WebLoad란?

어플리케이션의 성능(performance)과 확장성(scalability), 신뢰성(reliability)을 보장하기 위한 솔루션 → 성능 테스트

기능

많은 사용자들의 트래픽을 동일하게 재연하여, 다음 질문에 대한 답을 말 해줍니다.

① 부하 상황 하에서 애플리케이션의 기능은 정확하게 처리되고 있는가?

② 응답 시간이 고객의 목표 요구 사항에 맞는가 ?

③ 해당 애플리케이션이 처리할 수 있는 동시사용자수는 얼마 인가?

테스트

- 성능 : 테스트 조건에서의 어플리케이션의 속도
- 응답시간 : 어플리케이션의 응답시간
- 처리량 : 어플리케이션의 처리량
- 안정성 : 주어진 시간 동안의 안정된 성능
- 자원사용률 : 어플리케이션의 효율적인 자원 사용률

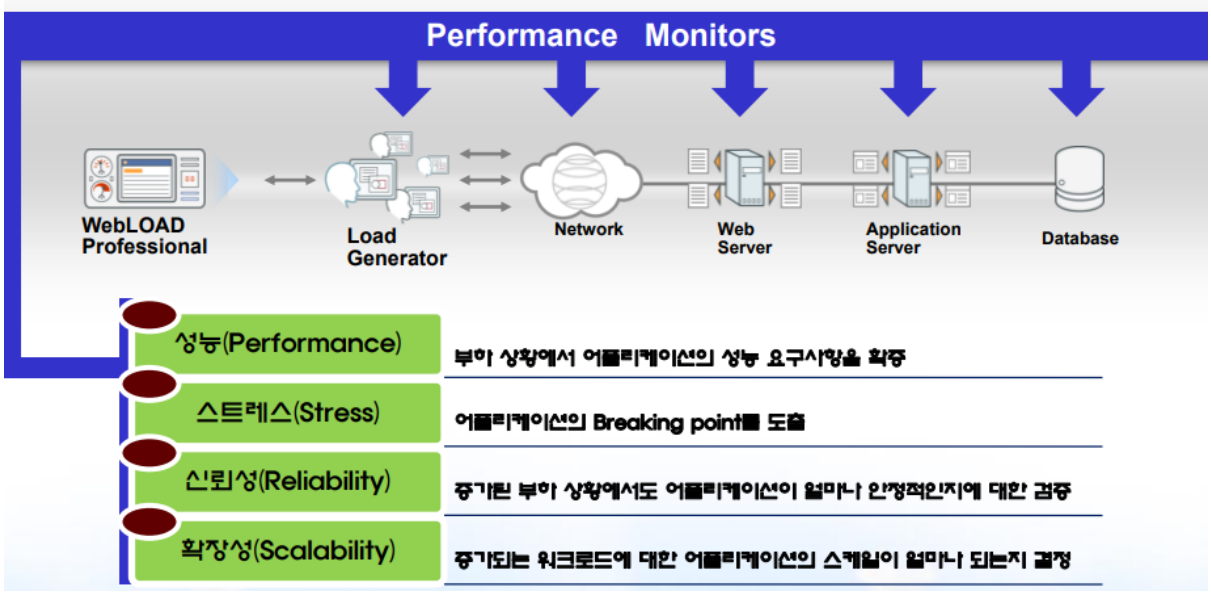
모니터링, 서버 부하, 실시간 결과 리포트,분석 등을 할 수 있음

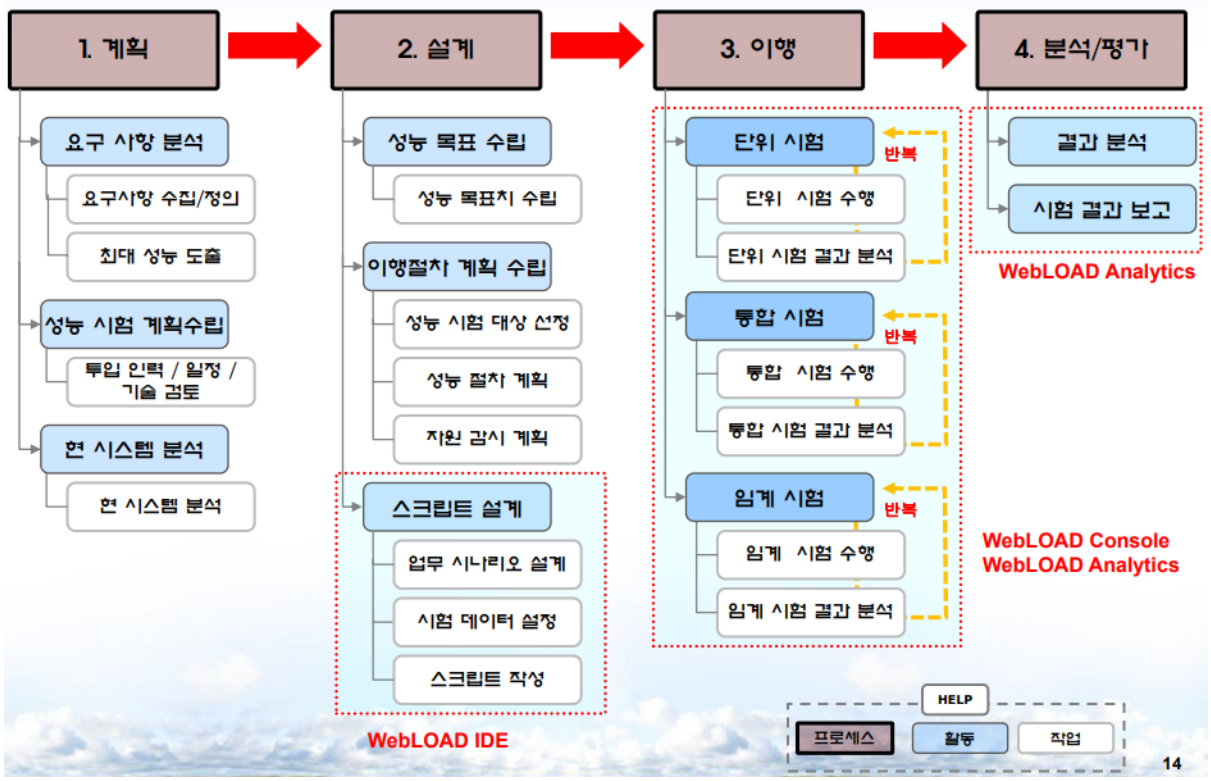
특징 :

IDE에 설치 가능

JS기반 스크립트 편집

