CS-438 - Decentralized Systems Engineering, Fall 2024

BalduchColab

A Peritext Realtime Implementation

Team	02
Authors	Ugo Jean Mario Balducci
	Yasmin Chaimaa Ben Rahhal
	Emile Janho Dit Hreich
	Emma Gaia Poggiolini

Version	1.0
Publish Date	December 2, 2024

Table Of Content

Introduction to the Project

In the rapidly evolving domain of collaborative editing, ensuring seamless, real-time interaction among multiple users while maintaining data consistency poses significant challenges. Peritext, a Conflict-free Replicated Data Type (CRDT) designed for rich text editing, offers a robust foundation for addressing these challenges. It provides a decentralized framework that allows concurrent modifications without the need for central coordination, making it a prime candidate for adaptation to modern collaborative environments.

This project aims to leverage the strengths of Peritext by adapting it to a real-time block editor collaboration model. Block editors, which organize content into modular blocks (e.g., text, images, or interactive components), have become a standard in contemporary content creation platforms. However, integrating CRDT-based solutions into block-based editors introduces unique technical and design considerations due to their modular structure.

The adaptation focuses on enhancing scalability, consistency, and user experience in collaborative block editing. By extending Peritext's capabilities to support block-level operations, this project explores solutions for challenges such as:

- *Efficient Conflict Resolution at the Block Level*: Ensuring consistent integration of changes made by multiple users to individual blocks and across interdependent blocks.
- *Granular Synchronization*: Optimizing network and computational overhead by synchronizing changes at the block level rather than the page level.
- *Real-Time Collaboration Features*: Maintaining low-latency updates and immediate feedback for users in dynamic editing environments.
- *Preserving Rich Text Semantics*: Adapting the rich text editing focus of Peritext to a block-based model without compromising on formatting and content fidelity.

The outcome of this adaptation is a hybrid approach, called **BalduchColab**, that combines the decentralized advantages of Peritext's CRDT foundation with the flexibility and modularity of block-based editing.

Specifications of the CRDT

Block Editor Schema

BalduchColab blocks are defined by specific schemas that include unique properties, default attributes, and content types. It is based on the block editor library BlocknoteJS used in the interface. This specification details the structure and functionality of each block type and the corresponding inline content schema.

Basic Blocks

- **Paragraph Block**

Type: `paragraph`
Properties:

`id` (string): Unique identifier for the block.

`props`: Includes default block properties such as background color, text color, and text alignment.

`content`: Array of `InlineContent` objects, allowing rich text formatting within the paragraph.

`children`: Array of nested Block objects for hierarchical structures.

- **Heading Block**

Type: `heading`
Properties:

`id` (string): Unique identifier for the block.
`props`: Combines default block properties with a specific level property:
`level` (1 | 2 | 3): Specifies the heading's level, representing titles (1), headings (2), and subheadings (3).
`content`: Array of `InlineContent` objects for styled text.
`children`: Array of nested Block objects.

- **Bullet List Item Block**

Type: `bulletListItem`
Properties:

`id` (string): Unique identifier for the block.
`props`: Includes default block properties.
`content`: Array of `InlineContent` objects.
`children`: Array of nested Block objects for sublists or hierarchical bullet points.

- **Numbered List Item Block**

Type: `numberedListItem`
Properties:

`id` (string): Unique identifier for the block.
`props`: Includes default block properties.
`content`: Array of InlineContent objects.
`children`: Array of nested Block objects for hierarchical structures.

- **Image Block** (Optional)

Type: `image`
Properties:

`id` (string): Unique identifier for the block.
`props`: Combines default block properties with additional attributes:
`url` (string): The image source URL.
`caption` (string): Text displayed as the image caption.
`previewWidth` (number): Pixel width for image previews, defaulting to 512.
`content`: Always undefined, as images do not contain text content.
`children`: Array of nested Block objects.

- **Table Block** (Optional)

Type: `table`
Properties:

`id` (string): Unique identifier for the block.
`props`: Includes default block properties.
`content`: A TableContent object representing structured table data.
`children`: Array of nested Block objects.

Default Block Properties

All blocks share a set of default properties (in `props`) that govern their appearance and alignment:

- `backgroundColor` (string): Background color of the block and nested blocks ("default" by default).
- `textColor` (string): Text color of the block and nested blocks ("default" by default).
- `textAlignment` ("left" | "center" | "right" | "justify"): Alignment of the block's text, defaulting to "left".

Inline Content

Blocks that support textual content (e.g., paragraph and heading) use `InlineContent` objects for fine-grained text styling and interactivity.

- **StyledText**

Type: `text`
Properties:

`text` (string): The displayed text.
`styles`: Includes default styles for text formatting:
    `bold` (boolean)
    `italic` (boolean)
    `underline` (boolean)
    `strikethrough` (boolean)
    `textColor` (string)
    `backgroundColor` (string)

- Link

Type: `link`
Properties:
`content`: Array of `StyledText` objects for link text with styles.
`href` (string): URL to which the link points.

