

Florida International University
School of Computing and Information Sciences



Software Engineering Focus

Final Deliverable

jTLEX 3.0: Visualization for the TLEX Algorithm

Team Members:

Luis Robaina
Ismael Clark
Emmanuel Garcia
Victoria Fernandez
Antonela Radas
Leandro Estevez
Ronald Pena
Jared Hummer
Pedro Diaz

Product Owner(s):

Mark A. Finlayson

Mentor(s):

Mustafa Ocal

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

This document describes the engineering work done as part of the Capstone project by the jTLEX team. TLEX is an algorithm that allows researchers in Natural Language Processing (NLP) and other fields to analyze TimeML annotated files to find inconsistencies, indeterminacies, and build timelines. The TLEX algorithm is currently implemented as a Java library named jTLEX. jTLEX was also developed as a Capstone project at FIU. jTLEX successfully implements the TLEX algorithm but lacks visualization tools that allow for ease of interpretation. This document explains the engineering efforts behind jTLEX 3.0, a work that aims to provide a web visualization tool for the TLEX algorithm.

Table of Contents

INTRODUCTION	5
CURRENT SYSTEM	6
PURPOSE OF NEW SYSTEM	7
USER STORIES	
IMPLEMENTED USER STORIES	7
PENDING USER STORIES	8
PROJECT PLAN	
HARDWARE AND SOFTWARE RESOURCES	9
SPRINTS PLAN	10
<i>Sprint 1</i>	10
<i>Sprint 2</i>	10
<i>Sprint 3</i>	10
<i>Sprint 4</i>	10
<i>Sprint 5</i>	10
<i>Sprint 6</i>	10
SYSTEM DESIGN	
ARCHITECTURAL PATTERNS	11
SYSTEM AND SUBSYSTEM DECOMPOSITION	14
DEPLOYMENT DIAGRAM	15
DESIGN PATTERNS	16
SYSTEM VALIDATION	18
GLOSSARY	19
APPENDIX	20
APPENDIX A - UML DIAGRAMS	20
<i>Static UML Diagrams</i>	20
APPENDIX B - USER INTERFACE DESIGN	21
APPENDIX C - SPRINT REVIEW REPORTS	25
APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS	28
REFERENCES	29

INTRODUCTION

TLEX (TimeLine EXtraction) is an algorithm developed by researchers Mustafa Ocal and Mark A. Finlayson from the School of Computing and Information Sciences at FIU. TLEX introduces a method for extracting a set of exact timelines from a TimeML annotated text [1]. Timelines are important constructs in Natural Language Processing (NLP), a timeline gives a total ordering of events and times. For example, in question answering (QA), which is an important task in NLP it is often required to have an understanding the overall temporal order of events in the text [1]. Timelines also greatly facilitate comparing sequences of events across documents, which is useful for cross-document alignment and event coreference. Finally, timelines are a natural way of summarizing and visualizing many texts [1].

TLEX, and all its utilities for the NLP researcher, were implemented as a Java library by Capstone students at FIU. The Java library that implements the TLEX algorithm carries the name of jTLEX. *jTLEX - A Java Library for Timeline Extraction* successfully implements the requirements specified by the TLEX algorithm and has been tested to verify correctness and performance criteria [2].

After successfully implementing the jTLEX library, the team behind the engineering of jTLEX (referred to as the *jTLEX team* in this document) realized that the output produced by the TLEX algorithm is difficult to understand in text form. This is where jTLEX 3.0 becomes the next logical step for the implementation of the TLEX algorithm to be completely useful.

In this project we engineered a solution that provides an easy-to-understand user interface and an extensible backend system to study the analysis of TimeML annotated files as implemented in the jTLEX library and specified by the TLEX algorithm.

Current System

The system before jTLEX 3.0 was a java library that implements the TLEX algorithm. This library (jTLEX) was engineered to be a formally correct and efficient implementation of the TLEX algorithm that can be used by any researcher in NLP or other fields. The jTLEX library collects many different java classes that implement different functionalities of the TLEX algorithm including Inconsistency detection, indeterminacy detections, partitioning, and timeline extraction.

The problem with the jTLEX system is that it lacks a powerful visualization tool. This makes the solution less attractive to the NLP researcher.

Purpose of New System

The new system, jTLEX 3.0, builds on top of the work done for jTLEX and aims at adding powerful visualizations to allow the NLP researcher to quickly understand the output of the TLEX algorithm. The visualization tool is engineered as a web application that runs on the user's browser and connects via HTTP requests to a backend system that contains the jTLEX library and serves as an interface from the user's browser to the TLEX algorithm.

