A Performance Estimation Model for Software Testing Tools

Pramod Mathew Jacob, Prasanna Mani

Abstract*: *Software engineering has vast dimensions in the current world. Most of the engineering streams use software for making their tasks and processes easier. Internet of Things (IoT) based applications has much relevance in the current scenario. IoT based application programs are used for industrial automation, sensing, component calibration and various other real time applications. Though IoT uses both hardware and software, it is mandatory to ensure the quality of the developed product. Software developers follow various software testing techniques to validate and verify the application software they have been produced. Testing can be done manually or using automated testing tools. There are many software automation testing tools available with different functionalities and performance capabilities. Our proposed model suggests a method to choose which software automated testing tool is best suited for validating an IoT application. We have experimentally evaluated our model for various software testing tools. The experimental results prove that our model can be useful for project managers to choose the right software testing tool for a particular application.

Index Terms*: *Automated testing, Software testing, Test scripts, Testing tools, Internet of Things (IoT)

I. INTRODUCTION

Software industry plays a crucial role in the smart technology era. Most of the devices and objects are interconnected and is communicating which other using various communication protocols [1]. Internet of Things (IoT) is the key technology behind this revolutionary change. IoT is a network of various connected objects which performs a particular task. IoT system uses typical software architecture to address the various real time issues like interoperability, heterogeneity, scalability and security. The various software architecture includes client- server, peer – peer, publish – subscribe and Representational State Transfer (REST). These architectural patterns can be chosen based on various selection models [2]. Based on the selected model, application is modeled and developed. The test engineers take the responsibility to validate the product by applying various IoT test procedures [3].

Software testing [4] is a quality assurance procedure to identify the faults in a program. Software plays a vital role in IoT based systems and is mandatory to ensure that the developed system is performing its intended functionalities. This process of verification and validation can be done in this testing phase. For an IoT based system, testing is done at various levels like component testing, modular testing, integrated testing and finally system testing [5]. Alpha and acceptance testing is usually followed for an IoT based software product. Though testing is a time consuming herculean task, most of the industries prefer automated testing than manual testing. Automated testing not only efficient but also economical. Various software industries follow different categories of automated tools for verifying their developed product. Manual testing is preferred for User Interface based systems whereas automated is followed for web based and real time applications, The various process involved in automated and manual testing is illustrated in Figure 1.

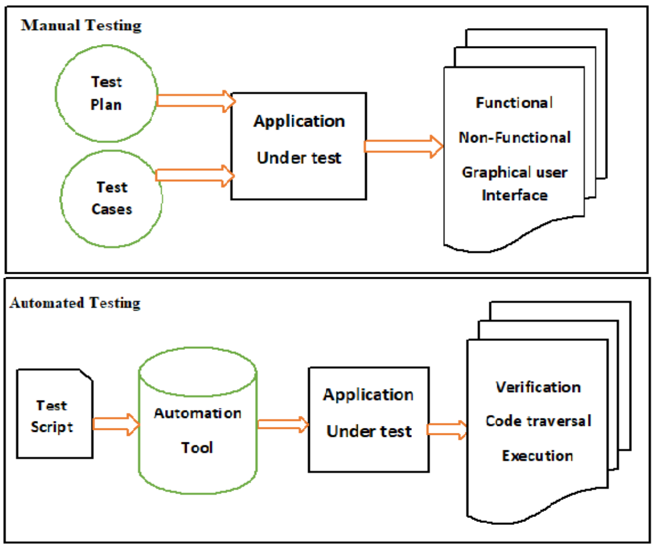

Figure 1: Process involved in automated and manual testing

Though IoT based applications are web based real time applications, automated testing tool can be used for better efficiency. But the critical challenge for a project manager is to choose the appropriate automated testing tool for a particular application. Our research work tries to resolve the following queries.

Q1: How can we choose the appropriate software testing tool for a particular application?
Q2: What are the various factors to be considered for choosing a best software testing tool?
Q3: How the efficiency and productivity of a test automation tool can be estimated?

