Paul Court

Assignment 1 SE 670 Spring 2021

Dr. Omar Al-Azzam

Due: 1/26/2021

Chapter 1: Why do we test software?

1. **What are some factors that would help a development organization move from Beizer's testing level 2 to testing level 4?**

There are several issues an organization will need to address to move up the levels of test process maturity. As I see it, at level 2 testers are focused on showing failure, which is an inherently negative activity and thought pattern. It is a necessary evil when testing, however it usually manifests a rift between developers and testers. At this level, if no failures are found, usually developers consider that a win for their side however, there are very seldom no issues with software, so testers have most likely not done their job with the detail required for a successful launch of the project.

At level 3, this idea that a successful test shows no failures is debunked. Risk is a bigger part of the equation. When a software runs "correctly" is when you may incur risk, finding ambiguities in the product near or after its "completion". A team mentality between testers and developers promotes premium overall work. This can be accomplished by hiring and retaining a quality workforce. Nurturing the relationship among testers and developers will allow a company to mature to level 3 on Beizer's scale.

To move beyond this to level 4, which defines testing as a mental discipline that increases quality, I believe an organization must not only foster a cohesive, wholistic approach to product development but they must also demonstrate this at the highest level. This will allow the detection and mitigation of errors, faults, and failures early on in the product life cycle and decrease time, frustration, and cost issues later in the proceedings. Leadership or management need to *model* harmony and teamwork to achieve this as an organization. You cannot have unity in an organization when the leaders do not demonstrate this on a daily level from the top. This is key to the overall success. When every member of the team has the mental discipline and focus to produce a successful, working product that tests out with the highest quality, your organization has arrived at level 4.

To summarize, there are several issues to tackle to move your organization from level 2 to level 4 on Beizer's scale of organizational maturity. Avoid viewing testers as negative. Employ practices that encourage hiring and retaining quality people for development and testing teams. Promote harmony among developers and testers, using a team concept. Finally, be sure management embraces these concepts and models them for the whole organization.

2. **What is the difference between software fault and software failure?**

It is a common misconception in discussion of software issues that bug, error, fault, failure, and other terms mean the same thing. For the most part, they all mean there are issues, and the program does not display the intended functionality. A fault in a software is a static defect in the software, a root cause

of the “symptoms” that are displayed in the software. The software will not perform the required function because of some design mistake. Whereas a failure is the inability of the system or component to perform according to its specification. This would be the actual “symptom” or inconsistency displayed by the product. The challenge of a good tester is to find as many of these failures before release so the manifesting fault can be determined and corrected as early in the process as possible.

3. **What do we mean by “level 3 thinking is that the purpose of testing is to reduce risk?” What risk? Can we reduce risk to zero?**

Once a product has reached the level 3 maturity, as far as testing goes, there is a risk that the software works well enough to appear “complete” and ready for release. Why is this a risk? If all functionalities check out to the point that the team feels confident in the product, there is still the underlying concerns that there will ultimately be failures. This would increase the risk to a business’ reputation when these failures are discovered by a customer. Some of these failures may not amount to much. However, the harm can come from cost of mitigation efforts, dissatisfied customers, downtime, and other negative impacts. Since there can never be a product that has zero possible failures, there will always be risk. Eventually, a product must be released. It is the job of developers and testers to decrease the risk, or at least reduce it in the areas that are essential to security, quality of user experience, future potential cost of mitigation and overall performance.

When risk is managed keeping the future impact of potential failures in mind, then developers and testers have done their job correctly and a quality product is the result.

