# react cop2

ono

2024 年 1 月 8 日

# 目 次

# 1 はじめに

## 1.1 研究背景

## 1.2 研究課題

## 1.3 研究目的

## 1.4 本論文の構成

# 2 関連研究

## 2.1 COP の概要

## 2.2 React COP

## 2.3 EventCJ に複合層を導入

# 3 提案手法

本研究では、関連研究にある複合層、多層の機能追加 react cop の改善点を洗い出し、それを解決するための機能を追加した。本章では、本研究で提案する手法について述べる。

ReactCOP では、レイヤーの活性化情報を useState を用いて管理している。state 変数はクラスのインスタンスを用いている。しかし、state 変数は参照型のため、state 変数の値を変更しても再レンダリングが行われない。そのため、レイヤーの活性化情報を更新しても、再レンダリングが行われない。これを解決するために、state 変数を参照型のクラスのインスタンスから、イミュータブルなデータに変更する必要がある。この変更は react cop を大きく刷新することと同等であるため、ReactCOP2 ではとして新しく実装することとした。

ReactCOP では、レイヤーのパラメーターと layer の活性化情報を別で管理している。そのため、レイヤーの活性化とレイヤーのパラメーターで 2 度レイヤー名を指定する必要がある。これは、レイヤーの管理が煩雑になる原因となっている。この問題を解決するために、レイヤーを一つのオブジェクトとして管理することとした。

以降では、ReactCOP2 の実装について述べる。ReactCOP で代替となる機能を提供するために、react cop2 では、以下の機能を提供する。ReactCOP2 では、カスタムコンテキストを用いて、レイヤーの操作や取得などの機能を提供する。react cop から改善された機能を以下に示す。

## 3.1 代替する機能

### (1) レイヤーパラムの設定・取得

React cop では、レイヤーパラムの設定・取得ができる useLayerParams というカスタムフックを提供している。このカスタムフックは、以下のように使用する。

```
1 // レイヤーパラム
2 const [getHoge, setHoge] = useLayerParams('hoge', ["Hoge"]);
3 getHoge() // hoge
4 setHoge('fuga',"Hoge")
5 getHoge() // fuga
```

useLayerParams の第一引数には、レイヤーパラムの初期値、第二引数には、レイヤー名を指定することで、レイヤーパラムの値を設定できる。レイヤー名は複数指定することができる。もしレイヤーが存在しない場合は、新しくレイヤーを定義する。これは、意図しないレイヤーが簡単に定義できてしまう。setHoge の第一引数には、レイヤーパラムの値、第二引数には、レイヤー名を指定することでレイヤーパラムの値を設定できる。getHoge の引数には、レイヤー名を指定することで、レイヤーパラムの値を取得できる。引数にレイヤー名を指定しない場合は、活性化したレイヤーのレイヤーパラムの値を取得する。ただし、活性化したレイヤーが複数ある場合は、最初に取得したレイヤーのレイヤーパラムの値を取得する。レイヤーの並び順は、レイヤー名を登録した順番である。

ReactCOP2 では、レイヤーパラムの設定ができる setLayerParams、取得ができる getLayersParam というメソッドを提供する。このメソッドは、以下のように使用する。setLayerParams の第一引数には、レイヤー名、第二引数にはレイヤーパラムの値を指定することで、レイヤーパラムの値を設定できる。getLayerParams の引数には、レイヤー名を指定することで、レイヤーパラムの値を取得できる。

### (2) レイヤーの活性化・非活性化

ReactCOP では、レイヤーの活性化・非活性化ができる useLayerManager というカスタムフックを提供している。このカスタムフックは、以下のように使用する。

```
1 const layerManager = useLayerManager();
2 layerManager.activateLayer("Float");
3 layerManager.deactivateLayer("Integer");
```

activateLayer の引数には、レイヤー名を指定することで、レイヤーを活性化できる。deactivateLayer の引数には、レイヤー名を指定することで、レイヤーを非活性化できる。

ReactCOP2 では、レイヤーの活性化・非活性化ができる activateLayer、inactivateLayer というメソッドを提供する。このメソッドは、以下のように使用する。activateLayer の引数には、レイヤー名を指定することで、レイヤーを活性化できる。inactivateLayer の引数には、レイヤー名を指定することで、レイヤーを非活性化できる。

