BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian ini mengacu pada beberapa peneliti terdahulu. Pertama adalah penelitian yang dilakukan oleh Hariyanto (2020), dengan judul penelitian **Software Quality Assurance pada Perusahaan Pengembang Perangkat Lunak Skala Kecil dan Menengah**. Penelitian ini adalah untuk mengetahui bagaimana perusahaan pengembang perangkat lunak skala kecil dan menengah melakukan proses SQA pada produk yang dikerjakan, serta nantinya perusahaan dapat mengetahui dan menerapakan standar SQA mana yang sesuai dengan kebutuhan pada perusahaan.

Penelitian kedua adalah penelitian yang dilakukan oleh Aulizar Arfan (2022), dengan judul penelitian **Software Quality Assurance dengan Metode Pengujian Black Box**. Penelitian ini merupakan black box testing yang dilakukan secara manual dan otomatis, untuk menemukan error/bug selama masa pengembangan. Hal berikut merupakan upaya untuk meminimalisir cacat produk ketika aplikasi akan digunakan oleh pengguna.

Penelitian ketiga adalah penelitian yang dilakukan oleh Erick F. Oechler (2016), dengan judul penelitian **Quality Assurance Risk-Based Optimization for Departments of Transportation**. Penelitian ini mengembangkan metode untuk mengoptimalkan material Quality Assurance program dan memberikan rekomendasi untuk alokasi sumber daya Quality Assurance yang tepat di berbagai jenis, ukuran, kompleksitas, dan metode pengiriman proyek.

Penelitian keempat adalah penelitian yang dilakukan oleh Alshaimaa Adel Tantawy Abdou (2009), dengan judul penelitian **Software Quality Assurance Models : A Comparative Study**. Penelitian ini melakukan studi banding antara delapan SQA terpilih model dan pendekatan, mengembangkan sistem berbasis pengetahuan untuk pemilihan model SQA dan penerapan. Menerapkan sistem yang diusulkan ke studi kasus nyata untuk diperiksa validitasnya.

Berikut merupakan daftar pustaka yang digunakan sebagai acuan serta referensi dari pembangunan penelitian yang bisa dilihat di tabel 1 berikut ini :

Tabel 1. Daftar Tinjauan Pustaka

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
Hariyanto (2020)	Software Quality Assurance pada Perusahaan Pengembang Perangkat Lunak Skala Kecil dan Menengah.	Kuantitatif	Baik perusahaan skala kecil dan menengah, terbukti telah menerapakan metode pada proses pengembangan perangkat lunak. Metode yang diterapkan setiap masing-masing perusahaan berbeda. Hal ini dipengaruhi juga oleh jumlah SDM

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
			dari perusahaan. Perusahaan level menengah dibantu oleh tools khusus dalam penerapan SQA, berbeda dengan perusahaan skala kecil yang mengandalkan cara manual dan berbasis dokumen.
Aulizar Arfan (2022)	Software Quality Assurance dengan Metode Pengujian Black Box.	Black Box	Pengujian black box mampu dengan baik menjalankan pengujian manual maupun otomatis dapat menemukan bug pada fungsionalitas dalam aplikasi.

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
Erick F. Oechler (2016)	Quality Assurance Risk-Based Optimization for Departments of Transportation	Review Literatur, Survei, Interview	Penelitian ini adalah untuk mengembangkan kerangka kerja untuk mengoptimalkan siklus hidup CoQ untuk DOT menggunakan model berbasis risiko. Pendekatan siklus hidup diadopsi karena DOT harus membayar upaya terkait kualitas yang terjadi sebelum dan selama konstruksi dan membayar untuk pengerjaan ulang atau rekonstruksi ketika material gagal selama

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
			melayani. Penelitian sebelumnya berfokus pada QA untuk kontraktor tetapi tidak ada penelitian sebelumnya menggunakan pendekatan berbasis risiko atau berfokus pada perspektif DOT.
Alshaimaa Adel Tantawy Abdou (2009)	Software Quality Assurance Models : A Comparative Study.	Kuantitatif	Menerapkan studi kasus pada sistem SQAMSI yang diusulkan menunjukkan bahwa sistem yang diusulkan menyarankan perusahaan E-SmartSoft model CMMI sebagai lebih

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
			model kualitas yang tepat untuk diadopsi sesuai dengan persyaratan dan kebutuhannya. Hasil ini kompromi dengan model aktual yang diterapkan di perusahaan. Di sisi lain, menurut laporan penilaian diri dari E-SmartSoft perusahaan pada CMMI Level 2 yang dihasilkan dari tahap implementasi sistem yang diusulkan, perusahaan E-SmartSoft bisa mendapatkan CMMI Level.

