ISSN: 2455 - 1597

ijergs
Open Access

International Journal of Engineering Research and Generic Science (IJERGS)
Available online at: https://www.ijergs.in

Volume - 5, Issue - 5, September - October - 2019, Page No. 15 - 20

Object Oriented Programming Testing In Java and Python and It's Pitfalls With Challenges

[1]Ankit kr. Taneja, Computer Science Department, Jaipur National University Jaipur, India

[2]Dr.Sonu Mittal, Computer Science Department, Jaipur National University Jaipur, India

E-Mail Id: ankit_taneja2002@yahoo.com, dr.sonumittal@jnujaipur.ac.in

Abstract

Object-oriented era is becoming increasingly famous in numerous one-of-a-kind contexts. The Object-orientated paradigm has been applied inside the areas of programming languages, databases, person interfaces, specification and design methodologies. Object-oriented languages are extensively carried out in enterprise, and numerous business programs are designed and evolved with object oriented era. As a result, the attitude toward item-oriented software program excellent has undergone a rapid alternate over the past years. Initially, the object oriented paradigm has been considered powerful enough to assure software program exceptional with none extra effort. Several analysis and layout methodologies state that a well-designed item-oriented device could only want minimum trying out. Unfortunately, although item-orientation enforces many critical programming concepts, consisting of modularity, encapsulation, and facts hiding, it is not sufficient to assure the great of software program products. Today, each practitioners and researchers are aware that object oriented software contains errors similar to traditional code. Moreover, object oriented systems gift new and special problems with respect to traditional applications, due to their peculiarities.

Keywords: Oops, Testing, Unit Testing, Real Time, Integration Testing.

1. Introduction

The object-orientated paradigm is based on the assumption (or instinct) that it is herbal to specify, design, and broaden a software program machine in phrases of objects. Such an assumption is justified by using the commentary that computer programs version real international entities together with their interactions, and human beings tend to see their environment in terms of objects. In a totally trendy way, we may also say that we practice the item-orientated paradigm each time we think about software program structures in terms of items and interactions between them. It is impossible to trace a borderline among what may be considered object generation a what cannot. There exist one-of-a-kind degrees of item-orientation, and one-of-a-kind classifications were proposed to mirror how a good deal a system implements the object-oriented paradigm.

2. Testing

Software verification is the hobby of establishing whether or not a program efficiently implements a given model. Verification techniques can be outstanding amongst static and dynamic analysis strategies. Static evaluation strategies do now not require the program below take a look at to be completed, at the same time as dynamic evaluation techniques do. Examples of static analysis strategies are formal proofs of correctness, code inspections, statistics-drift analysis. Examples of dynamic strategies are trying out strategies. Since this thesis focuses on trying out of object-orientated systems, in this bankruptcy we only don't forget the primary principles of software testing. At various levels of testing of object oriented software, techniques which can be applied are :

1. Unit Testing
2. Method Testing
3. Class Testing
4. Integration Testing
5. System Testing

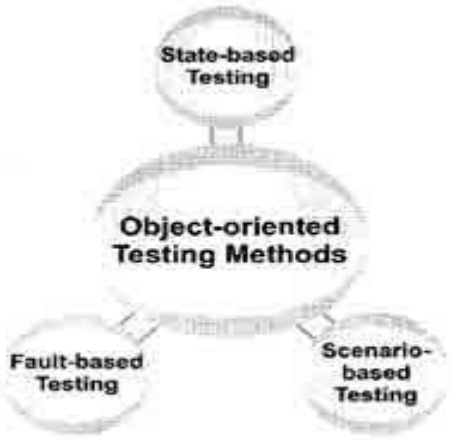
3. Unit Testing

The trying out of a single application unit, wherein the term unit can expect one-of-a-kind meanings relying at the precise surroundings. A unit can be a single procedure or module. Unit testing is characterized by using being generally carried on by using the programmer that without a doubt evolved the code and accordingly has a entire visibility and maximum perception at the code. Due to the excessive stage of under stand ability of the software program at this level, the selection of check information workout precise elements of the code is commonly a good deal less difficult in this situation than at some point of integration and machine checking out. The primary issues with unit testing is the development of the scaffolding (i.e. drivers, stubs, and oracles) making an allowance for real executing single devices in isolation, which may be a completely complex mission. In particular, in the case of item-oriented systems the development of drivers and stubs requires the “emulation” of lacking lessons, that may provide complicated functionalities whose conduct is depending at the interactions among them and are hence hard to emulate in a meaningful way.