Manuscript published on 30 June 2019.
***** Correspondence Author (s)
Pramod Mathew Jacob*, Assistant Professor in Providence College of Engineering, Kerala. Department of Computer Science & Engineering from Kerala University. India
Dr. Prasanna Mani, Associate Professor in Vellore Institute of Technology Vellore (Deemed to be University), Vellore, (Tamil Nadu) India

Though there are some models available for the same, they are incapable of answering all the above said research questions. We have analyzed the works done so far in this area and the summary is discussed in the next section.

II. RELATED WORKS

James B. Michael et al [6] evaluated various existing test metrics and modeled a new metric for choosing suitable software testing tool. They have derived various factors like user friendliness, maturity of the software tool, requirement of human intervention, management of testing tools and capability of generating test cases. Their model also consider the object oriented factors like maximum class structures that are allowed, tool response time and feature support count. All these factors are validated using LRDA test bed, Parasoft Test bed and the results are observed. Functional modeling and Unified Modelling language [7] are used for validating the requirements.

Pramod and Prasanna [1] have derived an IoT based testing model which comprises of various layers of testing strategies. This model consists of hardware and sensor level testing, communication layer testing, application layer testing and system level of testing.

Reetz et al. [8] proposed a new approach for testing IoT based application built on a code insertion methodology. It is derived using the semantic description of IoT based service. Their proposed architecture framework consists of test design engine and test execution engine as well as a sandbox environment, which has the capability of simulating the networking aspects of the system. Code insertion is done manually. However, this methodology lacks the validation of the hardware and physical things of an IoT based system.

Poston and Sexton [9] modelled a tool evaluation criteria which measures test quality, percentage of bugs identified etc. But their model fails to address tool easiness and time gor generating a test case.

Pawan Singh et al has derived some metrics for estimating the efficiency of testing tools [10]. They evaluated various factors like toughness of user interface, customer acceptance rate, tool age, end user convenience, tool correctness, functional metrics and coverage. The drawback of their work is, it requires some experience as well as needed some back history details of project being previously tested to estimate the values more accurately.

Rigzin Angmo and Monika Sharma has evaluated the open source web testing tools like Selenium and Watir [11]. They analyzed execution speed, easiness in using record and play back features, required number of steps for generating test cases and browser compatibility The focused on qualitative metrics rather than quantitative metrics..

Mustafa et al has classified the various available testing automation tools based on testing methodologies being used [12]. They have evaluated nearly 135 testing tools and categorized them over three types of software products (web application, application software, network protocol). Their analysis also points out which testing method has limited automation tools. But though there are many tools available for testing the same domain, this work fails to provide some evaluation criteria for choosing one among them.

T. E. J. Vos et al has proposed a framework for evaluating software testing tools and techniques which is successfully used for some case studies in EvoTest and FITTEST [13]. They derived metrics for evaluating Generated test cases count, Task applicability, Tool reliability, Test case coverage and many more. But this framework fails to evaluate the performance constraints of automated testing tools like test script generation time, test script evaluation time and much more. Our models evaluates script based parameters, user experience based metrics as well as performance parameters of a testing tool to make the tool selection much easier. The next section discusses about the basic terms related to software testing.

III. PROPOSED MODEL

Testing process is not same in all applications. Different companies follow different strategies and tools for similar software applications. Record and play back tools **[14]**, Tools which uses test scripts and other CASE (Computer Aided Software Engineering) tools are used for testing applications **[15]**. Our intention is to derive a test tool performance comparison model based on test scripts. Most of the IoT products consist of a web application and it is mandatory to generate test scripts for evaluating the various user functionalities. Test scripts are considered as our fundamental parameter for our performance evaluation.

Software metrics [16] is a mathematical value which conveys the degree of capability of some particular aspects of software. Software metrics are categorized into process metrics, product metrics and project metrics. Testing tool metrics can be considered as a project metric though the project managers evaluate tool performance and efficiency to choose the best testing tool appropriate for their project domain. Some existing works [17] discuss the performance of component based software systems. But those models fail to address the script based parameters of a software testing tool. Thus we formulate a set of metrics for evaluating script based parameters. The various testing tool metrics used in our metric model are classified into three categories as shown in Figure 2.

The various metrics in each category is discussed below.

