Requirement Engineering
Implementation of Process Mining Visualizations for Conformance Checking

Group 2

November 2024

Contents

1 Requirements Elicitation and Development

The first step in this phase of requirements engineering is to identify the stakeholders who will bring different perspectives on the system. Based on their viewpoints, we will collect the initial requirements. Any gaps in these preliminary requirements will be addressed through further analysis. The requirements will then be organized hierarchically, allowing specialized team members to focus on fulfilling specific requirements. Finally, we will present a functional model to visualize the processes. We will then provide an overview of various requirement management tools.

1.1 Stakeholders

The following stakeholders have been identified and will be consulted when gathering requirements:

- Project Team
- Customer
- User

1.2 Raw Requirements

1.2.1 Customer Requirements

1. The Software should implement static visualizations, which should provide insights into the conformance issues of a process.
2. Any visualization implemented should be saved in an image.
3. Visualization could be implemented in a variety of ways, such as:
    - Dotted chart and performance spectrum
    - Temporal Behavior Patterns (TBP) Chart
    - Conformance Heatmap with Concept Drift Over Time (CHCD Chart)
    - Activity Interaction Network (AIN)
    - Semantic Conformance Word Clouds
    - Sequence Embedding Conformance (SEC) Visualization
4. The software uses the PM4Py library[1].
5. The software works correctly and without errors
6. The library should have good documentation and should be easy to use

1.2.2 User Requirements

1. The visualizations must be intuitive while ensuring accuracy in representing the data.
2. The quality of the output should meet high standards, ensuring clear and precise results.
3. The techniques employed must be widely recognized, allowing users with prior knowledge to easily comprehend them or quickly learn them.
4. The software must be discrete, ensuring privacy and data integrity.
5. Clear and comprehensive documentation should be provided to guide users in utilizing the software effectively.

1.2.3 Developer Requirements

1. The software should be developed in Python, extending the PM4Py library to ensure compatibility with established process mining libraries.
2. The codebase should adhere to modular and encapsulated programming principles to facilitate future maintenance, scalability, and the addition of new visualizations.
3. The software should follow PEP 8 for better readability and consistency across the codebase.
4. Documentation should be maintained throughout the development process, including comments within code, a README file, and function-level documentation to support both current and future developers.
5. Developers should implement automated unit tests for key functions (e.g., data import, visualization rendering) to ensure reliability and prevent regressions during updates.
6. Error handling should be implemented to manage common issues, such as unsupported data formats or missing data fields, providing users with clear error messages.
7. Developers should perform regular code reviews and refactoring to maintain code quality and optimize performance, especially in data-heavy processes.

1.2.4 System Requirements

1. The software should be compatible with commonly used operating systems (Windows, macOS, and Linux) to ensure accessibility across diverse environments.
2. The system should allow easy installation and setup, with automated dependency management to simplify the installation of required libraries or packages.
3. The software should be capable of handling event log data in formats like XES and CSV, supporting files up to 1 GB to accommodate standard industry process logs.
4. The software should generate high-quality visual outputs suitable for presentations and documentation.

1.3 Requirement Analysis

After the proposition of the raw requirements, analysis showed they were clear, complete, unduplicated, concise, valid, consistent and unambiguous. The viewpoint of the BlaBla Soultions GmbH (especially Alessandro Berti) was intensively considered once more. As a result the requirements were broken down into 3 ordered ranks namely high, medium and low so that implementation workload can be divided adequately to team members throughout the development process, for designing and testing.

1.4 Allocation and Flow-Down of Requirements

The result of the requirement analysis in hierarchical structure will be provided in this section. Hierarchically classified requirements will define the interaction between applications and subsystems to fulfill the requirements of the software [2]

1.4.1 High Level

H1 The software shall offer static visualizations for conformance checking, enabling users to visually analyze process compliance and detect deviations.

H2 The software shall generate visualization outputs

H3 The software shall be compatible with Windows, macOS, and Linux, allowing broad accessibility across different operating systems.