### (3) レイヤーの活性化情報の取得

ReactCOP では、レイヤーの活性化情報を取得できる useLayerManager というカスタムフックは getLayerState というメソッドを提供している。このメソッドは、以下のように使用する。

```
1 const layerManager = useLayerManager();
2 const layerState = layerManager.getLayerState();
3
4 layerState.Float
5 layerState.Integer
```

layerState. レイヤー名で、レイヤーの活性化情報を取得できる。ただし、あらかじめレイヤーを活性化・非活性化していないと、レイヤーの活性化情報は取得でず、undefined が返される。

ReactCOP2 では、レイヤーの活性化情報を取得できる getLayer というメソッドを提供する。このメソッドは、以下のように使用する。getLayer の引数には、レイヤー名を指定することで、レイヤーの情報を取得できる。

### (4) レイヤーが活性化しているかどうかの判定

ReactCOP では、レイヤーが活性化しているかどうかの判定ができる useLayerManager というカスタムフックは isActiveLayer というメソッドを提供している。このメソッドは、以下のように使用する。

```
const layerManager = useLayerManager();
// レイヤーがactive かどうかを判定
layerManager.isActiveLayer("Float")// true or false
```

isActiveLayer の引数には、レイヤー名を指定することで、レイヤーが活性化しているかどうかを判定できる。レイヤーが活性化している場合は、true が返される。

ReactCOP2 では、レイヤーが活性化しているかどうかの判定ができるメソッドは提供しない。代わりに、レイヤーの活性化情報を取得できる getLayer メソッドから、レイヤーが活性化情報を取得すれば、レイヤーが活性化しているかどうかの判定ができる。TODO: ここにコードを書く

## 3.2 改善点

### (1) 改善点一覧

- layer の de/active 時に新しいレイヤーを定義できないようにする react cop では、layer の de/active 時に新しいレイヤーを定義できてしまう。そのため、意図しないレイヤーが簡単に定義できてしまう。またレイヤーの管理が煩雑になる。コレを解決するために、layer の de/active 時に新しいレイヤーを定義できないようにする
- layer params の値を入れるときに新しい layer を定義できないようにする
- typescript での実装
- テストの追加

### (2) layer の de/active 時に新しいレイヤーを定義できないようにする

ReactCOP では、layer の de/active 時に新しいレイヤーを定義できてしまう。layer の de/active 時に新しいレイヤーを定義できると意図しないレイヤーが簡単に定義できてしまう。またレイヤーの管理が煩雑になる。

ReactCOP2 では、layer の de/active 時に新しいレイヤーを定義できないようにする。これによって、意図しないレイヤーが簡単に定義できなくなり、レイヤーの管理が煩雑にならない。

### (3) layer の活性化条件を定義できる

## 3.3 機能追加

### (1) 機能追加一覧

- アクティブなレイヤーを取得する getActiveLayers: () =¿ Layers;
- レイヤーのパラメーターを取得する getLayersParam: (names: string[]) =¿ any[]; getAllLayersParams: () =¿ any; getActiveLayersParams: () =¿ any; getLayerParamsByGroup: (group: string) =¿ any;
- グループを指定してレイヤーを取得する getLayersByGroup: (group: string) =¿ Layers; getActiveLayersByGroup: (group: string) =¿ Layers; getInactiveLayersByGroup: (group: string) =¿ Layers;
- レイヤーを追加する addLayer: (layer: Layer) =¿ void;
- レイヤーを削除する removeLayer: (name: string) =¿ void;
- レイヤーの活性化を切り替える toggleLayer: (name: string) =¿ void;

- グループの設定をする setGroupConfig: (name: string, config: GroupConfig) =¿ void;
- レイヤーにグループを設定する setLayerGroup: (name: string — string[], group: string) =¿ void;
- レイヤーの活性化条件を設定する addDependency: (name: string, dependencies: DependencyGroup) =¿ void;

## 3.4 アクティブなレイヤーを取得する

ReactCOP でアクティブなレイヤーを取得するためには、レイヤーの活性化情報を取得し、活性化しているレイヤーを取得する必要がある。これは、レイヤーの管理が煩雑になる原因となっている。この問題を解決するために、アクティブなレイヤーを取得する機能を提供することで、レイヤーの管理が簡単になる。