1) Script Creation Time (SCT): It is the time taken to generate the script of an individual test case. It depends upon the complexity of the module or test case being tested, number of input fields being tested etc. SCT increases as either of the above said parameters increases.

$$SCT = SFT - SST \qquad (1)$$

SFT: Script_creation Finish Time

SST: Script_creation Start Time

2) Mean Script Creation Time (MSCT): It can be defined as the mean time to create test scripts. It is the average time taken to generate one test script. The tool with lesser MSCT value can be considered as an efficient tool.

$$MSCT = \frac{\sum_{i=1}^{n} SFTi - SSTi}{n} \text{ seconds/script} \qquad (2)$$

SSTi : Script_creation Start Time of i^th test script

SFTi : Script_creation Finish Time of i^th test script

n: Total number of scripts being created

3) Script Execution Time (SET): It is the time taken to execute a generated test script. SET value should be minimum for a good test automation tool.

SET = EFT – EST (3)

EST: Execution Start time

EFT: Execution Finish time

4) Mean Script Execution Time (MSET): It can be defined as the mean time to execute test scripts. It is the average time taken to execute a single test script. Tool with lesser MSET value is considered best.

$$MSET = \frac{\sum_{i=1}^{n} EFTi - ESTi}{n} \text{ seconds/script} \quad (4)$$

ESTi : Execution_script Start Time of ith test script

EFTi : Execution_script Finish Time of ith test script

n: Total number of scripts being executed

5) Mean Script Turn-around Time (MSTT): It is the mean time required for a tool to create and execute a test script for a particular test case. Tool with minimal MSTT value is preferably good.

$$MSTT = \frac{\sum_{i=1}^{n} SCTi + SETi}{n} \text{ seconds/test case} \quad (5)$$

6) Tool Productivity Rate (TPR): It can be defined as the number of test cases executed per hour. For a good testing tool, TPR value should be higher.

TPR =3600 / MSTT test cases/hour (6)

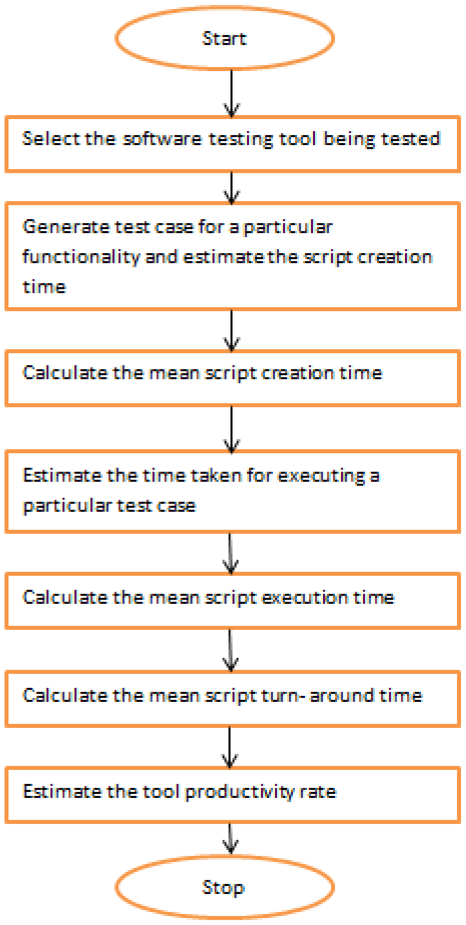
Start
Select the software testing tool being tested
Generate test case for a particular functionality and estimate the script creation time
Calculate the mean script creation time
Estimate the time taken for executing a particular test case
Calculate the mean script execution time
Calculate the mean script turn- around time
Estimate the tool productivity rate
Stop

Figure 2: Steps of Proposed Metrics Model

We need to derive some script based performance metrics for evaluating the tool performance, though the model focuses on script based testing tools. Some basic metric can be time taken for generating test scripts, time for executing test scripts etc. Our proposed pseudo code for estimating script based metrics is illustrated below. It can be applied on a selected module of a software project. A software module is a set of functions or methods desirably possessing high cohesiveness. Each module may perform a set of functionalities. Test cases should be derived for validating the functional behaviour.

