Test case generation and prioritization: a process-mining approach

Andrea Janes

unibz
Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

The situation in Italy/Europe

- Small and Medium Enterprises (SMEs) are the dominating form of organizations:
  - 99% of all European businesses are SMEs, and of all SMEs,
  - 9 out of 10 are so-called microenterprises (MEs), i.e., companies with less than 10 employees
- SMEs often do not have the resources to invest into software quality, e.g., writing test cases.
- Particularly for start-ups or small enterprises such costs become prohibitive, which often prefer to invest their time into the development of new functionalities instead of testing.
- Our work wants to investigate how we can reduce such costs on two levels: generating test cases and prioritizing them.

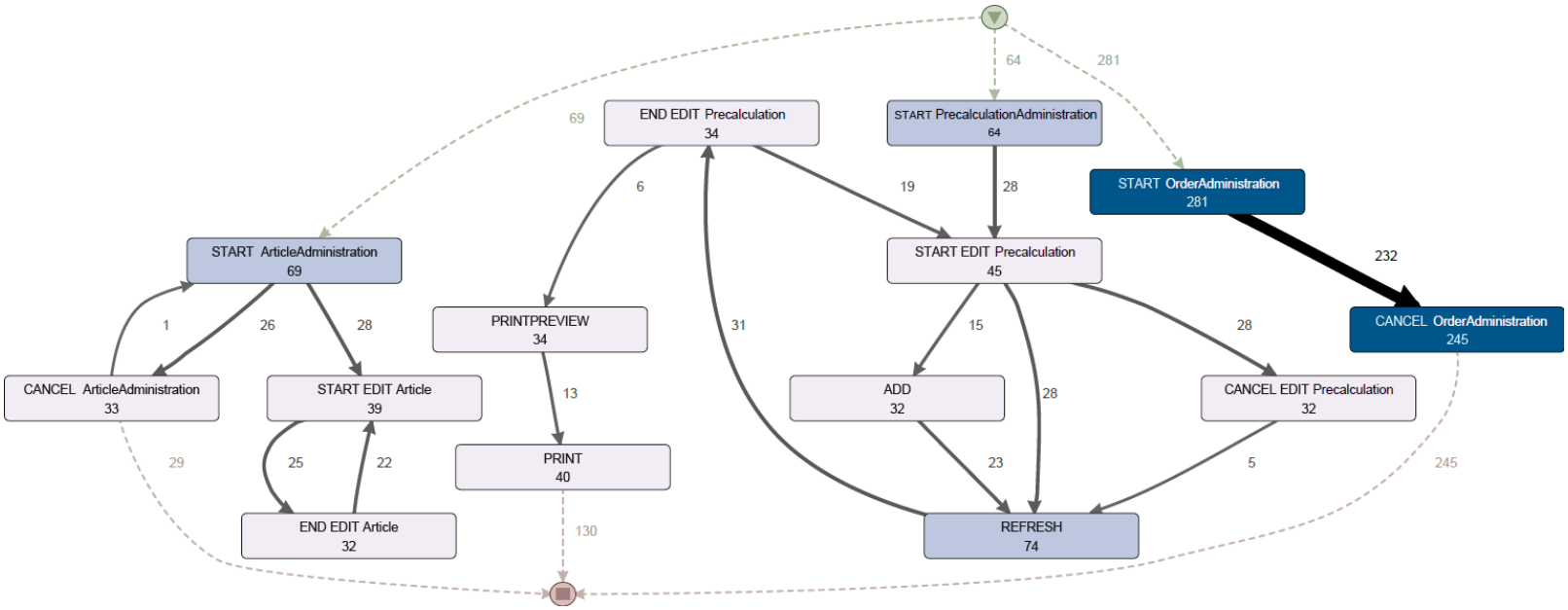
Test case generation

- One approach to generate test cases based on a mined process model is to use the identified user workflows, i.e., typical ways how the user interacts with software, which actions he or she performs, in which sequence actions are performed, etc. to guide the generation of unit test cases.
- This requires that the collected user interactions are collected at a granularity that allows the re-execution of every user action. Such test cases represent system test cases linked to UI features of the software under test. Technologies that allow the execution of such test cases are e.g., image-recognition-based GUI testing frameworks.

Test case generation – an approach

- We instrument an ERP software developed by a company in Italy in such a way that a log of the used (any type of interaction) UI controls is generated
- User interacts with software
- We obtain a log of all the activities at the user interface level
- Knowing the sequence of controls and the actions performed on those controls, we are able to reproduce the user interface interactions

A process-mining model of a collected log



Test case generation – an approach

- The obtained log of UI actions can be used to replay user behavior on an application of the same version where it was captured.
  - Either all behavior in a time frame
  - Traversals from source to sink
- In both cases the state of the application (in our case the DB) has to be restored at the beginning of the test execution
- In our first implementation, we re-execute the traced activities within a time frame.
  - Activities contain a number of user sessions
  - User sessions contain a number of transactions on customers
- If the UI changes, we define a sequence of actions that have to be executed whenever a specific UI element is encountered.

Test case generation – an approach

- Another approach is to use the mined process model to guide test case generators, e.g., Random Testing.
- Random testing uses random input values to evaluate if the tested software is able to perform without unexpected reactions.
- Such generators often face the problem that – since based on random input – obtain a low testing coverage because they are not able to invoke complex/deep functionalities.
- The obtained process model could be used to guide the generation of test values.

Test case prioritization

- When executing all test cases takes too long (Extreme Programming recommends 10 minutes as a maximum, otherwise programmers tend to execute them less often), test case prioritization becomes necessary, i.e., to select a reduced number of test cases, which will be executed regularly and which are the most likely to find critical defects in the software.
- Test cases can be prioritized according to various criteria, we propose to use the value of the feature they test.
- The perceived value is closely related to usage and the consequences of usage. In the case of software, this would mean that parts of the software that are used more often are indicators of value.

Test case prioritization – a approach

- We use the log obtained in the test case generation part
- We use the usage frequency as an indicator of value and as a criteria to prioritize test cases.
- Test cases that test code connected to the frequently used UI elements are given a higher priority than other tests
  - The connection between test cases and the code connected to a UI element can be defined
    - Manually
    - Obtain the coverage of each test case and prioritize test cases that cover the code executed in the context of the UI element
      - In .NET such code can be identified since its methods are prefixed with the ID of the UI element. Another possibility is to parse the source code and identify which code was added as event handler to which UI element.

Conclusion: About the industrial collaboration

- How did you get in contact with the industrial/academic partner?
  - A former thesis student of mine is working in this company
- How did you collaborate with the industrial/academic partner?
  - The student implemented the logging feature and obtained all the necessary permissions by his superior/collaborators. We analysed the data and are still collaborating to implement the developed prototypes. The student is responsible for the development in the company.
- How long have you collaborated with the industrial/academic partner?
  - Half a year for the development of the first prototype, still ongoing.
- What challenges/success factors did you experience when collaborating with the industrial/academic partner?
  - Success factor is to have an “allied” in the company. Otherwise it can be that the company does not perceives it as worth to collect data to improve something.