EMDA 簡略化の為のツール作成

福知山公立大学 情報学部情報学科

32245105　和田想

指導教員　橋田光代　准教授

提出日　2026 年 1 月 30 日

— 目次 ———

— 図目次 ———

— 表目次 ———

1. はじめに

ツールとは，目的を達成するための道具として用いられるコンピュータプログラムのことである．今回達成したい目的というのは，EMDA 分析の簡略化，つまり木構造の簡易作成である．この研究を始めた経緯として，私が 2 年生の時に EMDA 分析を行った際，PowerPoint で木構造を作成したのだが，膨大な時間がかかってしまい分析の進行が遅くなってしまった事から，この作業を簡単にすれば EMDA 分析自体にも触れやすくなるのではないかと考えたため，今回のツール作成に至った．開発環境としては，Python に標準搭載されている GUI ライブラリである Tkinter を使用した．

本稿では，EMDA の前身となった GTTM，そして EMDA の説明を踏まえた上で，プログラムの制作過程と最終的な機能，性能評価について述べていく．

2. EMDA について

2.1 音楽理論 GTTM とは

音楽理論とは，音イベントを時系列で構造解析する技術の事である．この技術を用いることで，時間の進行によって生じる音イベントをグループ分けした上で，重要な音を見つけることが出来る．Generative Theory of Tonal Music(GTTM) は，作曲家・音楽学者の Lerdahl と，言語学者の Jackendaff によって 1983 年に提案された音楽理論である．

特徴としては楽曲の簡約や木構造による表現が挙げられ，構造解析は，音楽の認知に必要とされる 4 つのサブ理論から成る．1 つ目はグルーピング構造であり，楽曲を楽曲中の動機や楽節といった単位のグループに分節し，分節された各グループの階層構造を定めるもの，2 つ目は拍節構造であり，各拍節レベルにおける強拍と弱拍を定めるもの．3 つ目がタイムスパン簡約であり，グルーピング構造の解析と拍節構造の解析をもとに重要な音の選出を繰り返し，ボトムアップに木構造 (タイムスパン木) を作っていくものである．?

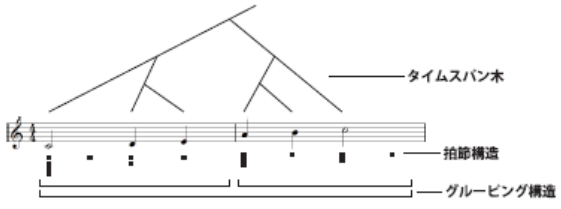

図 1: GTTM のタイムスパン木

タイムスパン木の構造では，枝 (branch) が幹 (stem) の従属部になっており，幹が枝より重要な音であることを表している．4 つ目に延長的簡約という，機能和声に基づく緊張-弛緩構造をトップダウンに分析するものがある．

2.2 EMDA

今回ツールを作成する目的である EMDA(Emotion Movement Design Annotator)?は，GTTM を音楽だけでなく様々な作品に応用したものであり，対象である音楽や物語の根幹となる部分を軸に木構造を作成し，それまでのメロディや事象が何に繋がっているのか，何が重要なのかを可視化したものである．分析の中で必要ではあるが，時間がかかってしまう木構造の作成という部分に焦点を当て，ツールを作成した．

3. 制作過程

3.1 「木」部分の作成

今回，最終的にツールで作成したいものは以下のようなものである．

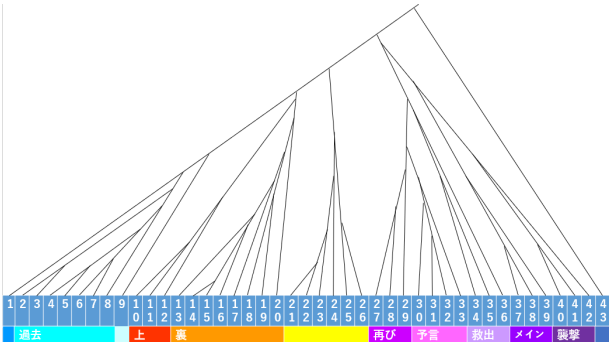

図 2: 以前作成した木構造

作成するにあたって，上部の木構造部分と下部のイベント部分で分けて考えることにし，順序としては木構造部分を作成した上で，テキストボックスをそれに連動させる形で追加するという手順で行った．

