# ARAMIS Interview Transcript

*June 16, 2017*

H = Harris Kristanto

A = Ana Nicolaescu, M.Sc.

F = Muhammad Firdaus Harun, M.Sc.

**H :** “Thank you very much for your time. Today, I would like to conduct a testing session for our approach which is called as CoSA. The aim of this testing is to validate our code smell activity, impact and the harmfulness score. First, I want to explain briefly about the mechanism behind CoSA.

There are two kinds of tool in our framework, the first one is the back-end which responsible to analyze all the information from the project. The information refers to the detected code smells, and the history of that code smells. Later, this information will be stored to the Historical Storage and then be visualized in our front-end tool.

As for your information, before this interview is conducted, I have run the back-end tool, so right now we will focus on the front-end tool.”

**A :** “How often do you run the back-end? Is the back-end triggered every time you change your project in the Version Control System? Or How is it done?”

**H :** “The back-end is triggered by the commit, so when the commit is occurred, the back-end analysis will be run automatically”

**A :** “Okay”

**H :** “Okay, now I will start to ask the General Questions. What is your name?”

**A :** “I am Ana”

**H :** “How long have you been working in the software development field?”

**A :** “Does it count when I was a student?”

**H :** “Only the professional experience”

**A :** “In a firm, I only worked for two or three years”

**H :** “How long have you known about the software development? It includes your student experience”

**A :** “Since 2005”

**H :** “As you may know that we have analyzed six ARAMIS project, thus I would like to ask you, which one of these projects that you have a big and huge knowledge?”

**A :** “Unfortunately, the projects that I know best are ARAMISMapper and ARAMISValidator which are not within these. As for the other one, I do not know very much detail. But, if I were to choose, I will choose the ARAMISHtrgInputModels”

**H :** “What was your role in this development?”

**A :** “I was the designer, the supervisor”

**H :** “In the development, did you code something or just reviewed the code?”

**A :** “I reviewed the code but I did not program the code”

**H :** “And, How long does it take to develop this program?”

**A :** “Four months and one developer”

**H :** “Based on the performed changes, what kind of changes that happened in the past? Is it bug-fixing? New Feature Implementation? Or?”

**A :** “They are mostly bug-fixing, code reviewing and not so many implementing new features”

**H :** “Okay and as the code-reviewer and developer, when you found some code smells what kind of information that you want to know that code smells?”

**A :** “The code smell types as if it is a code smell that I am not familiar with I would like to know what is that code smell, its factors (that led to identify the code smell). Generally, I would also like to see the severity (since it shows the smelliness), the impact (how does it impact the maintenance) and some refactoring recommendations.”

**H :** “Okay, so this is the end of the General Questions. Before we jump to the next session, I will tell you briefly about our front-end. In this page (the Evolution Form page), use can choose the code smell types which currently only focus on God Class, Brain Class and Data Class. They can also choose the code smell activity types which are the Strong Active, Stable Active, Ameliorate Active and Dormant”

**A :** “What does it mean?”

**H :** “I will tell you about these activities in the next page”

**A :** “Okay fine”

**H :** “Okay, so this is the next page (Evolution Detail page). It shows the details of the analysis including the total number of commits, total number of code smells, last analysis and last commit date.  
  
This page also shows the Harmfulness Mapping, I will explain the Harmfulness Mapping after I explained the Harmfulness Score.

At this section (Download As section), user can export the reporting in excel format. While for this section (Filter section), it is used to filter the table below.

And as I promised previously, I will explain about the Code Smell Activity Types. The explanation is provided in the form of example which you can open from this section (The Question Mark next to Activity Type section).

So, in this example, we have a line graph that shows the evolution of a god class in its six versions. The horizontal line refers to the version and the vertical line refers to the ATFD. ATFD is one of software metrics that is used to classify a god class. The number that is represented next to the point in the line graph, refers to the value of that ATFD in the corresponding version.

In respect to the code smell activity, in the first version, since it is suspected as a god class thus we classify its activity to Strong Active. In the second version, changes were performed, but the performed changes did not really modify the code smell severity thus this activity is classified as Stable Active. Next, as you can see the severity of that code smell is increasing a lot, then we classify this activity to Strong Active. And the next one is stable active, and so on.”

**A :** “Okay, I understood. But, how if I eliminated the code smell? It won’t appear in this or?”

**H :** “Yes, it won’t appear in this”

**A :** “Okay, but may be, I want to see all the code smells that I have eliminated. So, the list of the one that were code smells but are not code smells anymore might be interesting”

**H :** “In the current implementation we do not have this feature, however, I will put it as a suggestion for the next implementation”

**F :** “Why do you need the code smell that have been refactored?”

**A :** “Just to see the improvement of my code.”

**H :** “Thank you for your suggestions, the last one I want to mention about the Dormant which mean that there are no changes at all.

