**SketchWise**: Collaborative Tool for Online Whiteboarding

**ABSTRACT**

SketchWise is a collaborative tool for online whiteboarding that offers a unique variety of sketching tools for a wider range of expression and design using a comprehensive algorithm that accentuates concurrency techniques. There are several existing collaborative tools such as Sketchboard.io and RealtimeBoard, however the SketchWise project intends to improve the concurrency and latency issues found in existing applications. The motivation of this project is to thoroughly research different existing operational transformation techniques in order to implement the best possible architecture and algorithm.

**MOTIVATION**

While collaborative documents and text editors address the problem of editing a single document simultaneously, many do not address a very important type of document: a sketchpad. In the processes of brainstorming, teaching, drawing, sketching prototypes. etc., a collaborative platform for working on the same paper with a more varied visual scope beyond textboxes and shapes becomes essential. Therefore, we present **SketchWise**, a distributed platform that allows multiple users to simultaneously edit a blank canvas using a variety of visual components such as images, textboxes, a pencil tool, a color palette, different shapes, etc. This enables users to:

* Meet remotely and communicate
* Preserve and re-access old, private discussions
* Collaborate and design visually, clearly illustrate ideas
* Can be used academically or in the industry by teams or in schools
* Can be used by designers to sketch low-fi prototypes together

**UNIQUE SELLING PROPOSITION**

The purpose of SketchWise, compared to other collaborative whiteboarding tools, is to provide a base for our primary target market, which includes artists, designers, and industry professionals. Our secondary target market encompasses office and school use of a general whiteboarding tool with extensive capabilities. We appeal to this market by complementing the basic collaborative whiteboard tool with different types of brushes, pencils, and other online drawing utensils that enhance design work, as well as additional features to supplement their work such as pre-designed templates. Furthermore, we offer a better tool technologically through the robust algorithms planned to be implemented that ensure consistency and speed, detailed further in the remainder of this paper.

**HYPOTHESIS**

We will maintain eventual consistency in a real-time concurrent editing multi-user environment by employing the Generic Operational Transformation algorithm (GOT), adapted to work with pixel groups representing different strokes, and the client-server model of Google Wave.

**GOALS**

The primary goal of a real-time concurrent editing system is to maintain consistency in a shared space. We intend to achieve consistency by adhering to the following properties:

* Convergence: At the eventual or quiescence state, copies of the shared document are identical at all sites, using operational-transformation algorithms [19]
* Precedence or causality-preservation: For any pair of causally related operations, a and b, where a causally precedes b, a is executed before b at all sites [19]
* Intention-preservation: For any operation O, the effects of executing O at all sites is the same as the intention of O, and the effect of executing O does not change the effects of independent operations [19]

We plan to achieve the following technical goals upon the implementation of this project:

* Strong availability to load the most consistent board, which is possible because OT algorithms are optimistic, so the data is pushed through to the client immediately
* Fault tolerant to any number of contributor failures
* Fully consistent

Our goals for the product itself encompass the following:

* Design brushes, pencils, and other tools to enhance drawing experience
* Offer clean user interface
* Have features to complement the design experience

**TECHNIQUES**

In our research, we chose to implement an eventual consistency model so that we have strong availability, and the document can seen by all users at any given time. We examined three techniques for eventual consistency: operational transformation, differential synchronization, and commutative replicated data types. We will explore each of the following and demonstrate why we choose operational transformation.

**Operational Transformation:**

In OT, every change to the document is represented as an operation, which when applied to the current document, changes its state. In order to maintain consistency, OT transforms pairs of concurrent operations in the same current state such that when sets of these transformed operations are executed in different orders, the result is identical for all states. [19]

Pros:

* The most recognized technique implemented for real-time collaborative editing

Cons:

* Complexity increases as as the number of operations and users increases, exponentially for the former. [15]

**Differential Synchronization**

DS repeatedly diffs two documents and updates the necessary documents by applying found changes through patch operations. [15]

Pros:

* Because only two states are diffed, network impact is small. [15]

Cons:

* It is primarily used for systems such as Git and SVN, which do not handle complex merge conflicts. This is cannot be employed on our system. [15]

**Commutative Replicated Data Types**

In CDRT, there is no transformation of concurrent operations to maintain consistency. Instead, concurrent operations in CDRT are required to be associative, commutative, and idempotent, removing the issue of conflicts. [15]

Pros:

* Robust and scalable [15]

Cons:

* Has not been well established enough to be implemented for real-time collaborative editing

We ultimately chose OT, because DS cannot be applied to real-time collaborative editing and CDRT has yet to be. Because OT precedes both DS and CDRT is its creation, it has much more documentation and established algorithms available for us to utilize than DS and CDRT. Additionally, it is also currently implemented in Google Docs, one of the most well-known real-time collaborative editors used today.

