範囲に限らず様々なシナリオを検討することにより，研究目標２で定義し

た言語，言語変換を一般化および詳細化する．

以下作業項目ごとに研究内容について述べる．

【作業項目３ａ】 多様なシナリオの検討

様々なプロジェクトや開発手法を想定し，提案ツールの様々な活用シナリオを検討，列挙する．1 にて述べたように，本研究においてはテスト駆動開発，形式手法，品質保証テストの支援となることを想定しているため，それぞれの活用シナリオを検討する．このために，研究チームや協力者による議論のほか，書籍に掲載されているものなど各分野においてよく知られる例題についても検討する．

【作業項目３ｂ】 定義拡張

作業項目３ａの結果を踏まえて，それに対応する言語定義，言語変換定義およびツール機能を拡張（一般化・詳細化）する．これは，本研究独自の概念に関する一般的な定義のほか，構文糖衣やライブラリの充実も含む．

【作業項目３ｃ】 技術全体の確認・検証

これまでに構築したツールを用いて，例題の構築と確認を行い，様々な観点からの確認・検証を行う．すなわち提案言語および提案ツールについて，実際に利用することにより，確認・検証を行い，必要ならば修正や拡張に取り組む．

(2) 当初の到達目標と期待される効果

研究目標３により，本研究における技術的な取り組みは基本的に完成することになる．これにより，1 で述べた本研究の目指す課題解決につながる技術が得られるようになる．広く一般的に開発者が用いるためのツールやドキュメント整備はこの時点では不十分ではあるものの，研究チームや協力者による試行，検証，利用例の提示などが進められるようになる．特に，研究目標４におけるセミナーの準備について具体的に進めることができるようになる．ただし，2.1.1 にて述べたように技術的に難しい部分に対し，セミナー内容に特化した取り組みを研究目標４に関し行う可能性はある．

3.3.2 研究プロセスと成果

(1) 研究プロセス

研究目標３に対しては，以下のプロセスで取り組んだ．

① 様々な活用シナリオ，例題を検討，議論した（作業項目３ａ）．
② 必要となる言語拡張あるいはツール機能拡張に対するシナリオ（提案ツールに対するテストケース）を設定し，言語定義および言語変換定義，ツール機能に関する拡張を実装するというサイクルを繰り返した（作業項目３ｂ）．
③ これまで議論した例題やシナリオについて，構築した言語・ツールを適用し，様々な観点からの確認・検証を行った（作業項目３ｃ）．

(2) 具体的な研究成果（結果）

【中心となる構文の全体】

3.2.2 では中心となる構文について例を通して概観したが，ここではそのバリエーションも通して，最終的な構文，および関連するコマンドオプションについて形式的な定義を示す．ここではフィールドを対象に実行した場合については省略し，メソッドを対象に実行した場合に絞って説明を行う．

生成されるテストケースの集合を TS とする．各テストケース $t \in TS$ は，入力と出力の値により構成される．

$$t = \{i_1, \ldots, i_m, o_1, \ldots, o_n\} \in I_1 \times \ldots \times I_m \times O_1 \times \ldots \times O_n$$

入力値の定義域 $I_1, \ldots, I_m$ は，メソッドの引数および，実行前の変数値の型により定義される．ただし変数値は，少なくとも一つのアノテーションにおいて参照されているものに限る．同様に，出力値の定義域 $O_1, \ldots, O_n$ も同様に，戻り値および，実行後の変数値の型により定義される．ただしこれらの定義域は，実際には実装により，型の値に対する適切な範囲を定めることになる（整数型のビット幅など）．

ツールの実行モードとしては，valid，invalid，checkpost および checktest の 4 つが存在する．それぞれのモードに対し，各アノテーションの意味は以下のように定義される．

事前条件および事後条件がそれぞれ $PRE = \{pre_1, \ldots, pre_i\}$ ， $POST = \{post_1, \ldots, post_j\}$ と与えられているとする．このとき，

- valid モードにおいては，下記を満たすような TS を生成する．すなわち，事前条件も事後条件も満たす適切な入出力を表すテストケース（具体例）を挙げる．

$$\forall t \in TS . \Big(\big(\forall p \in PRE . p(t) \big) \wedge \big(\forall p \in POST . p(t) \big) \Big)$$

- invalid モードにおいては，下記を満たすような TS を生成する．すなわち，事前条件を満たさない入力を表す異常系のテストケース（具体例）を挙げる．事後条件については満たすことは要求しない．なお，満たさない事前条件に応じて例外を投げることの定義も提案言語では可能としたが，ここでは省略する．

$$\forall t \in TS . (\forall p \in PRE . \neg p(t))$$

- checkpost モードにおいては，下記を満たすような TS を生成する．すなわち，事前条件を満たす入力に対し，出力が事後条件を満たさないという，正しくない実行の具体例を挙げる．

$$\forall t \in TS . \Big(\big(\forall p \in PRE . p(t) \big) \wedge \big(\forall p \in POST . \neg p(t) \big) \Big)$$

テスト設計およびテストケースを与えた場合，上記に加えてさらに追加の条件が考慮される．テスト設計およびテストケースがそれぞれ $PART = \{part_1, \ldots, part_k\}$ ， $CS = \{case_1, \ldots, case_l\}$ と与えられているとする．このとき，

- valid モードにおいては，下記を満たすような TS を生成する．すなわち，テスト設計で与えられた場合分け，およびテストケースを含めるようにする．

$$\forall p \in (PART \cup CS).\exists t \in TS.p(t)$$

上記定義は，制約式の観点では場合分けとテストケースは同一に扱われることを示している．すなわち，場合分けと見なすかテストケースと見なすかは，記述をする開発者の受け取り方であり，内部の機構としては集合の要素数が複数か単一かはあまり関係がないということである．

invalid モード，checkpost モードに関しても，同様の定義を設ける．すなわち，それらのモード専用の場合分けおよびテストケースを与えることができる（例えば invalid モードに対して不適切な入力の場合分けや例を与えるなど)．これらについても意味は上記と同様であるため，省略する．

- checktest モードは，valid あるいは invalid と組み合わせて用いる．valid モードと組み合わせた場合，以下の条件をさらに追加する．すなわち，提示されるテストケースはすべて与えられたものであり，追加で生成されたものを含まないとする．これは与えたテストケースが，テスト設計に対して十分であるかどうかを検証することを意味する．

$$\forall t \in TS.(\exists cs \in CS.cs(t))$$

最後に以下のようにアサーションを与えることができる．

- テストケースが決定的であることを宣言することができる．決定的であるようなテストケースの集合を $DET \subset CS$ とする．このときこのアサーションをチェックするコマンドは，以下の条件を満たす反例の生成を試みる．

$$\exists det \in DET.(|\{t \in TS \mid IN(\mathrm{det})(t)\}| > 1)$$

ただし $IN(det)$ は制約式 det に対し，入力値を det と同様に縛るが，出力は任意の値を許すような制約式とする．テストケースの場合，各値の取り得る値に対する条件が論理積で示されていると仮定でき，このような制約式は自明に得ることができる．

- 各テストケースに対して，それが属するべき場合分けについて宣言することができる．この所属関係を表す関係を $BELONG \in CS \times PART$ とする．このときこのアサーションをチェックするコマンドは，以下の条件を満たす反例の生成を試みる．

$$\forall t \in TS.\exists (cs, part) \in BELONG.cs(t) \wedge \neg part(t)$$

【アプローチの総括】

以上の定義は，Java 言語に限らず，広くホーア論理に基づく操作（メソッド）の定義に対し，以下で述べる原則を踏まえたアプローチを提起するものである．これらの原則は必