H4 The software's interface shall be responsive

H5 The software shall prioritize ease of use, offering an intuitive interface and detailed documentation to lower the learning curve for new users.

H6 The software shall support import of event log data in standard formats like XES and CSV, ensuring compatibility with existing process mining tools.

H7 The software shall implement multiple visualization types (e.g., TBP, CHCD, AIN) to provide comprehensive insights into process behaviors and conformance.

H8 The software shall ensure data security, protecting sensitive process information from unauthorized access and adhering to data privacy standards.

1.4.2 Intermediate Level

I1 The software should include export functionality, allowing users to save visualization data in various formats (e.g., PNG,JPEG, CSV) to enable further analysis or reporting. (H1,H2,H7)

I2 The software should incorporate modular design principles, making it easy to extend or modify each visualization without impacting other components. (H4,H5,H7)

I3 Error handling should be robust, providing meaningful error messages for common issues (e.g., data format errors, missing values) and logging errors for troubleshooting. (H4,H5)

I4 The software should include automated tests for key functions, such as data loading, visualization rendering, and export, to verify accuracy and prevent regression. (H8)

1.4.3 Low Level

L1 This software should be able to load event log data. (I1)

L2 This software should be able to pre-process data for each visualization.(I2)

L3 (TBP) The library should have a function to extract temporal features.(I2)

L4 (TBP) should be able to aggregate behavior counts over time and create a scatter plot. (I2)

L5 (TBP) adjust dot color or size dynamically based on frequency.(I2)

L6 (CHCD) aggregate conformance scores over time intervals.(I2)

L7 (CHCD) pivot table to organize data for the heatmap.(I2)

L8 (CHCD) maps conformance score intensity to color, with higher conformance issues represented by warmer colors.(I2)

L9 (AIN) Generate pairs of activities from traces to capture transitions.(I2)

L10 (AIN) Aggregate frequencies and conformance scores for each transition.(I2)

L11 (AIN) create a directed graph, with edge thickness and color representing frequency and conformance scores.(I2)

L12 (SCWC)preprocesses text data with NLP tools to standardize and clean labels.(I2)

L13 (SCWC)calculate the frequencies.(I2)

L14 (SCWC)ht non-conforming words.(I2)

L15 (SEC) Convert each trace to a sequence of embeddings.(I2)

L16 (SEC) Generate trace embeddings.(I2)

L17 (SEC) reduce embeddings to 2D.(I2)

L18 (SEC) map conformance scores to color or size to each plotted trace.(I2)

L19 Implement function to save results as png meaningfully. (I1)

L20 Format error handling function. (I3)

L21 Checks for errors in outputs to ensure accuracy corrected it.(I3)