Pseudo code for estimating the script based metrics

```
Script_metrics_estimator()
 1.  INPUT Number of test tools as T
 2.  FOR all T
             CALL Test_script_generation()
             CALL Test_script_execution()
     ENDFOR

Test_script_generation()
 1.  INPUT No: of testcases as N
 2.  FOR all TESTCASE_ID 1 to N
         SCT = Generate_testscript(TESTCASE_ID)
         TOTAL_SCT= TOTAL_SCT + SCT
     ENDFOR
 3.  COMPUTE MEAN_SCT = TOTAL_SCT / N

Test_script_execution()
 1.  INPUT No: of testcases as N
 2.  FOR all TESTCASE_ID 1 to N
         SET = Execute_testscript(TESTCASE_ID)
         TOTAL_SET= TOTAL_SET + SET
     ENDFOR
 3.  COMPUTE MEAN_SET = TOTAL_SET / N

Generate_testscript(TESTCASE_ID)
 1.  Initialize START_TIME = CURRENT SYSTEM TIME
 2.  RUN test script creation process
 3.  IF (Test script creation = 'Successful') THEN
             SET FINISH_TIME = CURRENT SYSTEM TIME
     ELSE
             PRINT 'Unable to generate test script'
     ENDIF
 4.  COMPUTE SCRIPT_CREATION_TIME = FINISH_TIME – START_TIME
 5.  RETURN SCRIPT_CREATION_TIME

Execute_testscript(TESTCASE_ID)
 1.  Initialize START_TIME = CURRENT SYSTEM TIME
 2.  RUN test script execution process
 3.  IF (Test script execution = 'Successful') THEN
             SET FINISH_TIME = CURRENT SYSTEM TIME
     ELSE
             PRINT 'Unable to execute test script'
     ENDIF
 4.  COMPUTE SCRIPT_EXECUTION_TIME = FINISH_TIME – START_TIME
 5.  RETURN SCRIPT_EXECUTION_TIME
```

The above pseudo code estimates the Script Creation Time (SCT) and Script Execution Time (SET) for various test cases. From these parameters we can derive Mean Script Turn-around Time (MSTT), Mean Script Creation Time (MSCT) and Mean Script Execution Time (MSET) for various tools. From that we can derive metrics for tool productivity, tool reliability etc. Our model becomes hybrid by estimating metrics for tool learning time.

IV. EXPERIMENT EVALUATION AND RESULTS

We have applied our testing tool metrics on two software testing tools which supports automated script generation. We refer the tools as TOOL ABC and TOOL XYZ respectively. We have generated a test suite with five test cases. We have noted down the script creation start time, script creation finish time, script execution start time and script execution finish time. We entered this values in tables (Refer Table 1 and Table 2) to derive Script Creation Time (SCT), Mean Script Creation Time (MSCT), Script Execution Time (SET) and Mean Script Execution Time (MSET). The time shown in tables is in 24-hour format.

Table 1. Script Creation Time of TOOL ABC & TOOL XYZ

Tool id	TOOL ABC			TOOL XYZ		
Test case_id	SST HH:MM:SS	SFT HH:MM:SS	SCT (In Seconds)	SST HH:MM:SS	SFT HH:MM:SS	SCT (In Seconds)
1	11:40:12	11:42:53	161	02:13:10	02:15:03	113
2	12:21:13	12:25:32	319	02:17:12	02:20:55	223
3	10:22:15	10:24:16	121	03:10:05	03:11:08	63
4	09:00:01	09:00:59	58	11:12:45	11:14:51	126
5	09:50:13	09:54:15	242	12:59:32	13:03:35	183
	Total Script Creation Time of ABC		901	Total Script Creation Time of XYZ		708

Table 2. Script Execution Time of TOOL ABC & TOOL XYZ

Tool id	TOOL ABC			TOOL XYZ		
Test case_id	EST HH:MM:SS	EFT HH:MM:SS	SET(In Seconds)	EST HH:MM:SS	EFT HH:MM:SS	SET(In Seconds)
1	12:40:20	12:41:10	50	02:33:18	02:34:03	45
2	13:15:18	13:17:01	103	12:32:12	12:33:21	69
3	14:22:10	14:22:23	13	13:19:45	13:20:08	23
4	09:30:11	09:30:58	47	04:18:45	04:19:11	26
5	09:50:28	09:51:03	35	14:59:32	15:00:03	31
	Total Script Execution Time of		248	Total Script Execution Time of		194

