

**Investigation of software testing within multi layered architecture based debugging application**

by

Owen Skilton

Project submitted as part of the requirements for the award of MSc (Hons) Software Engineering (Web Development)

September 2021

Introduction

Within modern day software development teams, a tremendous amount of time is spent on the process of maintaining and debugging code. With the plethora of testing tools available to a developer its difficult to find the most efficient one to operate within that specific environment under those specific conditions. With the shift towards a multi-layered approach for almost every popular software package, a massive amount of time and money is spent investing in the testing and debugging approach. Minimizing this time deficit and attempting to maximise the time spent on applicable coding should be the goal of any organisation.

Achieving this goal is not as simple however. With current code maintenance timing trends similar to those in the 70s, [1] [2] it is questionable how with the evolution of software development as a whole has made little to no change in this overhead. With the number of microservices utilised within modern day projects combined with the multiple environments and languages spread across an application, it is a massive time investment and undertaking to ensure that all these dependencies have sufficient coverage in terms of their tests. It is incredibly difficult for a developer to monitor all the changes across multiple services and maintain a functioning application with such a large number of factors impacting the slightest change to these dependencies.

Automated testing is the most efficient and popular method for monitoring this, but with the sheer number of tools available, an evaluation on the ease of use and time investment to create these testing scripts needed can help minimize “dead time” spent maintaining and debugging code in the future.

My aim for this project is to analyse the most popular testing utilised by organisations such as the DVLA and provide a critical analysis on the ability and application as compared to similar tools available. The tools selected will be focused on the interaction of a multi-layer architecture based environment with an emphasis on abstraction within the microservices utilised. I will do this by creating an error based search engine that is able to scan through the most popular programming message boards and documentation to try and find the most accurate solution for a developers’ specific problem. This is to help find a solution to the problem of a developer’s time being spent more on debugging and maintenance, rather than creating tangible code that impacts an application. The creation of this application both helps to provide a solution to the debugging time investment as well as providing a platform in order to perform this analysis of the tools available to help it function.

Literature Review

Within the subject of testing there exist multiple different type of test that can be run. The most basic of which is known as unit test. The smallest piece of code that can be isolated within a program that can be tested in isolation. [3] These are usually only a few lines long and test for basic logic and data manipulation that return a quantifiable object. A step up we find integration testing, usually a combination of unit tests that form some form of interaction or the interaction itself of services within our application. [4] Finally, there is regression testing, usually a simulation of a user’s journey through an application and the branching paths able to be taken or the different outputs able to be achieved. [5] Integration testing is the largest and most time-consuming aspect but serves as an indication of the usability of the application.

The purpose of these testing practices is to ensure that the functionality created matches what was required from the customer. Testing attempts to minimize the down time during production by minimizing debugging efforts during the creation or editing of features. The higher the coverage of tests able to be created, the more confidence that exists within the quality assurance of the product being developed.

During the planning period for any project, the business and design teams will agree on a software design methodology that they will incorporate into a products development. It is important for the two teams to come to agreement on the methodology selected as it has large bearing on the delivery time estimation for a project as well as the approaches available during the development lifecycle. This has a profound effect on the approach and tools available to be used.

There are multiple methodologies that are used in industry today, but the most common of which is Agile. [6] Within Agile there are a multitude of different types of practice and methodologies again to select from. Selection is dependent on the project management’s decision on how best to implement an agile based development approach as well as the preferred method of execution and formation of the stories being created. Regardless on what development methodology is implemented, it will influence the type of testing approach that will be taken by the development team. 3 popular practices that are impactful on the testing approach for a product are:

* Testing-driven development (TDD)
* Domain-driven development (DDD)
* Behaviour-driven development (BDD)

Testing-driven development relies on software requirements being phrased in terms of test cases to be created before software is begun to be developed. This means that the tests that will be used to ensure code is implemented correctly will be created first. The developer will then implement the code changes to the software to pass these new test cases, this normally will be in in the simplest form and no code will be added beyond the tested functionality, after which the feature / edit can be considered completed and added. [7] A positive of a TDD approach ensures that the developer focuses on the requirements to the customer before making edits to the code. This creates incredible streamline and specific changes at a time. [8] However this approach is an enduring process as all requirements needed to be ironed out before any development can begin as well as during any new addition the current tests must be maintained and updated to enable new criteria.