## 3.5 レイヤーのパラメーターを取得する

### (1) 複数のレイヤーのパラメーターを取得する

ReactCOP では、複数のレイヤーのパラメーターを取得するためには、レイヤーのパラメーターを取得するためには、レイヤー名を指定してレイヤーのパラメーターを取得する必要がある。少し不便であるため、複数のレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

### (2) 全てのレイヤーのパラメーターを取得する

ReactCOP では、全てのレイヤーのパラメーターを取得するためには、レイヤーのパラメーターを取得するためには、レイヤー名を指定してレイヤーのパラメーターを取得する必要がある。少し不便であるため、全てのレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

### (3) アクティブなレイヤーのパラメーターを取得する

ReactCOP でアクティブなレイヤーのパラメーターを取得するためには、レイヤーが活性化しているかどうかの判定を行い、活性化しているレイヤーのパラメーターを取得する必要がある。これは、少し不便であるため、アクティブなレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

### (4) グループを指定してレイヤーのパラメーターを取得する

RectCOP では、そもそもグループの概念がないため、グループを指定してレイヤーのパラメーターを取得する機能は提供していない。しかし、グループを指定してレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

## 3.6 レイヤーを追加する

ReactCOP では、明確にレイヤーを追加する機能を提供していない。レイヤーを活性化させるときやレイヤーのパラメーターを設定するときに、もしレイヤーが存在しない場合はレイヤーを追加するという処理を行っている。これは、いつレイヤーが追加されるかわからないため、レイヤーの管理が煩雑になる。

ReactCOP2 では、明確にレイヤーを追加する機能を提供することで、レイヤーの管理が簡単になる。

### 3.7　レイヤーを削除する

必要がないレイヤーを削除することで、レイヤーの管理が簡単になる。

### 3.8　レイヤーの活性化を切り替える

ReactCOP では、レイヤーの活性化を切り替える機能を提供していない。レイヤーの活性化を切り替えるには、レイヤーの活性化情報を取得し、活性化しているかどうかの判定を行い、活性化している場合は、レイヤーを非活性化し、非活性化している場合は、レイヤーを活性化する必要がある。これは、少し手間であるため、レイヤーの活性化を切り替える機能を提供することで、レイヤーの管理が簡単になる。

### 3.9　グループの設定をする

ReactCOP では、そもそもグループの概念がないため、グループの設定をする機能は提供していない。グループとは、レイヤーをグループ化することである。グループを設定することで、グループ内のレイヤー内で活性化するレイヤーを取得するなどの機能を提供することができる。

ReactCOP2 では、グループの設定をする機能を提供することで、レイヤーの管理が簡単になる。

### 3.10　レイヤーにグループを設定する

レイヤーがどのグループに属するか設定することができる。

### 3.11　レイヤーの活性化条件を設定する

レイヤーの活性化条件を設定することができる。レイヤーの活性化条件とは、レイヤーが活性化する条件である。例えば、レイヤー A とレイヤー B があるとき、レイヤー A が活性化しているときに、レイヤー B を活性化するという条件を設定することができる。条件の指定方法は、レイヤーの活性化状態の論理積、論理和を指定できることである。

#### (1)　Typescript での実装

ReactCOP では、Typescript での実装をしていない。Typescript での実装をすることで、コードの可読性と保守性が向上する。

#### (2)　テストの追加

ReactCOP では、テストをしていない。テストは、コードの品質を保つために必要である。

### 3.12　評価方法

- 実装前と後で、できることの違いを比較する。

## 4　実装

大まかな構成は ReactCOP と同じである。useState を用いてレイヤー情報を管理する。レイヤー情報は Provider を用いて、子コンポーネントに渡す。子コンポーネントは、useContext を用いてレイヤー情報を取得する。

## 4.1 レイヤーの型

レイヤーの型は、以下のように定義する。

ソースコード 1: レイヤーの型

```
export interface Layer {
    name: string;
    isActive: boolean; // state
    group: string;
    params: any;
    dependencies?: DependencyGroup;
}
```

name はレイヤーの名前である。isActive はレイヤーが活性化しているかどうかを表す。group はレイヤーのグループ名である。params はレイヤーパラムである。dependencies はレイヤーが活性化する条件である。

## 4.2 代替する機能

### (1) レイヤーパラムの設定・取得

### (2) レイヤーパラムの設定

レイヤーパラムの設定は setLayerParams というメソッドを提供している。型は以下のように定義する。

ソースコード 2: setLayerParams の型

```
setLayerParams: (name: string, params: Record<string, any>) => void;
```