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
Usulan (2023)	Magang Bersertifikat Kampus Merdeka Inspek Terhadap Nomor Tiket Defect Pada Aplikasi Sekolah.mu Dalam Sebuah Sprint Dengan Metode Framework Scrum.	Framework Scrum	Dalam penelitian Inspek terhadap nomor tiket pada aplikasi dalam sebuah sprint dengan framework scrum, hasil yang diharapkan adalah identifikasi dan penanganan efektif terhadap defect pada aplikasi dalam sprint tersebut, sehingga dapat memperbaiki kualitas produk dan mempercepat pengiriman produk ke pasar. Dalam hal ini, tim Scrum akan dapat mengevaluasi

Penulis (Tahun)	Topik Penelitian	Pendekatan/Metode	Hasil
			kinerja mereka dan memperbaiki proses pengembangan produk di masa depan.

2.2 Dasar Teori

2.2.1 Software Quality Assurance

Software Quality Assurance adalah proses sistematis untuk memeriksa apakah sebuah software telah dikembangkan sesuai dengan kebutuhan yang telah ditentukan sebelumnya. Proses ini, bisa dilaksanakan oleh seorang QA Tester atau oleh seorang QA Engineer. QA Tester memiliki tugas utama melaksanakan pengujian terhadap perangkat atau emulator, membuat alur pengujian, serta membuat laporan hasil pengujian. Sementara QA Engineer biasanya bertugas untuk membuat porgram pengujian otomatis, membuat laporan pengujian, memberikan masukan atas aplikasi yang diuji, serta berkomunikasi dengan pihak-pihak yang berkepentingan, seperti pengembang UI/UX, back end atau product manager (PM). QA Analyst merupakan suatu profesi dari Software Quality Assurance yang berfokus untuk melakukan analisa mengenai prosedur, product requirement documents, dan requirements testing lainnya serta menuliskan hasil dari analisa tersebut ke dalam test cases pada produk yang akan dilakukan testing, serta melakukan testing produk software dan memastikan bahwa suatu produk software yang dihasilkan telah sesuai dengan requirement yang telah ditentukan dan melaporkan hasil dokumentasi dari hasil testing produk software yang sudah ditest. Untuk proses QA, dibutuhkan kemampuan-kemampuan:

- Mindset Pengujian
- Analisa & Pengujian Fungsional
- Perbaikan Proses
- Defect Management
- Pengujian Keamanan
- Pengujian Performa

- Otomasi
- User Acceptance Testing (UAT)

Sementara untuk menjalankan proses software QA, diperlukan antara lain:

- Laptop (Linux OS/ Mac)
- Device utk pengujian
- Pengetahuan pemrograman
- Terbiasa dengan Git
- Terbiasa dengan Agile (Scrum)

Pengujian terhadap software sendiri terbagi menjadi dua jenis:

1. Pengujian Manual

Pengujian ini biasanya dilakukan untuk mengecek aliran aplikasi, memeriksa cacat (desain atau pemrograman), pengujian di sistem operasi berbeda, serta uji migrasi dari versi aplikasi terdahulu

2. Pengujian Otomatis

Pengujian ini terdiri dari pengujian regresi, pengujian otomatis yang dilakukan pada malam hari, pelaporan otomatis (melalui email atau tool kolaborasi seperti slack), automated build, dan automated publish.

Untuk pengujian otomatis, bisa dilakukan di berbagai sistem: Continuous Integration (CI) Jenkins, Travis CI, Circle CI, dan lain-lain. Apabila terjadi kesalahan dalam pengujian biasanya karena ada permasalahan di sisi backend, di sisi Continuous Integration (adanya pembaruan library, misalnya) atau perbaikan pada aplikasi.

