付録６－１

教科「情報」の CBT 試験を TAO で実施するための PCI に関する調査研究業務（プログラミング問題）調査研究報告書

BPS 株式会社

2022 年 3 月 25 日

BPS co.,Ltd.
BEYOND PERSPECTIVE SOLUTIONS

aws partner network
Standard Consulting Partner

10823551
JISQ15001:2006 準拠

W3C® MEMBER

目次

1. はじめに

(独)大学入試センターでは大学入学共通テストへのCBT活用に向けて調査研究を行うため、大学・高校関係者をはじめとする教育関係者に向けて「情報I」の試験のために開発したCBTの機能を整備、提供することが求められています。

本業務では大学入試センターセンターより委託を受け、「情報I」の試験のためのCBTの機能のうち「プログラミング問題(短冊式)」の機能について、教育現場で広く活用されているCBTプラットフォーム「TAO」上で使用するためのPCI(Portable Custom Interactions)の開発及び調査研究を行います

2. 概要

TAO上で実行可能な「Custom Interaction」を開発した中で、開発上行き詰まった点や、仕様上の問題点、また、「プログラミング問題」を出題する目的にも沿った形で考えられる将来的な展望について、本報告書にまとめました。

3. インタラクション開発上の課題

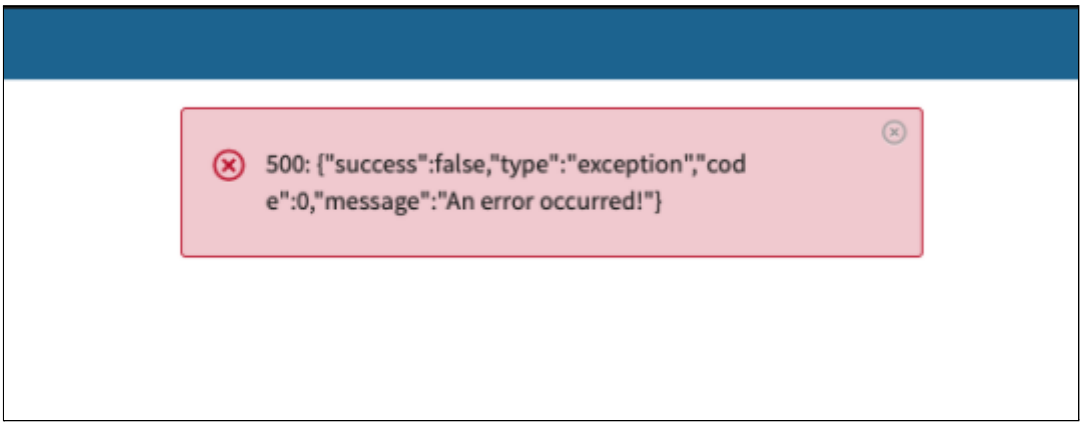
本報告書では、CBTプラットフォーム「TAO」上で使用するためのPCIの開発において
発見された不具合や、実装上の課題、また将来的な展望についてまとめます。

3.1 TAO上で実現できないこと、または注意すべき点

3.1.1 解答後レビューが表示されない

テスト完了後、採点者画面にて「レビューボタン」をクリックすると500番エラー
（図3.1.1参照）が返却され、解答内容のレビューが実施できない問題があります。

図3.1.1）



原因は不明ですが、本PCIと全く関連の無い環境下で、かつ標準インタラクションを用いた解答後
プレビューも同じ結果となることから、TAO特有の問題と考えられます。
この問題を回避するための方法は公式にも記されておらず、3.3.0-RC2では解答後レビューは機能
しないという結論に至りました。

3.1.2 解答は2回送信できない

TAOの仕様として、1つの問題に対して解答を2回以上送信することはできません。
必ず1問1答となる必要があります。

3.1.3 CSSのoverrideについて

Custom Interactionで配置する独自のオブジェクト（ただし、TAOのクラスを利用する）に
独自のスタイルを適用する場合、後から読み込んだ独自CSSにてクラスをoverrideしても
TAO標準のCSSの値を上書きできないことがありました。
この場合、独自で作成したPCIに存在する「Interactionを読み込んだ時に実行されるJavaScript」
において、対象クラスのCSSを上書きするか、HTMLのstyle属性にimportantをつけた上で強制的に上書きするかといった方法で、適用されることが確認できました。