木構造を作成する上で，必要な機能として自由に線を動かす機能が真っ先に思い浮かび，その部分から手を付けることにした．最初に，Tkinter のキャンバスに描いた左右 2 本の線分の「上端」だけをドラッグで動かせるようにし，線同士が交差または十分近づいたら，ドラッグした側の上端を相手線の中点へスナップして結合状態にしつつ，子側を中点から一定距離以上離して離すと結合解除できるようにした．

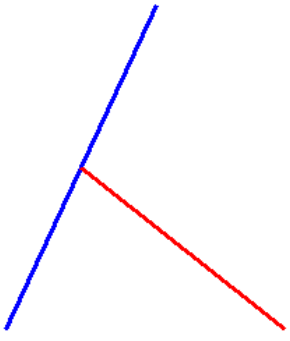

図 3: 2 本線の木構造

木構造自体はこの動作を繰り返していけば形になると考えた．次に，線の数を 2 本から 6 本に増やして動作させたが，この時の線はそれぞれの中点にしかスナップしないような仕組みになっていたため，1 つの幹に対して枝が 2 本吸着する形になった時に同位置に吸着するという問題が発生した．

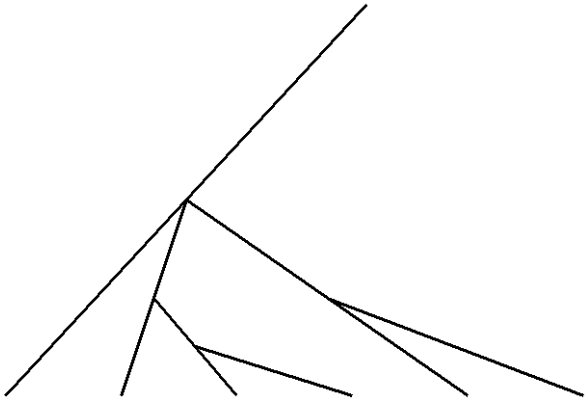

図 4: 中点に吸着する 6 本線の木構造

その問題を解決するべく，線を吸着させる部分を線の中点ではなく，t 値 (全ての線の数-1) で等間隔にスロットを作り，既に使用しているスロットには他の線が吸着しないようにした．

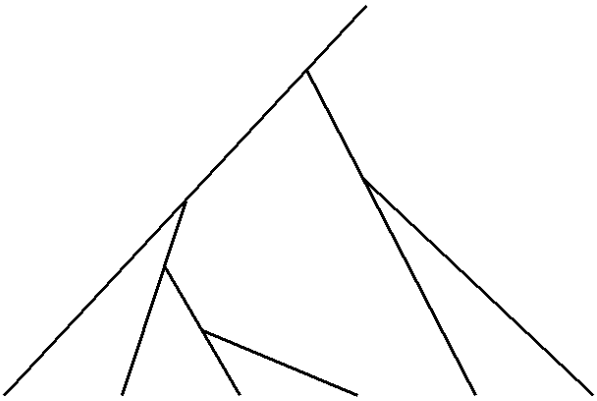

図 5: 各スロットに吸着する 6 本線の木構造

これにより，木構造を作成するための基礎的な機能は実装出来たと言えるが，とても便利といえるものではない為，さらに機能を追加した．

まず，イベントの数は作品によって変動する為，自由に線の本数を変えられる機能を追加した．テキストボックス内に本数を入力することで瞬時にその本数になる機能と，ボタンを押すことにより，右端 (一番新しい線) から線を追加，削除することが出来る機能の 2 種類である．また，PowerPoint で作業していた時に煩わしかった作業の 1 つである，イベント間にイベントを追加，削除するという機能を追加し，最終的には以下の図のようになった．