レイヤー名とレイヤーパラムを指定することで、レイヤーパラムの値を設定できる。レイヤーパラムはレイヤーの params に格納される。ReactCOP ではレイヤーが存在しない場合は、新しくレイヤーを定義でき、意図しないレイヤーが混入する可能性があった。ReactCOP2 では、レイヤーが存在しない場合は、エラーを返すようにした。

### (3) ひとつのレイヤーのレイヤーパラムを取得

ReactCOP では、レイヤーパラムの設定・取得ができる getLayerParam というメソッドを提供している。型は以下のように定義する。

ソースコード 3: getLayerParam の型

```
getLayerParam: (name: string) => any;
```

レイヤー名を指定することで、レイヤーパラムの値を取得できる。レイヤーパラムは何を設定しても良いため、any 型である。

このメソッドは、以下のように使用する。

```
getLayerParam('hoge')
```

### (4) 複数のレイヤーのレイヤーパラムを取得

複数のレイヤーパラムを取得したい場合は getLayersParam というメソッドを提供している。型は以下のように定義する。

ソースコード 4: getLayersParam の型

```
1 getLayersParam: (names: string[]) => any[];
```

レイヤー名を配列で指定することで、レイヤーパラムの値を配列で取得できる。レイヤーパラムは何を設定しても良いため、any 型である。

このメソッドは、以下のように使用する。

```
1 getLayersParam(['hoge','fuga'])
```

### (5) レイヤーの活性化・非活性化

レイヤーの活性化・非活性化は、activateLayer と inactivateLayer というメソッドを提供している。型は以下のように定義する。

ソースコード 5: activateLayer と inactivateLayer の型

```
1 activeLayer: (name: string) => void;
2 inactiveLayer: (name: string) => void;
```

レイヤー名を指定することで、レイヤーの活性化・非活性化ができる。具体的にはレイヤーの isActive を活性化状態なら true、非活性化状態なら false にする。ReactCOP ではレイヤーが存在しない場合は、新しくレイヤーを定義でき、意図しないレイヤーが混入する可能性があった。ReactCOP2 では、レイヤーが存在しない場合は、エラーを返すようにした。

### (6) レイヤーの活性化情報の取得

レイヤーの活性化情報を取得したい場合は getLayer というメソッドを提供している。型は以下のように定義する。

ソースコード 6: getLayer の型

```
1 getLayer: (name: string) => Layer;
```

レイヤー名を指定することで、レイヤー情報を取得できる。活性化状態はレイヤーの中に isActive として格納されている。

### (7) アクティブなレイヤーの取得

アクティブなレイヤーを取得したい場合は getActiveLayers というメソッドを提供している。型は以下のように定義する。

ソースコード 7: getActiveLayers の型

```
1 getActiveLayers: () => Layers;
```

#### (8) 複数のレイヤーのパラメーターを取得する

複数のレイヤーのパラメーターを取得したい場合は getLayersParam というメソッドを提供している。型は以下のように定義する。

ソースコード 8: getLayersParam の型

```
1 getLayersParam: (names: string[]) => any[];
```

レイヤー名を配列で指定することで、レイヤーパラムの値を配列で取得できる。

#### (9) 全てのレイヤーのパラメーターを取得する

全てのレイヤーのパラメーターを取得したい場合は getAllLayersParam というメソッドを提供している。型は以下のように定義する。

ソースコード 9: getAllLayersParam の型

```
1 getAllLayersParams: () => any;
```

#### (10) アクティブなレイヤーのパラメーターを取得する

アクティブなレイヤーのパラメーターを取得したい場合は getActiveLayersParam というメソッドを提供している。型は以下のように定義する。

ソースコード 10: getActiveLayersParam の型

```
1 getActiveLayersParams: () => any;
```

#### (11) グループを指定してレイヤーのパラメーターを取得する

グループを指定してレイヤーのパラメーターを取得したい場合は getLayerParamsByGroup というメソッドを提供している。型は以下のように定義する。

ソースコード 11: getLayerParamsByGroup の型

```
1 getLayerParamsByGroup: (group: string) => any;
```