3.1.4 qti.xmlの上書き

Custome Interactionをフィールドに配置し、問題・解答を作成した上で保存処理を実行
（「保存」をクリックするか、戻る等を押した際に表示される未保存の問題を保存するかどうか
のポップアップにて「Save」を選択する）すると、問題のXMLファイルがサーバ上の

/var/www/html/tao/data/taoItems/itemData/<識別子*ID>/itemContent/en-US/qti.xml*
※ /var/www/html/taoをインストール先のフォルダとした例

に保存されます。
通常の、例えば標準搭載されているInteractionでは、設定した問題は全てXMLタグに変更されて保存されますが、本PCIは性質上、設定した短冊型コードは全てHTMLとしてXMLファイルに保存しています（HTMLファイルの中のdata-question-state属性に短冊型コードの情報を保存）。

1回保存した後の2回目以降（編集時等）については、このdata-question-state属性の中身を
JSON Parseにより解析し、短冊型コードの再配置を行っています。この時、data-question-state
には1回目で保存した短冊型コードの元データとなるHTMLが保存されていることを確認して
います。

2回目以降に何らかの変更を加えて再度保存をすると、data-question-stateを含むHTMLタグが
消滅、問題として設定した短冊型コードの「配置」が全て消える事象が発生しました。

短冊型コードの「配置」はdata-question-state属性にありますが、解答に必要となる実行結果正答
や、JavaScriptについては、同XMLファイル内の

responseDeclaration > correctResponse > value

に保存されており、この値は消えることはありませんでした。

1回目も保存できる時と保存できない（消滅する）時があり、実際にサーバにPOSTしている値を検査しても、保存可否時において違いがないことが確認できています。
従って、サーバ上で保存処理を行っているTAOのヘルパ等に、InteractionのHTML内に存在するdata属性を正常に解釈できないなどのバグが内在する可能性も考えられます。

※本事象についてはTAO 3.3.0-RC02で確認しています

3.1.4 編集後の保存で保存データが消えることがある

問題を作成後、アイテムを編集し、再度「保存」ボタンを押すと作成した問題が全て消えることがあります。これは2回目以降の保存時に、問題データのうち短冊型ブロックを表示するためのデータ領域が空になってしまうことが原因で、空データで上書きされてしまうため、次回表示時に短冊型コードが消えてしまっている、という現象です。

常に上書きされるわけではなく、まれに（100回検査したところ12回は正常に保存できた）保存できることがあり、どのような条件で発生するかについての調査は結論が出せませんでした。

保存データが消える事がある問題については、発生条件が絞れており、回避し正常に保存させることが可能です。

3.1.4.1 検証

保存データが消える現象について、実際のSAVE時のPOSTデータ等を調査しました。
図3.1.4.1-1は正常に保存ができたときのPOSTデータです。「data-question-state」属性の中に短冊型コードの状態がJSON化されて送信されており、この内容でサーバ内データが保存されるため、次回展開時も短冊型コードを復元することができます。

図3.1.4.1-2は保存に失敗したときのPOSTデータです。
「data-question-state」の中身が空になっており、この内容でサーバ内XMLデータが上書きされてしまうため、次回展開時に空となってしまいます。

TAOにおける「save」クリック時に「A：saveしようとしている内容」と「B：その時の短冊の並び」を比較する処理を入れ、差があればAをBの内容で上書きする、という処理を入れようと試みました。しかし、TAOのsave処理よりも前に別の処理を入れることができず、こちらは上手くいきませんでした。

data-question-stateの中身が消えてしまう直接的な原因は判明いたしませんでしたが、
後述する「データの2回目以降の保存方法」という、TAOの異なるSaveAPIを用いて保存する方法でこの問題が回避できることから、TAOのSaveAPIになんらかの問題があると考えられます。