Page Example

To better understand the specifications for the block editing, check the *Annex 1*. It gives a page example that demonstrates the output of a list of blocks, how they are represented and styled.

CRDT Operations

During the editing of the page on a client, all the changes applied to the page are saved and converted to CRDT operations. These operations are heavily inspired by Peritext operations, but adapted to include the block editing, and some other additions.

- Add Block

Properties:
`opId` (string): Unique identifier for the operation, will also serve as block id.

`blockType` (string): The type of the block.

`afterBlock` (string): Unique identifier for the block before.

`parentBlock` (string): Unique identifier for the parent block. If parent block is not null and afterBlock is null, then it is the first children.

`props`: Struct properties related to the specific block.

```
{
    action: "addBlock",
    opId: "2@alice",
    afterBlock: "1@alice",
    parentBlock: null,
    props: {}
}
```

- Remove Block

Properties:
`opId` (string): Unique identifier for the operation.

`removedBlock` (string): Unique identifier for the block to be removed.

```
{
    action: "removeBlock",
    opId: "5@alice",
    removedBlock: "2@alice",
}
```

- Update Block

This operation enables to move the block, nest it, or change all its properties. If props is set, it will replace all the existing properties.

Properties:

`opId` (string): Unique identifier for the operation.

`updatedBlock` (string): Unique identifier for the block before.

`afterBlock` (string): Unique identifier for the block before.

`parentBlock` (string): Unique identifier for the parent block. If parent block is not null and afterBlock is null, then it is the first children.

`props`: struct properties related to the specific block.

```
{
    action: "updateBlock",
    opId: "2@alice",
    afterBlock: "1@alice",
    parentBlock: null,
    props: {}
}
```

- Insert Character

Properties:

`opId` (string): Unique identifier for the operation.

`blockId` (string): Unique identifier for the block concerned.

`afterId` (string): Unique identifier for the character before.

`character` (string): The character that was added.

```
{
    action: "insert",
    opId: "5@alice",
    blockId: "2@alice",
    afterId: "4@alice",
    character: "c",
}
```

- Delete Character

Properties:

`opId` (string): Unique identifier for the operation.

`blockId` (string): Unique identifier for the block concerned.

`removedId` (string): Unique identifier for the character concerned.

```
{
    action: "remove",
    opId: "6@alice",
    blockId: "2@alice",
    removedId: "5@alice",
}
```

- Add Mark

Properties:

`opId` (string): Unique identifier for the operation.

`blockId` (string): Unique identifier for the block concerned.

`start` (struct): Unique identifier for the character starting the formatting.

`end` (struct): Unique identifier for the character ending the formatting.

`markType` (string): The type of formatting.

`options` (struct): Options related to the format (color, url, ...).

```
{
    action: "addMark",
    opId: "6@alice",
    blockId: "2@alice",
    start: { type: "before", opId: "5@alice"},
    end: { type: "after", opId: "6@alice"},
    markType: "bold",
    options: {},
}
```

- Remove Mark

Properties:

`opId` (string): Unique identifier for the operation.

`blockId` (string): Unique identifier for the block concerned.

`start` (struct): Unique identifier for the character starting the formatting.

`end` (struct): Unique identifier for the character ending the formatting.

`markType` (string): The type of formatting.

```
{
    action: "removeMark",
    opId: "6@alice",
    blockId: "2@alice",
    start: { type: "before", opId: "5@alice"},
    end: { type: "after", opId: "6@alice"},u
    markType: "bold",
}
```

Data Structure

Pages

A user can create multiple pages that refer to an instance of the CRDT editor. Each page can be shared among the peers and contains a list of blocks representing the content. As so, the CRDT operations and their identification uniqueness are independent for each page.

Blocks

Inside the pages, the blocks represent the content. As shown above, they can have a lot of different types to represent different data. The merge of CRDT operations happens on two level: the page level that manages the blocks present in the page, and the block level that is specifically in charge of the content of the block.

Adaptation of Peerster

New Messages

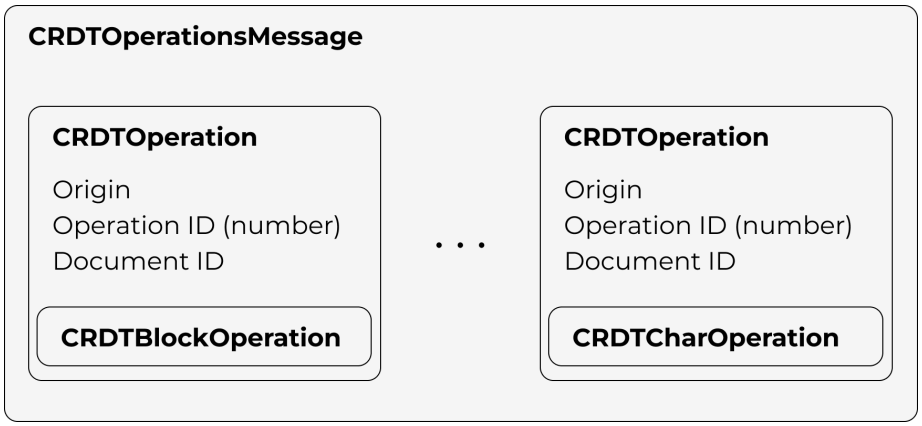
CRDTStatusMessage

This message contains a node's view on the state of the pages. By "view" we mean all the operations the node has processed so far. That is, for each page, for each known peer, the node's view contains the operation ID number of the last operation processed from that peer. The status allows peers to compare their views and get/send updates from/to other peers.

