

Problem

Qualitative temporal graphs that are derived from annotations on text (TimeML annotations) explicitly reveal only the partial orderings of events and times. For many purposes, however, a total ordering (a timeline) is more useful.

Unfortunately, timelines are rarely explicit in text, and usually cannot be extracted directly. Information from time annotated files can be used to construct a temporal graph by using a temporal representation language such as a temporal algebra or TimeML.

The TLEX algorithm specifies an exact, end-to-end solution, to the task of extracting multiple timelines from TimeML annotated texts. TLEX transforms TimeML annotations into a collection of main and subordinated timelines arranged into a trunk-and-branch style structure.

jTLEX puts the end-to-end solution proposed by the TLEX algorithm into an easy-to-use Java library that can be consumed as an API for many practical applications of timeline extraction from annotated TimeML files. For example, question answering often requires understanding the overall temporal order of events in a text. **jTLEX** can be used to accomplish this task.

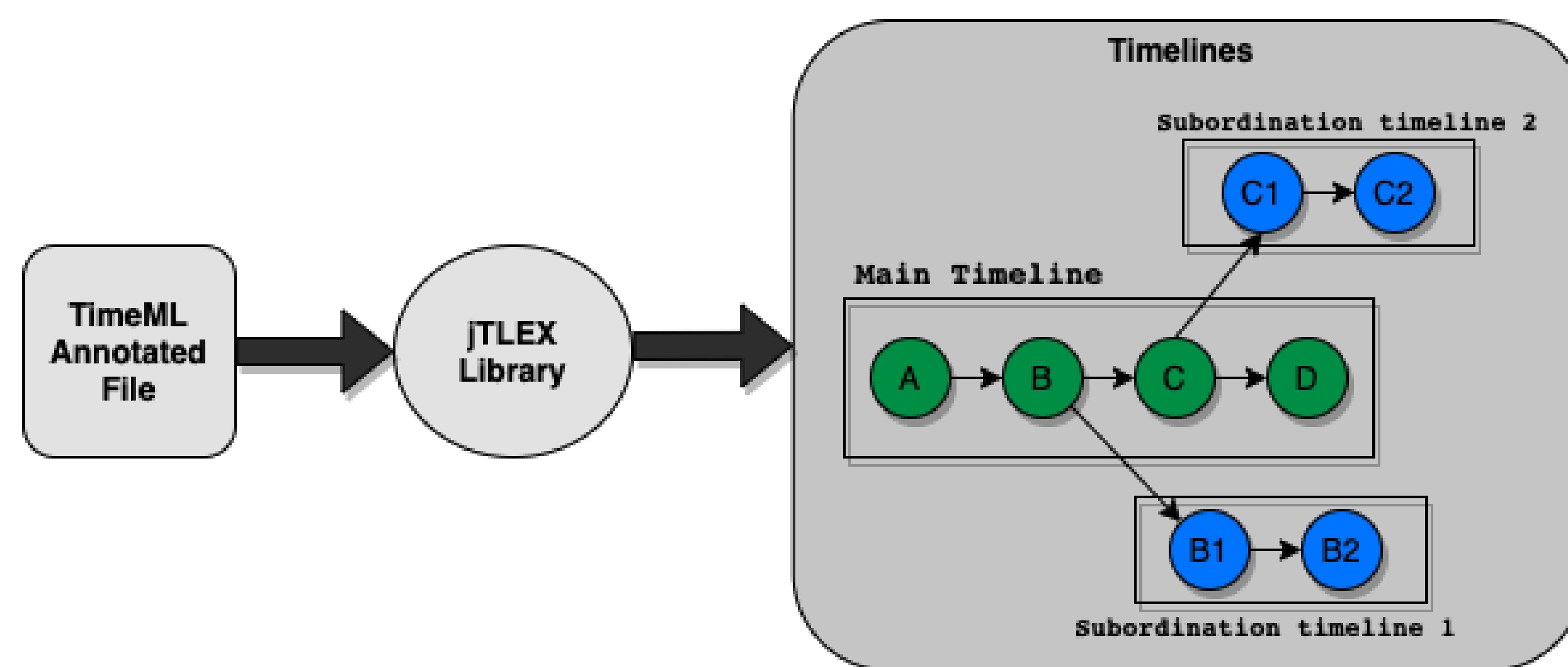
Current System

jTLEX puts the TLEX algorithm into an easy-to-use Java library. The current system takes a TimeML annotated file and parses it into a graph data structure (TimeML Graph) that allows jTLEX to represent the temporal information as a graph where nodes are events or times, and edges are temporal links that express relationships between these nodes. The TimeML graph is the basic unit of processing used by jTLEX to generate main and subordination timelines. To accomplish the translation from a TimeML graph into a set of structures with total ordering (timelines), jTLEX first partitions the graph into temporally connected subgraphs including subordinating subgraphs which represent possible, counterfactual or conditional situations. Each of these subgraphs is then transformed into a temporal constraint satisfaction problem (TCSP) which is solved by the Java Constraint Programming Solver (JaCoP). The solutions to the TCSP provide us with a total ordering for the events described in each subgraph which is then presented to the client user as a set of timelines ready to be analyzed or used in any application.