ずしも研究開始時には明確にはなっていなかったが，具体的な言語やツールの構築や仕様を通して明確になってきた．なお，操作に限らず状態遷移系に対しても同様のアプローチは定義でき，モデル検査技術などを用い実現できると考えられるが，本研究の範囲では実現していない．

性質による記述と例示による記述の融合：従来形式手法，形式仕様記述においては，性質(Property)，すなわち宣言的な命題により仕様記述を行ってきた．これはテスト設計(テストスイートに対する仕様）においても同様である．一方でテスト駆動開発からの発展として，例示による仕様（Specification by Example）というキーワードも用いられている．提案言語はこれらの双方を使い分けることを可能とする．さらに，双方を両方与え互いの確認に用いることも可能になっている．

仕様，テスト設計，テストケース（具体例）の混在した記述：上記と同様のことであるが，より具体的な実現として，仕様，テスト設計，テストケースを混在させ，互いに互いの確認に用いたり，相補的に用いたりすることができるようになっている．

早いフィードバックを与えるツール：与えた制約式に対し，その式の意味を具体例にて示す．これにより，何か記述を行った際に，様々な記述をそこから積み上げる前に，すぐに行った記述の意味を確認することが可能である．すなわち，逐次意味を確認しながら，段階的に形式的な記述を構築していくことができる．また，ある一通りの式を与えないとフィードバックを得られないということはないということでもある．これはテスト駆動開発において重要な原則である一方，仕様あるいはテスト設計について部分的にのみ形式的な記述を与えた場合でもツールを活用できることを意味する．

仕様を得るタスクおよびテストケースを得るタスク双方への活用：提案言語およびツールは，テストケース生成のためのものであるが，とらえ方によっては「仕様とテストを行き来する」ためのものであるとも言える．すなわち，必ずしも生成するテストケースが成果物であるとは限らず，その生成を通して仕様記述やテスト設計記述の妥当性確認を行っているとも言える．このように提案言語およびツールは，仕様記述やテスト設計を成果物とするタスクにおいても，テストケースを成果物とするタスクにおいても利用することができる．

例と反例双方の確認：与えた式を満たす場合だけでなく，満たさない場合も確認することができるようにする．特に部分的な条件（弱い条件）を与えた場合には，条件を満たす例は意図に合わないものが多くなる．その場合でも条件を満たさない反例を確認することで，どういう可能性をまずは排除したのかを確認することができる．これに限らず，条件が強すぎて意図しない場合が排除されているときにそれを反例の中から見つける可能性があるなど，条件を満たさない反例の確認には意義があると考えられる．

命題に対するテストケース・テスト駆動開発：テスト駆動開発では，まだ記述していない機能や観点に対してテストケース（具体例）を与えたとき，失敗する（Red/Fail）ことが開発サイクルの出発点となる．これはテスト駆動開発・テストファーストであるかに限らないが，プログラムと異なり仕様記述の場合，まだ記述していないことについては「どういう出力でも受け入れる」ととらえることができる．すなわち，非決定的な記述が可能である．これに対して，既存の仕様記述では不十分であることを明示するためには，ある

テストケースの値が受け入れられるというだけではなく，その入力に対して唯一の受け入れられる出力を定めているということをアサーションとしたい．提案言語はテストケースが決定的であることを宣言することができ，この要求に対応している．

【制約記述のための構文拡張】

各アノテーション内の条件記述部分については,基本的に Java 構文に準拠するようにしている．ただし以下のように拡張を行っている．

一階述語論理による式を記述可能としている．すなわち，集合の要素に対して,「すべての要素がある条件を満たす」，および「ある条件を満たす要素が少なくとも 1 つある」ことが記述できるようになっている．ただし Alloy にならい構文糖衣として,「ある条件を満たす要素がちょうど 1 つだけある」,「ある条件を満たす要素が存在しない」という記法も提供するようにしている．さらに，集合だけではなくリストを束縛の範囲に指定することもでき，その場合はリストの要素を含む集合に対する記述として扱われる．

Java の標準ライブラリに含まれる複合データ型，すなわち集合，リスト，写像（マップ）に対し，演算を容易に記述できる記法を提供している．Java ライブラリでは，例えば和集合を表現しようとすると，副作用のある（該当オブジェクトを書き換える）メソッドにより集合要素の追加を行うような逐次処理の記述が必要となる．これに対し，Alloy，あるいは VDM, B など一般的な形式仕様記述言語を踏まえ，集合和をプラス記号で表現できるようになっている．またこのように定義した演算については，演算子形式，メソッド形式双方の形式での記法を可能としている．例えば，仕様記述ではリストに対してインデックスの集合を得ることがよく必要となるが，これは `inds list` という単項演算子形式による記述も可能であり，また `list.indexSet()` というメソッド形式での記述も可能である．

そのほか，C などで用いられる def 同様のエイリアス定義などの構文糖衣を検討し，実装している．

【テスト設計のためのオプション】

場合分けは様々なテスト設計を表現することができるが，よく知られた技法に対しては直接的ではない．このためコマンドに与えるオプションにより簡易にテスト設計を指定することも可能としている．具体的には，値の上限あるいは下限が定数により与えられている場合に境界値を含めるようにするオプション,異常系のテストケース(invalid モード)において 1 つのテストケースに複数の異常が含まれないようにするオプション，テストケースの数を指定するオプションなどを実装している．

3.3.3 課題とその対応

研究目標３は実用的な言語とツールの構築に取り組むものであるため，構文およびツール機能において様々な要求あるいは現状の限界が顕在化してくる．また多くの詳細な実装を行うため，その十分なテストおよびデバッグについても限られた工数で十分に行うことは難しい.このため,研究目標５において実施セミナーにおいて本質的に重要である部分，および本研究の有用性などを評価するために重要な部分に特に焦点を当てて取り組んだ．そのほかの部分については，3.1.3 で挙げた性能の問題などと合わせて，研究目標４にお

けるセミナー準備の過程において，セミナー内容に特化して取り組むこととした.

3.4 研究目標４「セミナーの教材構築・準備」

3.4.1 当初の想定

(1) 研究内容

研究目標３で挙げたシナリオも踏まえ，セミナー教材を構築する．またこの過程で，提案ツールのタスク管理機能に関する外注を始められるようにする．これらにより，セミナーを実施できるようにする.このセミナーには下記の３つのシナリオを含めるようにする.

- テスト駆動シナリオ: 各コード部品スケルトンだけを基に可能なテストケース入力を洗い出し提示する．必要に応じ，簡単な場合分け記述，データ制約や入力制約も活用し，絞っていきつつ確認する
- 形式仕様シナリオ： 事後条件を与え，仕様記述の妥当性を確認しながら段階的に記述をしたり，期待結果も含めてテストケースを生成したりする
- 品質保証テストシナリオ: 品質保証のためのユニットテスト設計に必要な情報を一通り入力し，境界値分析，ペアワイズ法などの技法を適用させ，テストケースの入力部分を自動生成する

以下作業項目ごとに研究内容について述べる.

【作業項目４ａ】 セミナー向けシナリオ検討

作業項目３ａにて挙げたシナリオから，セミナーの実施時間を踏まえ適切なシナリオを選択，調整する．必要に応じて追加の調査や議論を行い，シナリオ候補を追加する．またそれらのシナリオについて，これまでに構築した提案ツールの性能や，限られたセミナー時間での実施などの観点から，実現可能性を検証し，候補を絞り込む.

【作業項目４ｂ】 タスク管理機能の仕様構築

作業項目４ａにて検討したセミナー内容も踏まえ，セミナーにおけるツール演習を滞りなく行うためのタスク管理機能の要件を検討し，仕様書を構築する．すなわち，提案ツールのユーザーが，複数のファイルを管理したり，3.3.2 で挙げたような複数のコマンドから選択して実行を行ったりするができるような機能を検討する．実装方針として，拡張が容易であり普及している IDE として，Eclipse の拡張として機能の構築を検討する.