Table 3. Derived metric values for TOOL ABC and TOOL XYZ

Metrics	Unit	TOOL ABC	TOOL XYZ
Mean Script Creation Time (MSCT)	seconds / script	180.2	141.6
Mean Script Execution Time(MSET)	seconds / script	49.6	38.8
Mean Script Turn-around Time (MSTT)	seconds / test case	229.8	180.4
Tool Productivity Rate (TPR)	test-cases / hour	15.66	19.95

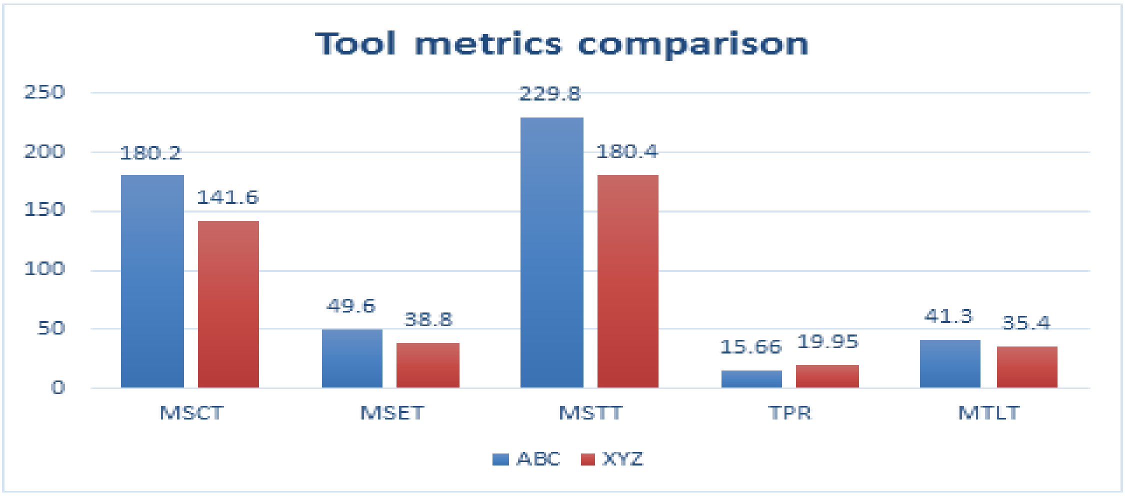

Figure 3: Performance evaluation of TOOL ABC and TOOL XYZ

The proposed metrics model is applied to the testing tools to derive various tool performance metrics. The graphical analysis of various performance metrics is portrayed in Figure 3. From the evaluation graph it is easy for the project manager to trade-off between various testing tools that can be applicable for the project domains. If they need a testing tool with lesser training time, then the project manager can choose a test automation tool with minimum MTLT value. If they require a tool with better productivity, then project manager can choose a tool with high TPR value. So this performance evaluation model can be used as a reference model for project managers or test engineers to choose a suitable automation tool.

V. CONCLUSIONS

Software plays a vital role in the development of an IoT based system. Most of the hardware components are remotely controlled and managed with the help of a software tool. So it is mandatory to verify and validate the developed software product. We uses automated testing tools to verify the developed product. Though there are many similar testing tools available in the market, it will be difficult for the project manager to choose the best one for their particular application. This crisis can be managed by our proposed model . Our model analyses the various parameters of a script based testing tools and the comparative analysis may help us to derive the best tool for testing a particular application. Our experimental analysis claims that this model can be used for choosing a best software testing tool from many. Our future work may focus on using the concepts of machine learning to choose the best suitable testing tool for an application.

ACKNOWLEDGMENT

We express our sincere gratitude to Providence College of Engineering, Chengannur, Kerala and Vellore Institute of Technology, Vellore, Tamil Nadu for the support provided. We are thankful to Mr. Hariharan R L and Ms. Jisha Elizabeth Mammen for the technical support provided in executing test cases.