CRDTOperationsMessage

A `CRDTOperationsMessage` contains one or more `CRDTOperation`. A `CRDTOperation` is a modification to a page that a peer wants to broadcast to everyone. As illustrated below, it contains the initiator of the operation, i.e. the peer's address that created the operation, a page id, an operation number, and the

underlying operation to be performed. A CRDTOperation can be seen as a wrapper around any kind of operations that can be performed in a document (adding a block, a character, ...), which all have their own properties. Therefore, once a `CRDTOperation` is sent by a peer, it can expect all the operations from the list to be eventually distributed and processed by every peer.



Annex 1: Page Example

Here is an example of how a page is structured in the block editor:


```
[
  {
    "id": "40924c93-2f92-4a80-8152-8665d9f474f6",
    "type": "heading",
    "props": {
      "textColor": "default",
      "backgroundColor": "default",
      "textAlignment": "left",
      "level": 1
    },
    "content": [
      {
        "type": "text",
        "text": "This is a heading block",
        "styles": {}
      }
    ],
    "children": []
  },
  {
    "id": "de1c8949-30f6-4b68-9a3d-b6d9226e72ae",
    "type": "paragraph",
    "props": {
      "textColor": "default",
      "backgroundColor": "default",
      "textAlignment": "left"
    },
    "content": [
      {
        "type": "text",
        "text": "This is a paragraph block with ",
        "styles": {}
      },
      {
        "type": "text",
        "text": "bold",
        "styles": {
          "bold": true
        }
      },
      {
        "type": "text",
        "text": ", ",
        "styles": {}
      },
      {
        "type": "text",
        "text": "italic",
        "styles": {
          "italic": true
        }
      },
      {
        "type": "text",
        "text": " and ",
        "styles": {}
      },
      {
        "type": "text",
        "text": "color",
```

```
        "styles": {
          "textColor": "yellow"
        }
      }
    ],
    "children": [
      {
        "id": "f6b736ba-b86c-403e-8ac1-ae2161e5c254",
        "type": "paragraph",
        "props": {
          "textColor": "default",
          "backgroundColor": "default",
          "textAlignment": "left"
        },
        "content": [
          {
            "type": "text",
            "text": "I am a child block",
            "styles": {}
          }
        ],
        "children": []
      }
    ]
  },
  {
    "id": "913d4177-7dc6-4855-a3f1-8a77859d9e6f",
    "type": "bulletListItem",
    "props": {
      "textColor": "default",
      "backgroundColor": "default",
      "textAlignment": "left"
    },
    "content": [
      {
        "type": "text",
        "text": "Bullet point",
        "styles": {}
      }
    ],
    "children": []
  },
  {
    "id": "d30e27e7-72f7-4d2e-bed1-d1c9f3588e38",
    "type": "checkListItem",
    "props": {
      "textColor": "default",
      "backgroundColor": "default",
      "textAlignment": "left",
      "checked": false
    },
    "content": [
      {
        "type": "text",
        "text": "Check list",
        "styles": {}
      }
    ],
    "children": []
  },
  {
    "id": "c76fa02b-efbe-44bb-9209-1708c39b7134",
    "type": "table",
    "props": {
      "textColor": "default"
    },
```

```
    "content": {
      "type": "tableContent",
      "columnWidths": [
        null,
        null,
        null
      ],
      "rows": [
        {
          "cells": [
            [
              {
                "type": "text",
                "text": "Cell1",
                "styles": {}
              }
            ],
            [
              {
                "type": "text",
                "text": "Cell2",
                "styles": {}
              }
            ],
            [
              {
                "type": "text",
                "text": "Cell3",
                "styles": {}
              }
            ]
          ]
        },
        {
          "cells": [
            [
              {
                "type": "text",
                "text": "Cell4",
                "styles": {}
              }
            ],
            [
              {
                "type": "text",
                "text": "Cell5",
                "styles": {}
              }
            ],
            []
          ]
        }
      ]
    },
    "children": []
  },
  {
    "id": "5531fc5b-426c-4a65-a545-9161c59cd11d",
    "type": "paragraph",
    "props": {
      "textColor": "default",
      "backgroundColor": "default",
      "textAlignment": "left"
    },
    "content": [],
    "children": []
```

```
    }
  ]
```

This corresponds to the page:

This is a heading block

This is a paragraph block with **bold**, *italic* and color

> I am a child block

- Bullet point

- [ ] Check list

Cell1	Cell2	Cell3
Cell4	Cell5	