【作業項目４ｃ】 教材構築・試行

セミナーの教材資料，および演習に必要なモデル等を作成し，試行を行うとともに，そのフィードバックを受け改善をする．この際，ここまでに構築したツールを用いる．本作業項目の前半では,初年度に構築した,作業項目２ｄで仕様を定め統合したものを用いる．最終的には作業項目４ｂにて仕様を定め外注した部分も含めて用いる．さらにこの過程において，必要に応じここまでに構築した言語やツールの拡張，特に技術的に困難である部

分や，本研究期間で取り組む想定で内部分について，セミナー内容に特化して取り組む.

【作業項目４ｄ】 実態調査内容検討・構築

セミナー実施時に行う調査内容について検討し，アンケートおよび演習結果評価を通した調査の準備を行う．定性的，定量的な評価ができるようにアンケートの評価項目などを定める.

(2) 当初の到達目標と期待される効果

研究目標４は，続く研究目標５にて実施するセミナーの準備である．直接的には，産業界の開発者・研究者に提案言語およびツールを試用してもらい，有効性や有用性に関する評価，および実用化のためのフィードバックを得るための準備が整うこととなる．間接的には，より踏み込んで提案言語およびツールの利用を検討するために，様々なユースケース,有用性が端的に示される例題,活用の上での考え方や使いこなすためのガイドライン，技術的限界などが明確になる．さらにこれらを踏まえ，セミナー内容に特化しての改良や強化を行うことにより，提案言語およびツールの実用性を向上することができる.

3.4.2 研究プロセスと成果

(1) 研究プロセス

研究目標４に対しては，以下のプロセスで取り組んだ.

① 様々なシナリオの特性やセミナー題材としての適正などの評価を行った（作業項目４ａ）.
② タスク管理機能の仕様書を構築した．3.2.2 で述べたように，この仕様書は統合機能の仕様書と合わせて外注を行うこととなった．（作業項目４ｂ）.
③ セミナーの例題や演習内容の試行を行い，それらの選択や調整を行うとともに，提案言語とツールの拡張や調整も行う.そのサイクルを繰り返す一方で,教材（スライド）をまとめていく（作業項目４ｃ）.
④ セミナーにおける評価項目について議論し，定めた（作業項目４ｄ）.

(2) 具体的な研究成果（結果）

【セミナーの主題】

1 で述べた背景と動機にもあるように，本研究はテスト駆動開発，形式手法，および品質保証テストの３つの分野に横断した技術や議論を通し，それぞれの課題を解決しようとするものである．またそれらの分野が互いに強く関連していることを認識し，総合的な課題解決の施策に取り組んでいく姿勢を得ることを促していくことも目指している．このことからセミナーの内容は，まず上記３分野それぞれの導入を行い，提案言語およびツールを用いた演習を行うものとなる．さらに，それら分野の相関や観点の違いについて端的に整理されており，それに基づき気づきや議論を促進するものであるべきである．後者の分野間の相関や違いについては，これまでも様々な議論もあるものの，合意が取れているような明確な言い回しがなされているわけではない．セミナーの例題や提案言語およびツー

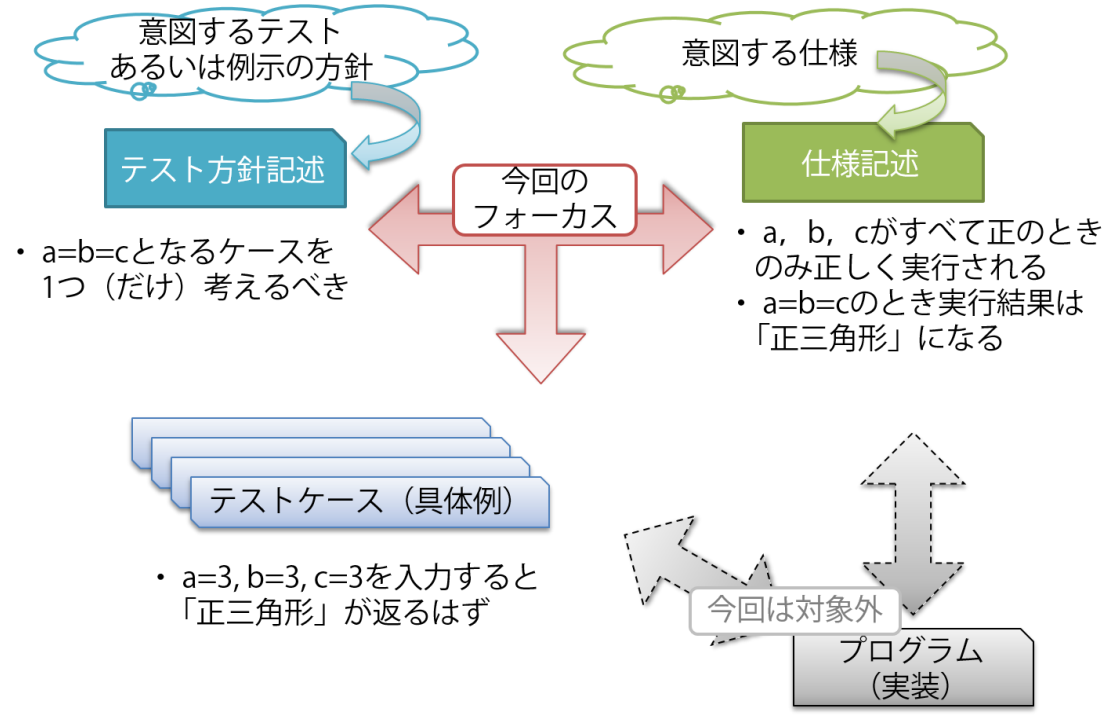


図 3-7 セミナーにて用いた扱う記述の概要図

ルの位置づけなど，具体的な内容との対応もとりながら，本研究としてしっかり整理する必要がある．

この点に関して準備，セミナーで利用した図の１つを図 3-7 に示す．この図においては，セミナーにて扱う記述項目を示している．また，Myers の三角形判定問題［38］を例にして，各記述項目の例も示している．なお，本報告書では「テスト設計」という用語で統一しているものについて，セミナーではより弱く「テスト方針」という言葉を用いている．この図に含まれる記述項目，すなわち仕様，テスト設計（テスト方針），テストケース（具体例）に対し，どれに重きを置いているか，何をインプット・アウトプットと考えているかなどの観点から，各分野の位置づけや比較，相関の議論を行った．各分野の概要を述べる際，あるいは例題や演習問題などで提案ツールを用いる際に，どの記述項目を与え，どの記述項目と照らし合わせているのかなど，同じ構造の図の上で随時示すことにより，用語のぶれなどなく一つの枠組みでの対比や議論が行えるようにした．例えば本研究で対象としている３つの分野，および 1.2 にて述べた課題解決アプローチについては，以下のように説明することができる．

- テスト駆動開発においては，テストケース（具体例）が「主役」となり，開発の目標を端的に示すものとして開発を導いていく．テストケースを適切に定めていくことが重要となり，かつ難しい問題である．これに対し仕様やテスト設計をもし明示的に記述しない場合でも，それらがテストケース定義の重要な根拠であり，検討や議論においては必ず考慮されているはずである．提案言語およびツールにおいては，仕様やテスト設計も明示的に記述し，テストケースの導出あるいは確認に用いることができるようになっている．
- 形式手法（特に形式仕様記述）においては，仕様が「主役」となり，厳密な記述を行