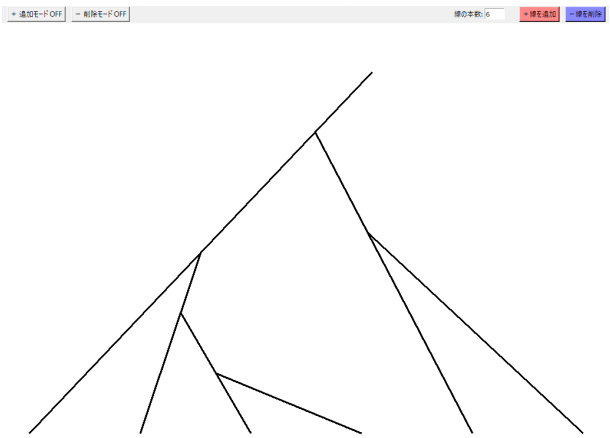

図 6: 木構造部分完成時のウィンドウ

各種ボタンを押すことによって，それぞれの機能が使えるようになっている．これにより，木構造部分だけなら本数が自由に変えられるツールが出来たと言える．

3.2 「イベント」部分の作成

次に，作成した木構造部分に連動させながらテキストボックスを追加する段階に入った．まずは線分同士の間にテキストボックスを入れ込むような形で作成した．

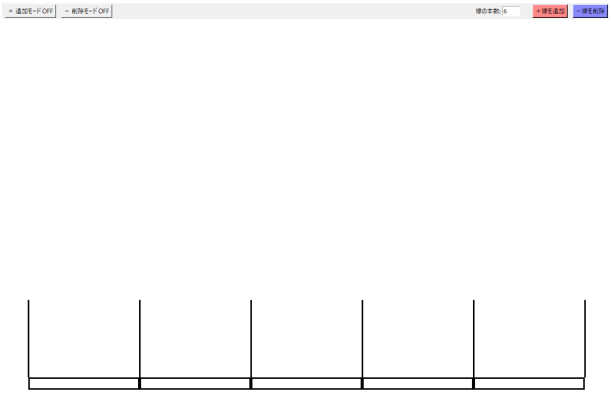

図 7: 上段テキストボックス追加

テキストボックスとその左上の角から伸びている線分が連動している仕組みになっており，例えば左から 2 番目の線を削除した時，共に左から 2 番目のテキストボックスが消えるようになっている．

最初に例として挙げた木構造を再現するのなら，それぞれのイベントをグループ分けするためのテキストボックスの追加も必要になってくる．グループ分けという目的に沿ったものにする為，初期状態は全てのイベントを包括する大きなグループがあり，それを分割させていくことで，イベントのグループ分けを再現しようと考えた．この考えのもと作成したのが以下の画面になる．

図 8: 下段グループ列分割前

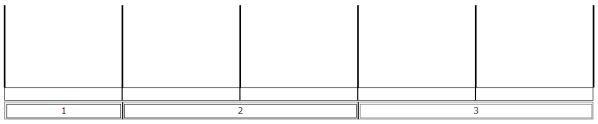

図 9: 下段グループ列分割後

このように，イベントテキストボックスの下部に 1 つの大きなテキストボックスを配置し，それを分割することでグループ分けとした．通し番号については分かりやすいように手動で入力したものである．他にもいくつか細かい修正や機能を追加した時の初期ウィンドウが以下の画像になる．

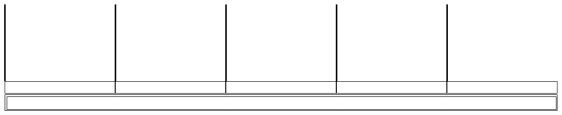

図 10: 下段グループ列追加後ウィンドウ

3.3 ユーザーフィードバック

ここまで作成してきたツールを実際に使ってもらい，不便な点や改善点を挙げてもらうテストを行った．対象ユーザーは同学年のゼミ生 5 人と顧問 1 人の計 6 人，調査方法はまず以下の画像と同じ木構造をツールを使用して作成してもらい，

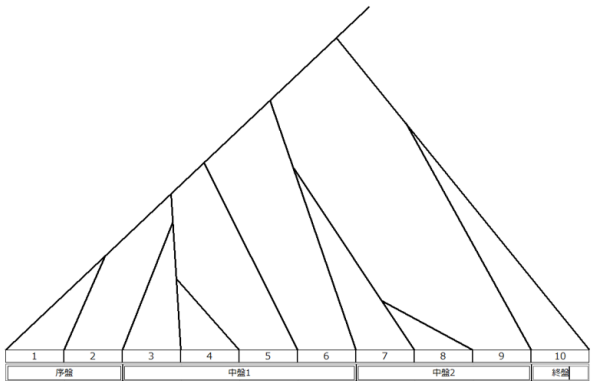

図 11: 調査用木構造 (編集前)