USER STORIES

The following section provides detailed user stories that were implemented in this iteration of the jTLEX project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

- As a jTLEX developer, I would like to Read Chapter 2: Scrum Framework and Chapter 4: Sprints of the "Essential Scrum" textbook.
- As a jTLEX developer, I would like to Read J.F. Allen's paper on timeline extraction.
- As a jTLEX developer, I would like to read and understand TimeML Annotation Guidelines.
- As a jTLEX developer, I would like to Read the TimeBank Corpus research paper.
- As a jTLEX developer, I would like to Read TLEX paper to understand the implementation of the TLEX algorithm.
- As a jTLEX developer, I would like to Parse each CNN (.tml) file into TimeML Graphs to get familiar with the jTLEX library code.
- As a user, I would like for each TimeML graph, partition it: Transform graphs to TCSPs, Extract timelines from TimeML graphs, Check and Detect Inconsistencies on TimeML graphs, Identify Indeterminacies on TimeML graphs,
- As a jTLEX developer, I would like to discuss addition of subordinate links into Graph objects to get familiar with the project's code.
- As a jTLEX developer, I would like to Create a basic React application to start the visualization efforts of the jTLEX library.
- As a user, I would like to be able to see a Graph Component in React, process a TimeML file and show a TimeML graph.
- As a user, I would like to have an "About" page (The About page must break down the main details of the TLEX paper to show the user how the algorithm works).

- As a jTLEX developer, I would like to create a SpringBoot application to work as the backend for the visualization part of the project.
- As a jTLEX developer, I would like to do research on Gradle to improve the backend application.
- As a jTLEX developer, I would like to Switch Maven Project to Gradle since it is a better tool in the development of the project.
- As a jTLEX developer, I would like to do literature review on JSON formats.
- As a jTLEX developer, I would like to convert jTLEX output into a JSON format output so it can be displayed by frontend.
- As a jTLEX developer, I would like to remove in/out links from the JSON output.
- As a jTLEX developer, I would like to test, document, and improve the jTLEX core library code.
- As a user, I would like to be able to upload a TimeML file to the application.
- As a user, I would like to be able to paste a TimeML file into the text box.
- As a user, I would like to be able to visualize each Partition of the graph.
- As a user, I would like to be able to visualize each Timeline.
- As a user, I would like to be able to visualize Inconsistency.

Pending User Stories

- As a jTLEX developer, I would like to Research available profiling tools in java libraries to better understand the performance of the jTLEX library.
- As a jTLEX developer, I would like to Perform code profiling of our java library (jTLEX).
- As a jTLEX developer, I would like to Report performance bottlenecks.
- As a jTLEX developer, I would like to Propose solutions to any performance issue.
- As a user, I would like to see how events in the timelines overall with each other.
- As a user, I would like to have a visualization of indeterminacies in the TimeML file.

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated agile development techniques and as such required the sprints to be accordingly planned. The sprint planning are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

Software:

- Eclipse IDE for Java Developers
- Subversion VCS
- Java SpringBoot
- NodeJS
- Gradle
- Java JDK
- Zoom (team communication)
- Trello (team sprint planning)
- Discord (team communication)

Hardware:

- MacBook Air (development and testing)
- MacBook Pro (development and testing)
- Dell Vostro (development and testing)

Sprints Plan

Sprint 1

- Setup Development environment.
- Review provided material about jTLEX.
- Research and compare visualization tools.

Sprint 2

- jTLEX Port/ Backend.
- jTLEX Core Testing, Documentation, and Refactoring.
- Decide with PO what must be visualized in the jTLEX frontend page.
- Create a sketch of the frontend page.
- Design a JSON structure that specifies all the data that jTLEX PORT must send to the frontend.

Sprint 3

- Visualization and Frontend.
- jTLEX Port (Backend).
- jTLEX Core Testing, Documentation, and Refactoring.
- Implementation of JSON Engine
- Create sample graphs in expected JSON format.
- Create a React application.
- Visualize the sample graph.
- Research on APIs.
- Create the jTLEX backend API.

Sprint 4

- Visualization and Frontend.
- jTLEX Port (Backend).
- jTLEX Core Testing, Documentation, and Refactoring.
- Implementation of JSON Engine
- Perform code profiling in jTLEX Core java library.
- Implement Graph Component in React to show a TimeML graph.
- Work on the “About” page.
- Handle Component to upload TimeML files in React.
- Create a component to copy/paste TimeML files into the text box.