### 4.3 レイヤーを追加する

レイヤーを追加するには、addLayer というメソッドを提供している。型は以下のように定義する。

ソースコード 12: addLayer の型

```
1 addLayer: (layers: Layer[]) => void;
```

レイヤーを配列で指定することで、レイヤーを追加できる。

### 4.4 レイヤーを削除する

レイヤーを削除するには、removeLayer というメソッドを提供している。型は以下のように定義する。

ソースコード 13: removeLayer の型

```
1 removeLayer: (name: string) => void;
```

レイヤー名を指定することで、レイヤーを削除できる。

### 4.5 レイヤーの活性化を切り替える

レイヤーの活性化を切り替えるには、toggleLayer というメソッドを提供している。型は以下のように定義する。

ソースコード 14: toggleLayer の型

```
1 toggleLayer: (name: string) => void;
```

### 4.6 グループの設定をする

グループの設定をするには、setGroupConfig というメソッドを提供している。型は以下のように定義する。

ソースコード 15: setGroupConfig の型

```
1 setGroupConfig: (name: string, config: GroupConfig) => void;
```

グループ名とグループの設定を指定することで、グループの設定をすることができる。

### 4.7 レイヤーにグループを設定する

レイヤーにグループを設定するには、setLayerGroup というメソッドを提供している。型は以下のように定義する。

ソースコード 16: setLayerGroup の型

```
1 setLayerGroup: (name: string | string[], group: string) => void;
```

レイヤー名とグループ名を指定することで、レイヤーにグループを設定することができる。

### 4.8 レイヤーの活性化条件を設定する

レイヤーの活性化条件を設定するには、addDependency というメソッドを提供している。型は以下のように定義する。

ソースコード 17: addDependency の型

```
1 addDependency: (name: string, dependencies: DependencyGroup) => void;
```

レイヤー名と活性化条件を指定することで、レイヤーの活性化条件を設定することができる。

## 5 評価

本章では、実装した機能の評価を行う。実装した機能によって何ができるようになったか、改善されたかを示す。

### 5.1 state 変数の変化

ReactCOP では、レイヤーの活性化情報を useState を用いて管理している。state 変数はクラスのインスタンスを用いている。しかし、state 変数は参照型のため、state 変数の値を変更しても再レンダリングが行われない。そのため、レイヤーの活性化情報を更新しても、再レンダリングが行われない。これを解決するために、state 変数を参照型のクラスのインスタンスから、イミュータブルなデータに変更する必要がある。この変更は react cop を大きく刷新することと同等であるため、ReactCOP2 ではとして新しく実装することとした。この実装によりレイヤーに何か操作を加えるたびに notifyUpdatedLayerState() を呼び出す必要がなくなりコードがより簡潔になった。

## 5.2 レイヤーの管理

ReactCOP では、レイヤーのパラメーターと layer の活性化情報を別で管理している。そのため、レイヤーの活性化とレイヤーのパラメーターで 2 度レイヤー名を指定する必要がある。これは、レイヤーの管理が煩雑になる原因となっている。この問題を解決するために、レイヤーを一つのオブジェクトとして管理することとした。この実装によりレイヤーの管理が簡単になった。

## 5.3 レイヤー操作機能の拡張性

ReactCOP では与えられたレイヤーの操作のみを行うことができる。しかしユーザーによっては、レイヤーの操作を拡張したい場合がある。ReactCOP2 ではレイヤーの操作機能を拡張するために、レイヤーを管理している useState の返り値である setLayers と layers をそのまま提供している。ユーザーはカスタムフックを自作で実装するよことにより、レイヤーの操作機能を拡張することができる。

## 5.4 グループの追加

ReactCOP では、レイヤーのグループを設定することができない。しかし実際にはレイヤーはグループに属していることが多い。例えば、晴れや雨などの天気のレイヤーは天気のグループに属している。私達が実際に認識していることを表現するためには、レイヤーのグループを設定することが必要である。グループを表現することにより私達にとって認識しやすくなる。ReactCOP2 では、レイヤーのグループを設定することができるようになった。

## 5.5 グループ内で活性化を排反にする

ReactCOP では複数のレイヤーが活性化している状態を許容している。場合によっては、複数のレイヤーが活性化している状態を許容したくない場合がある。同時に活性化を許容しているため ReactCOP のサンプルアプリである電卓アプリでは以下のように冗長な表現となる。

```
1 // ReactCOP
2 <Layer condition={layerState.Float && !layerState.Integer}>
```