この木構造を作成した後にイベント「6」とイベント「9」を削除，イベント「7」とイベント「8」の間にイベント「!」を追加，グループの分割位置を変える，といった追加の操作指示をすることで，木構造を作成する時に使うであろう機能をすべて使ってもらうという形で調査を行った．最終的に作成した木構造は以下のものである．

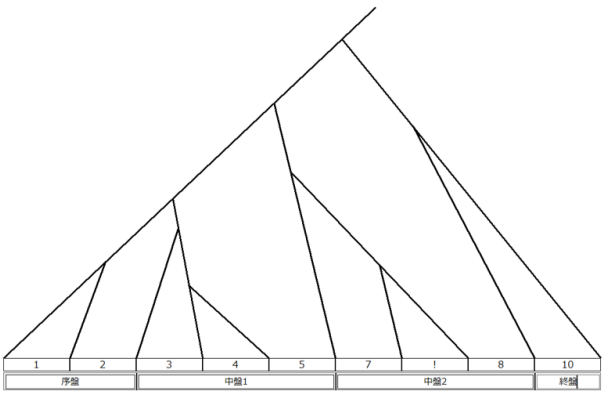

図 12: 調査用木構造 (編集後)

この木構造を作成する上での感想や改善点を挙げてもらい，それを元にプログラムに修正を加えていった．フィードバックとしては，直感的に操作が分かりづらい箇所がある，判定の拾い方次第では不具合のような挙動になることがある，などの自分で操作していた時には気づけなかった意見があり，テストプレイの重要性を実感した．

4. コード解説

4.1 概要

本プログラムは，Tkinter を用いて GUI を構築し，キャンバス上に複数の線分を配置・編集するツールである．線分の上端をドラッグして形状を変更できるほか，線分同士を吸着させることで階層構造を形成できる．また，線分間の区間にテキスト入力欄を自動配置し，区間ごとのテキスト入力を可能にしている．さらに下段には，複数区間をまとめる「グループ行」を設け，境界をクリックして区間の分割・結合を行うことが出来る．状態は JSON として保存・読み込みが可能であり，Redo / Undo にも対応している．

4.2 右端の隠し線

このコードの重要な設計ポイントとして，内部的に”右端の隠し線”を 1 本持つというものがある．self.lines には 内部本数が入るのだが，実際に画面に描画するのは self.lines[:-1] としている．理由としてはテキストボックスを線と線の間に生成するようにしているため，テキストボックスを N 個作るには線が N+1 本必要となるが，1 本の線に対し 1 つのテキストボックスが対応しているデザインにするため，右端の線を隠している．

4.3 Line クラス

これは主に線の状態と描画を担当しているクラスであり，線 1 本ずつに様々な要素を持っている．本ツールでは，線の x 位置をピクセルではなく 0〜1 の比率（世界座標）として保持している．これにより，ウィンドウサイズの変更やズームイン・ズームアウトを行っても，線の配置を保ったまま再描画できる．具体的には Line.map_x() / Line.unmap_x() により，世界座標と画面座標を相互変換している．表示範囲は view_min, view_max（世界座標）で表し，これを変更することでズームイン・ズームアウトを実現している．ズームでは表示幅が極端に小さくなることを防ぐために最小幅 _min_span を設けている．

線の位置の基準について解説していく．base_ratio は線の根元の x 座標を表しており，base_y_ratio は y 座標を表している．この時 y 座標はキャンパスの範囲を上端から 0〜1 とした時，0.9 固定としている．length はデフォルトの線の長さである．base_x，base_y は根元のピクセル座標，top_x，top_y は上端のピクセル座標を表している．

線をスナップする時の親子関係を表しているものとして，parent が親の Line，children を子の Line リストとして保存している．また，snap_t で親の線分上のどこにスナップしているか，used_t_values で親の線に対して，すでに使われた t を判別するようにしている．これは，3.1 でも述べた全体の線の数-1 を t 値としてそれに応じて等間隔にスロットを作り，既に使用しているスロットには他の線が吸着しないという仕様を実現するためのコードである．

ドラッグ操作に対応し，上端座標を更新，比率更新，再描画を行うのが move(dx,dy) である．また子線がある場合は follow_parent() により追従させており，これは親線の根元から上端の線分上で t=snap_t の位置に自分の上端を配置し，親の変形に追従するものである．