Adapun kriteria-kriteria dari sebuah desain framework pengujian otomatis adalah:

- Mudah digunakan

- Bisa/ mudah dalam pemeliharaan
- Scalable
- Dukungan browser atau peranti
- Adanya metrik dan pelaporan
- Dukungan terhadap pengujian yang dikehendaki
- Dapat dijalankan secara lokal mauputn remote (dari jarak jauh)
- Eksekusi paralel
- Mendukung fitur-fitur yang akan diuji
- Mendukung environment yang berbeda
- Dukungan Tool
- Batasan bahasa/ tool

Dan yang tak kalah penting dari hal-hal di atas adalah, QA tester maupun QA engineer harus selalu berpikiran terbuka. Dengan sikap terbuka untuk mempelajari hal-hal yang baru, diharapkan QA dapat memberikan kontribusi positif untuk menghasilkan produk software yang baik.

2.2.2 Testing

Di dalam functional testing, memiliki level atau tingkatan testing. Test level terdiri dari 4 tingkat yaitu sebagai berikut :

Unit Test

Unit test merupakan testing level untuk menguji bagian software terkecil. Tujuan dilakukannya unit test ini yaitu untuk menemukan error lebih cepat sehingga dapat mengantisipasi error yang lebih kompleks dikemudian hari. Unit test dilakukan dengan cara kolaborasi antara tim developer dan tim quality assurance. Dari tim developer yaitu melakukan pengujian terhadap prosedur atau fungsi dalam

sebuah program code. Sedangkan dari tim QA yaitu melakukan pengujian terhadap parameter dan response API, field data pada database, atau terhadap UI elements yang disesuaikan dengan kebutuhan development dan software requirement.

Ada dua teknik untuk melakukan unit test ini, yaitu dengan teknik manual atau automate. Untuk meng-automate unit test pada fungsi dari program code, yaitu menggunakan framework yang sesuai dan support dengan code atau bahasa pemrograman yang digunakan. Misalnya java dengan junit atau php dengan phpunit dan seterusnya. Sedangkan untuk menguji API dapat menggunakan Postman dan UI elements menggunakan Galen Framework.

Integration Test

Integration test yaitu pengujian dari gabungan atau integrasi antar unit code untuk menghasilkan flow scenario fungsi, fitur dan atau module tertentu. Contohnya register, login, search, dan lainnya. Integration testing dapat dilakukan dengan metode end-to-end testing, dimana end-to-end testing merupakan testing alur dari awal sampai selesai. Misalnya ketika melakukan register pada sebuah system, maka harus dipastikan flow, input dan output dari awal sampai selesai sesuai dengan requirement. Tujuannya adalah untuk memastikan bagaimana flow scenario tersebut dapat berjalan sesuai dengan requirement.

Hal yang harus diperhatikan dan dipastikan dari testing level ini yaitu bagaimana kesesuaian alur data pada database dengan response API dan tampilan di-UI, atau bagaimana alur integrasi antar API bekerja. Selain menguji integrasi antar unit, contoh lain untuk integration testing yaitu menguji integrasi antar software. Misalnya sebuah e-commerce yang menyediakan jual beli pulsa yang sistemnya disediakan oleh perusahaan pihak ketiga, maka harus dipastikan

bagaimana e-commerce tersebut dapat melakukan proses jual beli pulsa dengan software atau system yang terintegrasi.

Proses integration testing yaitu sebagai berikut.

- Mempersiapkan test scenario dan test cases
- Membuat automated test jika diperlukan
- Eksekusi test cases dan automated test script
- Membuat report dan melaporkan jika terjadi error atau ketidaksesuaian software dengan requirement
- Re-testing sampai software sesuai dengan requirement

Integration testing juga dapat dikombinasikan dengan pendekatan exploratory testing yaitu proses testing tanpa ada panduan atau test cases atau requirement tertentu. Hal tersebut dapat dilakukan untuk memvalidasi dan atau melengkapi test cases. Tools dan framework automated test pada tahap ini dapat menggunakan Cypress, TestCafe, dan lainnya.