Next, we will briefly examine the following significant operational transformation algorithms, dOPT, adOPTed, Jupiter, COT, Google Wave, GOT, and GOTO and demonstrate why we chose to use GOTO for our consistency maintenance in our system.

**dOPT**

If operation O is causally ready, meaning its execution will not violate the causality property, it is transformed against all operations in its history log that it is concurrently related to so that executions of transformed concurrent operations in different orders produce identical document states. It primarily relies on the TP1 transformation property:

* Given a pair of concurrent operations in the same document state, a and b, a o T(b, a) is equivalent to b o T(a, b) where T is the transformation function and a o b, signifies the sequences of operations containing a following by those containing b. [19]
* T, given a and b, returns a new operation that is applied after execution of the second parameter and preserves the intent of the first parameter. [19]
* T essentially computes a new operation tailored to each site whose application at each site can produce identical results across all sites.

dOPT fails, however, due to its priority marker associated with each operation with more than one client. [21]

**adOPTed**

Adopted another transformation property TP2 to improve the original dOPT algorithm:

* For every three concurrent operations a, b, and c in the same document state, T(c, a o T(b, a)) is equivalent to T(c, b o T(a, b))
* This helps solve dOPT’s problem of failing whenever an operation is concurrent with two or more dependent operations

adOPTed fails to preserve intention preservation. [21]

**Generic Operational Transformation**

To address the problem of intention preservation not solved in dOPT or adOPTed, GOT introduced the concept of context:

* Context of a document state: the sequence of operations executed on the initial document state to arrive at the current document state [19]
* DC(O), the definition context of O, is the context of the document state on which O is defined [19]
* EC(O), the execution context of O, is the context of the document state on which O is to be executed [19]
* Context equivalent relation U: Given two operations Oa and Ob, Oa and Ob are context equivalent iff DC(Oa) = DC(Ob). 2 [19]
* Context preceding relation ->, Given two operations Oa and Ob, Oa is context preceding iff DC(Ob) = DC(Oa) + [Oa] (where + expresses the concatenation of two lists). [19]

And exclusion transformation as opposed to the solely inclusive transformed that dOPT and adOPTed performed:

* Inclusion Transformation (IT): Transforms operation ‘a’ against context equivalent operation ‘b’ so that the impact of b is included, where the output is Oa’ and Ob causally precedes Oa’ and the intention of Oa is equivalent to Oa’ so that DC(Oa) = DC(Oa’) [19]
* Exclusion Transformation (ET): Transforms operation ‘a’ against context preceding operation ‘b’, so that the impact of ‘b’ is excluded, where the output is Oa’ and Ob is concurrent with Oa’, and the intention of Oa equals Oa’ so that DC(Oa) = DC(Oa’) [19]

The intention of an operation can be preserved if its definition context matches its execution context, DC(O) = EC(O), which can be achieved by IT and ET transformations. In order to achieve convergence, GOT utilizes both IT and ET transformations in an undo/do/redo scheme. GOT only supports basic string operations, however, and is missing the correctness proof of its algorithm [21]

**Generic Operational Transformation Optimized**

GOTO optimizes GOT by ensuring intention-preservation and convergence without the undo/ redo/scheme and reducing the number of IT and ET transformations. GOTO takes advantage of the post-conditions of TP1 and TP2 to do this and generalize the context relation definitions [21].

**Jupiter**

The Jupiter collaboration system consists of a centralized server that maintains a shared representation of the states of each object. This client-server architecture of the Jupiter system solves the dOPT problem since it only allows two-way communication. Any update that a client makes must be received and transformed by the server before it is propagated to all clients. After a client receives a propagation, it applies a transformation operation, if needed, and applies the changes to its local copy. This ensures that a causality violation can never occur. Jupiter deals with convergence with its transformation function described in the paper written at Xerox PARC [17]. Jupiter keeps a two dimensional state space graph to store all the possible operation transformation paths. Operations involved in a transformation must follow a context equivalent pre-condition. This means that the operations involved in the transformation must originate from the same starting state.

