# 不確かさを包容するテスト支援

渡辺啓介[†1] 深町拓也[†1] 鵜林尚靖[†1]
細合晋太郎[†1] 亀井靖高[†1] 渡邉卓也[†2]

ソフトウェア開発において，例えば未確定の設計や実装といった，既知の不確かさが存在する状態で開発を継続するとき，不確かさに対する複数の可能性のそれぞれに基づいて単体テストを行う際には，それぞれの可能性に対応してテストケースの複製が必要になる．これは，保守性や労力の観点からは問題がある．そこで，筆者らが提供してきたインターフェース機構 Archface-U，および統合開発環境 iArch-U を基盤として，アスペクト指向プログラミングにより単一のテストケースを柔軟に運用し，不確かさを包容するテストを支援する方法を提案する．

## 1. はじめに

ソフトウェア開発においては，曖昧さを取り除き設計を確実にすることが重視されてきたが，現実には不確かさ (Uncertainty) を避けることは困難であることが指摘されている[2]．不確かさは未知であるか既知であるかといった観点から複数のレベルに分類することができる[2] が，我々はこのうち未知の不確かさへの対応は難しいものの，未確定の設計やアルゴリズムのような，既知の不確かさに対応することは可能であると考えた．そこで我々は，後述するインターフェース機構 Archface-U[1]，および統合開発環境 iArch-U[1] を提供してきた．これらにより既知の不確かさを適切に記述し，また不確かさを抱えたまま実装を進めることが可能になった．

ところで，不確かさを抱えたまま開発を行うとき，単体テストに際して懸念される問題がある．例えば，複数の実装方法が考えられ，どの実装方法を採用するか決まっていないという不確かさが存在するとき，その中である特定の実装方法を用いる場合をそれぞれテストすることが必要になる．しかし，複数の実装方法に対して一対一にテストケースを作成することは労力がかかり，実装方法の候補が増える度にテストケースを作成する必要に迫られる．また，将来不確かさが解消される場合，その労力はむだであるといえる．

この問題に対応するため，我々は統合開発環境 iArch-U を拡張し，不確かさに基づいたテスト支援を試みた．拡張された iArch-U は，AspectJ[⋆1] の文法に基づいて実装方法の候補に対応するアスペクトを生成することで，単一のテストケースをそれぞれの実装方法に対して実行することを可能にする．

## 2. インターフェース機構 Archface-U

ここでは，我々が提案してきた技術 Archface-U について概要を述べる．Archface-U は，設計と Java による実装の仲立ちをするインターフェース機構である Archface[3] を不確かさが記述できるように拡張したものである．Component-and-Connector アーキテクチャに基づいて，開発者は Archface-U にコンポーネントインターフェース，およびコネクターインターフェースを定義することができ，前者にはクラスとメソッドの宣言を，後者にはメソッドの呼び出し関係を記述することができる．

Archface-U と対になる統合開発環境 iArch-U は，Eclipse のプラグインとして実装されており，Archface-U に定義された各インターフェースに Java ソースコードの実装が従っているかどうかを検査し，その結果を開発者に提供する．

Archface-U には 2 種類の不確かさを記述することができる．1 つ目は，いくつかのコンポーネントの候補の内どれを採用するか分からないという不確かさで，これを Alternative と呼び，メソッドを {} で囲むことで表現する．2 つ目は，あるコンポーネントを採用するか分からないという不確かさで，これを Optional と呼び，メソッドを `[]` で囲むことで表現する．

文法の説明のため，**図 1** に P2P ファイル共有システムにおける Archface-U の記述を例示する．コンポーネントインターフェースは Java のインターフェースの文法で記述される．この例では，`Node` クラスには `start` メソッドが宣言される必要があるが，`restart` メソッドは Optional であるため，宣言して

