Testing Report

1. Category Partition Method

The category partition method is a powerful technique allowing test engineers to identify distinct categories of inputs that might lead to erroneous behaviour of the System Under Test(SUT), in a structured and comprehensive manner. In the present report the SUT is the Java Collections Framework, which is part of the Oracle Java Software Development Kit. To modularise the testing process, independent *functional units*[1] have been selected: the *sort, rotate* and *min* methods implemented as part of *Collections.* Independent test procedures and data will be developed for each functional unit in accordance with the ECSS guidelines(ECSS-E-ST-40C – 5.5.3.2).[2] Subsequently, the choices identified in this stage will be used as inputs for these methods.

For the three parameters provided for the functional units(list, distance, collection), three categories of inputs have been selected based on the properties of the valid input data: Content, Size and Data Type. Since the *collection* data type is extended by the list, set and map types, only sets will be discussed further*(Figure 3)*. This has been decided due to the property of sets to exclude duplicates from a given array of items, which often leads to interesting implications during sorting, but was also selected because lists are independently discussed in the context of *rotation*. Array-based structures such as lists and sets were categorised by their content(numbers, letters, symbols etc) and size, while the distance of rotation was

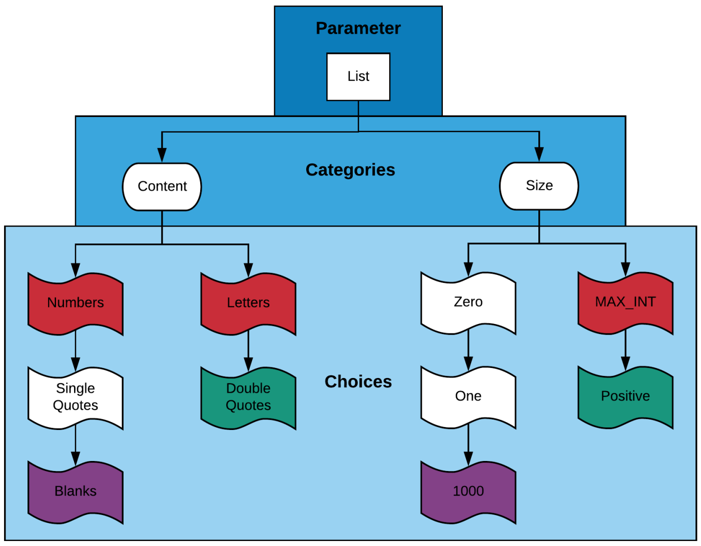

Figure 1: Choice derivation for List

characterised by value and data type*(Figure 1).*

ID	Parameter	Category	Choice
1	List	Content	Numbers
2	List	Content	Letters
3	List	Content	Single Quotes
4	List	Content	Backslash
5	List	Content	Blanks
6	List	Size	Zero
7	List	Size	Positive
8	List	Size	MAX_INT
9	Distance	Content	Positive Integer
10	Distance	Content	Negative Integer
11	Distance	Content	Letter
12	Distance	Content	length(list)
13	Distance	Content	Zero
14	Distance	Data Type	Int
15	Distance	Data Type	Double
16	Distance	Data Type	Char
17	Collection(Set)	Content	Alphanumeric
18	Collection(Set)	Content	null
19	Collection(Set)	Content	min(long)
20	Collection(Set)	Content	Single Quotes
21	Collection(Set)	Content	Double Quotes
22	Collection(Set)	Size	Zero
23	Collection(Set)	Size	MAX_INT
24	Collection(Set)	Size	Positive

Table 1: Choice listing

As shown in *Figure 1,* the categories are divided into choices, which represent concrete types of values that can be taken by the inputs and will be used to derive the test cases. **The basis for choice selection was anticipating what**

types of inputs might exhibit undue behaviour when reordered or shifted, such as elements being merged together due to nested quotes becoming adjacent or blank spaces collapsing at the end of lists.