うことにより，様々な検証を通し，開発の中心となる仕様の品質を高めることができる．一方で，宣言的な命題による記述（性質の記述）において，十分な記述を行うこと,その妥当性について確信を持つことは難しい.提案言語およびツールにおいては，その検証あるいは妥当性確認のために具体例（テストケース）や反例を用いることができるようになっている．

- 品質保証テストにおけるテスト自動生成・テスト設計支援ツールにおいては，テスト設計および必要な仕様を与えることにより，煩雑で大量のテストケースの記述を，一定の基準を満たすように行うことができる．一方で，適切にテスト設計を厳密な記法で与えることや，意図に応じたテストケースがすべて含まれるようなテスト設計を与えることも難しい．提案言語およびツールにおいては，部分的であってもテスト設計や仕様を与えたり，具体的なテストケースも与えたりしながら，テスト設計を支援していくことができる．

いずれにおいても本研究の提案言語の特徴としては，3 つの記述項目をすべて自由に与えることができるようになっていることである．

図 3-7 に示した記述項目という観点のほか，図 1-2 にて示したようにテストケース（具体例，インスタンス）とそれらを分ける境界線という図も共通して用いた．すなわち仕様やテスト設計は，「性質による記述（Specification by Property)」であり，様々なテストケース（具体例，インスタンス）を正しさや場合分けの観点で分ける境界線であると考えることができる．一方で，テストケース（具体例，インスタンス）を直に記述することにより，その境界線を意識させる，確信を持たせることもできる．これは「例示による記述（Specification by Example)」である．これらは排他的ではなく，境界線を厳密に，一意に意図通りに描こうとこすること，境界線に仕切られた領域にある具体例を考えることとは「行き来をする」必要がある．この点が提案言語およびツールを位置づける重要な観点であること，またキャッチーな言い回しも意識して，セミナーの副題は「仕様とテストを行き来してアジャイル・形式手法・テスト自動生成を考える」とした．

【セミナーの構成】

当初の想定においては，研究期間後の活用も見据え，セミナーは 1 日版，より長い 1.5 時間×7 コマ（大学院や実施機関で行っているトップエスイー［33]での集中講義に相当）版など数個のものを準備する予定であった．しかし後者はトップエスイーなど既存のプログラムにおいて,時間割調整の上導入するなどしないと参加者を集めることが難しいため，研究期間内においてはまず 1 日版の構築に集中することとした．

これに応じ，イントロダクションの後，3 つの分野に関する概説と演習を行い，最後にまとめや議論を行うことから，個々のセッションは 1 時間から 1 時間半程度になる．限られた時間ではあるが，各分野における技術や動向の深い解説を行うものではないため，注意深く設計をしてこの時間配分にて教材の構築に取り組むこととした．

なお，最終的なセミナーのスライド資料は表紙やリファレンスなども含め 233 枚となった．

【セミナーで用いる例題の検討】

セミナーで用いる例題については，これまでの過程で調査したソフトウェア工学全般，形式手法，テスト駆動開発，品質保証テスト設計の文献などから候補を挙げた．なお，それらの一部は，これまでの取り組みにおいて試行対象あるいはテストケースとして用いられており，すでに提案言語・ツールでの試行や検証が済んでいたものである．なお，それらの過程では，形式手法に詳しい研究者が，一見簡単な問題に対し十分でない論理式を与えたこともあった．このような経験も踏まえて例題を選んだ．

セミナーの時間が限られていることを踏まえ，本研究において独自あるいは本質的ではない構文要素などの説明や,問題設定自体の説明などに時間を割かないことが重要である．演習自体にも長い時間を避けないため，複数ファイルを参照したり，リファレンスを参照して様々な演算子を活用したりすることも難しい．一方で，論理式の誤りや例示によるその修正などが実感できる，個人ごとにテスト設計に差があり確認や議論の重要性が実感できる，といった些細ではない問題が望ましい．

以上を踏まえ，セミナー資料も構築しつつ，それと整合性があるように例題の選択，および詳細な演習問題の設定および調整などを行った．

【言語・ツールのセミナー内容に応じた洗練・強化】

提案ツールの実装のうち，3.1.3 で挙げたような，高度な技術あるいは多くの実験やデータを要するチューニングなどについては,セミナーで用いる例題に特化して取り組んだ．シンボリック化については特にこの時期に多くの取り組みを行っている．またツール機能のテストや，特に様々な記述のバリエーションを与えたときの動作確認についても，セミナーで用いるものに特に重点的に取り組んだ．

【最終的なツール実装】

作業項目４ｂで扱ったタスク管理機能については，一般的な仕組みを外注により実現している．例えば実行を開始する際には，Eclipse におけるファイル一覧管理画面（Project Explorer）においてコンテキストメニューを開き，該当ファイル内で定義されたコマンド一覧から実行するものを選ぶ．その後，研究チーム側で構築した機能，結果表示 GUI などを随時呼び出すようになっている．この部分は一般的な設計であると考えられるため，詳細は割愛する．

【セミナーにおける評価項目】

本研究における提案言語およびツール，またセミナー自身の評価を得るため，セミナーの受講者に対して以下のような内容に関するアンケートを採ることとした．

5. 一般的な背景：業務内容
6. 本研究が論じている 3 分野それぞれに関するとらえ方：自身にとっての興味と経験の有無と程度，セミナーを受けた上での自身にとっての評価
7. 提案言語およびツールの評価：有用であると評価する特徴，期待できる活用方法，改善すべき点
8. セミナーの評価：有用であると評価する特徴，改善すべき点

上記のうち 1 および 2 の前半部分は，受講者の層，特徴について把握するためのもので

ある.2 は,本研究の目的に分野に横断した視点や課題解決の施策を促すことがあるため,セミナー受講前後で意識の変化があるかどうかを調べる狙いがある．例えば，形式手法は高信頼システムのためのものであり自身に関係ないと考えていたが，原則やアプローチは本質的に有用であると考え直した，といった意識の変化を促せることを期待している．3 および 4 は，本研究の評価に関わるものである．これらの選択肢に関しては，本研究のアプローチの特徴である「総合的・融合的な視点」「部分的でもかまわない」といった考え方に対する評価，各分野における課題解決に対する評価，および意識変化や教育観点からの評価を行えるようにした.

その他演習の結果についてもある種の実験結果としてとらえ，例えばスキル評価としての利用可能性などを検討することが考えられた．しかし，今回のセミナーでは短い時間での演習となり，演習の結果を左右する要因が受講者の特性など以外にも多くあるため，演習の結果を詳細に分析することは行わず，定性的な分析と議論にとどめることとした.

3.4.3 課題とその対応

3.3.3 で述べたとおり，汎用性のある言語およびツールの実装を試みる以上，多くの要求や実装の改善点，あるいは実装の問題（バグ）などが現れる．またセミナーを実施する上でも，しっかりしたドキュメントが利用できることが望ましい．これらについてはセミナーの内容に特化して取り組んだものの，理想的にはその範囲に限らず継続的に時間をかけて取り組むべきことであった.例えばセミナーの演習を踏まえて,「こういう書き方をしてみよう・ああいう問題に使ってみよう」といった要望に答えきれるほどのドキュメントは用意できず，今後の課題として残った.

3.5 研究目標５「セミナー実施・実態調査・研究評価」

3.5.1 当初の想定

(1) 研究内容

研究目標４で構築したセミナーを実施することにより，提案ツールの実証評価を行うとともに，可能であれば背景・業務の異なる開発者のスキル傾向などの実態も調査する．セミナーは実施機関にて，累計 60 名程度の参加者数を目処に複数回開催する.

以下 2.2 において概観した作業項目ごとに研究内容について述べる.