4.4 SnapHierarchyApp クラス

ツール全体の状態・UI・イベント・保存/復元などをまとめて持っているクラスである．

上段テキスト欄の説明からしていく．上段テキストの入力欄は「線と線の間」に自動で作られるようになっている．隣り合う line[i] と line[i+1] の間が 1 つの区間になり，その区間に 1 つ Entry が置かれる．そして，線を増減した時にテキストがズレないように各入力欄は各テキスト左側の線の line_id である left_line_id をキーにしてテキストを保存・復元している．これにより線の配列が変わったとしても，どの区間の文字かを保ちやすくしている．さらに，左右矢印キーで入力欄間を移動できるよう，Entry を順序リストにまとめて _focus_prev/_focus_next でフォーカス移動するようにし，マウスで入力欄をクリックする手間を省けるようにしている．

次に下段グループ行について，下段は複数区間をまとめるグループ分けをするための行である．group_boundaries に分割位置（各線分の位置）を持っておき，_rebuild_group_row() が，その境界に合わせて Entry を作り直す仕組みになっている．

分割モードが ON のときは，下段をクリックするとクリック位置に近い境界候補を探し，その境界が既にあれば削除，無ければ追加

という形でグループの分割と結合が出来る．

下段の文字も上段と同じく left_line_id をキーに保存するため，分割位置を変えて Entry が作り直されてもグループがそのまま保持されるようになっている．

各モードについての説明をしていく．追加モードは add_mode で定義されており，追加モードを起動することで背景色を変化させ，show_insert_guides() により追加ガイドを表示している．このガイドをクリックしたときに，グループの境界が右にずれ，テキスト間に新しいテキストを挿入しているように見せている．削除モードは delete_mode で定義されている．追加モードと削除モードは排他的であり，片方が ON になった時もう片方は OFF になるよう設計している．削除モードを起動すると show_delete_guides() により削除ガイドが表示され，こちらも背景色が変化する．先ほど追加する時はグループの境界を右にずらしていると述べたが，削除モードでは削除対象の境界を削除した後，それより右側の境界を 1 つ左にずらしている．分割モードについて説明していく．split_mode で定義されており，分割モードで境界を付けるには最低でも内部で 3 本以上の線が必要になるため (2 本ではテキストボックス単体となる)，_handle_split_click_local では len(self.lines) ¡ 3 の場合に何もしないよう処理を施している．分割モードを起動している時に全線の base_x を取得し，クリックしたときに距離が対象範囲に近かった場合，分割処理を行うようになっている．追加モードと削除モードは排他的であるのに対し，分割モードは独立している．

Redo / Undo 設計について，過去状態の保存先である undo_stack と Undo した状態を戻すための redo_stack を用いて，現在の状態を状態の変更前にデータとして積んでおくことで可能にしている．

JSON で木構造データをどのように保存しているかを解

説していく．JSON には状態を保存しておく必要があるのだが，この状態には様々な要素がある．

状態に含めている要素

・線（Line）の集合

id（line_id：永続 ID）

base_ratio（根元 X の世界座標比率）

top_ratio_x, top_ratio_y（上端位置の比率）

parent_id（親の line_id）

snap_t（親の線分上の吸着位置 t）

・上段テキスト

top_texts (left_line_id をキーにした辞書)

・下段テキスト（グループ行）

boundaries（境界のインデックス配列）

texts_by_left_id（left_line_id のテキスト）

・表示範囲（viewport）

view.min, view.max（世界座標での表示範囲）

これらの情報をファイル内に保存することで復元している．例として，次の木構造を挙げる．

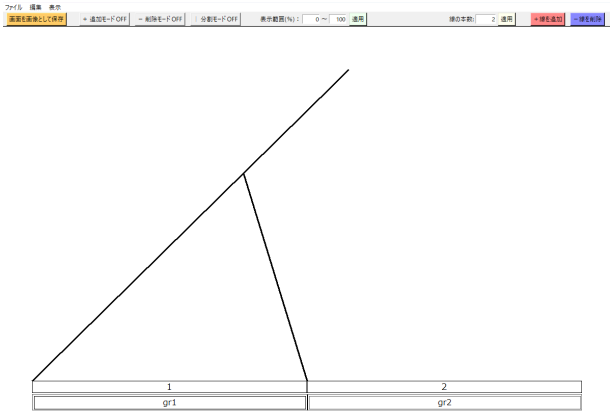

図 13: JSON ファイル確認用木構造

この木構造を保存した時の JSON ファイルは以下のようになる．line_id をキーにする事で，線の追加や削除でインデックスが変わってもテキストが保持される仕組みになっている．