**Google Wave**

Google Wave’s operational transformation algorithm and client-server model are based off of Jupiter. One of the primary goals for Google Wave is to have a basic but efficient server. In its client-server model, clients are required to wait for an “ACK” from the server before further operations are communicated to the server. When the server acknowledges an operation, it means that it has transformed the client’s operation, applied it to the server copy, and broadcasted the transformed operation to all the involved clients. Google Wave also follows optimistic concurrency. This means that local operations at the client site are executed without being delayed until an “ACK” from the server. The client and server may have different intermediate operational transformations before reaching a converging state. Further information on techniques such as server operational transformation prediction that are also employed in Google Wave can be found in the whitepaper series [18]. One setback to Google Wave’s implementation is the lack of intention preservation, which we intend to eliminate with the GOT algorithm.

**Evaluation and Decision of Technique**

We chose to employ the GOT algorithm because it is the only known solution with convergence, causality-preservation, and intention-preservation and we chose to use Google Wave’s client-server model.

**Client-Server Model**

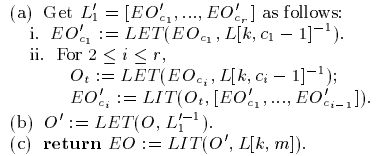
Our client-server model is based off of the Google Wave and Jupiter architecture. Our design includes a single central server similar to that of Google Wave. Clients will follow the non-blocking property where operations are executed without delay while still waiting for an “ACK” message from the server. The server will maintain a queue of operation requests and will apply, then broadcast each change one by one. When a client receives an “ACK”, it will check to see if it needs to apply any further transformations locally and then execute the changes to the local copy. Although local changes continue at the client site, it will not send further operations to the server until an “ACK” for the previous operation has been received. The server is required to filter all operations and propagate to all copies in order to ensure eventual consistency. However, the latency between the a client’s update and the propagation from the server to all the client copies is a factor to consider when evaluating the system architecture though we chose to focus on solving concurrency over latency.

**Algorithm**

Here we will give a preliminary overview of our operational transformation algorithm GOT from reference [19].

The algorithm operates as follows:

* O: a causally-ready operation
* L: the list of operations [EO1 ; EO2 ; :::; EOm] in EC(O).
* EO: the execution form of O.
* LET denotes an ET transformation on L
* LIT denotes an IT transformation on L
* L-1 denotes an inverse of L
* Scan L[1; m] from left to right to find the first operation EOk such that EOk is concurrent with O. If no such operation is found, then return EO := O.
* Otherwise, scan L[k + 1; m] to find operations causally preceding O. If no single such operation is found, then return EO := LIT (O ; L[k; m]). [19]



**Handling Insertions and Deletions**

From our research in operational transformation, we will use established string transformation functions [22] for our all text editing operations. Because our whiteboarding supports not only text editing but also a wide range of artistic editing, we will need to define insertion and deletion functions for lines, shapes, in all thicknesses and sizes. We will define our document, on which operations are applied, as a n x m board of pixels, where each (row, col) tuple is the position of a pixel. We will define a new inclusive and exclusive transformation function for every unique pair of editing operations, i.e. (delete line, draw line), (insert text, delete shape), etc.). For insertions and deletions, it will take in a range of pixels ordered by insertion time, represented by a list instead of a single position p, and c, a color for non-text operations.