Sprint 5

- jTLEX Core Testing, Documentation, and Refactoring.
- Perform code profiling in jTLEX Core java library.
- Implement Partitioning of a TimeML graph.
- Implement Timeline frontend component.
- Implement Inconsistency frontend component.
- Implement Indeterminacy frontend component.

Sprint 6

- Finish documentation of the project.
- Create an installation guide for the project.
- Create posters, presentations, and demo videos.
- Complete final deliverables.

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

The architecture of the jTLEX 3.0 system includes two main components: The jTLEX frontend, which acts as a service requester, and the jTLEX backend, which is the service provider. These two components communicate via HTTP requests. Another important part of the system architecture is the jTLEX JAR file which aggregates the classes that jTLEX (the TLEX java library) implements. The jTLEX JAR file is consumed from the system's SpringBoot backend and the output is rendered with interesting visuals on the users' web browser running the TLEX frontend application.

Architectural Design	
Name	Description
jTLEX Frontend	The jTLEX Frontend is the client side of our visualization project, the user here inputs a TimeML file or copies the text content of a TimeML file, then the user can observe the results of processing the TimeML file using the TLEX algorithm implemented in jTLEX.
jTLEX Backend	The jTLEX Backend is the server side of our application, is in charge of sending the given TimeML file to the jTLEX JSON Engine core library to be processed, and also sending the result back to the jTLEX frontend
jTLEX JSON Engine	The JSON Engine is in charge of getting the result of processing the TimeML file from the core jTLEX library and transforming it to a JSON file to be displayed by the jTLEX Frontend.

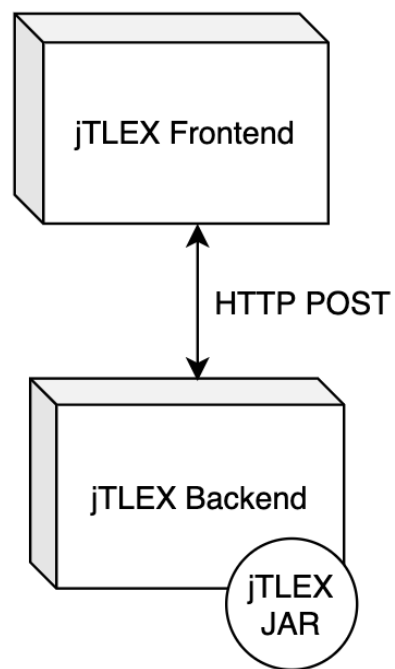
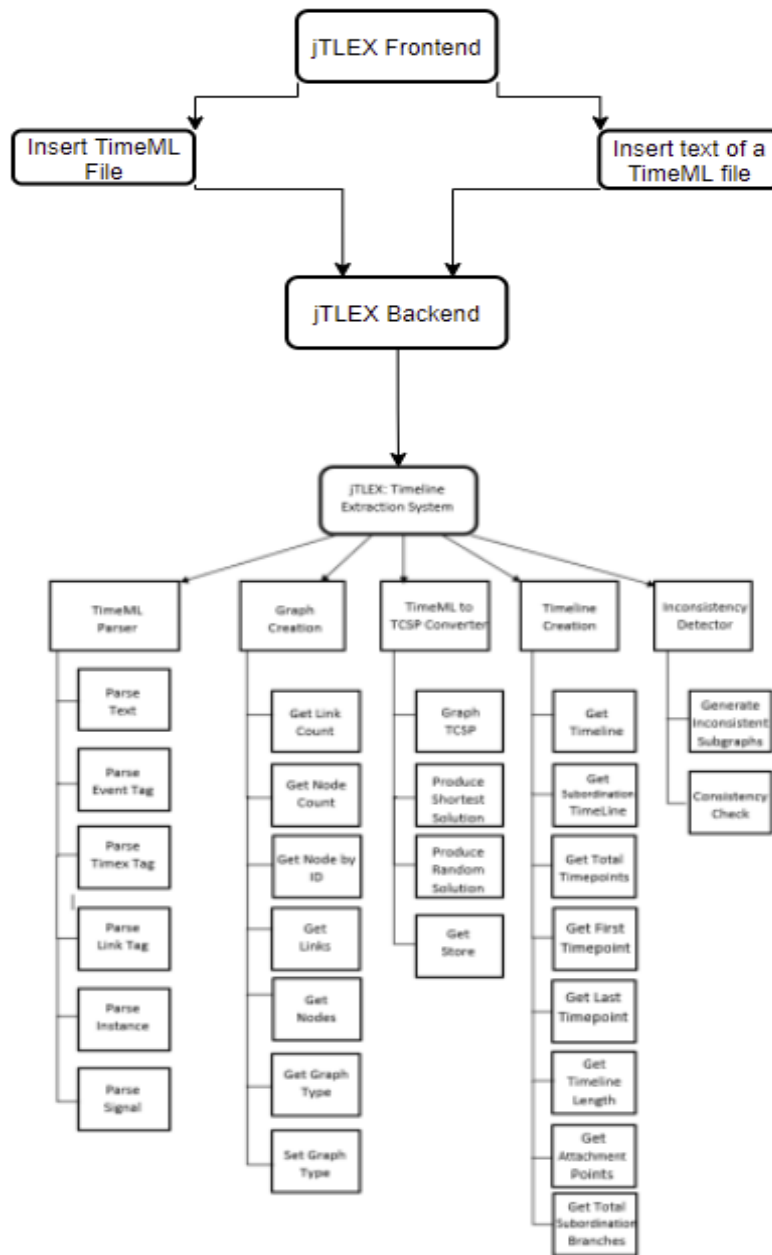