Domain-driven development focus on the formation and language of the software, and therefore its tests, revolve around the business-based domain. This entails that classes and events represent business objects that wish to be implemented such as firstApplication or refundEvent. This is especially useful in complex domains where a multitude of models and events exist in the same space allowing for easier understanding of logic flows through a project. In order to achieve this though a large amount of encapsulation is required to ensure that the models created are accurate to the business domain specification. [9]

Behaviour-driven development can be considered a combination of both TDD and DDD. The idea involves that the software development process should involve both the business and technical interest. BDD specifies that test are to be created in terms of the desired behaviour specified by the business domain. BDD also facilitates the collaboration of developers and business analysts for the creation of user stories which follow a specific narrative of the specified role being impacted by the story and the outcome that is requested. [10] This is then converted into testing acceptance criteria known as scenarios which follows the format of GIVEN: an initial condition, WHEN: An action taken that triggers and event, THEN: the required outcome of the story. Test are formed using these scenarios using what is known as domain-specific languages. These are languages that are written in English readable sentences that follow the acceptance criteria format. [11] One tool that is incredibly popular is cucumber which utilise gherkin as a language parser to the ruby code. It is commonly used to test UI interactions and events.

Ultimately the testing methodology that a team or individual decides to integrate into their development process will have a large restriction on the type of tools available to be utilised thus it is important to critically evaluate the most efficient for a particular project in order to maximise the potential delivery time and minimize overhead.

For a development methodology to be selected, the business and development team need to have the project architecture defined first. Without a concrete decision being made on the architecture, the two would not be able to select the most appropriate methodology without understanding the structure of the project. During this definition phase of the project an architect will decide on the software tiers approach they will take, non-layered or multi-layered.

Non-layered architecture simple refers to an application where all the services and components exist entirely on one server. Both the backend data persistence and logic manipulation exist within the same location as the front-end page responses. [12] There are some positives to this architecture design, mainly it is a very cost-effective measure for running a web application for small scale interaction. Server cost is minimized as well as effective resource utilization in terms of storage capacity. [13] A negative of this single server utilisation can be an availability issue if the server is down. None of the services hosted on it, if they are used by other external applications, can be accessed while down as well as the entire application and the data stored it itself. A security risk could be exposed if sensitive data is being persisted as a single attack to this server could expose all the personal information stored whereas abstraction of these layers would ensure a breach to one service would not impact the others.

Multi-layered architecture refers to the software separation of a products service and features. The most common multi-layered architecture is the 3-tiered architecture. This is the abstraction of the presentation, Application and data management layers both functions and functionality. [14] This separation is created in order to allow greater freedom to editions within each layer without impacting other services or functionality. These layers are defined as follows:

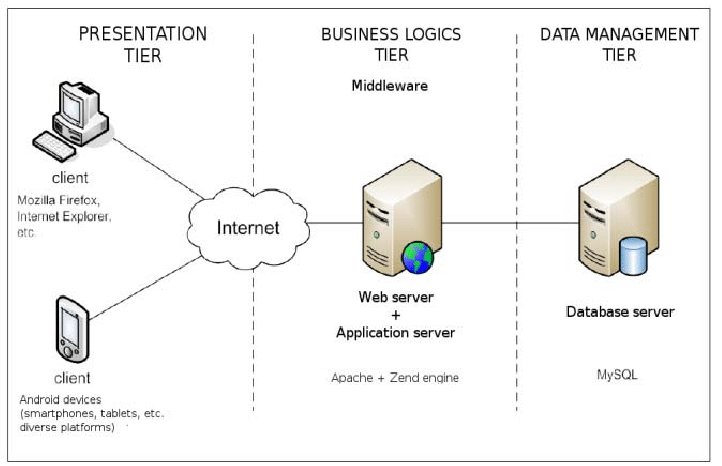
* The presentation layer:
  + The topmost layer through which the GUI is represented, the user interacts though any valid terminal which displays the content usually though a web browser or application. This is normally done through HTML and CSS with a combination of JavaScript for some simple logical operations. The presentation layer will communicate with the application layer to pass information through methods such as HTTP requests. [15]
* The application layer:
  + This layer acts as the middleman that receives data from the presentation layer form the user and applies the business logic rules responsible for providing the functionality to the application. The completed manipulated data may either be passed back to the presentation layer either in the response to the HTTP request or pass the data through to the management layer for persistence. [15]
* The data management layer:
  + The lowest level of the traditional 3 tier architecture. This layer is responsible with the storing and retrieval of data concerning to the application. Data can be stored on any form of device or media that can facilitate storage as long as mechanisms are in place to facilitate storing and retrieving such as APIs. The application layer will utilise the appropriate API when storing or retrieving data from the data storage device such as a server by using HTTP requests. [15]