Although string operations can become complicated and require position shifting in the case of concurrent operations with overlapping insertion and deletion position, we will not run into that problem with lines and shapes as overlapping operation positions will just result in the overlay of these figures. Our primary concern in overlaying will be keeping track of the order of executions of operations. The order of executions is handled in IT and ET transformations is handled as follows for a given pair of concurrent operations Oa and Ob. IT(Oa, Ob) will:

* Compare the pixel or text ranges of both operation ranges
* Find Oa’s operation range in the document state after Ob has been executed, thus transforming Oa to Oa’ (an assumption made by the transformation property TP1)

**Algorithm Advantages**

* Satisfies convergence, causality preservation, and intention-preservation properties
* Established an operational transformation undo which resulted in faster and more efficient undo operations
* Implemented in several well-known editing programs [21]
* Supports string transformations

**Algorithm Disadvantages**

* Not proved theoretically correct for all scenarios [21]

**EXPERIMENTS**

After analyzing potential solutions to satisfy our technical goals, we determined that our initial prototype should prioritize concurrency management. Due to the aforementioned needs of the application, we determined that the best algorithm to use is GOT. This encompasses the causal ordering that we deemed to impose, as well as maintaining this eventual consistency model across numerous clients. We adapt the GOT algorithm to work with events that are arrays of pixels being covered instead of strings being inserted or deleted, as described in the handling insertions section above. We test the following cases on a 10 computer cluster with various line drawings as the events, including insertion and deletion. We use a single server dedicated to propagating the changes it receives in the order determined after applying the operational transformation algorithm. We measure whether the changes eventually appear, in the correct order, on all the clients. This essentially entails recording qualitative data regarding its consistency when drawing different orderings of lines on multiple clients and making sure the ultimate overlapping is correct.

|  |  |
| --- | --- |
| **Experimental Cases** | **Results** |
| Three independent line stroke events (E1,E2,E3) exist on the document, the new operation (O) draws a line crossing all three, creating a causal dependency.  E1->O, E2->O, E3->O | In our experimental run of this scenario, we tested to see whether or not E1, E2, and E3 overlay O. The result was that the overlapping pixels between E1 and O were filled with O’s stroke since E1 occurs before O. We achieved the same results from E2 and E3, which were successfully overlayed by O’s strokes. |
| One line stroke already exists on the document. Operations E2, E3, and O happen simultaneously and must all show up on all clients.  E1->O, E2 || O, E3 || O | In this case, E1 appears under O. Since E2, E3, and O occur concurrently, we explored a few different cases. In the case that E2, E3, and O do not overlap, the lines eventually are executed on every client copy. In the case that E2, E3, and O are overlapping strokes, we notice that the server’s transformation function randomly chooses which stroke overlays which. In our experimental run, E2 overlays E3 and O. This ordering is maintained across all client copies. |
| E1, E2, and E3 are independent operations. E1 and E3 causally precede O (line drawings intersect/conflict) and E2 is concurrent with O.  E1->O, E2 || O, E3 -> O | Order is imposed on events E1, E3, and O. We observe that because of the timestamp, the local machine displays E1 before O and another machine displays E3 individually. Once this information is communicated to all the clients, the overlap is executed such that E1 and E3’s lines appear under O’s line. Since E2 and O do not have a dependency but are concurrent, both show up eventually on the other client. We also tested the edge case where the concurrent operations overlap - during which the transformation function seems to randomly choose between the two and display that on top, but the results are still consistent in terms of ordering across all the clients. |

These findings show that concurrency management matches our hypothesized results when using the GOT algorithm. We also observe that the model we chose to implement, in its most basic form, provides eventual consistency and all the clients mirror the most updated copy at the end of all the operations. Some of the undefined behavior, such as randomly choosing the shape to display for concurrent operations, show the drawbacks of GOT. However, since all the client copies reflect that decision, this doesn’t threaten the overall functionality of SketchWise.

**RELATED WORKS**

**RealtimeBoard** is a Virtual Whiteboard & Remote Collaboration tool available on desktop and mobile. It is more focused on templated collaboration, like thought boards, stories, post-it notes as opposed to free style drawing. It is integrated with Google Drive, Dropbox, Slack, and daily tools, and has comments, chats, and video [1]

**AWW App** is an online Whiteboard for Real-time Visual Collaboration, which can be run on any browser, computer, tablet, or mobile phone (great if not everyone is working on the same device). It allows you to log in to privatize your session and invite others to join. Its design is intuitive, featuring undo/delete buttons, swipe erasers like an actual board, allowing boards to be saved and accessed at any time, and offering premade business templates. It has a plugin and is able to draw shapes/lines, upload pictures [2].