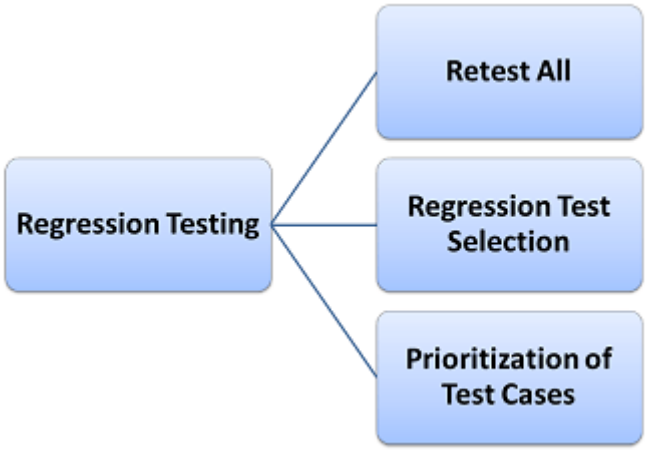
System Test

System test adalah pengujian yang dilakukan untuk cakupan yang lebih besar. Pengujian ini dilakukan ketika integration test selesai. System test merupakan pengujian yang dilakukan secara keseluruhan terhadap sebuah software. Misalnya dalam satu development life-cycle akan membuat fitur transaksi seperti pembelian atau pembayaran, maka yang harus diperhatikan dalam pengujian ini adalah module atau fitur lainnya tidak boleh ada error dikarenakan impact dari development fitur baru tersebut.

Salah satu jenis pengujian yang dapat dilakukan pada system test ini adalah regression test. Regression test adalah pengujian software untuk memastikan bahwa

penambahan atau perubahan code baru tidak mempengaruhi code yang sudah ada. regression test ini dapat dilakukan dengan beberapa cara yaitu sebagai berikut.

- Dilakukan dengan mengeksekusi semua test case yang ada, mencakup software secara keseluruhan
- Dilakukan dengan mengeksekusi sebagian test case yang berkaitan dengan penambahan fitur
- Dilakukan dengan mengeksekusi semua test case yang mempunyai prioritas/level tertentu, seperti gambar berikut.



Gambar 1. Regression Testing

Tujuan dilakukannya system test yaitu memastikan sistem atau software secara keseluruhan dapat berfungsi sebagaimana mestinya. Oleh karena itu, system test dilakukan sebelum update software ke environment yang digunakan oleh user.

Untuk estimasi waktu system test harus disesuaikan dengan jenis test yang digunakan. Khusus untuk functional testing, dapat melihat dari banyaknya test case yang dieksekusi, serta estimasi jika terjadi error atau ketidaksuaian requirement. Tetapi seharusnya error yang terjadi pada level ini sudah minim, karena impact yang ditimbulkan oleh existing module sudah dianalisis sebelum development dimulai. Jika terjadi error yang tidak ter-capture ketika analisis impact, maka

menjadi tanggung jawab bersama (mulai dari product manager, software engineer, quality assurance dan seterusnya yang terlibat di dalam tim development) untuk menyelesaikannya hal tersebut.

Output yang dihasilkan dari system test ini yaitu laporan hasil eksekusi test case, misalnya berupa grafik atau hal yang mudah dipahami oleh stakeholder. Laporan ini mencakup semua hasil test secara functional testing dan non-functional testing. Laporan tersebut bertujuan untuk menyimpulkan apakah update system sudah layak atau belum untuk diteruskan ke environment yang digunakan user (staging atau production).

Acceptance Test

Acceptance test adalah tahap uji terakhir pada functional testing level. Tahap ini QA secara umum tidak terlibat langsung, karena yang bertanggung jawab pada tahap acceptance test adalah product manager atau tim yang berhubungan langsung dengan user yang akan menggunakan software. Tim product manager langsung berhadapan dengan user untuk melakukan pengujian terhadap software untuk memvalidasi dan memverifikasi apakah software sudah sesuai dengan alur bisnis yang diinginkan. Acceptance test juga sering disebut dengan user acceptance test atau UAT.

Hal yang perlu diperhatikan dalam acceptance test ini yaitu software yang akan diuji sudah melewati tahap uji sebelumnya, kemudian mempersiapkan dokumentasi serta alur test yang akan dieksekusi menyesuaikan dengan user story dan alur bisnis. Ketika sedang melaksakan acceptance test, behavior user dalam menggunakan software juga perlu diperhatikan. Tujuannya adalah untuk mengevaluasi apakah fitur tersebut sudah nyaman digunakan atau masih butuh improvisasi.