【作業項目５ａ】 企画・提案

セミナーの実施日時など最終的な企画と募集文面を定め，実施機関で行っている教育プログラム「トップエスイー」[33]や国内外の会議などに対してセミナー開催の企画提案を行う.

【作業項目５ｂ】 実施

これまでに構築されたツール全体を用い，セミナーを実施する．またその際に，提案言

語とツール，セミナーに対する評価について調査を行うとともに，可能であれば開発者のスキル傾向などの実態調査も行う．

【作業項目５ｃ】 調査結果まとめ

セミナー開催時に行ったアンケート結果および演習の結果を集計する．定量的，定性的な分析を行い，結果の意義や意味について考察を行う．

【作業項目５ｄ】 研究全体評価

セミナー開催時に行ったアンケート結果や，これまでに行った議論や考察を踏まえ，研究成果全体に対する評価を，提案の独自性，有効性，有用性など様々な観点からまとめる．

(2) 当初の到達目標と期待される効果

実施計画書に基づき記載する．

3.5.2 研究プロセスと成果

(1) 研究プロセス

研究目標５に対しては，以下のプロセスで取り組んだ．

① セミナーの企画と提案を行った（作業項目５ａ）．
② セミナーを実施し，その中で評価・調査も行った（作業項目５ｂ）．
③ セミナー時のアンケート結果および演習の結果を集計し，考察を行った（作業項目５ｃ）．
④ 評価とそのまとめを行った（作業項目５ｄ）．

(2) 具体的な研究成果（結果）

【セミナーの企画と参加募集】

研究責任者も参加する，産業界向けに様々なセミナーなどを提供しているNPO法人「トップエスイー教育センター」に[39]おいて，セミナーの実施について提案を行い，承認を得ることができた．その後公開した参加募集の内容について以下に引用する．
（本報告書執筆の段階では http://topse.or.jp/2014/11/2243 にて公開されている．）

日時：
2014年12月5日(金)，7日(日)
ともに9:30-17:30（受付開始：9:00）
※5日と7日の内容は同一です。

会場：
国立情報学研究所 国立情報学研究所 20階 ミーティングルーム(2009/2010)

概要：
ますます困難となるソフトウェア開発の課題に対し，以下のように様々な立場・視点からの取り組みがなされています.
・ アジャイル開発のための，テスト（動作例）を軸としたテスト駆動開発やその発展手法
・ 厳密な言語による記述に基づく仕様の厳密化やツールによる検証（形式手法）
・ 様々な技法を用い系統的に行う品質保証テストやその自動生成ツール
これらの分野は互いに関係ない，時には相反するものだと見なされているかもしれません．しかし，これらの分野はすべて，

「ソフトウェアが実現すべき機能」を定義する，そしてその定義を用いて確かに実現がされていることを検証・確認する

という共通の課題に向き合っているものです．このため，これらの分野それぞれで積み上げられた技術や考え方は，元となる思想が異なるからこそ，互いの課題を解決する可能性があります.

本セミナーでは，ソルバー（モデル発見）ツールを用いて，「仕様を基にテストケース（動作例）を生成や確認する」，あるいは「テストケース（動作例）を基に仕様を確認する」演習を行います．この演習は上記の 3 つの分野にまたがって行うもので，それぞれの分野での上記課題に対する考え方への入門となるとともに，それら分野の相互関係・相互補完について広い視点から学び，議論する内容となります.

本セミナーで用いるツールは，情報処理推進機構（IPA）のソフトウェア工学分野の先導的研究支援事業（RISE)の委託研究により試作したものです．Java 言語で書いたデータ構造やインターフェースの記述に対し，仕様や場合分けのヒント，動作例などを付加することで，手軽にソルバー（モデル発見）の技術を，インクリメンタルに，インタラクティブに活用できるようになっています.

【注意事項】
・ ツールをインストールした Windows PC を用意する予定です.
・ テスト駆動開発（TDD）やその発展（受け入れテスト駆動開発 ATDD など），VDM や Alloy などの形式仕様記述手法，PICT などのテスト自動生成ツールなどの知識・経験があれば，より理解が深まりますが，必須ではありません（これらの分野への導入をしつつ，相互の関係について考える内容となっています).

講師：
・石川 冬樹 准教授（国立情報学研究所）

定員：
35 名(先着順)
※定員に達し次第，申し込みを締め切らせていただきます。

参加費：
無料

主催：
NPO 法人 トップエスイー教育センター

協力：
国立情報学研究所 GRACE センター

上記のように，1 日のセミナーを同一の内容で 2 日実施した．開催場所の都合および，演習時のツール利用へのサポートの難しさなどから定員を設けている．

この参加募集は，NPO 法人トップエスイー教育センターの会員や過去のセミナー等の参加者，および実施機関が運営している産業界向け教育コース「トップエスイー」の受講生・修了生にメールにて送付された．そのほか，twitter/Facebook での案内も行った．後者に関しては様々な方々にも参加募集情報の共有にご協力いただき，多くの人々の目にとまるよう参加募集を行うことができた．

結果として，両日ともに定員に達し募集を打ち切ることとなった．

【セミナーの実施】

予定通り 2 日のそれぞれセミナーを実施した．内容については準備に関する 3.4.2 にて述べたとおりである．基本的には順調に進んだが，参加者が演習に深く取り組んでいたために演習時間を予定より長くとったり，参加者の演習の様子を見て補足を行ったりするなど，多少想定より時間をかけてしまった．これとセミナー開催箇所の都合等により，アンケートについては後日の回答でもよいこととした．これにより回答率が下がってしまった．設問により異なるが，60 名中 35 名程度から回答を得ることができた．

【アンケート結果：参加者の背景】

回答者のうち 30 名は産業界からの参加であり，残りは大学院生および大学等の研究者であった．産業界からの参加者に対し普段の業務を複数選択可で尋ねたところ，SE（要求分析や仕様化などの担当）とプログラマーが最も多かった．品質保証担当や CEO・マネージャーはそれぞれ数名以下であった．ただし，SE，プログラマー，品質保証などを複数挙げている参加者，明確に分離・分担していないという参加者も 10 名以上いた．アジャイル開発あるいは中小企業などにおいて小さな人数で取り組んでいる参加者も多かったと考えられる．

本研究が対象としている形式手法・形式仕様記述，アジャイル開発・テスト駆動開発，テスト技法･テスト自動生成の 3 つについて，これまでの興味や経験についても調査した．その結果を表 3-2 に示す．

表 3-2 セミナー参加者の背景

これまでの取り組み	形式手法・形式仕様記述	アジャイル開発・テスト駆動開発	テスト技法・テスト自動生成
興味があり仕事の一貫でも活用していた	7	19	18
興味があり書籍やセミナーで勉強はしたが､仕事では使っていなかった	23	15	16
興味があるが勉強や活用はしていなかった	7	4	5
聞いたことはあるが興味はない・あまり関係ないと思っていた	2	2	1
名前を聞いたことがあるくらいだった・名前も知らなかった	1	0	0

アジャイル開発・テスト駆動開発を仕事で用いている，あるいは勉強しているという人の割合が高い．これは twitter や Facebook による参加募集において，情報共有にご協力いただいた方々の影響が大きいと考えられる．また形式手法・形式仕様記述を仕事で用いている人も数名おり，勉強したこともある人が大半である．これは参加募集メールが，実施機関にて行われた多くの形式手法セミナー・講義に出席したことがある人に送られたという観点もあると考えられる．ただしそうではない参加者が大半であることから，実施機関や IPA などで行われているセミナー等において，形式手法・形式仕様記述の知名度が高まっているものだと受け取れる．ただし，参加募集の方法が偏っていることから，国内の参照会における動向などを示すものでは決してない．テスト技法・テスト自動生成については，産業界においては関心が高い分野であるため，仕事の対象としている，あるいはそうでなくても勉強しているということは当然の結果であろうと考えられる．