†1 九州大学
†2 エヂリウム株式会社
⋆1 https://eclipse.org/aspectj/

```
interface component Node {
  void start();
  void cancel();
  void completed();
}
uncertain component uNode extends Node {
  [void restart();]
  {void share(File file);,
   void bigDataShare(File file);}
}
interface connector cP2PSys {
  Node = (Node.start -> Node.cancel -> Node);
}
uncertain connector ucP2PSys extends cP2PSys {
  Node = (Node.start -> Node.completed ->
    [uNode.restart] -> Node.cancel -> Node);
}
```

**図 1** P2P ファイル共有システムの Archface-U による記述例

もしなくてもよい．コネクターインターフェースは FSP(Finite State Process) を基礎とした文法で記述され, この例では, `start` メソッド中で `cancel` メソッドが呼び出される必要がある一方，`restart` は Optional であるため，`start -> completed -> restart -> cancel` か，`start -> completed -> cancel` のいずれかの呼び出し関係が成り立てば良い．

## 3. 不確かさを包容するテスト支援の実現

ここでは，前掲した例に基づき，我々が今回提案するテスト支援の方法を説明する．P2P ファイル共有システムの開発において，ファイルの共有を開始するメソッドとして，`share` と `bigDataShare` のどちらを採用するかが決まっていないという不確かさが発生したとする．図 1 の `uNode` に着目すると，この不確かさは Alternative として記述されている．このとき，実装においては `share` と `bigDataShare` のいずれかが宣言されている必要があるが，前述したように，テストケースを作成するとき，そうした可能性のそれぞれに対してテストケースを作成するのは良い方法ではない．そこで，AspectJ の文法に基づいて，それぞれの可能性に即したアスペクトを自動的に生成することを考える．すなわち，`share` と `bigDataShare` のうち，`bigDataShare` を宣言するという可能性に対しては，execution ポイントカットを用いて，`share` の処理を `bigDataShare` の処理に置き換えるアスペクトを生成する (**図 2**).

例えば，メソッド `share` についてのテストケースを既に作成しているときに，`share` の代わりに `bigDataShare` を用いることが検討された場合，生成されたアスペクトをテストケースに適用することで，

```
@Around("execution(void Node.share(File))
  && this(__this) && args(file)")
public void share(Node __this, File file) {
  __this.bigDataShare(file);
}
```

**図 2** 自動的に生成されるアスペクトの例

そのテストケースを編集することなく，ファイルの共有を開始するメソッドとして `bigDataShare` を用いる場合をテストすることができる．

我々は iArch-U を拡張し，以上に述べたようなテスト支援を可能にした．すなわち，不確かさに対する特定の可能性について，それに即したアスペクトを自動生成することができ，その操作は全て Eclipse の GUI 上で行うことができる．

## 4. 今後の課題

第一に，複数の不確かさに対する全ての可能性に同時にテストを行うことが考えられる．しかしこの場合，可能性の数が膨大となるので，適切に取捨選択した上でテストを行うことを検討する．

第二に，これまでの議論では，テストケースにおける前提条件や期待結果が不確かさに対するどの可能性においても同一であることを前提としているが，実際にはそれらが異なる場合が考えられるので，場合によって切り替える機能を加えることを検討する．

**謝辞:** 本研究は，文部科学省科学研究補助費 基盤研究 (A) (課題番号 26240007) による助成を受けた．

## 参 考 文 献

1) Fukamachi, T., Ubayashi, N., Hosoai, S. and Kamei, Y.: Modularity for Uncertainty, *Proceedings of the Seventh International Workshop on Modeling in Software Engineering*, pp.7–12 (2015).
2) Perez-Palacin, D. and Mirandola, R.: Uncertainties in the Modeling of Self-adaptive Systems: a Taxonomy and an Example of Availability Evaluation, *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, pp.3–14 (2014).
3) Ubayashi, N., Nomura, J. and Tamai, T.: Archface: A Contract Place Where Architectural Design and Code Meet Together, *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pp.75–84 (2010).