The abstraction of the application layer creates our multi-tiered architecture. By separating the environments for all the varying services and products it allows more flexibility in terms of functionality and scalability without creating dependency conflicts with other interactions within the application. [16] The reason for the rise in popularity of multi-layered is down to this ability to make edits to these microservices to be as simple as possible. The most appropriate application for this however involves products that require faster network transmission minimizing traffic overheads as well as application with multiple services dependencies that can be spread across multiple servers in order to achieve this increased performance.

Within each layer of a multi-tiered application there exists specific testing tools that can cover the different types of testing branches and languages available to each layer. For example, a HTML and JavaScript based GUI would test if certain components were rendered to the screen successfully after a user action. A useful tool for testing UI interaction is Selenium, an open-source automated testing framework used to validate web applications within a browser. Multiple languages can be used create selenium test scripts making it incredibly flexible across applications. [17]

Multiple testing tools are able to be utilised within the application layer for different functions, some are framework specific to test syntax formation and linting such as RSpec within a ruby on rails-based framework, a behaviour driven tool used to lint ruby code. [18] Others involve tools such as Cucumber are used to formulate acceptance criteria tests within a ruby on rails application. [19] Dependent on the software selected for the application layer will largely dictate the tools available to a developer to adequately cover both unit and integration-based tests.

Finally, testing the data management layer involves either establishing a connection and disconnecting from the database itself, or testing the API channels and repositories accessible through the application layer. An example would be testing the response from a spring-based repository using a Java testing library known as REST-assured. It is commonly used to test JSON based responses form API calls involving the methods GET, PUT, POST, PATCH, DELETE. [20]



*Figure 1, Example of 3-tiered architecture approach* [21]

All these different types of tests that need to be created in accordance with the framework used, combined with a scripting tool usually formed in a script-based environment is incredibly time consuming. What if the product owner wants a small application that utilises a multi-layered application but with minimal number of features and only doing so to maximise security? Is the investment into an automated test environment worth it? At what point should a project skip the investment and relay solely on manual based tests?

In simple terms, from an economical point of view, it makes sense to adapt to an automated test based environment only when the cost of automation equates to less than the cost for the equivalent coverage of a manual approach for executing the same tests, the same number of times, that the automated test would execute during the developments life time. [22]

The cost of manual based testing is difficult to quantify. It can be argued this can either be the salary per hour of a developer or tester that would have to execute this or a base rate for the time spent during the development period that will impact the product delivery date. With most projects working under a methodology that focuses on adaption and fluidity the number of instances that a test will be run can be difficult to estimate. If the customer request edits to certain features this can dramatically impact the cost evaluation for the use of manual testing which is why most organisation will utilise an automated approach on as many cases as possible in order to both ensure complete coverage, and the inevitability of changing requirements has a minimizing impact on delivery times. The inverse can be applied however for a case involving automation testing. If a change of requirements is requested to certain features than that would make multiple testing scripts obsolete meaning the time spent creating them is ultimately wasted. Rudolf Ramler and Klaus Wolfmaier performed an in-depth analysis on the cost balancing efforts involving the use of manual vs automated testing. [23] They introduce the concept of opportunity cost, instead of the focus on minimizing the overall cost, shifting the focus to maximizing the benefits of testing involving the 2 approaches. By estimating the benefit on a test case in regard to its contribution with risk mitigation for the specific component, it provides a more honest representation between the 2 alternatives.

Similar studies have been performed that specifically analysis the tools available for application layer-based testing. More specifically the microservices within an application layer that facilitates functionality to a product. Juan P. Sotomayor et al [24] compared the run time for multiple testing tools within a riding sharing based application specifically for regression and integration-based tests. The purpose of this investigation was to find the most efficient testing tools that possess the same coverage on an identical terminal. The utilised multiple tools that span across multiple languages and testing methodologies but the most common were Java with a HTTP request-based method for unit and integration-based testing. Specific case studies performed within the investigation found some tools within a different framework were found to be over 18,000% more inefficient while others only had a discrepancy of 5%. This investigation however focuses strongly on the efficiency-based aspect of execution of these tests rather than the creation and useability of the tools presented and no mention of the method to improve the testing process other than the terminal used to execute them.