Fig 1. Simplified architecture diagram for the jTLEX visualization system

System and Subsystem Decomposition



Deployment Diagram

The jTLEX visualization application has only been deployed on local development machines, including the following systems: MacBook Air 8 GB 1.6 GHz, MacBook Pro, Dell Vostro. The jTLEX visualization application is engineered to follow a client-server architecture communicating using HTTP requests. The following are diagrams that illustrate the deployment of jTLEX visualization system both locally for development and testing, and remotely once the system is production ready

Local Development Deployment

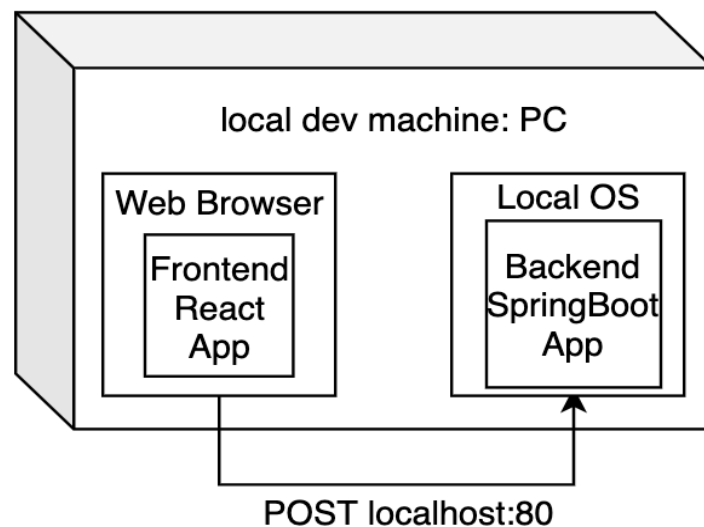


Fig 2. Deployment of jTLEX visualization system locally

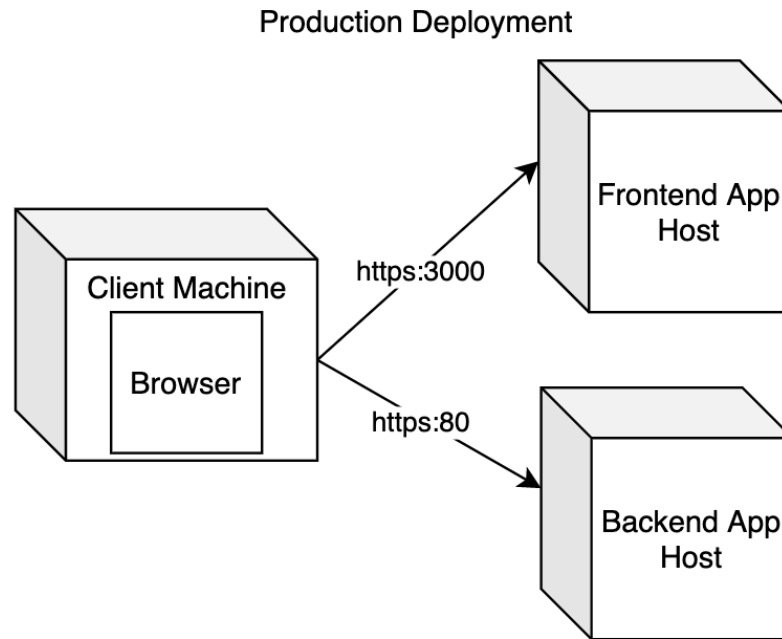
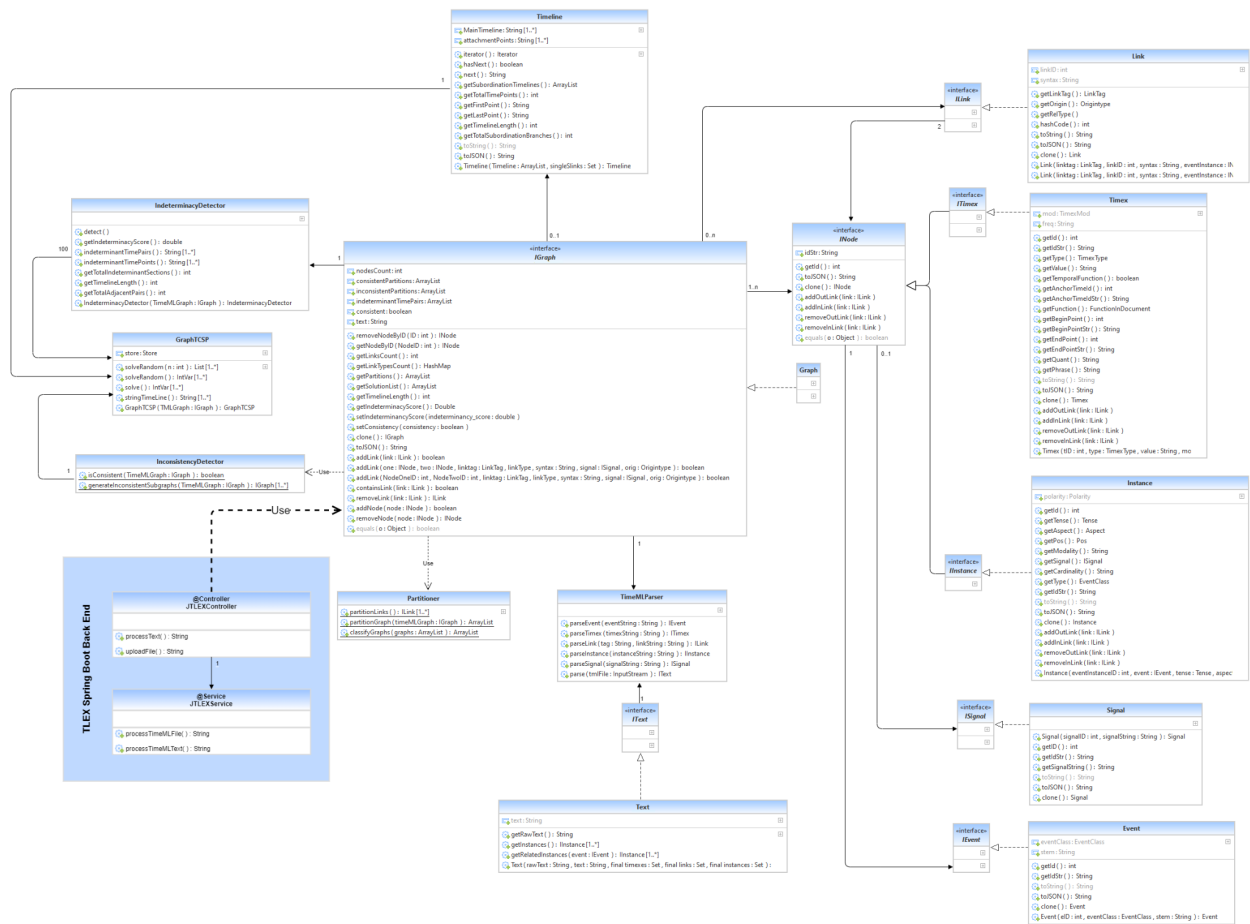


Fig 3 Projected production deployment architecture

Design Patterns

The following diagram illustrates most of the classes that embed the functionality of the jTLEX system on the backend. The Java SpringBoot backend system servers HTTP requests by interacting with the jTLEX core library of code via the IGraph interface.