最後に，スクリーンショット機能について説明していく．これは GUI 全体ではなく，キャンバス領域のみを画像として保存し，木構造の状態を資料用に出力できるようにしたものである．Pillow の ImageGrab を用いてキャンバスの画面上絶対座標である winfo_rootx/y とサイズ winfo_width/height から切り出し矩形である bbox を算出し，その領域のみを取得して画像保存する仕組みになっており，保存時には PNG と JPEG を選択することが可能である．

```json
{
  "lines": [
    {
      "id": 0,
      "base_ratio": 0.05,
      "top_ratio_x": 0.5683333333333334,
      "top_ratio_y": 0.10981912144702842,
      "parent_id": null,
      "snap_t": null
    },
    {
      "id": 1,
      "base_ratio": 0.5,
      "top_ratio_x": 0.39555555555555555,
      "top_ratio_y": 0.37295434969853575,
      "parent_id": 0,
      "snap_t": 0.6666666666666666
    },
    {
      "id": 5,
      "base_ratio": 0.9500000000000001,
      "top_ratio_x": 0.95,
      "top_ratio_y": 0.7054263565891473,
      "parent_id": null,
      "snap_t": null
    }
  ],
  "top_texts": {
    "0": "1",
    "1": "2"
  },
  "group": {
    "boundaries": [
      1
    ],
    "texts_by_left_id": {
      "0": "gr1",
      "1": "gr2"
    }
  },
  "view": {
    "min": 0.0,
    "max": 1.0
  }
}
```

図 14: データ確認用 JSON ファイル

5. ツールの機能一覧

ここまでどのようにツールを作成してきたか，プログラムをどのように組んだのかについて説明してきた．これらを踏まえた上で，最終的にどのような機能が実装できたのかを紹介していく．プログラムを起動した直後の初期画面は以下のようになっている．

このように，本プログラムの画面は上部の操作パネルと下部のキャンバス領域から構成されている．操作パネルには，画像保存，モード切り替え，表示範囲指定，線数の操作ボタンを配置している．キャンバス領域には，複数の線分と，線分に対応する上段テキスト入力欄，線群の区間に対応する下段グループ入力欄を配置している．これらにどのような機能があるか，操作方法も踏まえながら挙げていく．

図 15: ツール起動時の初期画面

5.1 線の基本操作

各線は下端（根元）と上端（先端）からなる．下端は画面下部に固定され，上端のみを移動可能である．変更したい線の上端を左クリックし，押下したままマウスを移動すると上端を任意位置へドラッグ出来る．

木構造を作成するには，線の親子関係を作る必要がある．方法としては，子にしたい線の上端をドラッグし，親候補の線分上 (t 値でされたスロット) へ上端を近づけ，十分近い状態でマウスボタンを離すと，子線上端が親線へ吸着し，親子関係が作られる．この時，子線は親線に追従するようになる．

子線を再度ドラッグし，親線から離れた位置で離すと親子関係が解除される．解除後，その線は単独の線として扱われ，親に追従しない．

5.2 上段テキストボックス

対象のテキストボックスをクリックし，通常のテキスト入力として文字列を編集できる．また，入力中に左右キーを用いることで，隣接するテキストボックスへフォーカスを移動できる．また，テキストボックスはそれぞれ左上の線と連動している．

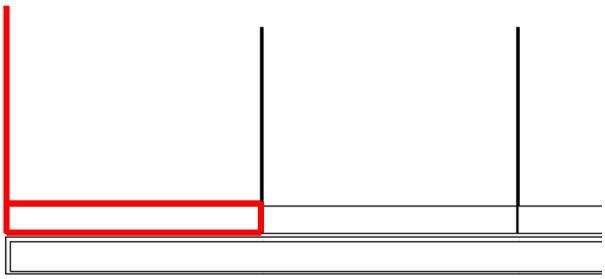

図 16: 線分とそれに対応したテキストボックス

5.3 下段グループ列

上段テキストボックスの下部にグループ入力欄が配置されている．下段グループ列は，線全体をいくつかの区間に分割し，各区間へ代表名やカテゴリ名を付ける用途を想定している．