図3.1.4.1-1）正常に送信されたときのPOSTデータ

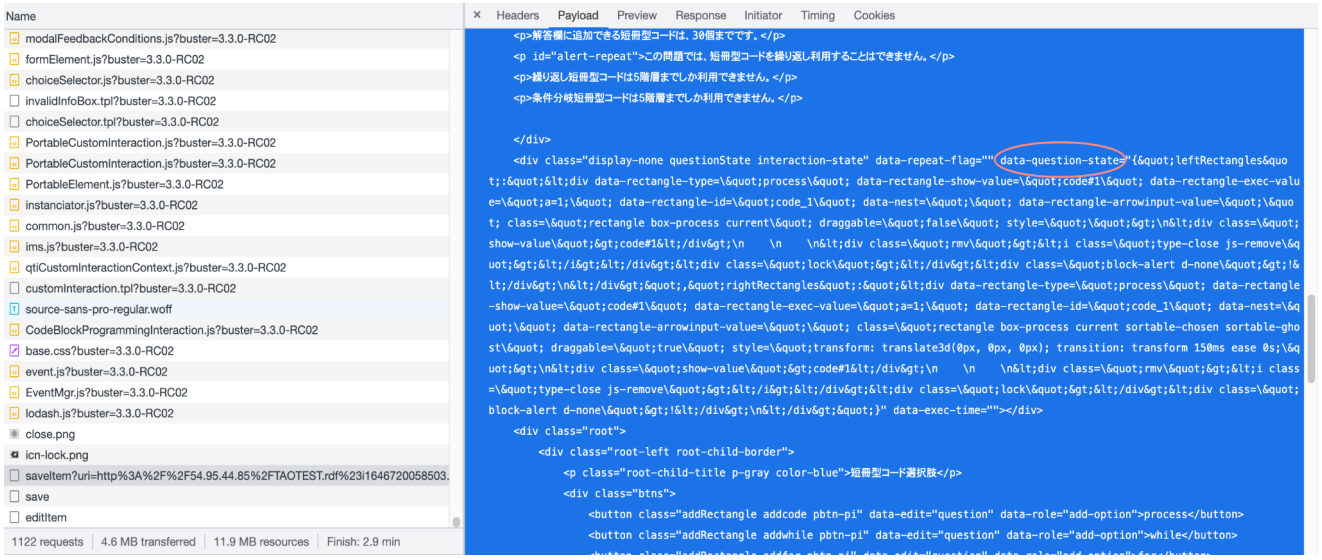
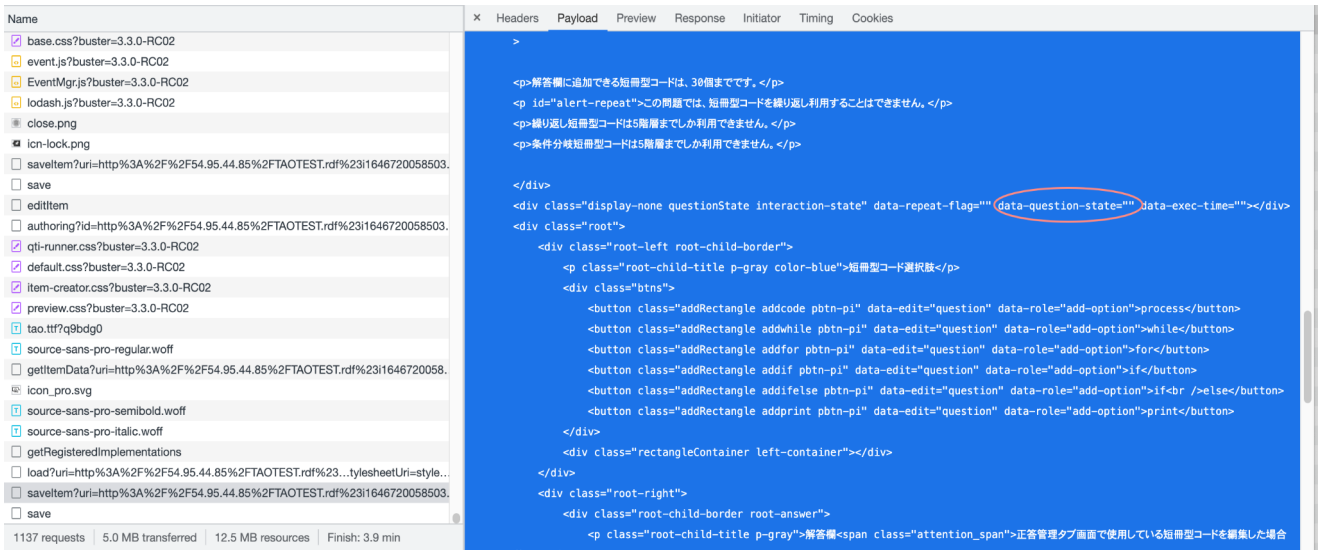