SYSTEM VALIDATION

VERIFICATION

Test File	Purpose	Method	Preconditions (Input)	Expectation	Result
(1) wsj_0006.tml	Show correct functioning for file upload	Manually upload TimeML file using GUI form	Valid TimeML File	Correct Result Displayed	TimeML Graph displayed
(2) wsj_0006.tml (text)	Show correct functioning for pasted text	Manually paste TimeML annotated text using GUI textbox	Valid TimeML annotated text	Correct Result Displayed	TimeML Graph displayed
(3) sample_incorrect.tml	Detect incorrect content of file	Manually upload TimeML file using GUI form	Invalid TimeML File	Illegal Argument Exception Thrown	Illegal Argument Exception Thrown
(4) sample_incorrect.tml (text)	Detect incorrect TimeML format for pasted text	Manually paste TimeML annotated text using GUI textbox	Invalidly annotated TimeML text	Illegal Argument Exception Thrown	Illegal Argument Exception Thrown

GLOSSARY

TimeML file: A text file annotated following TimeML annotation guidelines.

NLP: Natural Language Processing.

TCSP: Temporal Constraint Satisfaction Problem.

Appendix A - UML Diagrams



Appendix B - User Interface Design

The following screenshots illustrate details of our design choices for the user interface.

TimeML Graph Interface Component

The main component of the TLEX algorithm is the TimeML Graph, a graph constructed from the input TimeML File. We designed a user interface component to render the TimeML graph with useful statistics about components of the graphs including number of nodes, links, and types of links. The links on the graph are color-coded based on link types and the nodes are clickable. Clicking a node displays more information about that node. The TimeML graph component enjoys of a mini map which becomes useful when graphs are large to get an understanding of the dimensions of the object.

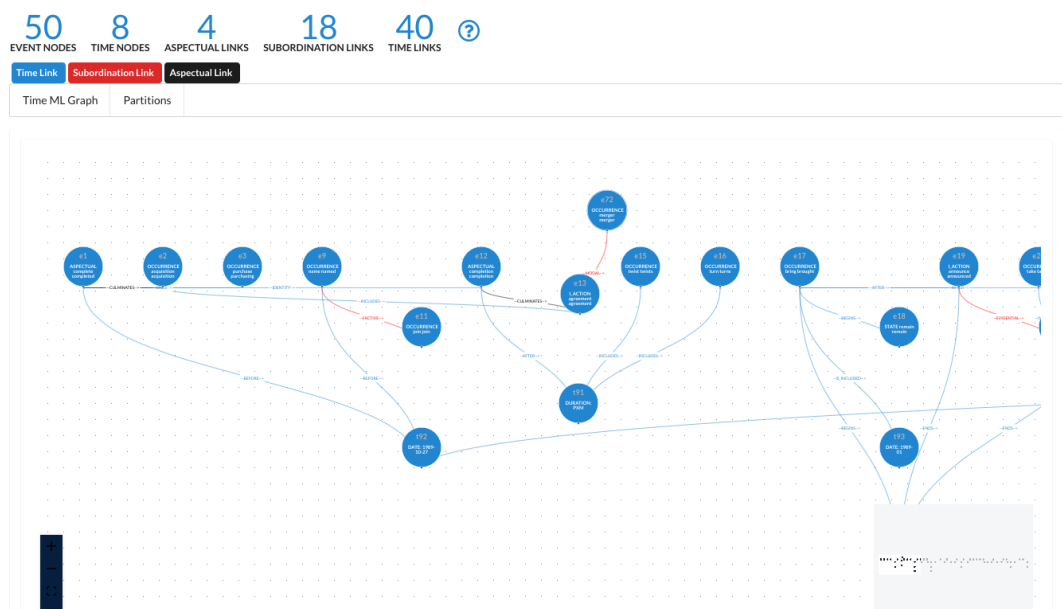


Fig 5. Rendering of a TimeML Graph

Partitions Component

TimeML graphs are partitioned as part of the pipeline of steps described in the TLEX algorithm. We designed a user interface component to render partition subgraphs with the following features:

1. A menu list of partitions is available for the user to switch among partitions
2. A label indicates if the partition currently rendered is a consistent partition or an inconsistent partition.
3. Partition subgraphs have all the same functionality of the main TimeML graph including: Color coded and labeled edges and clickable nodes.