Another case study to analyse is the one performed Bharat B Konka [25] that analyses the testing tools and method utilised within Volvo and specifically how the introduction of a new safety standard as effected the companies’ testing practices. This is predominately done through the interviewing of employees that span across 2 departments and 6 different groups within Volvo who consist of both software developers and researchers working within different methodologies defined as “structured” to “Ad hoc”. They found varying results for a structured based methodology that consist of multiple of test-driven development work. What is surprising that that some groups choose not to utilise regression or unit-based testing. Half of the projects investigated found that no software debugging tools were utilised. This shows that within an automobile domain setting, an incredible fast-growing sector for software development, there as a multitude of ways to achieve more efficient manners of development but that some teams are able to function with an unconventional approach.

A management report conducted in 1976 [1] found that surveying Great Britain found that 40% of time spent during development is on code maintenance. This is understandable considering the time this report was created. Within an era of much inferior technology in terms of execution times and testing tools available. However, a survey conducted by Stripe in 2018 [2] concluded that during an average 40 hour working week, a developer will spend 42% of time allocated on maintenance alone, either in terms of technical debt or remedying bad code. With the tools available to the modern-day developer how has this figure remained constant? Testers today can not only automate the testing performed but as well the automation for error logging and discovery of bugs. A hypothesis to this could be down to the time spent to form the scripts to execute these tests are not necessarily worth the time saved running them manually. Another could be that software today contains an unbelievable amount of microservices that require maintenance and constant editions to adapt to the ever-changing specification of a product.

It is estimated that the global cost for software development is over $1 trillion USD. A quarter of which is estimated to be spent on the wages for the programmers themselves. [26] An estimation by the university of Cambridge during an investigation into the time saving achieved by utilising a Reverse Debugging tool found that developers spend almost half of their time debugging software during their development based activities. This equates to over £150 billion dollars alone being spent on the process of debugging errors within and application.

With the growing rise of software development across the world, companies and entities within an invested interest will attempt to create tools in an attempt to decrease this deficit and increase overhead and delivery times for future projects. My goal is to create a development tool that can speed up this debugging process and investigate the different options during the development process to the testing tools available for each aspect of the architectural design. Any percentage of time that a developer is able to focus more on tangible software creation rather than the debugging will directly equate to a more streamlined project and thus a decrease in overhead.

Project plan

To help attempt to decrease the time spent debugging and increase productivity, as well as investigate the testing tools available during the development period, I plan to create a tool that is able to be used alongside standard development practices that is able to facilitate as an error message based search engine. The tool will utilise existing APIs for the most common error solution based sites as well as the documentation for that specific language if applicable to find the most applicable solution to the developer’s problem. The aim to is to point the developer in the direction of a solution as fast as possible to minimize the down time for debugging. A focus during development will be around the application of the testing tools within a multi-layered based environment and an analysis on the usability and appropriateness of the tools utilised. In order to be critical in the tools used I plan to use a combination of BDD and TDD for the developmental process as I believe that these to provide the best support in the utilisation and analysis of the testing tools being scrutinised.

The project will exist within a 3 layered based approach with the presentation being focused around react based framework due to its dynamic responsiveness and the multitude of libraries that facilitate the testing of component based rendering and functionality such as Jest. This will be combined with ruby on rails application layer to apply the business rules and help enable the API requests to the data management layer within a spring based framework. The main features to achieve during this investigation to enable the testing functionality an account based system, users are able to keep track of previous errors searched and favourite pages. Search engine based feature specifying a language and error message and returning the most similar solutions. Voting system for the responses from the API calls to help find the most useful solutions to specific scenarios. A comment based system to enable user interaction on error solutions. Finally, a documentation dictionary that is able to help developers find documentation more rapidly, for example, if a developer wanted to know the functions available to data structures within ruby, they would be directed to the correct documentation and page index to help speed up the learning / development process.

I plan to use multiple testing tools within each layer of the project, for example the presentation layer will revolve around jest for component unit testing with react exclusively but will also utilise selenium to simulate user journeys within the UI. This will be combined with cucumber within the ruby layer to automate the business manipulation of the data and interact with the selenium based tests as well as its own independent capybara based acceptance tests. And finally, the spring layer will use java based libraries such as Junit and JSONassert for unit based testing and JSON assertions. At each stage a linting tool will be used such as rubocop, chutney, ESLint. A finally overall tool for the project is SonarQube. SonarQube provides feedback on code inspection for smells, bugs and security vulnerabilities as well as providing a statistical analysis based approach on testing in the form of automatic reviews.