The data in *Table 1* indicates that the baseline input for list and set choices is represented by alphanumeric values, intended to test the core functionality of the

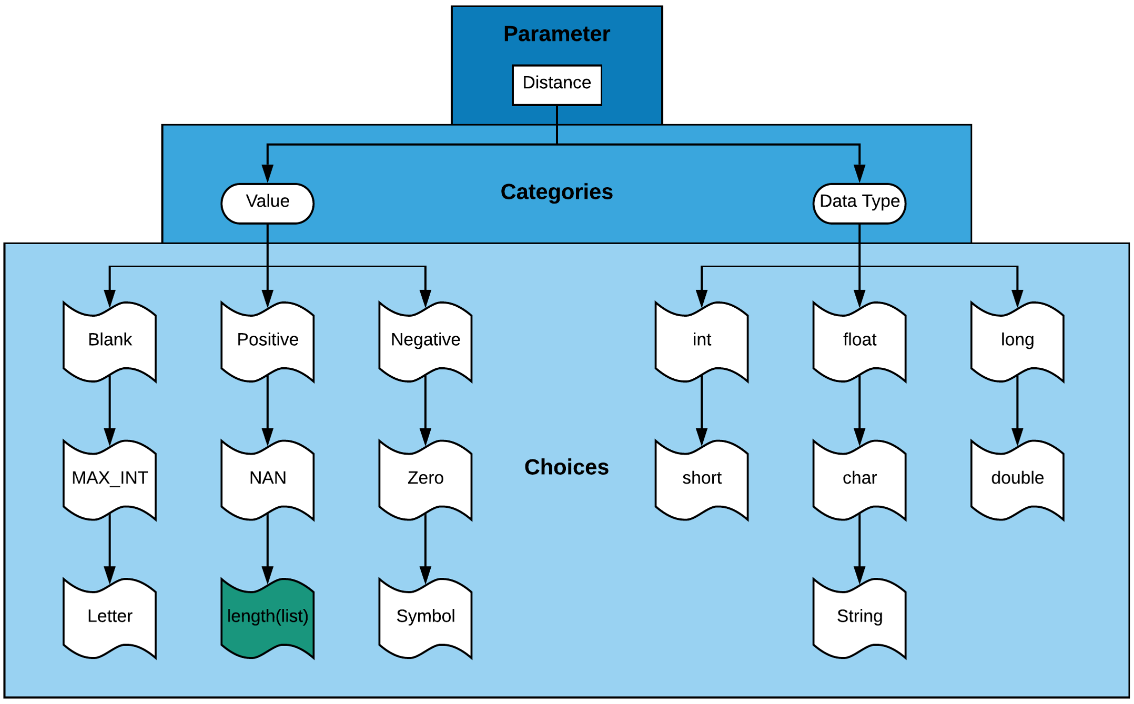

Figure 2: Choice Derivation for Distance

program for “reasonable” user inputs. These are subsequently combined with special characters such as single/double quotes, escape characters or blank spaces, which are known to have lead to unexpected behaviour. [3] Of special interest are also boundary values, which test the functionality of a system at the limits of a specific structure and represents the starting point for stress testing. The boundaries considered in the present study were: MIN_INT, ZERO and MAX_INT.

Characteristic for the *rotate* method is the case when the rotation distance coincides with the length of the list(*Figure 2*). This input makes the function periodic and is therefore a valuable foundation for a metamorphic relation. Also important are numbers that are defined as infinitely repeating decimals(eg. one third), but which have to be stored by the system as a finite decimal number(double), as a result the system might not interpret expressions such as “three times one third” as an integer. However this was considered more a property of double-precision numbers than of the function implementation. It was also observed that the function admits rotation by a single character enclosed within single quotes, but not double quotes.

This is an exception allowed by the SUT and therefore a vulnerable part of the system, so it will be exploited. The *String* type was excluded since it was always find to trigger an exception.