Jenkins, Selenium 및 기타 여러 도구와의 통합을 내장하고 있음





출처 : http://www.netsign.co.kr/default/img/product/doc/webload.pdf

https://ko.myservername.com/webload-review-getting-started-with-webload-load-testing-tool

Metric 정리

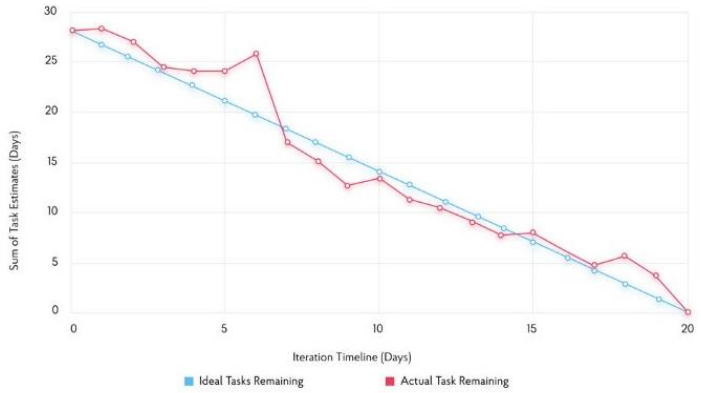
1. Agile Metric

Sprint burndown –

팀이 현재 반복에서 남아 있는 작업의 양과, 수행해야 할 Task의 양을 시각화할 수 있습니다.

현재 일정이 제대로 되고 있는지 시각화하여, 확인 하고 Sprint 조정가능

스프린트 번 다운 차트는 스프린트 진행 상황을 전통적인 표현입니다. 스프린트 동안 매일 현재 스프린트에 대해 예약된 작업을 완료하는 데 남은 시간을 제공합니다. 이 그래프는 팀이 스프린트를 완료 할 일정에 있는지 여부를 즉시 보여줍니다.