図3.1.4.1-2）保存に失敗したときのPOSTデータ



3.1.4.2 データの2回目以降の保存方法

図3.1.4.2-1に示した通り、保存時にナビゲーションメニューの「Items」を押した際に表示されるポップアップメニュー（図3.1.4.2-2）の「SAVE」から保存をすると、正常に保存されます。

図3.1.4.2-1）保存方法。ナビゲーションメニューのItemsから保存を行う。

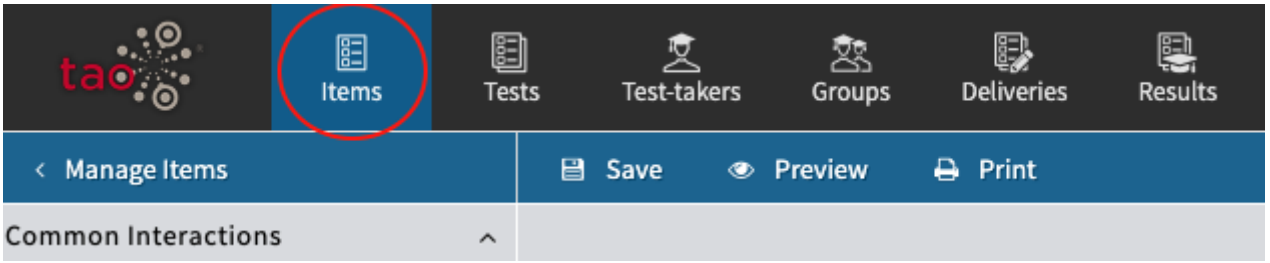
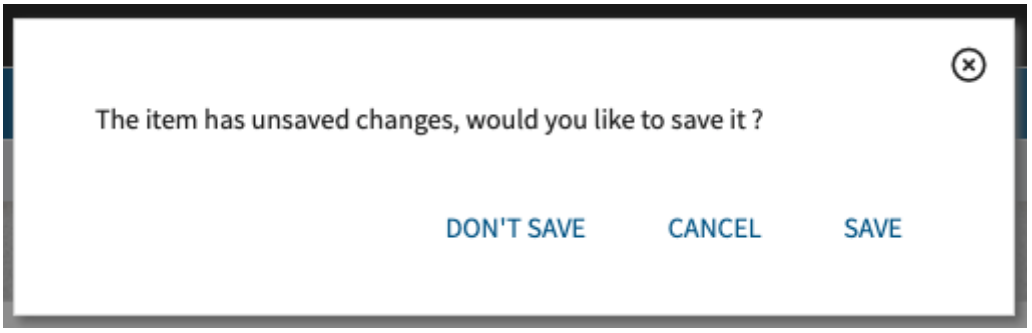


図3.1.4.2-2）Itemsを押した際に表示されるポップアップメニュー。



このように、メニューバー「Save」ボタンから行う保存と、ItemsのSAVEボタンから行う保存で保存結果が異なることから、この両者の保存は異なるTAO APIを使用していると考えられ、また「どちらのAPIを選択して保存するか」といったAPIリクエストはPCI上では実装不可であることから、本課題はTAO特有の課題と考えられます。

3.1.5 特定の文字を入力するとそれ以降の文字が消える

テキストフィールドに　<（半角小なり）を入力すると、それ以降の文字が消えることがあります。これは、TAOにおける「プロパティ欄のtextareaに対する入力文字をHTMLとして自動解釈されていること」が原因であると考えられます。

＜検証方法＞
本PCI、もしくはTAO標準のカスタムインタラクションであるBrainStorm等、プロパティ欄でtextareaによる入力欄を持ち、それがインタラクション上に反映されるものをドロップします。
プロパティに”<”から始まる何らかの文字を入力すると、”<”以降の文字が消えることが確認できます。

本PCIであれば短冊型コードを選択したときに表示される「表示内容」欄に”<"から始まる文字を何か入力します。

“<”の後に半角スペースを入れることで表示されますが、TAOにおけるtextareaの扱いをPCI側から変更することはできないため、本課題はTAO特有の課題と考えられます。

3.2 実装上の問題点

3.2.1 ドロップダウンリストについて

短冊型コードの表示内容をJavaScript以外の，例えば日本語疑似言語で表すと，ドロップダウンの論理演算子や算術演算子のみがJavaScript標準になってしまうことになり，結果的に論理演算子や算術演算子はドロップダウンリストでは問えないということになります。