少なくとも私の感覚としては、電卓のモード選択には Float かつ Integer が同時に活性化している状態は存在しないため、Float が活性化していることのみを condition に設定したい。ReactCOP2 では、グループ内で活性化を排反にすることができるようになった。

```
1 // ReactCOP2
2 <Layer condition={Float.isActive}>
```

このように、グループ内で活性化を排反にすることにより、冗長な表現を簡潔にすることができる。

## 5.6 複合層・多層

ReactCOP では、複合層・多層を実現することができない。複合層・多層は、レイヤーの活性化状態を条件として設定することで実現できる。ReactCOP2 では、複合層・多層を実現することができるようになった。

## 5.7 テストの追加

ReactCOP ではテストを行っていなかった。テストは、コードの品質を保つために重要である。ReactCOP2 では、テストを追加した。

## 5.8 型の追加

ReactCOP では、型を定義していなかった。型は、コードの品質を保つために重要である。ReactCOP2 では、型を追加した。

### (1) レイヤーパラムの取得

ReactCOP では、レイヤーパラムを取得する機能が存在した。ReactCOP ではレイヤーの活性化状態とレイヤーパラムを別で管理している。必然的にそれらを取得するためのメソッドも別々に用意する必要があった。

ReactCOP2 では、レイヤーの活性化状態とレイヤーパラムを一つのオブジェクトとして管理している。そのため、レイヤーを取得すればレイヤーの活性化状態とレイヤーパラムを取得できる。代替する機能として getLayerParam というメソッドを提供している。代替機能として用意したものの、レイヤーを取得すれば十分な必要がないかもしれないと考える。getLayerParam のメリットとしては、レイヤーではなく直接レイヤーのパラムを取得できることである。しかし、ライブラリを使用する上でたくさんのメソッドがあると、ライブラリの理解が難しくなる。そのため、レイヤーを取得するメソッドで十分であると考える。

## 5.9 レイヤーの追加

ReactCOP では明確にレイヤーを追加する機能は存在しなかった。レイヤーを追加するためには、レイヤーの活性化状態などの設定時にもしレイヤーが存在しない場合は新しくレイヤーを追加するという処理を行っていた。これは意図しないレイヤーが混入する可能性があるた。レイヤーを操作することとレイヤーの追加を分けることで、意図しないレイヤーが混入する可能性を減らすことができる。

## 5.10 レイヤーの削除

ReactCOP では明確にレイヤーを削除する機能は存在しなかった。必要ないレイヤーが存在すると、レイヤーの管理が煩雑になる。レイヤーを削除することで、レイヤーの管理が簡単になる。

## 5.11 レイヤーの活性化の切り替え

ReactCOP では、レイヤーの活性化の切り替えは、もしレイヤーが活性化している場合は非活性化し、非活性化している場合は活性化するという処理を行っていた。ReactCOP2 では、レイヤーの活性化の切り替えを行うためのメソッドを提供している。コードベースで見ると以下のようになる。

```
1 // ReactCOP
2 layerState.hoge ? layerManager.deactivateLayer("hoge") : layerManager.activateLayer("
      hoge");
3
4 // ReactCOP2
5 toggleLayer('hoge')
```

ReactCOP2 のほうが、コードが簡潔になり読みやすくなっている。

## 5.12 レイヤーの活性化

レイヤーの活性化には activeLayer というメソッドを提供している。レイヤーを活性化するという意味では、activeLayer より activateLayer のほうが適切である。activeLayer では、アクティブなレイヤーを取得するという意味と間違える可能性がある。この研究を引き継ぐ際には、activeLayer を activateLayer に変更することを推奨する。

### 5.13　レイヤーの非活性化

レイヤーの非活性化には inactiveLayer というメソッドを提供している。レイヤーを非活性化するという意味では、inactiveLayer より deactivateLayer のほうが適切である。inactiveLayer では、非アクティブなレイヤーを取得するという意味と間違える可能性がある。この研究を引き継ぐ際には、inactiveLayer を deactivateLayer に変更することを推奨する。

### 5.14　今後の課題

本研究では ReactCOP を大きく刷新し ReactCOP2 を実装した。本章で述べた問題もいくつか残っている。まだアプリケーションに導入して検証を行っていない。ReactCOP との機能の比較により、ReactCOP2 の機能が優位であることは議論したが、実際にアプリケーションに導入して検証を行う必要がある。

## 6　まとめ

## 7　参考文献