Velocity –

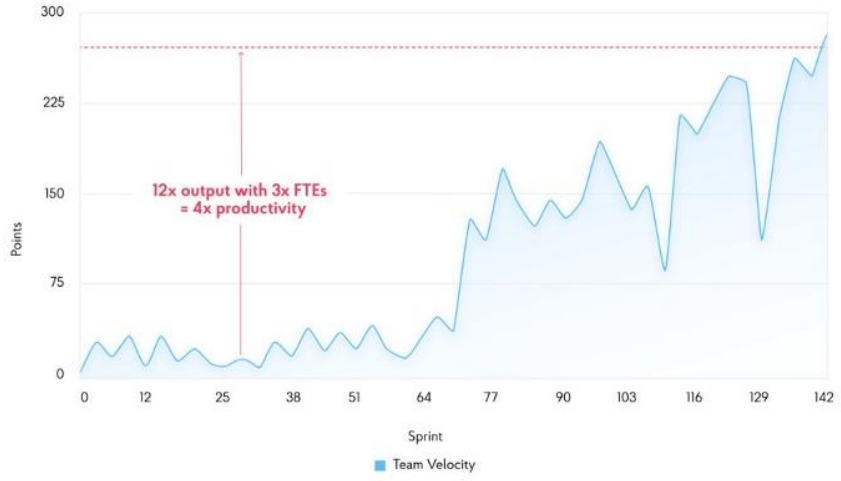
개발팀이 얼마나 기능을 빨리 구현하는지 나타내는 속도. 이로 인해 미래에 이 팀이 어느정도 속도로 개발 할 수 있는지 예측할 수 있는 지표로 사용됨

속도

이것은 스프린트에서 수행 한 작업량을 표시합니다. 이는 팀이 향후 스프린트에서 수행 할 수있는 작업 부분을 추정하는 데 도움이됩니다. 장기간에 걸쳐 속도를 추적하면 약정과 비용이보다 정확하게 일치하는 것을 알 수 있습니다.

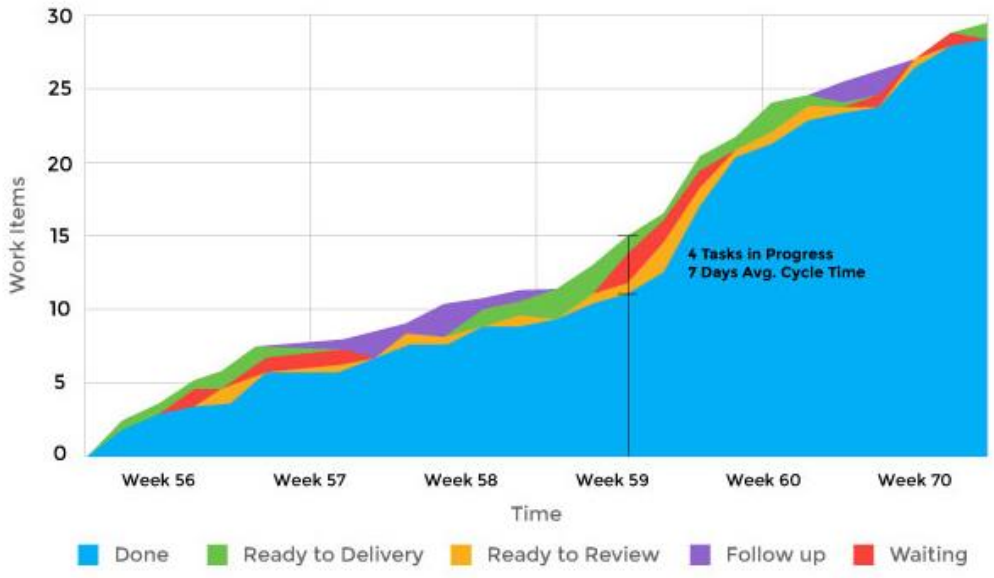
이 지표는 개별적이며 각 팀마다 다릅니다. 인위적으로 속도를 높이려는 시도는 신뢰를 약화시키고 팀과 경영진 간의 투명성을 떨어 뜨릴 수 있습니다.