いずれにしても，アジャイル開発・テスト駆動開発，あるいは形式手法・形式仕様記述の導入・勉強など，非常に意識の高い参加者が集まったと見なすことができる．これは本セミナーの参加募集が示唆する先進性（「ソルバーの活用」などの文言）や，そもそも平日あるいは日曜日に丸一日セミナーに参加する層であるということを考えても，妥当な解釈であろう．

【アンケート結果：各分野に対する意見】

次に本セミナーを終えて，各分野についてどう考えるかについても調査した．その結果を表 3-3 に示す．

表 3-3 参加者の各分野に対する受け取り方

自身の意見	形式手法・形式仕様記述	アジャイル開発・テスト駆動開発	テスト技法・テスト自動生成
技術（手法・ツール）として自分の業務に重要であり活用をしたい	16	24	26
技術（手法・ツール）は直接関係ないあるいは難しいかもしれないが、考え方・観点・原則は自分の業務に重要であり勉強・考慮・反映したい	17	13	11
業務にはあまり関係ないが興味深いので勉強・議論したい	6	3	2
自分にとってそれほどあるいは全く重要ではない・関係がない・興味がない	1	0	1

いずれの分野に関しても，ほとんどがポジティブな意見を持っていた．形式手法・形式仕様記述については，これまでも様々な調査などで言われている通り，初期の導入コストや，高信頼性のための特別な技術であるという間隔などから，今すぐ業務で活用したいという人の数は比較的少ない．

本研究の狙いとして，前述のセミナー前の取り組みや印象に対し，セミナー受講後に意識が変わるということがあった．しかし本セミナーの受講生はもともと意識が高く，様々な技術に対してポジティブに活用方法を産み出していくことができるような人たちであったとも考えられる．このため，この意識変化という観点について，アンケートから直接見える形での実証はできなかった．ただし後述するように，意識を変える・視点を広げるために提案ツールおよびセミナーが有効であることについては参加者の多くが合意している．

【提案言語・ツールの利点】

提案言語およびツールについて，そのよい点をどう考えたかを調査した．これは複数選択可能な形式とした．その結果を表 3-4 に示す．

表 3-4　提案言語・ツールの利点

A. テスト駆動・形式手法・テスト技法の考え方を 1 つのツールで扱える点	22
B. 何か部分的にでも書くとすぐにツールを動かせる点	22
C. 形式手法やソルバーツールとしては取っつきやすい点	17
D. 様々なアプローチを織り込み視点を広げる・学ぶよい機会になる点	16
E. 入力内容を変え様々なタスクをある程度こなせる汎用的なツールである点	14
F. 「テスト一部自動生成」など軽い導入がしやすい点	14
G. その他	1

基本的には多くの特徴について一定数の合意を得られている．人によって受け取り方，他のツールに対して考えている問題などが異なるため，受け取り方がばらけることは当然であると考えられるが，特に重要でないという観点はなさそうである．回答項目 A，B は，本研究の特徴である「分野横断的な取り組みを支援する点」，「とにかく部分的な記述でもツールを動かし早いフィードバックが得られる点」という点であるが，特に多くの参加者に合意を得られている．D は実作業の支援というよりも意識向上・教育観点からの評価であり，ツールに関する評価としては本来二次的なものではあるものの，本研究の狙いの一つである．この観点からの有用性に合意している参加者が一定数いた．G の「その他」については，「テストケースの妥当性を確認できる，それらしいデータを出せる」ということが挙がっていた．

【提案言語・ツールの課題】

提案言語およびツールについて，どのような点を改善あるいは強化すべきかを調査した．これも複数選択可能な形式とした．その結果を表 3-5 に示す．

表 3-5 提案言語・ツールの課題

A. 実行速度の改善	18
B. UI の洗練(記述の自動補完・情報の表示など)	14
C. UML・Excel など広く使われている表記との相互変換	8
D. その他	8
E. 直接扱える問題の拡張(操作列や部品の統合など)	3
F. 特定の利用法に絞って機能・UI の特化	3

G. 出力結果の数や多様性などのチューニング	2

A では,3.1.3 で挙げている性能が最も改善を要する点だということが指摘された.3.1.3 で述べたように，この点については今回の実施しなかった様々なソルバーの使い分け，より踏み込んだ実装の工夫などが必要になる．一方で関連した懸念であった G についてはそれほどの指摘はなかった．これはセミナーで扱った演習が比較的簡単なものであったからと考えられる．B，C は一般的にツールの完成度として求められるもので，これは工数をかければ達成できることであると考えられる．E は，本研究の期間内で扱う対象が狭すぎないかという観点であったが，特に大きな問題とは考えられなかったようである．F は本研究の特徴に関するトレードオフとして,「いろいろできるが，個別のタスクでやりたいことに踏み込んでできない」ということが懸念としてあったが，それほど問題とは考えられなかったようである．D の「その他」については，ドキュメントの充実や，利用環境の充実などがあった.多くは B,C と同様に,工数をかけてしっかり取り組むというものであった.ただし，ソフトウェアモデル検査やランダムテストとの融合，Web での提供など，さらなる高度なツール化のアイディアなどもあったため，参考にしていきたい.

【提案言語・ツールの利用法】

提案言語およびツールについて，どのような目的・タスクでの利用が効果的であるかについて調査を行った．これも複数選択可能な形式とした．その結果を表 3-6 に示す.

表 3-6 提案言語・ツールの活用方法

A. 中堅者の視点を広げる教育・概念整理・意識向上	15
B. テスト駆動開発におけるテストケースの整理や議論	14
C. 初心者の基礎的な考え方・技術への入門・教育	12
D. ドメイン分析や仕様化における制約の洗い出しや確認	11
E. 品質保証におけるテスト設計の補助	10
F. 特定なタスクによらず個人でのロジック整理・確認	9
G. プログラムに対する（自動処理できる）定型コメント	7
H. 既存の形式手法を使う準備としての記述の厳密化や確認	7
I. 論理的にややこしい問題への手軽なソルバー適用	4
J. その他	1

A，C は意識向上・教育観点からの評価であり，ツールに関する評価としては本来二次的なものではあるものの本研究の狙いの一つである．初心者向けか中堅者向けかで意見が分かれているものの，いずれにしても非常に高い評価が得られていると考えられる.

B，D，E，H は特定のタスクに関する支援の有用性である．形式手法を使うという前提の H は数が少ないが，それぞれの利用法について一定の評価が得られている．特に，今回のセミナー参加者の多くがアジャイル開発・テスト駆動開発を実践しているが，それらの参

加者が B を選んでいる点はよい評価であると見なせる．F，G，I は特定のタスクというよりも，電卓や Excel のように汎用的な小道具として用いるような方向性である．この方向でも一定の評価が得られている．J の「その他」はシンボリック実行によるテストとの組み合わせを挙げており，高度な技術である．

以上のように，参加者のそれぞれが，自分なりの使い方をイメージすることができる，柔軟性が高いツールであるという解釈ができる．

【セミナーにおいてよかった点】

セミナーに関しよかった点について，調査を行った．これも複数選択可とした．その結果を表 3-7 に示す．

表 3-7 セミナーのよかった点

A. 現場で役に立つ考え方・視点・原則の取得	19
B. 先端技術の体験・未来やあるべき姿の議論	17
C. 様々な思想・位置づけの異なる技術に対する広い入門	17
D. ソルバー技術の体験	12
E. 様々な思想・コミュニティー間の関係に関する整理・議論	12
F. 現場で役立つ技術の習得	4
G. その他	2