4. Integration Testing

The testing of person gadgets enables in disposing of local faults, however does not exercise the interactions amongst different units. Integration trying out is the pastime of exercising such interactions by means of pulling together the distinct modules composing a gadget. It is characterized by way of involving distinct interacting units that have been in general evolved by using different programmers. In this situation the code remains seen, but with a better granularity. Faults that can be found out by means of integration trying out include interface problems, missing functionalities, and unforeseen aspect-results of procedure invocation (as a ways as traditional procedural programming languages are involved). The above are only some examples of all the feasible issues that may arise in the course of integration of a software machine. In unique, many issues are language specific, or specific to instructions of languages. Before choosing an integration checking out approach, it is thus very essential to keep in mind the class of issues the check must deal with. For instance, when using a strongly typed language, many distinctive interface mistakes as the ones related to the wrong kind of parameters in a method call can statically be identified and removed. The essential difficulty in integration trying out is the choice of an integration order, i.E., the order wherein the distinctive gadgets, or modules, are incorporated. It is possible to pick out 5 primary strategies as a long way as the mixing order is worried, particularly, pinnacle-down, bottom-up, large-bang, threads, and critical modules. The pinnacle-down integration approach is the only wherein the combination begins with the higher module in the hierarchy defined via the use relation among modules, i.E., it starts with the module that is now not used by another module inside the system. The different modules are then brought to the machine incrementally, following the use hierarchy. In this manner, there's no need for drivers, however complex stubs are wanted. The backside-up integration method is the only wherein the combination begins with the decrease modules in

the use hierarchy, i.E., it starts with the modules that do not use any other module in the machine, and keeps by way of incrementally adding modules which are using already tested modules.



Object-oriented Testing Methods

5. Challenges of Testing Object-Oriented Systems

The predominant trouble with checking out item-orientated systems is that preferred trying out methodologies may not be beneficial. Smith and Robson [7] say that cutting-edge IEEE trying out definitions and suggestions cannot be implemented blindly to Object Oriented checking out, because they comply with the Von Newman version of processing. This model describes a passive save with energetic processor performing upon the store. It calls for that there be an oracle to determine whether or now not this system has functioned as required, with evaluation of performance in opposition to a defined specification." They also gift the subsequent definition of the testing process: "The technique of exercise the exercises provided with the aid of an object with the goal of uncovering mistakes inside the implementation of the routines or the nation of the object or each."Smith and Robson say that the procedure of trying out Object Oriented software program is extra hard than the traditional method, considering packages aren't executed in a sequential manner. Object Oriented additives can be combined in an arbitrary order; as a consequence defining test instances becomes a look for the order of exercises in an effort to reason an error. Shipman and Newton[8] agree that the country-primarily based nature of Object Oriented structures will have a poor impact on testing. Siepmann and Newton country that the iterative nature of growing Object Oriented systems calls for regression testing between iterations. Smith and Robson kingdom that inheritance is tricky; since the only manner to check a subclass is to flatten it through collapsing the inheritance structure till it seems to be a single class. When this is achieved, the checking out effort for the super elegance isn't always applied; consequently, duplicated checking out takes vicinity.

6. Running The Code

Python can be run from the command line; greater with no trouble, you may run it from inside an integrated improvement surroundings (IDE) inclusive of Eclipse. I opt to use an IDE due to the numerous productiveness enhancements they

offer: code technology, unit trying out, bundle and module advent, and so forth. Getting started with Python and Eclipse is straightforward: Install Eclipse after which use the Eclipse Marketplace to install the Py Dev plug-in. Create a Python (or PyDev) module, and you're ready to begin growing your Python code. Of route, it is even simpler to run the Java code in Eclipse, due to the fact the default installation already includes aid for Java. And permit's now not forget all of the ancillary Java productivity enhancements: code crowning glory, code technology (getters, setters, constructors, and so forth.), refactoring, and so on. Regardless of your language preference or programming model (OO versus procedural or functional), there may be no denying that the usage of a contemporary IDE such as Eclipse is a chief productiveness enhancement. This type of device allows agile development inside the shape of code technology, refactoring, and device integration through plug-ins. One exciting element of a contrast among OO code in exclusive languages is the commonality between such languages. Python OO code isn't always hugely one-of-a-kind from equal code in Java. This will be taken into consideration a bonus of using OO capabilities inside the multi-language era, assisting programmers to provide good code. Simpler code is usually properly received through upkeep programmers and manufacturing guide team of workers.

7. Procedural Vs Object-Oriented programming

One desirable way of describing something new is to examine it with something vintage. Most atmospheric and oceanic scientists have had enjoy with procedural programming, so we'll begin there. Procedural packages look at the sector in terms of two entities, "records" and "capabilities." In a procedural context, the 2 entities are break away each different. A feature takes information as enter and returns statistics as output. Additionally, there's nothing customizable about a feature with appreciate to information. As a result, there aren't any boundaries to the use of a characteristic on numerous varieties of statistics, even inappropriately. In the actual global, however, we don't assume of factors or gadgets as having these functions (statistics and features) as separate entities. That is, real international items aren't(typically)simply data nor merely functions. Real global items rather have both "nation" and "behaviors." For example, human beings have nation (tall, brief, and so forth.) and behaviour (gambling basketball, going for walks, and so forth.), regularly both on the same time, and, of path, in the identical person. The goal of item-orientated programming is to imitate this in terms of software, so that "items" in software have two entities attached to them, states and behaviour. This makes the conceptual jump from actual-world to applications(optimistically)less of a leap and greater of a step. As a end result, we can extra easily enforce thoughts into instructions a laptop can apprehend.