속도는 또한 팀의 백 로그 작업 능력을 결정합니다. 이는 시간이 지남에 따라 변경 될 수 있으며 일관된 성능을 보장하기 위해 모니터링해야합니다. 흐름의 예측 가능성이 감소하면 개발 문제와 워크 플로의 변경 필요성이 명확하게 추적됩니다.



Cumulative flow –

신속한 변화를 위한 프로세스의 병목 현상을 시각화할 수 있습니다. 특히, 테스트 리소스가 적절한지, 테스트로 개발 주기가 느려지는지를 팀이 시각화할 수 있도록 지원. 쉽게 말해서, 속도 측정 및 어디서 정체되고 있는지 좀 더 구체적으로 나타냄



2. Test Metrics

소프트웨어 테스팅 메트릭은 테스트 활동을 측정하고 모니터 할 수 있는 한 방법이며, 테스트 팀의 테스트 진척과 생산성 그리고 테스트 대상 시스템의 품질에 대한 "통찰(insights)"을 제공한다.

13. 통과된 테스트 케이스 비율(Passed Test Case Percentage)

$$\text{통과된 테스트 케이스 비율} = \left(\frac{\textit{통과된 테스트의 수}}{\textit{총 실행된 테스트 수}} \right) \times 100$$

14. 실패한 테스트 케이스 비율(Failed Test Case Percentage)

$$\text{실패한 테스트 케이스 비율} = \left(\frac{\textit{실패한 테스트의 수}}{\textit{총 실행된 테스트 수}} \right) \times 100$$

15. 중단된 테스트 케이스 비율(Blocked Test Case Percentage)

$$\text{중단된 테스트 케이스 비율} = \left(\frac{\textit{중단된 테스트의 수}}{\textit{총 실행된 테스트 수}} \right) \times 100$$

16. 수정된 결함 비율(Fixed Defects Percentage)

$$\text{수정된 결함 비율} = \left(\frac{\textit{수정된 결함 수}}{\textit{보고된 결함 수}} \right) \times 100$$

20. 중요 결함 비율(Critical Defects Percentage)

$$\text{중요 결함 비율} = \left(\frac{\textit{중요 결함 수}}{\textit{보고된 총결함 수}} \right) \times 100$$

21. 개발팀이 결함을 수정하는데 걸린 평균 시간

$$\text{개발팀의 결함 수정 평균 시간} = \left(\frac{\textit{버그 수정에 소요된 총 시간}}{\textit{버그 수}} \right)$$

22. 특정 시한 동안 테스트 실행 건수(Number of test runs per time period)

$$\text{특정 시한 동안의 테스트 실행 건수} = \left(\frac{\text{테스트 실행 건수}}{\text{총 시간}} \right)$$

23. 테스트 설계 효율성(Test design efficiency)

$$\text{테스트 설계 효율성} = \left(\frac{\text{설계된 테스트 수}}{\text{총 시간}} \right)$$

24. 테스트 검토 효율성(Test review efficiency)

$$\text{테스트 검토 효율성} = \left(\frac{\text{검토된 테스트 수}}{\text{총 시간}} \right)$$

25. 버그 발견율 또는 테스트 시간당 결함 수(Number of defects per test hour)

$$\text{버그 발견율 또는 테스트 시간당 결함 수} = \left(\frac{\text{총 결함 수}}{\text{총 테스트 시간}} \right)$$

26. 테스트당 버그 수(Number of bugs per test)

$$\text{테스트당 버그 수} = \left(\frac{\text{총 결함 수}}{\text{총 테스트 수}} \right)$$

27. 버그 수정을 테스트하는데 드는 평균 시간(Average time to test a bug fix)

$$\text{버그 수정 테스트에 드는 평균 시간} = \left(\frac{\text{모든 결함을 수정 후 재테스트 하는데 소요된 총 시간}}{\text{총 결함 수}} \right)$$

필요해 보이는 Metrics들만 정리했는데 꽤나 많아보임. 이렇게 10몇개 정도는 시각화해야될 듯. 테스트 진척과 생산성, 테스트 품질에 대한 것들을 파악할 수 있음

출처 : https://www.sealights.io/agile-testing/test-metrics/

https://grapevine9700.tistory.com/340

https://ichi.pro/ko/jung-yohan-aejail-peulojegteu-gwanli-kpi-mich-meteulig-nix-united-145672629282386

https://www.plutora.com/blog/agile-metrics