**SketchTogether** is a real time whiteboard for the web that is similar to AWW App [3]

**Sketchboard.io** is an online Sketch Diagramming Whiteboard for teams integrated on Slack [4].

**Web Whiteboard and Whiteboard Fox** are both similar to AWW App [16]. They are basic whiteboards that can be used in the browser with basic drawing/visual tools. Web Whiteboard also allows for text editing with a built in text editor [5].

**Conceptboard** is similar to RealTime Board, but much more visual-based, like an online mood board. It can upload files, images, and draw, and also has board chat [6].

**ZiteBoard’s** unique component is a line smoothing and shape recognition algorithm. Ziteboard will instantly recognize your intention and smooth the rough drawing of a circle or square [7].

**Twiddla’s** focus is drawing and writing directly on existing web pages, files, and images [8].

**Google Docs** is a well renowned real time, collaborative tool for text and documents, but does not mimic an authentic white board. Its algorithms for concurrency, however, are very crucial and adaptable to our design [9].

**ShareLaTex** is a web application that allows users to work together on a single version of a LaTex document. Its main advantage is that there is no need to download packages manually, and a chat option is available to communicate with collaborators. One disadvantage is the storage of projects - an improvement would be cloud storage for LaTex projects. It is open source and available on github [10].

**Scribblar** is marketed primarily as an education and tutoring tool [11]

**Limnu** most accurately simulates a real life board in respect to erasing, brush strokes, etc. It is the most fluid, most responsive, and fastest updating of the above [12].

Operational transformation is an algorithm applied to merge multiple states of a document in a conflict free manner [13]

**“Real World Challenges to Collaborative Text Creation”** explores issues involving editing collaboratively including editing in a browser, tracking changes, undoing, merging changes [14].

**“Real Time Collaboration Technology Roundup”** discusses operational transformation,

differential synchronization, and CRDT techniques and how they’re used [15]

**REFERENCES**

[1] RealtimeBoard. <https://realtimeboard.com/>

[2] AWW App. <https://awwapp.com/>

[3] SketchTogether. <https://sketchtogether.com/>

[4] Sketchboard.io. <https://sketchboard.io/>

[5] WhiteBoard. Fox. <https://whiteboardfox.com/>

[6] Conceptboard. <https://conceptboard.com/>

[7] ZiteBoard. <https://ziteboard.com/>

[8] Twiddla. <https://www.twiddla.com/>

[9] Google Docs. <https://www.google.com/docs/about/>

[10] Share Latex Github repository. <https://github.com/sharelatex/sharelatex>

[11] Scribblar. <https://scribblar.com/>

[12] Limnu. <https://limnu.com/>

[13] Agarwal, S. Building a real-time collaborative editor using Operational Transformation. Link: [https://medium.com/@srijancse/how-real-time-collaborative-editing-work-operatio nal-transformation-ac4902d75682](https://medium.com/@srijancse/how-real-time-collaborative-editing-work-operational-transformation-ac4902d75682)

[14] Wilm, Johannes & Frebel, Daniel. (2014). Real-world challenges to collaborative text creation. 10.1145/2723147.2723154. Link: [https://www.researchgate.net/publication /276206598\_Real-world\_challenges\_to\_collaborative\_text\_creation](https://www.researchgate.net/publication/276206598_Real-world_challenges_to_collaborative_text_creation)

[15] Irisate. Real Time Collaboration Technology Roundup. Link: <https://irisate.com/collaborative-editing-solutions-round-up/>

[16] Web Whiteboard. <https://webwhiteboard.com/>.

[17] David A. Nichols, Pavel Curtis, Michael Dixon, and John Lamping, “High Latency,

Low-Bandwidth Windowing in the Jupiter Collaboration System.”

[18] David Wang, Alex Mah, and Soren Lassen, “Google Wave Operational Transformation,”

July 2010.

[19] Sun, Chengzheng and Clarence S. Ellis: “Operational Transformation in Real-Time

Group Editors: Issues, Algorithms, and Achievements.”

[20] Operational Transformation, the real time collaborative editing algorithm. Link:

[https://hackernoon.com/operational-transformation-the-real-time-collaborative-editing-al](https://hackernoon.com/operational-transformation-the-real-time-collaborative-editing-algorithm-bf8756683f66)

[gorithm-bf8756683f66](https://hackernoon.com/operational-transformation-the-real-time-collaborative-editing-algorithm-bf8756683f66)

[21] Analysing different Operational Transformation algorithms for collaborative editing.