An ideal future for the project would be a tool that is able to be linked to individual projects such as drone, another tool I plan to use to help automate regression testing, that is able to track and monitor developers that receive similar error codes. An example would be when services are updated or API keys get altered, these can sometimes exist in untrackable git files meaning the relay of information of changes has to be done manually. A user on the project would be able to search through the projects error code history and view that multiple 403 errors have been issued recently and a comment system within the search engine for that project would be able to point the developer on where to acquire the new data from. This is a very specific example but an ideal scenario for the application is to help speed up the software development process by helping share solutions to similar problems. Doing so will help save time during a project’s lifetime and consequently, help reduce the costs involved.

References

1. E. B. Swanson. The Dimensions of Maintenance. 2nd international conference on Software engineering. 1976, p. 492-497.
2. Stripe. The Developer Coefficient. [Online] Available from: <https://stripe.com/gb/reports/developer-coefficient-2018> [Accessed 07 Sep 2021]
3. D. Vrandečić, A. Gangemi. Unit tests for ontologies. OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Springer, Berlin, Heidelberg. 2006, p. 1012-1020.
4. H. Yuan, T. Xie.Substra: A framework for automatic generation of integration tests. International Workshop on Automation of Software Test. 2006, p. 64-70.
5. M. Pezze, M. Young. Software testing and analysis: process, principles, and techniques. John Wiley & Sons. 2008.
6. R. Hoda et al. Agile project management. New Zealand computer science research student conference. 2008, Vol. 6, p. 218-221.
7. K. Beck. Test-driven development: by example. Addison-Wesley Professional. 2003.
8. J. W. Wilkerson, J.F. Nunamaker, R. Mercer, R. Comparing the defect reduction benefits of code inspection and test-driven development. IEEE transactions on software engineering, 2011, 38(3), p. 547-560.
9. D. Thomas, B. M. Barry. Model driven development: the case for domain oriented programming. Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. 2003, p. 2-7.
10. C. Solis, X. Wang. A study of the characteristics of behaviour driven development. 37th EUROMICRO Conference on Software Engineering and Advanced Applications. 2011, p. 383-387.
11. A. Van Deursen, P. Klint. Domain-specific language design requires feature descriptions. Journal of computing and information technology. 2002, 10(1), p. 1-17.
12. J. Levinson. Understanding Application Architecture: An Overview. Building Client/Server Applications with VB. NET: An Example-Driven Approach. 2003, p. 1-22.
13. Bitnami. Single-Tier Vs. Multi-Tier Architecture. [Online] Available from: <https://docs.bitnami.com/google-templates/singletier-vs-multitier/> [Accessed 07 Sep 2021]
14. O. Bimbe, L.M. Encarnacao, A. Stork. A multi-layered architecture for sketch-based interaction within virtual environments. Computers & Graphics, 2000, 24(6), p. 851-867.
15. A. Aarsten, D. Brugali, G. Menga. Patterns for three-tier client/server applications. Proceedings of Pattern Languages of Programs. 1996, 4(6).
16. M .Richards. Software architecture patterns. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Incorporated. 2015, Vol. 4.
17. Selenium. Selenium. [Online] Available form: <https://www.selenium.dev/> [Accessed 04 Sep 2021]
18. Rspec. Behaviour Driven Development for Ruby. [Online] Available from: <https://rspec.info/> [Accessed 07 Sep 2021]
19. Cucumber. Tools & techniques that elevate teams to greatness. [Online] Available from: <https://cucumber.io/> [Accessed 07 Sep 2021]
20. Rest-assured. Rest-assured. [Online] Available from: <https://rest-assured.io/> [Accessed 07 Sep 2021]
21. S. Caballé, F. Xhafa, J. Raya, L. Barolli, K. Uchida. Building a software service for mobile devices to enhance awareness in web collaboration. International Conference on Intelligent Networking and Collaborative Systems. 2014, p. 369-376.
22. [C. Schwaber, M. Gilpin. Evaluating automated functional testing tools. Forrester Research.](http://www.programmersparadise.ca/content/images/promotions_2007/ibm/PP_forrester_rft_eval_0205.pdf) 2005.
23. R. Ramler, K. Wolfmaier. Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. International workshop on Automation of software test. 2006, p. 85-91.
24. J. P. Sotomayor, S. C. Allala, P. Alt, J. Phillips, T.M. King, P. J. Clarke. Comparison of runtime testing tools for microservices. 43rd Annual Computer Software and Applications Conference. 2019, Vol. 2, p. 356-361.
25. B. B. Konka. A case study on Software Testing Methods and Tools. 2012.
26. T. Britton, L. Jeng, G. Carver, P. Cheak, T. Katzenellenbogen. Reversible debugging software. Judge Bus. School, Univ. Cambridge, Cambridge, UK, Tech. Rep. 2013.