Now, I will tell you about the Harmfulness Score. The Harmfulness Score is computed by the Code Smell Activity and the Impact. We provide the four quadrant of that Harmfulness Score, which are the Very High, High, Medium and Low. Very High refers to high activity and impact, High refers to high activity but low impact, Medium refers to low activity but high impact and Low refers to low activity and impact.

We also want to mention that in this testing we labeled Strong Active as 3, Stable Active as 2, Ameliorate Active as 1 and Dormant as 0”

**A :** “Okay”

**H :** “Now, I want to ask you the next questions session. As I mentioned before, we have two kind of factors to compute the harmfulness score, the code smell activity and Impact. Do you think that these factors are enough to compute the harmfulness score?”

**A :** “How do you compute the Impact?”

**H :** “The impact is the product of Complexity and Most Critical Symptom. To know these most critical symptom we refer to Radu’s research. For example, the most critical symptom for god class is ATFD.”

**A :** “Okay, then I don’t have any other better idea how to compute the harmfulness, so I think it is okay”

**H :** “Then, I also have mentioned about the Harmfulness Mapping which has four quadrants. Do you think that this is appropriate to have this kind of classification? Or do you want to have it more detailed?”

**A :** “I don’t think you need to make it more detail. For me, it is even a bit too much. I will focus on the Very High and High since Very High represents something that you need to act upon it right now, and High, it means that it is almost harmful, so it is not good as well. For the Low impact, it is there, but not so important since we don’t need to care so much about it.”

**H :** “Okay, we will go to the next question. Right now, we have found these code smells, does the highest harmful code smell was really hard to maintain and review (according to your experience)?”

**A :** “This class (the most harmful code smell) is responsible to transform the boxes and line from the eclipse project to ARAMIS perspective description format. And the thing is, he (the developer) put all the implementation in this single class as well as the whole logic. I didn’t understand why he (the developer) put everything in the same place. He delegated a lot but in the end, he created the whole functionality in this class. He also has another class (that similar with this class) if I am not mistake, where he computed the frequency.

Anyway, yeah I recognized it as a god class”

**H :** “From these code smells, we have computed their Harmfulness Score. So, do you think that it is correctly computed? In the sense of, does the most harmful code smell was much harder to deal with in the past compare to other code smells. In other words, do you think that the harmfulness score can express your experience in reviewing your code?”

**A :** “Yes, as we can see it from the code, the most harmful code smell is much more complex, it has more information, etc. So yeah, I agree with it. And I also agree with the harmfulness score computation (where the second harmful code smell supposedly placed in the second highest and so on”

**H :** “And, do you think that the most harmful code smell has a high change frequency in the past rather than the other code smells?”

**A :** “Almost, I am not sure whether the first one or the second one that is most frequently changed. I think it is the first one, since it has more business logic in it”

**H :** “So, based on these information, will you have an idea to choose the refactoring candidates in the future?”

**A :** “Yes, I will refactor the ARAMISSwitch (the third harmful code smell)”

**H :**  “Why do you want to refactor ARAMISSwitch instead of LineImplementation (the most harmful code smell)?”

**A :** “Because, the problem is clearly visible. For the LineImplementation, it will be a bit harder. I will need more time to examine that class. Thus, if I were to do something very fast and I don’t have enough time, then I will definitely refactor the ARAMISSwitch first. It is just because of the practical reasons.”

**H :** “Thank you and this is the end of the second session. In the next session, I would like you to have a user experience to use the front-end, especially in the ‘View Evolution’ part (since it is the focus of this session)”

(**A** using the system)

**H :** “Now we will start the next session. In these evolution graph, what kind of information that you can gather from?”

**A :** “It shows the changes that happened, some refactoring was done that why this level is decreasing. And a new feature is implemented because it became worst again. I think I can understand a little bit how you worked on the project by looking at this. But I am a bit confused with the Activity Evolution Graph since it is represented in the form of numerical value instead of Strong Active, Stable Active and so on

Yeah, I think these two evolution graphs show the same interpretation. So, what I can see here that I cannot see above?”

**H :** “The graph above (the Impact graph) actually shows the badness level of a code smell. For example, we have three classes (A, B and C) that exhibit a code smell and its Impact is 0, 1.5 and 3 respectively. So, the evolution graph of class A, B and C will be like this (show the example in the paper).”

**A :** “Okay, if that is the case, it is not so intuitive for me. I didn’t understand from the beginning that the Impact refers to all code smells. That is why I didn’t see the difference between the Activity and Impact”

**H :** “Okay, and for the last question, Do you think that the Harmfulness Mapping can help you to identify the next refactoring candidates?”

**A :** “Sure, I will look the code smells in the bad place first (Very High and High). We can the big overview first before we see the detail of each code smells in the table below.”

**H :** “So, that is all our testing session. Thank you very much for your contribution”

**A :** “Your welcome, nice interesting.”