下段グループの区間分割は，「分割モード」で行う．分割モードを有効にすると，下段グループ列の境界を置きたい，または削除したい付近をクリックした時，境界の追加・削除が可能になる．

図 17: 分割モード ON 状態

下段グループの各入力欄も，上段と同様にテキストボックスをクリックして編集，左右キーによる隣のグループ欄へのフォーカス移動が可能となっている．

5.4 追加モード

「追加モード」を ON にすると，線の下端付近に追加ガイドが表示される．利用者はガイドをクリックすることで，クリック位置に新しい線とそれに連動したテキストボックスを挿入できる．イベント間に新たなイベントを追加したい時に使用する．

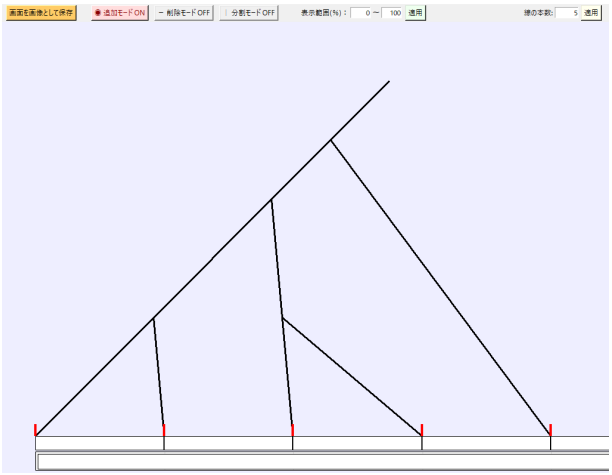

図 18: 追加モード ON 状態

5.5 削除モード

「削除モード」を ON にすると，各線の上端に削除ガイドが表示される．削除したい線のガイドをクリックすることで，その線とそれに連動したテキストボックスを削除できる．削除候補が分かりやすいよう，ガイドにカーソルを合わせた時に連動するテキストボックスの枠も強調表示される．

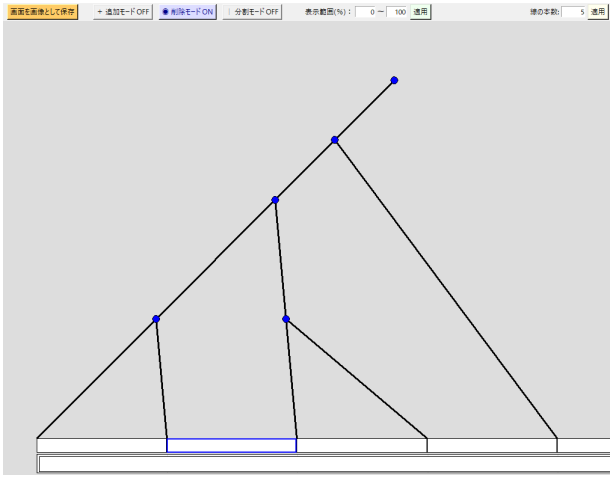

図 19: 削除モード ON 状態

5.6 線数の指定と増減

線の本数は，操作パネル右側の「線の本数」入力欄に数値を入力して適用することでも変更できる．また，「＋線を追加」「－線を削除」ボタンにより，1 本単位の増減も可能である．

5.7 表示範囲操作

マウスホイールによりズーム操作を行う事が出来る．ズームはマウスカーソル位置を基準として行われるため，注目したい領域を中心に拡大できる．右クリックのドラッグ操作により，表示範囲を上下左右へ移動できる．キーボードの左右キーでも左右操作が可能である．操作パネルの「表示範囲（%）」により，現在の表示範囲が確認できる．表示メニューから初期範囲へ戻すリセット機能を使用できる．

5.8 Redo / Undo と初期化

線の移動，接続，追加・削除などを操作履歴として復元できる機能である．Ctrl+Y により「Redo」，Ctrl+Z により Undo を実行する．編集メニューでも実行が可能である．また，編集メニューには初期状態へ戻すリセット機能も実装している．このリセットは編集操作として扱われるため，Undo によりリセット前へ戻すことも可能になっている．

図 20: 編集メニュー選択時

5.9 状態の保存・復元

4.4 で述べたように作成した構造と入力内容をファイルとして保存，再読み込みにより復元できる機能である．ファイルメニューから実行が可能であり，選択後にファイルの保存場所，ファイル名などを決める．