REFERENCES

1. P. M. Jacob and P. Mani, “A Reference Model for Testing Internet of Things based Applications”, *Journal of Engineering, Science and Technology (JESTEC)*, Vol. 13, No. 8 (2018) ,pp. 2504-2519.
2. P. M. Jacob and P. Mani, "Software architecture pattern selection model for Internet of Things based systems," in *IET Software*, vol. 12, no. 5, pp. 390-396, 10 2018. doi: 10.1049/iet-sen.2017.0206
3. B. Beizer, “Software Testing Techniques”. London: International Thompson Computer Press, 1990.
4. Glenford J. Myers, “The Art of Software testing”, Second Edition, Wiley India Edition.
5. B. Beizer, “Black Box Testing”, New York: John Wiley & Sons, Inc., 1995.
6. J. B. Michael, B. J. Bossuyt and B. B. Snyder, "Metrics for measuring the effectiveness of software-testing tools," *Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on*, 2002, pp. 117-128. doi: 10.1109/ISSRE.2002.1173225
7. P. M. Jacob, Muhammed Ilyas H, J. Jose and J. Jose, "An Analytical approach on DFD to UML model transformation techniques," *2016 International Conference on Information Science (ICIS)*, Kochi, 2016, pp. 12-17.
8. Reetz, E.S.; Kuemper, D.; Moessner, K.; and Toenjes, R. (2013). How to test iot-based services before deploying them into real world. *Proceedings of the 19th European Wireless Conference.* Guildford, United Kingdom, 1-6.
9. I. Altaf, J. A. Dar, F. u. Rashid and M. Rafiq, "Survey on selenium tool in software testing," *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*, Noida, 2015, pp. 1378-1383. doi: 10.1109/ICGCIoT.2015.7380682
10. R. Angmo and M. Sharma, "Performance evaluation of web based automation testing tools," *Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference* -, Noida, 2014, pp. 731-735. doi: 10.1109/CONFLUENCE.2014.6949287
11. Van Vleck, T., “Three Questions About Each Bug You Find”, ACM Software Engineering Notes, vol. 14, no. 5, July 1989.
12. SriivasanDesikan, Gopalaswamy Ramesh, “Software Testing: Principles and Practices”, Pearson.
13. Pramod Mathew Jacob and M. Prasanna, “A Comparative analysis on black box testing strategies,” International Conference on Information Science – ICIS –’16, Kochi, India, 2016
14. Rajib Mall, “Fundamentals of Software Engineering”,Third edition, PHI
15. http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm
16. Ian Sommeriele, “Software Engineering”, Addison Wesley.
17. Pressman, “Software Engineering –A Practitioner’s Approach”.
18. http://www.softwaretestingclass.com/software-testing-life-cycle-stlc/
19. http://istqbexamcertification.com/what-is-software-testing-life-cycle-stlc/

20. http://www.softwaretestingclass.com/automation-testing-vs-manual-testing/
21. https://www.infosys.com/it-services/validation-solutions/white papers/documents/choosing-right-automation-tool.pdf
22. http://www.softwaretestinghelp.com/software-test-metrics-and measurements/
23. https://en.wikipedia.org/wiki/HP_QuickTest_Professional
24. https://en.wikipedia.org/wiki/Selenium_(software)
25. http://www.seleniumhq.org/download/
26. https://saas.hpe.com/en-us/download/uft
27. https://peopleorbit.vit.ac.in/hr_login.asp
28. Harpeet Kaur, Gagan gupta, "Comparative study of automated testing tools: Selenium, QTP and Test Complete" Journal of Engineering Research and Applications www.ijera.com ISSN : 2248-9622, Vol. 3, Issue 5, Sep-Oct 2013, pp.1739-1743

AUTHORS PROFILE

Pramod Mathew Jacob has completed his B. Tech in Computer Science & Engineering from Kerala University. He possesses M. Tech in Software Engineering from SRM Institute of Science and Technology Chennai. He is working as Assistant Professor in Providence College of Engineering, Kerala. Presently he is pursuing PhD in Vellore Institute of Technology Vellore. He has a teaching experience of 5 years and Research experience of two years. He has published 8 papers in various international journals and conferences. His areas of interest include Software Engineering, Software Testing, and Internet of Things.

Dr. Prasanna Mani has completed his MS in Computer Science from Anna University. He received his doctorate in Software Engineering from Anna University. Presently he is working as Associate Professor in Vellore Institute of Technology Vellore (Deemed to be University), Vellore, Tamil Nadu. He has published nearly 25 papers in various national and international journals. He is guiding Research scholars in the area of Software testing and is an eminent reviewer of various international journals. He also authored a book for cracking interview questions of C programming. His area of interest includes Software Engineering, Software Testing, Internet of Things etc.