4.

a. **Write a Java method with the signature public static Vector union (Vector a, Vector b). The method should return a Vector of objects that are in either of the two argument vectors.**

```
//****** Start of Java Method, union for SE 670 Assignment 1 Question 4 a)********************
public static Vector union(Vector a, Vector b) {
//Initalizes a union vector of type Object.
  Vector<Object> uVect = new Vector<Object>();
//Initializes an iteratior as "it" and initializes it for Vector a.
  Iterator it = a.iterator();
//This loop will store the items in the vector a into the union vector,
//uVect and check to see if vector a has any repeated items.
  while (it.hasNext())
  {
        Object thisItem = it.next();
    if (!uVect.contains(thisItem))
```

```
        { //If the union vector doesn't already have the item, add it!
        uVect.add (thisItem);
        }
  }
//Establishes an iterator for Vector b.
        it = b.iterator();
//This loop will store the items from vector b into the union vector if they are not already there.
  while (it.hasNext())
  {
        Object thisItem = it.next();
    if (!uVect.contains(thisItem))
        { //If the union vector doesn't already have the item, add it!
        uVect.add (thisItem);
        }
  }
  //Returns the finished union vector.
    return uVect;
}
}
```

b. Upon reflection, you may discover a variety of defects and ambiguities in the given assignment. In other words, ample opportunities for faults exist. Describe as many possible faults as you can.

There were many faults that I encountered as I wrote this method. First, I have not programmed in Java before and encountered several issues that were initially more at level 1 and 2. When developing the main program and passing the Vectors as objects, I did find errors more on par with faults. For example, I found that if I included duplicate entries in either Vector, they were not omitted. At first, I wrote the method to handle integer data. There will be issues if different data types are used. As I thought about test cases for other types of data, I realized I neede to change the code to allow for the handling of any type of data entered. A surprising non-issue is problem of addressing a null Vector. I thought I might have to account for that with an If statement "If(!a.null) or something of that nature. Simply leaving the new Vector unfilled did work. Leaving both unfilled also returned the expected result of a null union Vector.

Other than encountering the usual syntax issues related to programming in a language you have never used before, these would be the only failure issues I can foresee. Whenever I get to testing a program, I feel it would be nice to have a team approach. I value other's input to what test cases would be necessary for a complete coverage of the issues that could arise. It is difficult to test code you have written.

c. Create a set of test cases that you think would have a reasonable chance of revealing the faults you identified above. Document a rationale for each test in your test set. If possible, characterize all of your rationales in some concise summary. Run your tests against your implementation.

Test Case Documentation

Case #	Objective	Vector Entries	Expected Result	Actual Result
1	Simple working of the union of two vectors.	Vector a = [1, 2, 3] Vector b = [4, 5, 6]	Union = [1, 2, 3, 4, 5, 6]	The union vector for test case 1 is: [1, 2, 3, 4, 5, 6]
2	Does the method account for repeated entries in Vector a and vector b?	Vector c = [1, 2, 3] Vector d = [3, 5, 6]	Union = [1, 2, 3, 5, 6]	The union vector for test case 2 is: [1, 2, 3, 5, 6]
3	Does the method allow for the change from integers data to character data?	Vector e = [‘a’, ‘b’] Vector f = [‘c’, ‘d’]	Union = [a, b, c, d]	The union vector for test case 3 is: [a, b, c, d]
4	Does the method allow for repeated entries with character data?	Vector g = [‘a’, ‘b’] Vector h = [‘b’, ‘a’]	Union = [a, b]	The union vector for test case 4 is: [a, b]
5	Does the method allow for one entry in one vector and none in the other?	Vector i = [] Vector j = [1]	Union = [1]	The union vector for test case 5 is: [1]
6	Does the method allow for mixed data entries?	Vector k = [‘a’, ‘b’] Vector m = [1, 2]	Union = [a, b, 1, 2]	The union vector for test case 6 is: [a, b, 1, 2]
7	Does the method eliminate duplicate entries in an array?	Vector n = [‘a’, ‘b’, ‘a’] Vector p = [‘c’, ‘d’, ‘c’, ‘c’]	Union = [a, b, c, d]	The union vector for test case 7 is: [a, b, c, d]
8	Does the method allow for larger amounts of vector entries, negative values, and decimal values?	Vector q = [1, 2, 3, 4, 5, 6, 8, -1, 4, 3.2, 8] Vector r = [3, 5, 6, 12, 1000]	Union = [1, 2, 3, 4, 5, 6, 8, -1, 3.2, 12, 1000]	The union vector for test case 8 is: [1, 2, 3, 4, 5, 6, 8, -1, 3.2, 12, 1000]
9	Does the method allow the union of two null vectors?	Vector u = [] Vector v = []	Union = []	The union vector for test case 9 is[]
10	Does the method handle string data?	Vector w = [“Ford”, “Chevy”, “Toyota”, “BMW”]	Union = [Ford, Chevy, Toyota, BMW, The, sly, brown, fox, jumped, over, the]	The union vector for test case 10 is: [Ford, Chevy, Toyota, BMW, The, sly, brown,

		Vector x = [“The”, “sly”, “brown”, “fox”, “jumped”, “over”, “the”, “BMW”]		fox, jumped, over, the]

Table 1: Test case summary of tests for Method Union of two Vectors.