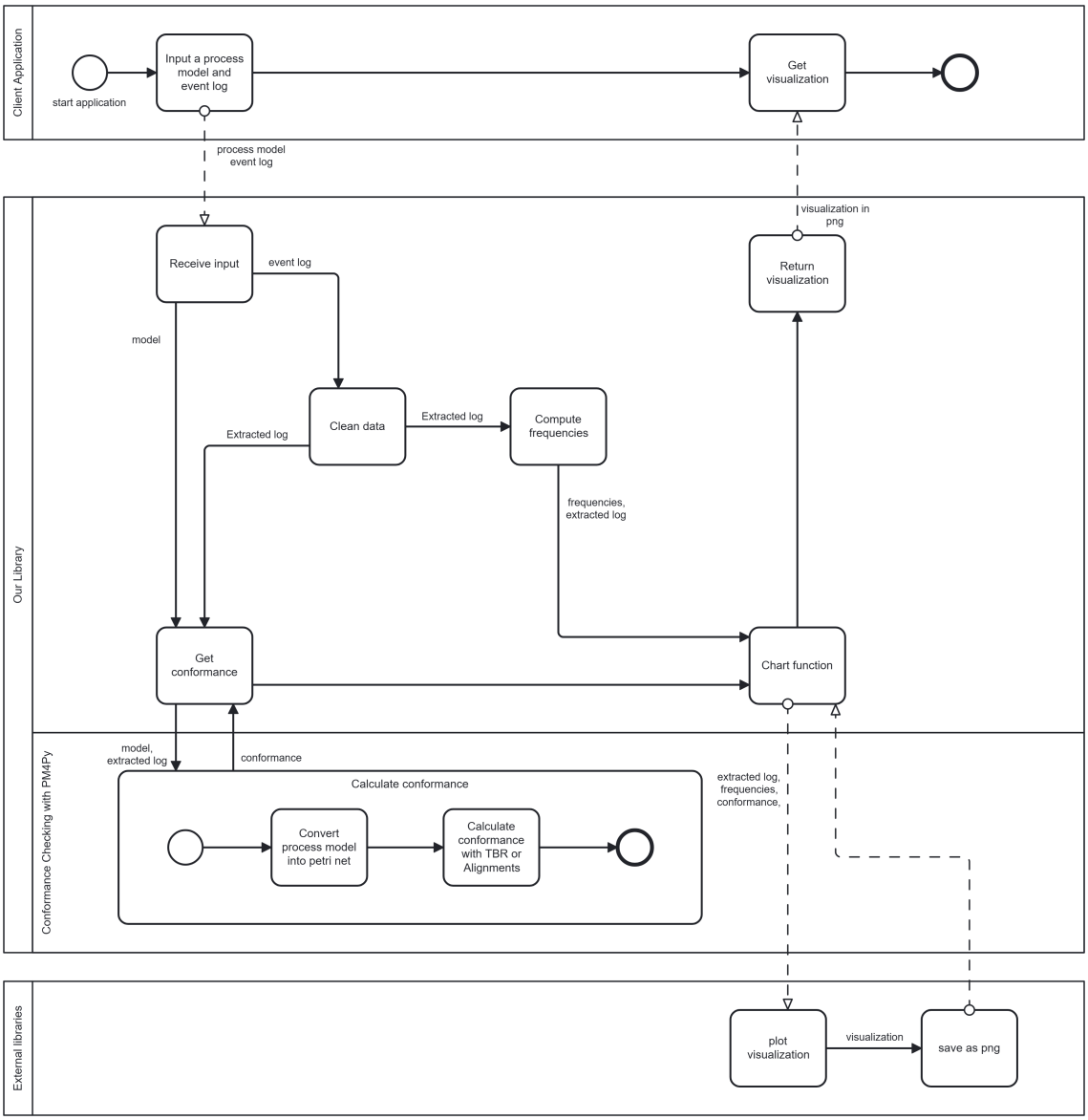
L22 Implement unit tests for L20,19,1,2.(I4)

2 Functional Model

Our functional model is designed to visualize processes involved in producing the required products. This diagram provides valuable insights to all parties engaged in the project, facilitating a clear understanding of the workflow and interactions.
The model includes three primary actors:

- **Client Application**: This is a user-friendly interface that allows clients to interact with the visualizations easily. It also serves as an extension of the PM4Py library, providing a tool that other developers can integrate into their own projects for enhanced process mining capabilities.
- **Our Software**: This is a set of functions that generates a variety of visualizations based on a given event log. It enables users to examine and analyze process behaviors, conformance issues, and other aspects of the data, aiding in effective process evaluation and improvement.
- **External Libraries**: We utilize external libraries for data preprocessing, graphic plotting, and model training for certain advanced visualizations. These libraries allow us to streamline complex tasks such as data handling, machine learning, and visual representation, making the software both efficient and scalable.

2.1 The Model



3 Validation and Verification of Requirements

To improve the quality of the project requirements, it was essential that all parties involved reach a consensus on the accuracy and clarity of each requirement. Our team met with BlaBla Solutions GmbH (contact person: Alessandro Berti) to gather feedback and ensure that the requirements align with user needs. Based on the feedback provided, we made the following refinements:

- **H4**: Originally stated that the software should be "responsive"; this was clarified to ensure responsiveness across various screen sizes and devices for optimal accessibility.
- **H2**: Initially described general visualization needs but was updated to specify that execution results must be visualized in a static format, such as PNG, for consistent output across all platforms.