Summary

jTLEX handles all TimeML relations and achieves perfect accuracy modulo the correctness of an underlying TimeML graph. jTLEX checks the TimeML graph for consistency, but goes further by automatically identifying inconsistent subgraphs, which allows them to be manually corrected. jTLEX outputs one timeline for each temporally connected subgraph, including subordinated timelines which represent possible (modal), counterfactual, or conditional situations. These subordinated timelines are connected to the main timeline in a trunk-and-branch structure. We also present a method for detecting which sections of the timelines have indeterminate ordering.

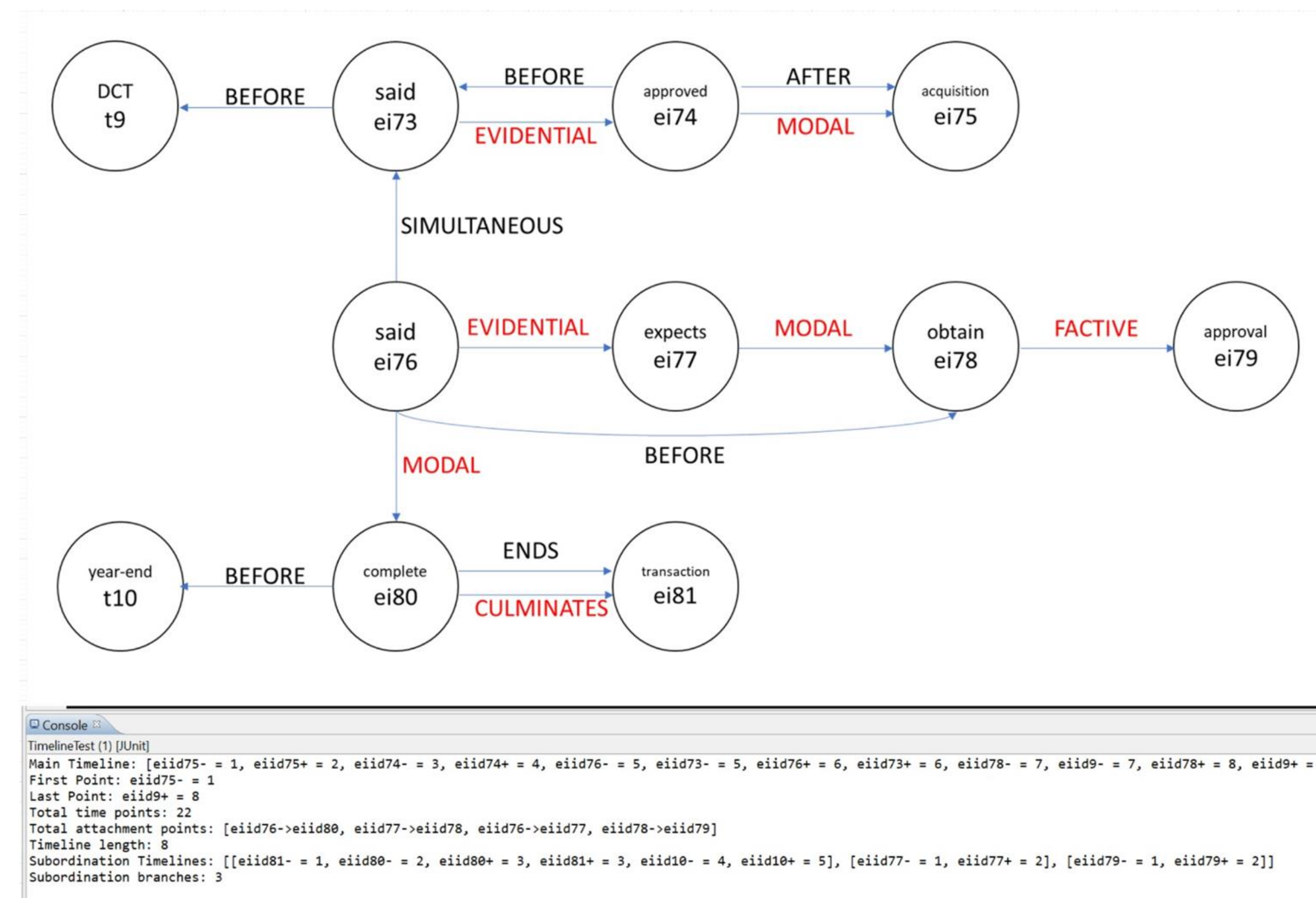
System Design



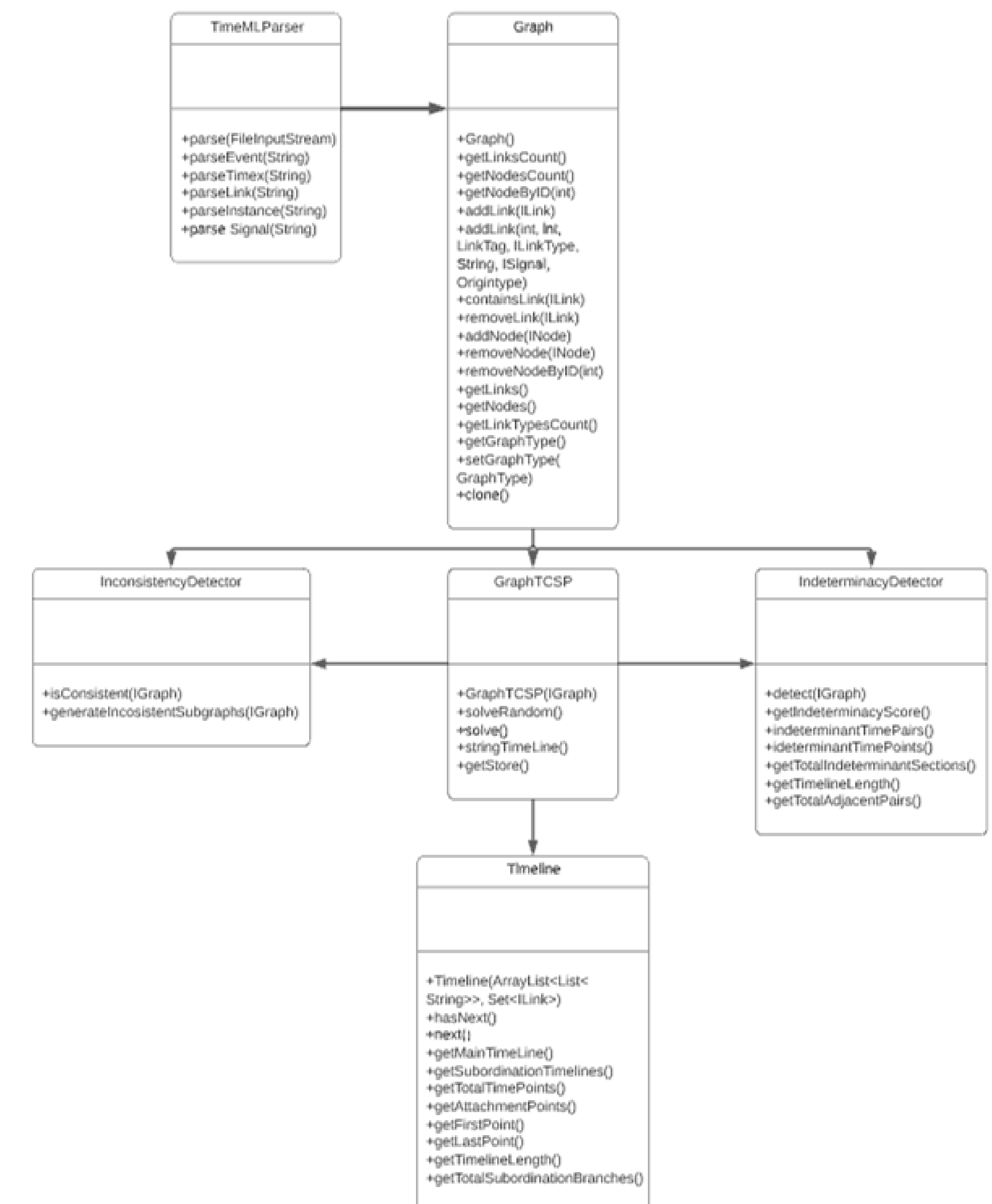
Verification

Test Case ID	JTLEX-SYSTEM-TEST
Purpose	To validate the running time of the TLEX algorithm on a corpus of TimeML annotated files.
Preconditions	A preexisting corpus of TimeML annotated files exists and its accessible for the jTLEX library
Inputs	A corpus of TimeML annotated files.
Expected Result	jTLEX should process the corpus of TimeML files and produce timelines in less than 10 seconds.
Result	PASS
Test Case ID	JTLEX-SUBORDINATION-TEST
Purpose	To validate that jTLEX is able to find subordination timelines from a TimeML graph.
Preconditions	A TimeML graph generated by parsing an TimeML annotated file with known subordination links exists.
Inputs	A TimeML graph instance with subordinations links passed into the partitionGraph method in IGraphPartitioner.java
Expected Result	jTLEX should process the TimeML graph and output a set of disjoint TimeML graphs corresponding to the main and subordinations timelines.
Result	PASS

Screenshots



Object Design



Implementation



The **jTLEX** library was implemented using Java code developed within the Eclipse Integrated Development Environment and tested using the JUnit testing framework. To effectively support team collaboration, we managed versions of **jTLEX** with subversion VCS.

Acknowledgement

We thank Mustafa Ocal for assistance, cooperation and mentorship that we received throughout this process.