Link: [https://hackernoon.com/analysing-different-operational-transformation-algorithms- for-collaborative-editing-60fcc49ef24b](https://hackernoon.com/analysing-different-operational-transformation-algorithms-for-collaborative-editing-60fcc49ef24b)

[22] Sun, Chengzheng, David Chen, and Xiaohua Jia: “Reversible inclusion and exclusion

transformation for string-wise operations in cooperative editing systems.”

[23] Collaboration Software Analysis Pie Chart Image.[https://financesonline.com/collaborat](https://financesonline.com/collaboration-software-analysis-features-types-benefits-pricing/)

[ion- software-analysis-features-types-benefits-pricing/](https://financesonline.com/collaboration-software-analysis-features-types-benefits-pricing/)

**SketchWise: Business Plan**

**Mission**

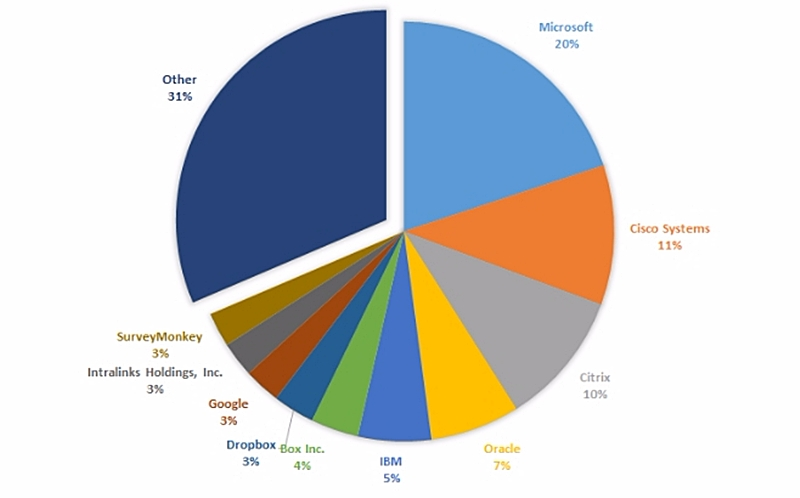
Sketchwise strives to become a common fixture in all offices and classrooms for all artists, designers, and industry professionals as a collaborative editing tool used for professional and educational purposes. Sketchwise aims to improve visualization of planning and design in order to improve workflow and communication within teams. It aims to provide a variety of tools in order to enhance design work and aid the learning process. Sketchwise runs on personal computers, mobile devices, and tablets.

**Product**

Sketchwise is a distributed editing tool used for collaborative whiteboarding in academic, professional, and artistic settings. Its user interface consists of either a blank screen or pre-made templates ranging from business to artistic models. The platform will provide options of using different visual elements such as textboxes, drawing tools, images, shapes, colors, diagrams, lines, various brush strokes and writing utensils, etc. The product will be built using advanced distributed systems technology that enables concurrent editing, consistent viewing by all clients, and low latency that enhances availability.

**Opportunity & Industry**

SketchWise has a niche it can potentially occupy in several fields including company meetings with remote attendees, design process complements, and professional design industry users looking for more individual interactivity than smart boards offer.

The following chart illustrates the current market share of the collaborative software industry. Identifying the software provided by each individual holder shows that many of these softwares provide a single, sectional purpose, but do not sustain when combining functionalities. Furthermore, many are branded with general use, but do not offer specific tools necessitated by certain markets. By providing a platform that combines the functionality of various document inscribing tools, while providing tools specific to its target markets, SketchWise has the potential to capture a respectable market share of the internal tools in its sector. [23]

**Market**

Our target market is design professionals, students, and industry professionals in a collaborative, digital setting. While SketchWise has many competitors in the collaborative document editing realm, SketchWise is also unique in its advents to its specific target markets. Namely, SketchWise provides many different types of visual elements necessary to complete a holistic whiteboarding experience, generating thoughts and ideas fluidly by allowing various forms of expression, not restricted to textual or free-form drawing elements. Furthermore, SketchWise’s internal technology provides significant improvements in terms of consistency complemented with low latency as well as concurrency control to provide a stronger service and a more seamless online collaboration experience.