今回のセミナーは研究活動にて構築したツールを使うものであり，F のように「今すぐ使える・役立つ」と見なされることは，ツールの完成度などの観点から難しい．一方で，A から E までは様々な視点から，教育効果や，視点を広げることや分野をまたがることが促されたことについて一定の評価を受けている．G の「その他」においては，特定技術の習得，提案ツールの演習を挙げていた．

【セミナーにおいて不十分だった点】

セミナーに関し不十分であった点について，調査を行った．これも複数選択可とした．その結果を表 3-8 に示す．

表 3-8 セミナーの不十分であった点

A. 時間	14
B. 全体の関係や位置づけに関する整理・議論	9
C. 独自の概念や記法・ツールに関する導入・説明	8
D. 言語・ツールの使いやすさ・インターフェース	7
E. ツールの根本的な技術の能力・達成度	6

F. ツールを用いた演習の量	4
G. 全体的な資料の完成度	3
H. 個々の分野に関する導入・解説(あまりにも略しすぎ)	2

A にある通り，3 つの分野を概説し，しかも新しいツールを使うという内容に対し，1 日のセミナーは短かったと思われる．B，C，F，H についてもより長い時間のセミナーであれば改善できる点である．ただし今回は H に関する指摘は少なかったが，本セミナーの位置づけとして個々の分野の深掘りは期待していなかったであろうこと，意識が高くすでに各分野の勉強経験がある参加者が多かったためであると考えられる．一方，D，E はすでに述べたツールの完成度に関するものである．G はセミナー固有の点であり，資料については反復的に改良を続けていく必要がある．

【その他】

総まとめとなる質問として,「提案ツールを今後使ってみたいですか？」という質問も行った.これに対しては,「帰ったら引き続きいろいろ試してみたい･使い道を考えてみたい」が 28 名となり，予想以上によい結果であったととらえている．より多いと予想していた回答である「方向性はわかるけどもっと実装がよくなったら」は 10 名であった.「別に（ちょっと方向性が違う・など)」が 1 名だけであった.

そのほか，すでに示した業務内容やこれまでの取り組み経験によって，提案ツールやセミナーに対する回答が異なるかどうかも調べたが,特筆すべき有意な差は見られなかった.

演習問題の解答については簡単な分析のみを行ったが，これについても傾向などは特に見られなかった．ただし，セミナー中のやり取りで有用な感触は多く得られたため，今後深掘りした実証も行っていきたい．例えば，テスト駆動開発を行う状況で，ある機能の実現に対して必要であるテストケースを検討してもらったところ，人によって非常に大きな差が出た．1.1 で挙げたテスト駆動開発におけるテスト設計の難しさ，本研究により支援される根拠の明確化やそれに応じた議論の重要性が示されていた.

3.5.3 課題とその対応

以上で述べたように，今回のセミナーは実施時間，参加者の層，ツールの完成度などにおいて不十分であった側面があった．しかし 1 日のセミナーは参加者を広く募る上では適切であるため，ツールの改良も続けながら継続的に開催をしたい．一方で，実施機関における産業界教育プログラム「トップエスイー」[33]における講義として提案するなど，より長い時間をかけられるようにする，より深掘りできる演習データなどを集めることも目指したい.

4 考察

4.1 判明した効果や課題と今後の研究予定

【意義・効果のまとめ】

本研究の根幹は，本質的には表裏一体である仕様やテスト設計に関する一般規則・性質・命題と，テストケース・シナリオ・具体例とを照らし合わせることの支援を実現したことにある．これにより 1.3 にて述べたように，前者に主眼をおく分野やタスク，後者に主眼をおく分野やタスクの支援を行う意義が得られると期待される．アンケート結果（3.5.2）にも見られるように，提案言語・ツールの活用方法は，開発者の背景や想いに応じて様々なものを挙げることができ，要求仕様の洗練，テスト駆動開発におけるテスト設計の補助，単体テスト工程での品質保証テスト設計補助など，組織・プロジェクト・課題意識に応じた活用ができるようになっている．

また本研究は同時に，一見異なる方向性として捉えられがちな分野が本質的につながっていることを示すものである．このため，意識向上や教育にも有効性が高いと考えられる．このこともアンケート結果（3.5.2）に示されている．

本研究では技術の研究開発だけでなく，産業界の技術者を対象としたセミナーを構築し，実施した．これにより，技術の実証評価を行うとともに，意識向上や教育の観点から今すぐ産業界に展開できる成果を得ることができた．

以上は本研究の開始時点においてもある程度想定していた意義である．一方，本研究の開始時点においては，どのような特色を追求すべきであるのか，あるいは特色をどのような言葉や対比で主張していくのかはそれほど明確ではなかった．本研究は，先端的な技術の具体化・実用化のための枠組みで実施された．このため特に学術的観点からは，言語およびツールについては，本質的にはありがちな単なる Alloy のラッピングに過ぎないというところもある．しかし実際に Alloy を出発地点にして研究を進め，テスト駆動開発やその発展系に関する調査を踏まえて融合的な考え方を追求し，少なくとも概念的，既存研究や異なるパラダイムのとらえ方としては有意に異なる原則・アプローチを整理することができた．

- 性質による記述と例示による記述の融合
- 仕様，テスト設計，テストケース（具体例）の混在した記述
- 早いフィードバックを与えるツール
- 仕様を得るタスクおよびテストケースを得るタスク双方への活用
- 例と反例双方の確認
- 命題に対するテストケース・テスト駆動開発

また技術的には具体化・実用化であるとしても，産業界の技術者に試用してもらい，本節の最初に述べたように様々な活用方法における有用性に関する評価を得たことは非常に大きな意義があると考えている．正確には上記の原則・アプローチのうち最後のものについては，新規性の高いものでありセミナーでは扱っていない．それ以外の原則・アプロー

チについては，個々に関する調査を詳細に行ったわけではないものの，提案言語・ツール，およびセミナーに対してよい評価を得ることができたと判断している（3.5.2）．

このうち特にセミナーについては，今すぐに展開（ある意味での実用化）が可能であるため，その有用性を確認できたことは意味があると考えている．

【今後の活動】

3.1.3 にて技術的な課題として述べ，実際にも 3.5.2 においてアンケート結果に表れたように，実行速度を高めるための本格的な技術開発が重要な課題である．これは先進研究として取り組んでいく．

一方で 3.5.2 においてアンケート結果に表れたもののうち，実用化のためのフォーマット変換，ユーザビリティーの向上，ドキュメントの充実などについては，むやみに工数をかけていくことは難しいものの，可能な範囲で取り組んでいく．

そのためにも，本研究で行ったものの改訂版となる 1 日セミナー，あるいはより長い 7 コマなどの講義についても，しっかり改訂を進めながら継続的に行っていきたい．

4.2 研究成果の産業界への展開について

上記のようにセミナーは定期的，継続的に開催していき，成果の周知に努める．またより踏み込んだ活用に向けて，研究責任者や研究チームが共同研究などを行っているパートナー（企業あるいは個人）とともに活用に取り組んでいく．具体的には，まずセミナーの受講を通して本研究の成果を知ってもらうことがスタート地点となる．研究期間内に開催したセミナーにおいては集客に成功しており，セミナー自体の教育効果についても一定の評価が得られたことから，セミナーを受講してもらうという第一歩は比較的容易であると考えている．提案言語・ツールの活用については，本報告書でこれまで述べてきたように，仕様明確化の支援，テスト駆動開発におけるテスト設計の視点，品質保証におけるテスト設計の支援など，いくつかの活用方法が考えられる．しかしいきなり組織的に導入し取り組むことは難しく，また研究の範囲外となる一般的なツール機能などについてもそういった実用化には十分とは言いがたい．このためまずは，電卓や表計算ソフトのように，個人がちょっとした論理の確認に活用するような形を考えるのがよいかもしれない．セミナーの受講者の中には，今後の活用方法などをブログに掲載してくれた方もおり，こういった方々の活用経験を通し，ツール機能の充実や活用プロセスの例示などを進めていくのがよいであろう．

