Conference Paper Title*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Design

A. AI Criteria

The overall process that was found when performing an epic breakdown is a combination of clustering related requirements, specifications, and details along with the further breakdown of complex info into smaller stories and tasks. Clustering can be performed in either supervised or unsupervised learning. Supervised learning requires the existence of a data set that has been correctly labeled. Although a general data set of decomposed epics can be generated, it was deemed that the range of topics an epic can encompass was too large to have a trained model that would accurately cluster the content of an epic.

Additionally, an early assumption made about the decomposition process is that the text processing techniques should not rely upon the existence of historical project information, whether that be in the form of previously completed epics or sprints. As a result, unsupervised clustering techniques were only considered as they allowed for the decomposition process to be applicable to new projects, which lack history, and projects that have not been topically available to train a clustering model off of.

The last criteria for the text processing approaches under consideration was that no additional information should be generated through AI. This was due to the previously discussed lack of data, specific to

Identify applicable funding agency here. If none, delete this.

the project domain, that could be used to potentially generate stories and tasks.

B. Epic Decomposition

C. Story Optimization

D. Task Generation

Task generation was identified as the process of breaking down a requirement into its most simplest form. Based on the criteria laid out in I-A, simplification on the existing input of requirements could be completed by deconstructing complex sentences into simple sentences.

The first process to break down sentences was to use parts-of-speech tagging to remove any unnecessary words from the sentence; this was done by identifying stopwords, since simple sentences contain only one subject and predicate. Therefore, using the word dependency we were able to determine which words were connected to the subject. By making sure the sentence contained one subject and verb we were able to create a complete simple sentence. Everytime a new subject was found that meant a new sentence was needed. Each simple sentence generated from the stories can then be suggested as tasks.

E. Flow

F. Relationship visualization

II. Implementation

A. Frontend

As part of the Atlassian Design Guidelines, a frontend library, Atlassian User Interface (AUI), is provided to create new UI elements to match Jira's style and user experience. Needed components that were missing from the AUI were crafted by the provided Design Guidelines from Atlassian.

Each of the processes use the same suggestion module, with variations depending on the need for user inputs in the case of epic decomposition and story optimization.

B. Backend

The backend served two primary functionalities: generating suggested issues and populating selected suggestions into Jira. Each process has its corresponding suggestion generation and suggested issue populator.

1) Epic Decomposition: The Epic Decomposition process was implemented using k-means clustering over a dynamic clustering technique, such as DBSCAN or Mean-shift. Dynamic clustering techniques were unable to be implemented in a manner that yielded consistent meaningful clusters across many epic inputs. This was a result of an inability to filter noise such that the clusters were neither completely disjoint or entirely overlapping. Thus, k-means, a centroid-based clustering algorithm, was used at the cost of selecting a default number of stories to generate. The default number of stories generated is five but the user can select between 2 and 10 stories using a newly added slider in the UI for the Epic Decomposition process.

2) Story Optimization: A new update to the story optimization is the slider integration. The user can choose to increase or decrease the degree of connectivity between sentences. The slider displays options from 0 to 10, which maps to values from 0.65 to 0.75 as threshold values on the backend. The slider integration was a feature that was added based on mentor feedback. A parameter is exposed to give the user more control of the output. The exposed parameter is the degree of connectivity between the sentences. It was determined after impact analysis that it is better to implement it this way because it generates more meaningful results tailored to each user. This also eliminates the need for having a fixed threshold number, since the optimal threshold number can vary between inputs.

3) Task Generation: Initially the natural language toolkit (NLTK) library was selected for implementing task generation, but it lacked the need for parts-of-speech tagging. The main feature that NLTK lacked was word dependency. Word dependency is important to this process, since this was the main feature used to create simple sentences. Word dependency returns a pair of words, identifying the dependency between the two. The new python library being used is Stanza. This library allows for parts-of-speech tagging, word dependency, and tree parsing. By having part-of-speech tagging and word dependency, the creation of simple sentences from complex and compound sentences was easier to implement.

Component	Options	Choice	Reasoning
Epic Decomposition	• KMeans • Meanshift • DBSCAN • OPTICS	Mean-shift	All of the mentioned clustering algorithms are dynamic as they do not require the number of clusters as input. Mean-shift only requires the size of the region to search through (bandwidth), which can be estimated, where all other options require arbitrary values to create clusters.
Story Optimization	• TF-IDF • SIF • Word2vec • Cosine simularity • Word mover's distance	SIF with word2vec and Cosine similarity	TF-IDF is the most simple for word embedding. SIF with word2vec can achieve a higher accuracy than TF-IDF with the custom word2vec model that is relevant to the topic. WMD is far more expensive to calculate, especially on longer texts. Cosine similarity can give the same performance without losing too much semantic similarity.
Task Generation	• Sentence classification	Sentence classification	There is a need to split up the different types of sentences, such as complex and compound, into simple sentences, in order for them to be manipulated into tasks. To do this, sentence classification is the necessary first step.