- **L5 and L8**: These requirements were initially stated broadly as "adjust dot characteristics" and "map conformance scores to color," respectively. They were refined to specify that dot color, size, and heatmap intensity should dynamically represent frequency and conformance scores, enabling better visual differentiation of data

4 Requirement Management Tools

4.1 Jira

Jira is a versatile project management tool by Atlassian, well-suited for agile teams. It provides flexible tools like Scrum and Kanban boards for planning and tracking tasks, making it ideal for workflows. Jira integrates seamlessly with other tools such as trello, allowing for a unified development process. Additionally, its robust reporting features help teams monitor progress and make data-informed adjustments, ensuring efficient project management and collaboration.

4.2 Wrike

Wrike is a flexible project management tool that supports customization and enhances team collaboration. It allows teams to tailor workflows, views, and workspaces to their specific needs, providing a centralized platform for clear communication and project visibility. Wrike's automation features, such as automatic approvals and templates, reduce manual tasks and boost productivity, helping teams stay focused on key priorities.

4.3 Jama Software

Jama Software is an enterprise-grade platform designed to manage requirements, risk, and tests. It provides tools for capturing and communicating requirements, tracking progress, and analyzing impacts. Teams can collaborate in real-time, review requirements, and ensure traceability across the project lifecycle. Its web-based interface allows users to access the tool from anywhere, making it suitable for distributed team

4.4 Helix RM (formerly TestTrack)

Helix RM is a requirements management tool designed to support traceability, change control, and collaboration in complex projects. It offers a centralized platform to capture, manage, and track requirements across the development lifecycle. It integrates seamlessly with other tools like Jira, Git, and Jenkins, making it a strong choice for teams using DevOps practices.

5 Phase Review

5.1 Lingjing Zhou

This phase presented new challenges compared to the previous one, but our team remained motivated and focused, collaborating closely to meet our goals. I gained a clearer understanding of the project requirements and the direction we're heading in. Our team's communication has been excellent, with everyone contributing and supporting one another, which made this phase feel productive and rewarding. I'm looking forward to the upcoming coding phase, where we'll start bringing our ideas to life.

5.2 Hanna Yashchuk

In this milestone, we encountered several challenges, including the busy schedules of team members, a heavy workload, and some uncertainty about our tasks. However, I truly appreciate how efficiently and effectively our team addressed each issue. Communication within the team was excellent; even without in-person meetings, we successfully discussed everything and established a solid workflow plan. I'd also like to highlight the team's commitment to maintaining a healthy work-life balance. Everyone stayed dedicated without needing to overwork. Overall, this milestone was a success.

5.3 Maulik Jain

This phase was more challenging than the last, but our team was incredibly focused, working around the clock to make this milestone achievable. Everyone approached their tasks with a positive attitude, and we successfully completed everything on time. This phase also gave us a much deeper understanding of what lies ahead, which makes me even more excited as we step into the coding phase.

5.4 Uy Sa Huynh

This phase was a little more challenging than the first phase, but it provided me with essential insights of how our software should work and what is needed. I am very happy with our teams' communication, we get along really well and everyone helps each other. I am satisfied with how our work turned out and I am exited of the upcoming weeks where the actual coding of our project happens.

5.5 M. Ali Agharazi

This phase was more challenging compared to the previous one especially within the short time frame. However the team effectively delegated the task and the workload was by no means overburdening for anyone. We were able to understand our tasks and our requirements. Our team's communication has been excellent and our usage of different platforms has made it possible to raise issues to scrum masters and resolve them immediately.The team was dedicated and everyone did their best.

References

[1] Alessandro Berti, Sebastiaan van Zelst, and Wil Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science, 05 2019.

[2] Dhirendra Pandey, Ugrasen Suman, and A.K. Ramani. An effective requirement engineering process model for software development and requirements management. pages 287 – 291, 11 2010.