表示上は['or', 'and']や['または', 'かつ']などにして、コードに変換する際に['||','&&']になるような仕組みに出来ないか検討しました。
例えば、['or':'||', 'and':'&&']のような形式で、表示文字とコード文字をそれぞれコロン区切りで設定するなどが考えられます。

本PCIにおいて利用する実行言語（TAO上で動作するCustom Interaction上の実行言語）については、全てJavaScriptで記載する必要があります。短冊型コードによって問題を作成した後の処理として、JavaScriptに変換し実行実行結果を得る挙動となりますが、JavaScriptに変換する際、JavaScriptの元のコードとなる「実行コード」プロパティにおいてもJavaScriptにて記載する必要があります。

セレクトボックスの処理については、「¥selectBox_<id(3桁)>[array(カンマ区切り)]」とすることで、array内の配列要素が選択できるセレクトボックスが生成されます。

例）
¥selectBox_012[1,2,3,4];
↓
[1 ▼]（1～4まで選択が可能）

これを配列内要素をコロン区切りで実装する場合、以下のような実行コードとすることが考えられます。

¥selectBox_012['または:||', 'かつ:&&', '足す:+', '引く:-']

現在、selectBoxを判定する正規表現は

/\¥selectBox_([a-zA-Z0-9]{1,3})\[.+?\]/

となっています。
この時、[]内に収められているカンマ区切りの文字をsplitで処理しセレクトボックス用配列に格納していますが、現状は配列内の文字列については詳細なバリデーションチェックを
行っていません。

ここで上記のように「または」という日本語表記と、実際の演算子を紐付ける処理を入れる場合、配列内の文字列に存在する特定の区切り文字（ここでは「:」（コロン）とします）で分割して配列に挿入する処理を実行し、表示はarray(0), 実行コードはarray(1)とします。この「実行コード」が実際のJavaScriptコードに変換されることになりますが、JavaScriptの記述に準拠していない形式の演算子が記載されていた場合、エラーを表示する必要があります。これらを全てのselectBox内の配列においてチェックし、さらに実行コードを含む場合と含まない場合が混合するようなケースも考えられますが、全てのパター

ンに対応して正常に表示させ、かつ正常に動作させることは全例外パターンの列挙が必要となるため現実的ではありません。

また、受検者によるインジェクションコードの挿入によって問題の正解が抜き取られる状態や、必ず正答となるような状況が作られてはなりません。
テキストボックスにおいては、このような不正なインジェクションコードの入力を禁止するため、以下の演算子の入力を禁止しています。

/(\,|\'|\"|\!|\=|\/|\+|\-|\~|*|\%|\<|\>|\||\&|\?|\^|\(|\))/

セレクトボックスについては作問者によって設定するため、入力禁止ワードの設定は設けていません。
「:」区切りで演算子の選択を可能にした場合、意図せずに不正ワードの入力を許可してしまう問題となってしまう可能性があり、やはりセレクトボックスにおいても入力禁止ワードの設定は必要不可欠と考えられます。

この2点の問題点につき、本件については本開発では実施しないものとしました。

3.2.2 ネスト数に関する課題

現在、FOR、WHILE、IF、IF-ELSE、といったネスト構造をもつ短冊型コードの「最大ネスト数」については、本PCIでは「最大5個まで」としています。
技術的には、6個以上、例えば20個や30個、100個もネスト構造として持たせることは可能ですが、実行時に全てのネスト構造を解析し、エラーチェックおよびjson化の処理を実行しています。

ネスト構造が深くなればなるほど解析処理や、ユーザインタフェースの処理（ブラウザのブラウジング）に負荷がかかり、ブラウザが停止してしまい、テストが中断されてしまうという可能性も考えられます。

一般的なプログラミングの開発方針として、ネスト構造が深ければ深いほどコードが複雑化し、また保守性や可読性が著しく低下するため、あまり良いプログラミング開発とは言い難いです。
さらに、6個以上設定すると解答欄からネストを含む短冊型コードがはみ出てしまったり、画面右からはみ出してしまった場合新たなネストを追加できない・テキストボックスに入力できない、削除ボタンも押せなくなるなど見た目上の問題点も多く、現状では枠内に収まる5個を最大値と
することでこれらの問題を回避しました。

とりわけネスト構造が6個以上続くようなプログラミング問題は作成されないという前提で開発を進めましたが、今後そのような問題を作成する必要が出た場合、画面の幅といったユーザインタフェース的な側面が大きく影響してくるため、注意が必要です。