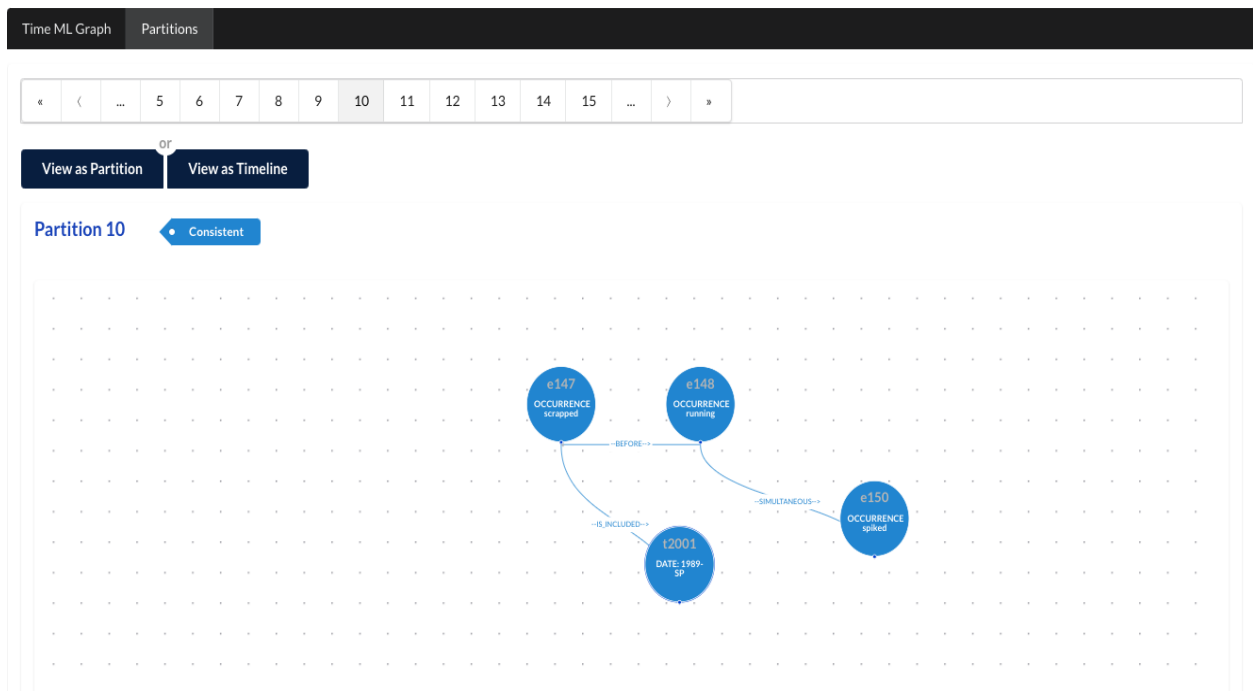


Fig 6. Example rendering of a partition subgraph

Timeline Component

One of the main uses of the TLEX algorithm is its ability to extract timelines from TimeML annotated files. We designed a timeline user interface components that allows the user to render a timeline from each consistent partition of the graph.