Test Case Result verification.

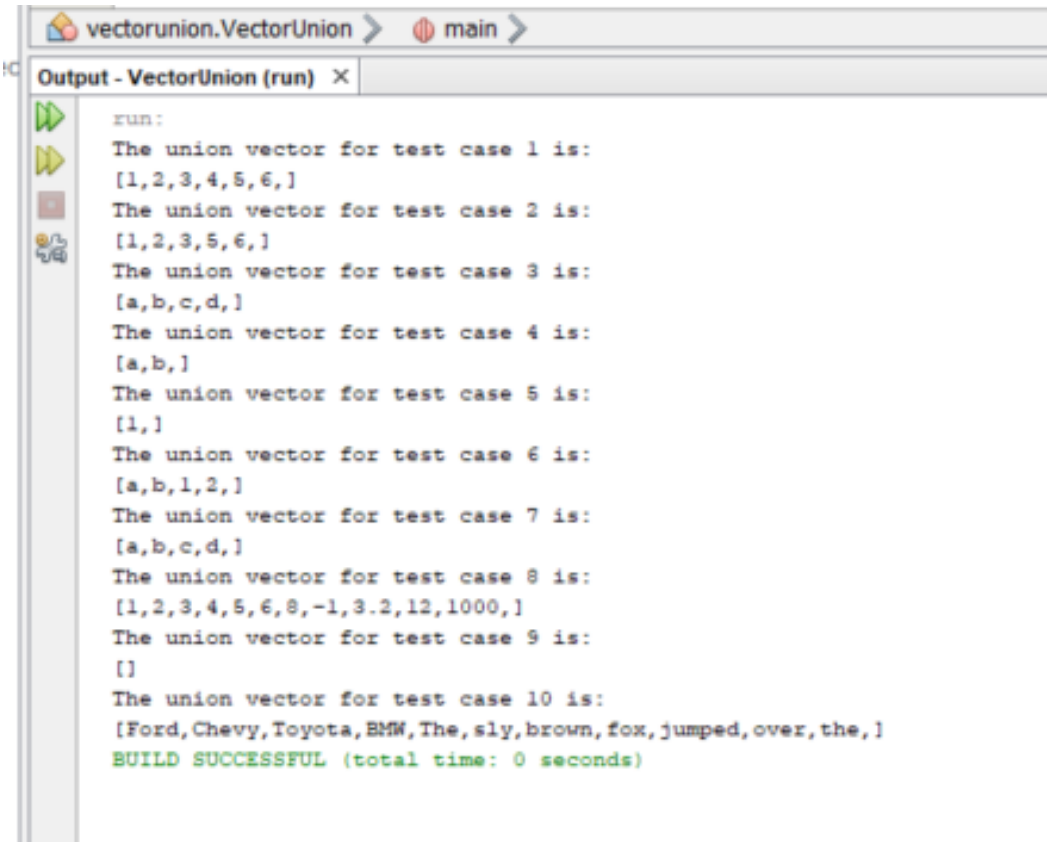

Figure 1: Snapshot of main program output to verify the results of the test cases.

In summary, I am sure that there could be more extensive testing of this method. I feel I covered the problematic areas well and am happy with the results. I could not get the comma that follows the last item of the union Vector be eliminated and calculated that as a risk of losing points. (Reminder: I have not programmed in Java before.) This output presentation was better than others I had tried. I weighed these options and chose this as the best alternative.

As any software, I am sure that there are some issues that I did not address. However, with this code, there is little risk involved in failure to perform correctly so any problems pose little inherent concerns.