最後に，本研究の成果および経験を受けて，産業界の方々に期待すること，一緒に取り組んでいきたいことについて述べる．産業界には，本研究のセミナーに限らず，多くのセミナーや講義，イベントなどに参加していただき，非常に積極的に情報の収集や発信を行っている人もいる．そういった方々にはこれまでも，本研究においても，多くの議論を通し様々なフィードバックをいただき，非常に感謝をしている．その一方，「テストが担当だからテストのイベントに出る」というようなやや狭い視点での活動を行っている人や，そもそも外部のイベントや勉強会などにほとんど参加していない，できていない人もいる．今回の研究で実施したセミナーなどをきっかけとして，表面的な分野の名前や，今自身が知っていること，自社でのやり方に捕らわれず，広い視点をもって一緒に様々な議論を行うように一歩一歩踏み出してもらう助けになれればと考えている．

こういった活動を通して，研究開発の場と開発の現場のよい相互作用を産み出していくようにしていきたい．

参考文献

[1] [オンライン]. Available: http://www.agilemanifesto.org/.

[2] K. Beck, Test Driven Development: By Example, Addison-Wesley Professional, 2002.

[3] F. Steven , P. Nat, Growing Object-Oriented Software, Guided by Tests, Addison-Wesley Professional, 2009.

[4] A. Gojko, Specification by Example: How Successful Teams Deliver the Right Software, Manning Pubns Co, 2011.

[5] [オンライン]. Available: http://cukes.info/.

[6] J. Fitzgerald, P. G. Larsen, P. Mukherjee, N. Plat, M. Verhoef , 酒. 寛（訳）, VDM++によるオブジェクト指向システムの高品質設計と検証, 翔泳社, 2010.

[7] 来間啓伸 , 中島震（監修）, Bメソッドによる形式仕様記述, 近代科学社, 2007.

[8] J.-R. Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.

[9] “The Java Modeling Language (JML) Home Page,” [オンライン]. Available: http://www.eecs.ucf.edu/~leavens/JML/index.shtml.

[10] “Frama-C,” [オンライン]. Available: http://frama-c.com/.

[11] 情報処理推進機構, “「形式手法適用調査」報告書：IPA 独立行政法人 情報処理推進機構,” [オンライン]. Available: http://www.ipa.go.jp/sec/softwareengineering/reports/20100729.html.

[12] “Alloy,” [オンライン]. Available: http://alloy.mit.edu/alloy/.

[13] D. Jackson, 中島震, 今井健男, 酒井政裕, 遠藤侑介 , 片岡欣夫, 抽象によるソフトウェア設計－Alloyではじめる形式手法－, オーム社, 2011.

[14] PictMaster, “PictMaster プロジェクト日本語トップページ - SourceForge.JP,” [オンライン]. Available: http://sourceforge.jp/projects/pictmaster/.

[15] “Model-based Testing with SpecExplorer - Microsoft Research,” [オンライン]. Available: http://research.microsoft.com/en-us/projects/specexplorer/.

[16] “CEGTest - 原因結果グラフからテスト条件を作成するツール,” [オンライン]. Available: http://softest.jp/tools/CEGTest/.

[17] “ZIPC Tester｜製品情報｜状態遷移表設計で品質向上 キャッツ,” [オンライン]. Available: http://www.zipc.com/product/ZIPC_Tester/.

[18] “Xtext - Language Development Made Easy!,” [オンライン]. Available: http://www.eclipse.org/Xtext/.

[19] S. Black, P. P. Boca, J. P. Bowen, J. Gorman , M. Hinchey, “Formal Versus Agile: Survival of the Fittest,” *IEEE Computer,* 第 巻 42, 第 9, pp. 37-45, 2009.

[20] S. M. Suhaib, D. A. Mathaikutty, S. K. Shukla , D. Berner, “XFM: An Incremental

Methodology for Developing Formal Models," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 第 巻 10, 第 4, pp. 589-609, 2005.

[21] S. Abdul Khalek, G. Yang, L. Zhang, D. Marinov , S. Khurshid, "TestEra: A tool for testing Java programs using alloy specifications," 著: *The 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, 2011.

[22] 岡. 楠. 森恵弥佳, "Alloy Analyzer を用いた表明に関する欠陥の検出手法 -JML による表明記述に対して-," 第 巻 30, 第 3, pp. 187-193, 2013.

[23] T. Nelson, S. Saghafi, D. J. Dougherty, K. Fisler , S. Krishnamurthi, "Aluminum: Principled Scenario Exploration Through Minimality," 著: *The 2013 International Conference on Software* , 2013.

[24] C. Acheco , M. D. Ernst, "Randoop: Feedback-directed Random Testing for Java," 著: *The 22nd ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications Companion (OOPSLA 2007)*, 2007.

[25] K. Sen, "Concolic Testing," 著: *The 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*, 2007.

[26] 石川冬樹, "サービス指向コンピューティングにおける合意に基づいた協調的な移動性 (研究会推薦博士論文速報・ソフトウェア工学研究会推薦)," *情報処理学会論文誌*, 第 巻 49, 第 6, p. 678, 2008.

[27] F. Ishikawa, N. Yoshioka , S. Honiden, "Developing Consistent Contractual Policies in Service Composition," 著: *The 2007 IEEE Asia-Pacific Services Computing Conference (IEEE APSCC 2007)*, 2007.

[28] F. Ishikawa, R. Inoue , S. Honiden, "Modeling, Analyzing and Weaving Legal Interpretations in Goal-Oriented Requirements Engineering," 著: *The 2nd International Workshop on Requirements Engineering and Law (RELAW'09)*, 2009.

[29] F. Ishikawa, R. Kawai (Inoue) , S. Honiden, "Modeling and Analyzing Legal Interpretations for/by Requirements Engineering Approaches," 著: *The 6th International Workshop on Juris-informatics (JURISIN 2012)*, 2012.

[30] "ClouT | ClouT : Cloud of Things for empowering the citizen clout in smart cities," [オンライン]. Available: http://clout-project.eu/ja/.

[31] 石川冬樹, 山本佳代子 , 本位田真一, "物理的相互作用に着目した, スマート空間の形式仕様記述と検証," *情報処理学会論文誌*, 第 巻 52, 第 1, pp. 220-232, 2011.

[32] T. Kobayashi, F. Ishikawa , S. Honiden, "Understanding and Planning Event-B Refinement through Primitive Rationales," 著: *The 4th International ABZ 2014 Conference*, 2014.

[33] "トップエスイー | サイエンスによる知的ものづくり教育プログラム," [オンライン]. Available: http://www.topse.jp/.

[34] F. Ishikawa, K. Taguchi, N. Yoshioka , S. Honiden, "What Top-Level Software

Engineers Tackles after Learning Formal Methods - Experiences from the Top SE Project," 著: *The 2nd International FME Conference on Teaching Formal Methods (TFM 2009)*, 2009.

[35] 石川冬樹 , 荒木啓二郎（監修）, VDM++による仕様記述, 2011.

[36] "日科技連 | ソフトウェア品質 | SQiP 研究会," [オンライン]. Available: http://juse-sqip.jp/workshop/index/.

[37] "GRACE センター | 先端ソフトウェア工学・国際研究センター," [オンライン]. Available: http://grace-center.jp/.