For the *min* method, an input of special interest would be the lower boundary of the numeric range, namely the smallest integer that can be stored in a *long* variable. It is expected that for an arbitrary number of alphanumeric values added to the set, the global minimum should not change from the original MIN_INT. This approach allows one to take advantage of the properties of both metamorphic testing and boundary value analysis [4]. Additionally, values that could be considered equivalent by the search algorithm were included, such as single and double quotes or blank spaces and null values. Of special importance for stress testing are cases approaching the memory limits of the computer's hardware, such as increasing the length of the set until the size approaches MAX_INT. On the other hand, testing an empty set for a minimum is known to throw an exception, therefore it will not be verified more than once.

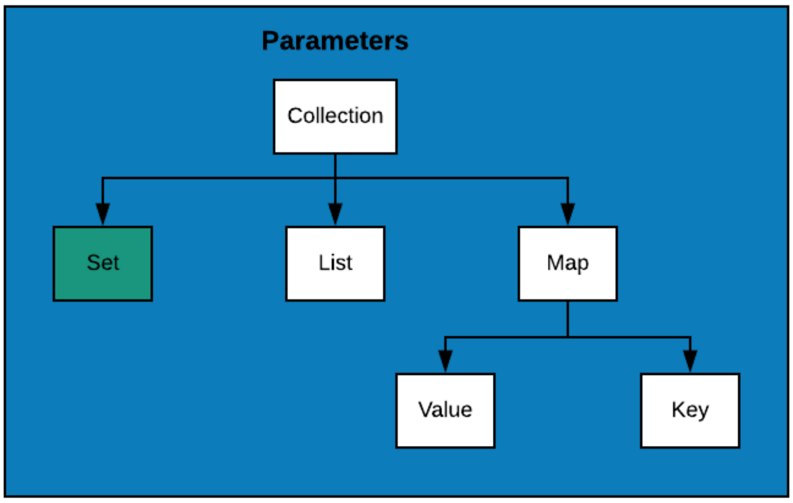

Figure 3: Parameter Selection for Collection

2. Test Cases

To begin with, constraints are applied to the choices identified in the previous step in order to avoid having to repeatedly test conditions that would not lead to any distinct results. This allows the testing engineer to avoid having to implement test cases for all the possible combinations of choices. For example, rotating an empty list by a range of distances would still lead to the same result, as would happen when trying to sort an empty list. Similarly, inputs that would lead to an exception being thrown, such as rotating a list by a string, were excluded. A substantial number of choice combinations are also mutually exclusive, such as having an empty set with alphanumeric values. Consequently, three test cases will be selected for each function based on their probability to cause erroneous behaviour.

For *sort* it was assumed that the function works as expected for homogenous data types, such as sorting only numbers or only characters; then, the choices were combined in order to obtain inputs on increasing levels of complexity:

- alphanumerics(eg: 23argre43555)

- alphanumerics with single quotes(eg: 23ar’gre43’555’)

- alphanumerics with blanks and single quotes (eg: ‘23ar gre 4355 5)

As a result, by starting with an initial alphanumeric string and adding blanks or quotes in the same positions for each string and then sorting the list, one would expect one to one mapping of the sorted alphanumeric array and the sorted array that contains special characters.

For *rotate*, choices have to be selected for each one of the two parameters(list and distance). The choices for the list parameter were preserved as before(alphanumerics with blanks and quotes) but escape characters were added. The distance parameter was then varied by considering integers, floating points and characters:

- rotate alphanumerics with blanks, single quotes and escape characters by an integer distance(“df2a2er23’/dl’\”,2)

- rotate alphanumerics with blanks, single quotes and escape characters by an fractional distance(“df2a2er23’/dl’\”,1/3)

- rotate alphanumerics with banks, quotes and escape characters by a char distance(“df2a2er23’/dl’\”,’c’)

In order to test the functionality of *min* there are immediate advantages in starting with a set that contains the the global minimum for a range of possible values, hence the MIN_INT constant, which is the minimum value that can be stored in a *long* variable, has been added to most test cases. Additionally, null values and double quotes have been added to the alphanumeric variables in order to monitor any immediate changes after ordering, shuffling or scaling certain variables prior to applying the *min* function:

- positive length and elements containing alphanumerics + MIN_INT

- alphanumerics, single and double quotes and MIN_INT

- alphanumerics, double quotes and NULLs

---

3. Metamorphic Relations

Metamorphic testing is a powerful approach for verifying systems where the source code for certain components is unavailable, hence black-box methods have to be resorted to. This technique is focused on identifying transformations which, when applied to the input, result in a consequential transformation on the output. The metamorphic relations used throughout this report were found by first considering general properties of functions such as periodicity, injectivity, surjectivity, evenness etc in order to find relationships between the outputs of functions, which must hold regardless of the changes on the input. This resulted in possible transformations which, when applied to the input, would lead to results which are equivalent with the original input.

For sorting, it was considered that a well implemented sorting algorithm would deliver the same output regardless of the initial conditions in the input, as long as the input remains the same. Therefore, no matter how much one would shuffle a list, the sorted output should always be the same. The reasoning behind this was that when elements containing special characters such as quotes or blanks might merge together or collapse when they become adjacent, hence leading to either less elements or a different ordering.

The second metamorphic relation for sorting was found based on general properties of functions, namely if a function is not injective prior to sorting(has duplicates), it should not be injective after sorting(unique elements). It was thought that for a list having multiple blanks and nulls might merge them during sorting, which should happen for sets but not for lists. There were also attempts to exploit the weaknesses of the quicksort algorithm for primitive data types but the solutions would not have been trivial.

Rotation is a periodic operation with period *N*, where *N* is the length of the list*;* this property was exploited in determining the first metamorphic relation. Therefore, no matter how many times a list is rotated by *N* the input and output should coincide.

Similarly, rotation is bijective, hence the operation can be done and undone while expecting the initial input to remain the same. Specifically, by rotating a list by an arbitrary distance in one direction and back by the same distance, one would expect to end up with the same output as the original input. It was considered that undue behaviour might occur when elements containing blanks or quotes are placed at the start or end of a list, therefore collapsing blanks or nulls, which would result in the list being different when rotated back to the initial condition.

The metamorphic relation for the minimum operation was found based on the property that for a given set of inputs, there exists a value which is the global minimum for the whole range admitted by that set. For example, *min(long)* is the smallest value that can be stored in a long variable, therefore by starting with a set containing only the global minimum, regardless of what elements one could add, the output should always be the initial element. It was thought that elements starting with escape characters or blanks might lead to undue behaviour in the binary search algorithm when placed in the same set with alphanumeric elements.

The last relation was found by making use of different functions and implementations in order to obtain the same result. By finding the minimum in an arbitrary set, then sorting the set from highest to lowest and finding the minimum again, one would expect the same result. It was again expected that elements containing special characters might result in the minimum merging with adjacent elements.

Results and conclusions

After extensive testing, none of the expected scenarios were found to lead to a system failure. The only cases that resulted in the system exhibiting erroneous behaviour occurred when the input was altered such that it was no longer a valid input for that function, for example when the length of the string used as a distance for the *rotate* method exceeds one, the system will throw an exception. These conclusions point to the robustness in implementation of the Java Collections Framework.

References

[1] Ostrand, T. J., and Balcer, M. J. 1988. The category-partition method for specifying and generating fuctional tests. Comm.of the ACM 31(6), 676-686.

[2] ECSS-E-ST-40C – 5.5.3.2 Software Unit Testing a. “The supplier shall develop and document the test procedures and data for testing each software unit.”

[3] M. Martin, B. Livshits, and M. S. Lam. Finding application errors and security flaws using pql: a program query language. In ACM SIGPLAN Notices, volume 40, pages 365–383. ACM, 2005.

[4] Khan, M.E. Different approaches to white box testing technique for finding errors, International Journal of Software Engineering and its Applications, 3 (vol. 5), 2011, pp. 1-14