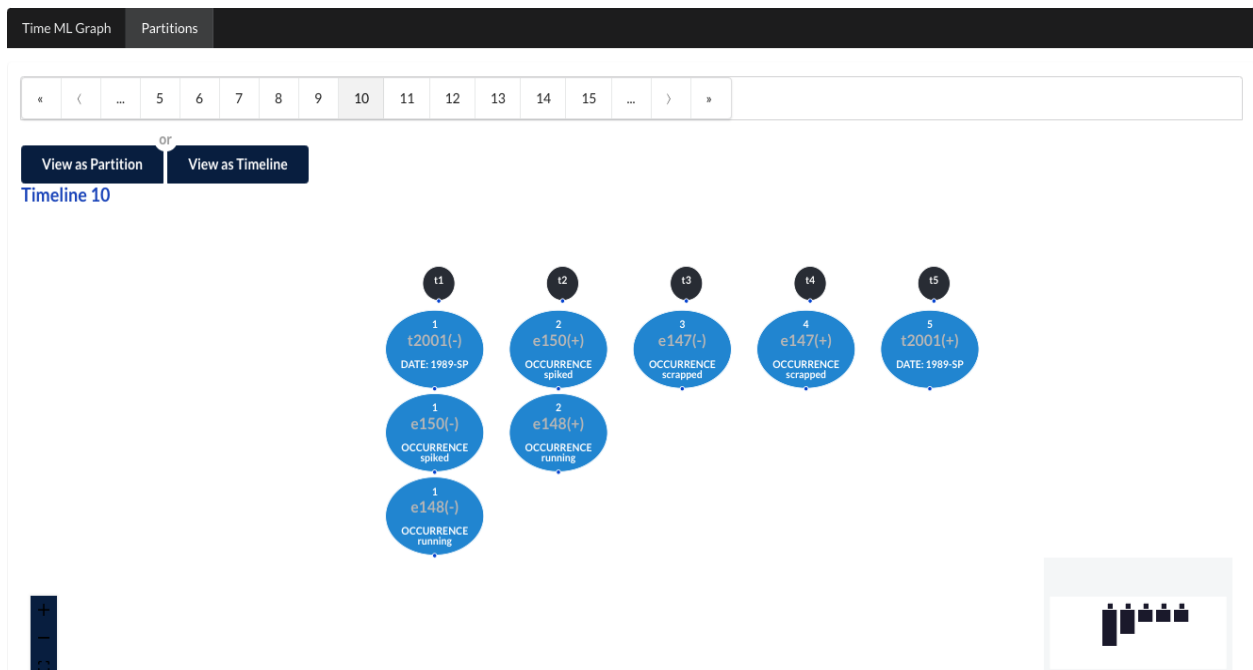


Fig 7. Example timeline

Landing Page

We designed a landing (about) page that educates the user about the inner workings of the TLEX algorithm. Noticed our choice of colors follows the colors of Florida International University.



Fig 8. Landing page displays information about the algorithm

Appendix C - Sprint Review Reports

Sprint 1 Review

Ismael Clark, Jared Hummer, Luis Robaina, Antonela Radas, Ronald Pena, Leandro Estevez, Victoria Fernandez, Pedro Diaz, Emmanuel Garcia

Date: May 29, 2021

What was completed during this sprint?

- The team was successfully able to install all the software tools needed for the project development.
- The team read all the documentation about jTLEX.
- The team has some recommendations for visualization tools to be used.

Sprint 2 Review

Ismael Clark, Jared Hummer, Luis Robaina, Antonela Radas, Ronald Pena, Leandro Estevez, Victoria Fernandez, Pedro Diaz, Emmanuel Garcia

Date: Jun 12, 2021

What was completed during this sprint:

- The team was able to compile a set of important data we want to be able to return from our backend API.
- Documentation is nearly complete. There were originally >175 problems. Now we are down to 10.

Stories that carry to the next sprint

- -

Sprint 3 Review

Ismael Clark, Jared Hummer, Luis Robaina, Antonela Radas, Ronald Pena, Leandro Estevez, Victoria Fernandez, Pedro Diaz, Emmanuel Garcia

Date: Jun 27, 2021

What was completed during this sprint:

- Implemented a first version of the TLEX frontend page.
- Endpoint for uploading time ML file .
- Endpoint for uploading time ML file as a string.
- Graph.toJSON() output

Stories that carry to the next sprint

- Documentation
- Testing
- Refactoring / fixes

•

• Sprint 4 Review

Ismael Clark, Jared Hummer, Luis Robaina, Antonela Radas, Ronald Pena, Leandro Estevez, Victoria Fernandez, Pedro Diaz, Emmanuel Garcia

Date: Jul 11, 2021

What was completed during this sprint:

- The graph visualization canvas now has clickable nodes, and a map to see the whole graph.
- Finished implementation of project in Gradle.
- There is now integration between the front end and the backend
- Added exception handling in the backend

Stories that carry to the next sprint

- Code profiling the core java library
- Finish integration with back end
- Add more graphing functionality

Sprint 5 Review

Ismael Clark, Jared Hummer, Luis Robaina, Antonela Radas, Ronald Pena, Leandro Estevez, Victoria Fernandez, Pedro Diaz, Emmanuel Garcia

Date: Jul 22, 2021

What was completed during this sprint:

- Errors in InconsistencyDetector.java are resolved.
- Timeline lengths now match expected output.
- Team demo product to the Product Owner
- Final styling modifications were made to the frontend.

Stories that carry to the next sprint

- General unit tests
- indeterminacyDetector.java ambiguity

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document, and other documents

The jTLEX visualization application contains a set of documents to facilitate the installation and use of the system. The following documents can be found alongside the submission of this documentation:

1. User Manual: **User Manual.pdf**
2. Backend Installation Guide: **Installation Guide jTLEX Frontend Application.pdf**
3. Frontend Installation Guide: **Installation Guide jTLEX Frontend.pdf**

REFERENCES

- [1]. *TLEX: A Formally Correct Method for Extracting Exact Timelines from TimeML Temporal Graphs*, by Mustafa Ocal and Mark A. Finlayson
- [2]. *jTLEX - A Java Library for Timeline Extraction* by Emmanuel Garcia, Luis Robaina et al.