**SEARCH ENGINES!!!!**

One side of application crawls stack overflow and other error message sites + enable own crawler that is able to crawl documentation sites in general and reject its own webpages and perform its own analysis on each retuned result to decide whether applicable to searched key words:

* <https://www.youtube.com/watch?v=AFigoBtWNQ0>
* Google custom search engine returns a large amount of web pages. Filter through internally and decide on most applicable result based on accuracy to key word searched?

History of search engines / crawlers:

* Historically measured using fixed search queries and then a panel of people judge the accuracy of the results:
  + [1,3,4,5,8,9]
  + <https://www.cs.uic.edu/~liub/searchEval/SearchEngineEvaluation.htm>:
  + 1. J. Bar-Ilan, Methods for measuring search engine performance over time, Journal of the American Society for Information Science & Technology, 53(4), p.308-319, 2002.
  + 3. H. Chu, and M. Rosenthal. Search engines for the World Wide Web: a comparative study and evaluation methodology. Proceedings of the 59th annual meeting of the American Society for Information Science, 1996.
  + 4. M. Gordon and P. Pathak, Finding information on the World Wide Web: the retrieval effectiveness of search engines, Information Processing and Management: an International Journal, 35(2), p.141-180, March 1999
  + 5. D. Hawking, N. Craswell, P. Bailey, K. Griffihs. Measuring search engine quality, Information Retrieval, 4(1), 2001.
  + 8. M-C. Tang, and Y. Sun. Evaluation of Web-based search engines using user-effort measures. Libres 13.2, 2003.
* It is easy to find a huge number of relevant pages to almost any query. Thus, page relevance is no longer a major issue. The evaluation of usefulness can only be done based on queries derived from the user's personal information needs and his/her personal perception of the returned results to the queries. (C&P JOB, put into own words)
* 2 types of queries:
  + Navigational query:
    - Usually only one satisfactory result or unique page
  + Informational query:
    - Few or many appropriate results. User trying to find info related to a subject with varying levels of relevance.

Modern day implementation of crawlers:

* Examples:
  + Trivago
  + Kayak
* Purpose
* Future implementations
* Importance

Important characteristics of web crawler:

* <https://www.accenture.com/us-en/blogs/search-and-content-analytics-blog/10-criteria-evaluating-search-engine>
* Things like:
  + Content processing
  + Scalability
  + Search relevancy
  + UI useability

Evaluation of a web crawler:

* Different evaluation types:
  + Response time under stress testing?
  + Accuracy of results
    - Number of pages crawls compared to returned results
* Evaluation of existing crawlers? If able to find info
* <https://en.wikipedia.org/wiki/Precision_and_recall>
* <https://en.wikipedia.org/wiki/F-score>
* <https://www.researchgate.net/profile/Ashish-Kumar-Maurya/publication/276206236_URL_Ordering_based_Performance_Evaluation_of_Web_Crawler/links/555c088e08ae8f66f3ade28e/URL-Ordering-based-Performance-Evaluation-of-Web-Crawler.pdf>
* <file:///C:/Users/Compl/Downloads/Performance_Evaluation_of_Web_Crawler.pdf>
* <file:///C:/Users/Compl/Downloads/Feature_evaluation_for_web_crawler_detec.pdf>

Limitations of a crawler:

* Key limitation for web crawler
  + Crawl limit
* <https://www.oncrawl.com/technical-seo/limitations-of-a-crawl/>
* <http://ndl.ethernet.edu.et/bitstream/123456789/41318/1/241.Information%20Resources%20Management%20Association.pdf#page=345>

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.896.2014&rep=rep1&type=pdf>