# **Report on Evolution and Maintenance of Specmate Tool**

**Anusha Mogili (P No: 970626-8167)**

**Description of your contribution:**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Tasks** | **Description** |
| 1 | Sprint 1 | Building workplace, analysing Working of specmate tool, Learning angular & typescript, Specmate Web folder analysis.  Contribution-25% |
| 2 | Sprint 2 | Identifying and Analysing code smells. Implementing the testing using mocha and chai plugin. Finally, Analysing the impact caused by refactored code on maintainability. Contribution-25% |
| 3 | Sprint 3 & sprint 4 | Analysing new features to be implemented and Testing the implemented features to know its behaviour & compatibility with other features. Evaluating the maintainability with the help of five attributes .  Contribution-25% |

**SPRINT 1:**

**1.1 Reflections on comprehensibility**

1. Analysing the source code to understand overall working of specmate needs a consistent time.
2. Having a clear idea on source code helped in feature implementation bit faster.
3. Comments help in understanding the source code easily.
4. Analysing the code structure had helped in knowing the architecture of specmate much better

**SPRINT2:**

**2.1 Good maintainability aspects:**

1. specmate testing tool is flexible as new updates and changes are integrated without any conflicts. As adding new feature does not require in modifying other components, we can assume that modularity of the specmate tool is easier and high. In some components reusability can be find. From all these attribute, we can evaluate maintainability of specmate tool as high.

**2.2 The code smells that are already taken care of in the code:**

* Some of the code smells that have already been taken care are:
* Divergent change and shotgun surgery, which are opposite to each other with small resemblance. Any of these cases are not found in specmate tool, which indicates proper code organization, reduced duplicate code, simplified analysis through the code is provided.
* Other code smell is refused bequest, which not found and enables proper inheritance in each class. By this we can obtain the code clarity level in specmate tool.

**2.3 Issues:**

* long method code smell found in object.ts component, which leads to low understandability level and hard to maintain. Long methods may become the place where duplicate code is found.
* primitive obsession- changing an object to primitive type, which consumes more space and reduces performance. These type of code smells leads to low level of maintainability and bad code structure, which need to be refactored.

**2.4 Severity of the code smell and implication it will have on the Specmate code maintenance:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Issue** | **Severity** | **Effect on maintainability- degree** |
| 1 | Object.ts.equals Complexity | Severe | Methods with more number of loops increases complexity. Which leads to the need of code optimization as to decrease code complexity and to decrease the level of analyzability. |
| 2 | ChangeFields Complexity – length | Severe | Due to more no of loops and nested conditions, performance of the system will be decreased as complexity increases, which leads to decrease in maintainability. |
| 3 | Return type void Used as parameter to another function (promise.resolve) | Severe | This type of bugs is false positive and type casting to other data types doesn’t help in removing bug. It doesn’t effect maintainability in any aspect. |
| 4 | refactor this function so that its implementation doesn't duplicate | Severe | Duplicate code leads to increase in complexity and level of understandability decreases as the code is repeated. Impact on maintainability decreases. |
| 5 | Getter - Global | Medium | Changeability and modifiability are affected due to in proper reference to the property. Complexity is increased as confusion arises whether the injected object or property object should be referred. As maintainability of link actions module also decreases. |
| 6 | Arrangement of CSS file - button: focus | Medium | The arrangement of CSS file button to focus creates better usability and customer satisfaction with clear visual display. Where reusability and understandability can be increased but unnecessary focus can make webpage look awful. |
| 7 | Element-tree: CSS | Low | Elements tree is not a defined selector. So, this can lead to unpredictable behavior. This type of bug leads to low level of understandability and analyzability is also decreased with increase in complexity. |
| 8 | Font-family | Low | For specmate tool sanserif font is accepted for better representation which in turn increases analyzability and maintainability. |
| 9 | Not used variables | Low | Unnecessary variables are parameters leads to increase in complexity and decrease in analyzability. |
| 10 | Unnecessary Boolean | Low | Removing Unnecessary Boolean will help in increase performance and in turn improves maintainability. |
| 11 | String object - string cast | Low | Change an object to primitive type. This type of bug reduces performance and consumes more space. Analyzability and understandability are affected. |
| 12 | Undefined parameters | Low | Ambiguous parameters lead to confusion and redundancy increases and understandability decrease. Which is bad practice of maintainability. |
| 13 | Unnecessary cast | Low | typescript automatically cast variables and there is need to specify separately. Which increases performance and reusability. So, it should be removed to increase maintainability. |
| 14 | Void return | Low | methods with return values are often coded as void return and in turn there is no value returned. This leads to unnecessary complexity. This type of bugs decreases maintainability and signifies poor level of programming. |

**2.5 Steps for improving the quality of specmate**

* In order to improve the code quality:
* Firstly, I have analyzed the specmate code structure, mainly web folder. To understand the interconnections between components and to find components that are to be refactored or implement new feature.
* Secondly, I have analyzed the code smell, its effect on specmate tool and how can it be refactored to improve the code quality and maintainability
* Third, implemented the code i.e. needed to remove the code smell or feature to be implemented and tried to test using mocha to understand, whether it is compatible with specmate tool.
* Finally, I have recorded the impact of code smell or new features on maintainability to know which attributes are affected mostly by it.

**2.6 Description of the smells/bugs**

* **Refactor this getter so that it actually refers to the property '\_model'/ property '\_requirement' (M)**
* In additional.info.service.ts component, model and requirement private variables are declared. These variables are accessed by using getter method in link.action.component.ts and in the same component, model and requirement variables are declared as public.
* Which are unnecessary variables and are not used in any of the components. So, removing of these variables can remove the bug and improve the code quality.
* **Impact:** This type of bugs creates unnecessary confusion and leads to low level of understandability. By refactoring, the modularity can be increased and analysability will be easy with reducing complexity. Modification of this code helps in increasing maintainability level.

**2.7 Ensuring the refactoring works:**

We tried to use mocha and chai plugin to ensure that the changes don’t affect the behaviour of the system and whether it is compatible with other changes are not.

**SPRINT 3&4:**

1. **Reflection on the analysability of the Specmate code.**

* Analysability of specmate code is moderate, because of unclear interconnections between the components. In some cases, it is taking moderate amount of time to analyse the components that are to changed and this is due to less comments used in specmate tool. It is harder to know, the usage of particular component I.e. its function, behaviour or role it is performing in this application.
* From Overall analysis, we can derive that it is affecting maintainability level with decrease in analysability.

1. **Feature implementation**

* **No: Feature 7**
* **Variant 2:** User can upload multiple backgrounds, select and configure them. And selected backgrounds should display on the login page for every reload.
* This feature has been implemented using afterviewinit lifecycle hook, and an array is used to store the images that have been uploaded by using textbox button, the array is actually saved in session and retrieved for displaying randomly.
* **Impact:** This feature can increase customer satisfaction and reusability. The disadvantage of this feature implementation is uploading and storing of multiple background images leads to decrease in performance. The maintenance of this feature is somewhat moderate, due to decrease in performance, increase in complexity.