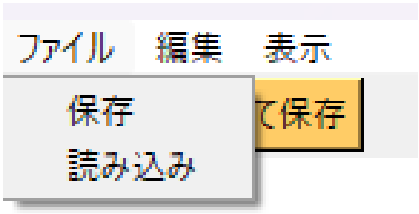

図 21: ファイルメニュー選択時

5.10 スクリーンショット

キャンバス領域のみを画像として保存するスクリーンショット機能である．操作パネルの「画面を画像として保存」を押し，保存先とファイル名を指定すれば画面の内容 (木構造) が画像として出力される仕組みになっている．

6. 性能調査

今回ツールを作成した目的は，木構造作成の簡略化及び時間の短縮である．そこで，PowerPoint を用いての作成とツールを用いての作成でどれくらい作業効率が上がったのかを調べることにした．

6.1 調査方法

PowerPoint とツールで，ランダム生成した木構造を 5 つずつ作成し，時間を計測するという手段を取った．線の親子関係は完全ランダム，上段テキストボックスには 1 から通し番号を振り，下段グループ列は分割位置をランダム，分割後のテキストボックスには gr1，gr2…といった通し番号が振られるようにした．この時，線の数を 8〜12 本で 1 本ずつ増やすことで，似た木構造がランダム生成される確率を減らした．また，PowerPoint とツールの作業を交互に行う事で，木構造作成作業に出来るだけ慣れが生じないように工夫した．

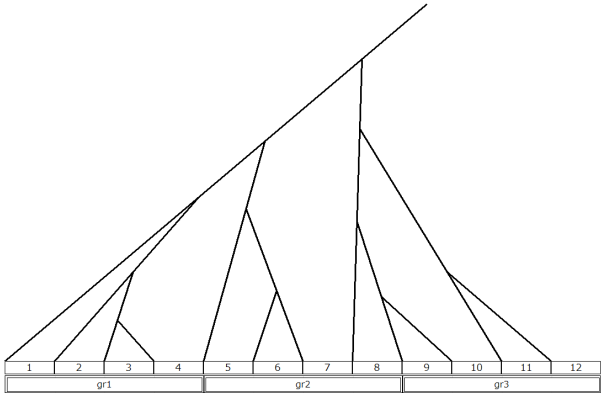

図 22: 計測時にツールで作成した木構造 (n = 12)

6.2 調査結果と考察

5 回ずつ測定した結果が以下の表になる．

	PowerPoint	ツール
1 回目(n=8)	5:41	1:35
2 回目(n=9)	4:07	1:58
3 回目(n=10)	4:27	1:50
4 回目(n=11)	4:40	1:45
5 回目(n=12)	4:42	1:44

表 1: 計測結果

1 回目の PowerPoint の時間が極端に長いのは表の出し方や線の配置に苦戦したためであるが，ツールを使用すればそのような事態も起きないと考え，記録に含んでいる．時間を比較した時に，PowerPoint に比べてツールを使用した時の時間は 2〜3 倍速くなっている事が分かる．また，PowerPoint を使用して木構造を作成する場合，線を 1 本動かすだけでも親子関係にある線や，全体の見え方を考慮した時の周りの線も 1 本ずつ手作業で移動させなければいけない．今回はそのパターンが無かったためこのような時間差になっているが，あった場合はさらに時間の差が広がると考えられる．この結果より，今回作成したツールは，EMDA 分析における木構造の作成を大きく簡略化するものになったと言えるだろう．

7. おわりに

本研究では，EMDA 分析において大きな負担となりやすい「木構造の作成」に着目し，その簡略化と作業時間短縮を目的とした支援ツールを開発した．

私自身が PowerPoint で木構造を作成した際に感じた煩わしさは解消されたものの，使い方次第では不足している機能や不具合なども見えてくると考えられるため，ユーザーフィードバックを継続的に取り入れながら適宜改良していきたい．ツールがどのくらい作業を簡単にしたか，という点も作品や対象人物によって変化するため，テストデータを増やし，有用性を明確にしていきたいと考えている．

今回，EMDA 分析を行う際に不可避だが時間がかかる作業である木構造を対象に，専用のツールを作成してきた．本ツールが，EMDA 分析の作業効率を高めるだけでなく，これまで EMDA 分析をしていなかったユーザーにとって，分析へのハードルを下げる一助となれば幸いである．