TABLE I
AI DESIGN CHOICES

C. Communication

The plugin communicates from the Jira interface to the backend processes through AJAX calls which are received by Flask. When a web panel is opened for any of the three processes, a message is sent to generate and return the suggestion issues. The Jira panel will function asynchronously with a loading animation appearing until the resulting suggestions are received. Once the results are received, populated in the suggestion box, and the desired suggestions are selected by the user, then a message is sent containing which suggestions to then create within Jira.

III. Verification and Validation

A. Demonstration

1) Complete Decomposition Scenario: Frank is a software requirements analyst, and he wants to speed up the process of breaking an epic full of requirements into smaller user stories. For this purpose, he installs the AI4Agile plugin to Jira. Frank creates a new epic, puts the requirements for his Spreadsheet Application in as plaintext sentences, and clicks the Decompose Epic button.

Now, Frank looks at the suggestions the AI came up with. If he decides he doesn't like any of these suggestions, he can click the Ignore All button to go back to his Spreadsheet Application epic. If Frank wants more or fewer stories, he can adjust the value on the slider and it will refresh the results. To edit a story, he can click on its text box, then save or cancel those edits. To ignore a story suggestion entirely, he can leave its box unchecked. Once Frank is happy with his resulting user story or the set of stories he wants, he clicks Create Selected.

After refreshing the webpage, the newly created user stories that Frank approved are visible under the heading of the Spreadsheet Application epic.

If Frank wants to continue the process of breaking up epics, he can go into one of the user stories and click Optimize User Story. Depending on the size of the user story already, it might be broken down into multiple smaller stories, or left alone if it's already optimized.

Once the Story Optimization Suggestions are generated, Frank has the same options as with the previous stories: to select or deselect stories via the checkboxes, make edits, or ignore all suggestions. In addition to those options, Frank can choose to adjust the connectivity slider to change how closely related items need to be in order to stay with the same story.

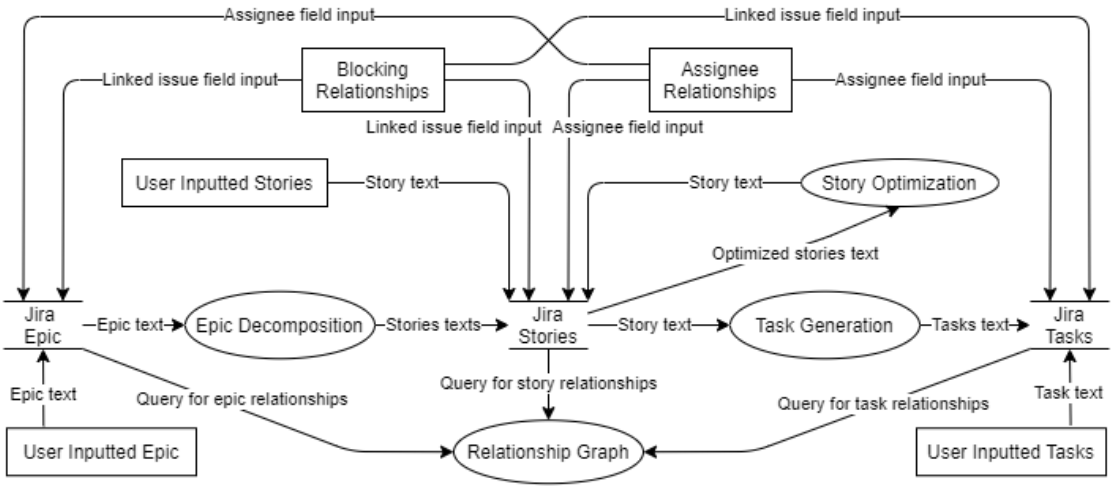

Fig. 1. Data Flow diagram showing the various entry points, sources of data, and processes within Jira and AI4Agile