3.3 将来的な拡張性

3.3.1 関数の実現

短冊型コード内への関数処理の展開については、プログラミング問題を扱うべく本PCIにおいて
重要度の高い機能と言えます。

関数型プログラミングの問題を出題する際、関数の考え方やコーディング方法について習得しているかは非常に重要であり、プログラミングを使う上で欠かせない知識とも言えます。

一方で、「関数の実現」について、本PCI上、とりわけ「TAO」というテストプラットフォーム上で実現するためには、様々な問題が発生します。
TAO-PCIが「AMD(*Asynchronous Module Definition*)」という非同期モジュールの定義方法に則った形で書かれており、（PCIの動作に必要な）独自の関数を書くために、まず関数の動作に必要となる定数を定義する必要があります。
例えばjQueryのイベントを利用する場合、jQueryの元となるファイルを関数の定義箇所に記載し、それを「$」として継承し、関数内で利用が可能となります。

最も問題となる点は、本PCIはJavaScriptの問題作成のためのプラットフォームではなく、あくまでプログラミングの体系的な考え方や記法を、メタ言語に近い形で出題できるようにするものであるため、あまりにもJavaScriptの作法に寄り過ぎてしまうと、JavaScript以外の問題が出題できなくなってしまいます。

関数の記法は各言語によって大きな差は無いものの、TAOというプラットフォーム上で関数をメタ言語的に扱うのは実装上高いハードルがあり、詳細な仕様策定・例外チェックが求められると思われ、本開発では扱わないものといたしました。

3.3.2 固定短冊の挿入禁止プロパティの設定

固定短冊型コードのプロパティに対し「前後に対する短冊型コードの挿入を禁止する」設定ができるようにすることで、解答欄に初期配置した短冊型コードを利用し、さらに発展的な問題を作成することが可能となる。

例えば、

短冊型コードA
空白
空白
短冊形コードB

と配置された解答欄において、現在では短冊型コードAよりも前、もしくは、短冊型コードBよりも後に新たな短冊型コードを配置することが可能ですが、これを禁止することにより、
問題に制約を設定し、出題者の意図やねらいを解答者に問うような発展的な問題の作成が可能となります。

ただしこの機能を実現する上で実装上難しい問題が存在します。
「固定化した短冊型コードの後に存在する領域」に対するドラッグアンドドロップ禁止処理は、現在のUI関連処理をまとめたJavaScriptコードでは対応していません。そのため、新たに「ドラッグアンドドロップを禁止する領域を設定できる処理」の開発が必要です。
また「ネスト内に存在する短冊型コードの場合、そのネスト内の対象固定短冊型コード以降とするのか、ネスト外も範囲に含めるのかの判断」が必要であるなど、細かな仕様の策定が必要になる他、例外的な処理への対応も複雑化します。

これらの問題により、本件では対応しないものとしますが、機能としては前述のように出題のスケーラビリティを確保する上で重要な機能と思われるため、将来的な展望として記載いたします。

3.3.3 Undo, Redoの必要性

本PCIで作成した問題の解答時に、Undo（直前の操作を取り消し、一つ前の状態に戻す）およびRedo（Undoした内容を取り消し、一つ元の状態に戻す）については、機能として備わっていません。本PCIを用いた試験はWebブラウザ上で実施するため、直前まで実施した内容を元に戻したいといった操作は頻繁に利用されるものと考えられます。ブラウザの戻るボタンであるブラウザバックなどで「戻る処理」をユーザが行った場合、問題開始前の状態に戻ってしまい、最初から試験をやり直すこととなってしまいます。これではユーザビリティがかなり低下するため、やはりUndo, Redo機能は重要であると考えられます。

しかしながら、CustomInteractionは全てJavaScriptベースで実装されるため、データの保存は基本的にブラウザが実装するマシンスペックに依存します。
Undo, Redoについては、「1つの操作が行われるたび、どのような状態だったかをブラウザ内の保存領域に保存し、それをJavaScriptでバージョン管理をする」といった処理が必要となりますが、どのような環境下でもこのバージョン管理機構を作動させなくてはならないため、ローカルストレージに対応したブラウザであるか、バージョン管理の最大値をどのように設定するかなど、詳細な仕様策定が必要と考えられます。さらに、TAOプラットフォーム上でそのような管理が可能かどうかなどの検証も必要となるため、本課題については将来的な展望として記載いたします。