8. On Going Work

In that context, our cause is to examine new models (and check standards for these fashions) or edition of classical ones that do not forget object-orientedness : practical and behavioural modelling. New features like encapsulation, inheritance, polymorphism, dynamic binding or generosity need to be addressed in such fashions. This first step in our work could be made without being concerned about the testing degree (unit testing, integration trying out, elegance trying out, ...) or the testing approach so that it will be carried out in step with the studied model (deterministic trying out, statistical checking out, ...).

In that path we are inquisitive about extension of nation-machines (for object-orientedness) in order that checking out techniques primarily based on kingdom-machine will be carried out, or extended. At the present time, work is made to use State charts in item-orientated layout .Typically, Object chart transitions correspond to country-changing strategies of a category and object attributes define Object chart states. With Object charts, you can described the inheritance relationships. This method addresses inheritance, dynamic changing, affiliation relationships. These tactics and the model added are very close due to the fact they are trying to deal with the equal things from the same graphical languages : finite-state machines. Furthermore, we are inquisitive about using item-orientated analysis and design methodologies in general .Then we are inquisitive about applying statistical testing to object-orientated programs due to the fact several case research have already confirmed the excessive fault revealing power of this method for procedural packages

We additionally need to cope with the problem of the programming language. Each object-orientated programming language (like Java and python) implements differently some capabilities of the item-orientated paradigm. Hence, the query of whether or not or now not checking out has to be distinct from one to another remains an open problem.

9. REFRENCE

1. Barbey et al. 1994] S. Barbey, M. Ammann and A. Strohmeier, “Open issues in testing Object-Oriented Software”, in Proc. of the European Conference on Software Quality, pp.25767, 1994
2. Barbey et al. 1996] S. Barbey, D. Buchs and C. Péraire, “A Theory of Specification-Based Testing for Object-Oriented Software”, in Proc. of the 2nd European Dependable Computing Conference, (Italy), pp.303-20, 1996.
3. Beizer 1990] B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, New York, 1990.
4. Binder 1994] R. V. Binder, “Design for Testability in Object-Oriented Systems”, Communication of the ACM, 37 (9), pp.87-101, 1994.
5. Binder 1996] R. V. Binder, “Testing Object-Oriented Software: a Survey”, Journal of Software Testing, Verification & Reliability, 6, pp.125-252, 1996.
6. Chow 1978] T. S. Chow, “Testing Software Design Modeled by Finite-State Machines”, IEEE Transactions on Software Engineering, SE-4 (3), pp.178-87, 1978.
7. Coleman et al. 1992] D. Coleman, F. Hayes and S. Bear, “Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design”, IEEE Transactions on Software Engineering, 18, pp.9-18, 1992.
8. Doong & Frankl 1994] R. Doong and P. G. Frankl, “The ASTOOT Approach to Testing Object-Oriented Programs”, ACM Transactions on Software Engineering and Methodology, 3(4), pp.101-30, 1994.
9. Duran & Ntafos 1984] J. W. Duran and S. C. Ntafos, “An evaluation of random testing”, IEEE Transactions on Software Engineering, SE-10 (4), pp.438-44, 1984. [Fiedler 1989] S. P. Fiedler, “Object-Oriented Unit Testing”, Hewlett-Packard Journal, pp.6974, 1989.
10. Harel 1988] D. Harel, “On Visual Formalisms”, Communications of the ACM, 31, pp.514-30, 1988.
11. Thévenod-Fosse & Waeselynck 1996] P. Thévenod-Fosse and H. Waeselynck, Towards a Statistical Approach to Testing Object-Oriented Programs, LAAS, (To appear in Proc. 27th Int. Symposium on Fault-Tolerant Computing (FTCS-27), (Seattle, USA), June 1997), N°96481, December 1996.

12. Thévenod-Fosse et al. 1995] P. Thévenod-Fosse, H. Waeselynck and Y. Crouzet, “Software Statistical Testing”, in Predictably Dependable Computing Systems (B. Randell, J.-C. Laprie, H. Kopetz and B. Littlewood, Eds.), ESPRIT Basic Research Series, Springer Verlag, 1995.
13. Turner & Robson 1993a] C. D. Turner and D. J. Robson, Guidance for the Testing of ObjectOriented Programs, Computer Science Divison, School of Engineering and Computer Science (SECS), University of Durham, Technical Report, N°TR 2/93, June 1993a.
14. Turner & Robson 1993b] C. D. Turner and D. J. Robson, “The State-Based Testing of ObjectOriented Programs”, in Proc. of the Conference on Software Maintenance, (I. C. S. Press, Ed.), (Los Almitos, California, USA), pp.302-10, 1993b.
15. Waeselynck 1993] H. Waeselynck, Verification de Logiciels Critiques par le Test Statistique, Doctoral Dissertation, Institut National Polytechnique de Toulouse, LAAS Report N°93.006, 1993.