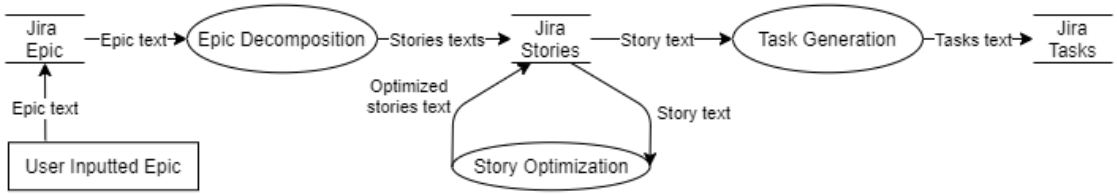

Fig. 2. Simplified Data Flow diagram outlining the flow of the decomposition processes

Now there are five user stories, since Story 4 was optimized into two separate stories. To continue the decomposition process, Frank opens a story and clicks Generate Tasks.

As before, the options include selecting or deselecting individual suggestions, editing, and ignoring all suggestions. Once he's happy with the tasks, he clicks Create Selected.

The tasks have now been created, and linked to their parent story to indicate a blocking relationship. All Frank had to do was make some decisions and maybe edits, and now he's got one story fully decomposed.

2) Relationship Visualization Scenario: Madison is a scrum master, and she wants to take a closer look at some of the tasks to make sure the developer assignments make sense. To do this, she uses the AI4Agile Issue Relationship Graph feature by navigating to a certain epic, user story, or task, and finding the graph portion. Now, Madison can see all the available relationships between the task she has selected and other tasks, stories, and the epic they came from. She is better able to decide whether the team members she assigned to these pieces make sense given the relations between them.

B.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g".

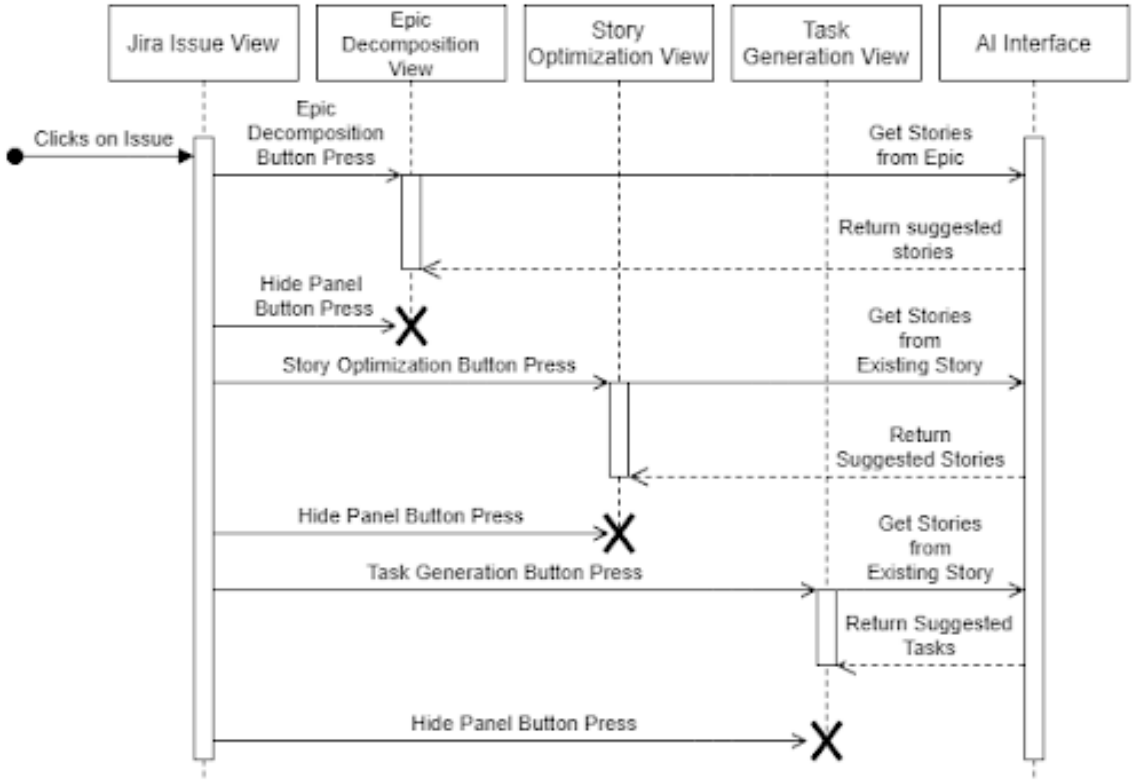

Fig. 3. Suggestion web panel sequence diagram representing the flow of messages between the frontend Jira web page and AI4Agile backend

Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.