2.2.3 Story Point

Story Point adalah ukuran atau estimasi untuk mengerjakan sebuat product backlog atau sebuah kerjaan. Estimasi terhadap rumitnya, resikonya, lamanya, dan banyaknya sebuah pekerjaan. Setiap tim dalam sebuah projek memberikan nilai poin berdasarkan kompleksitas, jumlah,ketidakpastian pekerjaan, dan resiko kegagalan. endekatan estimasi dengan story point adalah alternatif pendekatan estimasi selain pendekatan man-hour yang biasa digunakan di proyek konvensional. Berikut ini adalah contoh ukuran estimasi story point yang sering digunakan. Yang paling terkenal dan biasa digunakan di sprint planning adalah teknik planning poker. Yang pointnya adalah mengikuti pola Fibonacci, yaitu 1, 3, 4, 8, 13, 21, dan seterusnya.

Teknik Estimasi	Ukurannya
Planning Poker	1,3,5,8,13,21,dst
Ukuran T-Shirt	XS,S,M,L,XL,XXL
Bucket System	0,1,2,3,4,5,8,13,20,30,50,100,200
Dot Voting	Titik sebagai tanda estimasi dari tiap anggota tim
Affinity Estimation	Estimasi dengan mengumpulkan story yang sejenis

Gambar 2. Teknik Estimasi pada Story Point.

Secara umum, estimasi ini mempunyai banyak manfaat, antara lain:

- Memudahkan tim untuk mengukur kemampuan tim dan tugas yang akan dihadapi/diselesaikan.
- Memudahkan tim dan product owner untuk menentukan seberapa banyak value yang bisa di deliver di setiap sprint.

Story point bermanfaat untuk tim pengembang dan product owner, antara lain:

Untuk tim pengembang (developers):

- Tim mendapatkan pemahaman yang lebih baik tentang apa yang diminta dari mereka, sehingga lebih mudah mengembangkan strategi penerapan yang baik.
- Tim tidak akan membuat rencana berlebihan, sehingga mereka memiliki kesempatan yang lebih baik untuk menyelesaikan setiap task dan increment.
- Tim tahu berapa banyak yang harus direncanakan dalam sprint, sehingga membuat mereka bekerja dengan efektif dan efisien.
- Tim dapat membuat perkiraan yang masuk akal tanpa harus berkomitmen pada jangka waktu tertentu.

Untuk product owners:

- Product owners dapat menilai laba atas investasi atau return of investment (ROI) atau nilai sebuah produk dengan lebih baik.
- Dapat lebih memahami risiko teknis yang terkait dengan produk yang berukuran lebih besar.
- Memperkirakan delivery product jangka panjang ke client dengan lebih baik.

Setiap anggota tim harus berhati-hati dalam menentukan nilai dari setiap story yang akan dikerjakan, berikut ini adalah beberapa tips yang dapat lakukan, antara lain:

- Jangan memberikan story point untuk tugas-tugas yang sangat mudah atau kecil.
- Pembuatan story point adalah upaya tim, bukan hanya perorangan. Pastikan tim tetap terlibat, dan semua orang berkontribusi.
- Jangan membuat semua task memiliki nilai berdasarkan rata-rata seluruh story point.
- Jangan biarkan satu variabel mempengaruhi keseluruhan proses story point.

2.2.4 Framework Scrum

Scrum adalah suatu kerangka kerja sederhana yang banyak digunakan untuk pengembangan produk yang kompleks. Dalam scrum, pengembangan software difokuskan pada iterasi dimana di setiap iterasi terdapat value yang dapat diberikan kepada customer. Framework ini tidak mengharuskan analisa kebutuhan dari seluruh sistem ditetapkan di awal, sehingga memungkinkan untuk produk terus menerus dikembangkan. Scrum juga memungkinkan adaptasi pada perubahan kebutuhan di tengah-tengah pengembangan.

Dalam Scrum, terdapat beberapa peran, yaitu:

- Product Owner

Product Owner adalah pihak yang bertugas untuk mengumpulkan dan menganalisa kebutuhan dari stakeholders. Product Owner juga bertanggung jawab pada product backlog.

- Scrum Master

Scrum Master adalah pihak yang berperan untuk memastikan seluruh prosedur diikuti dan semua agenda berjalan lancar. Scrum master merupakan orang yang paling memahami cara kerja metode scrum dalam menangani suatu proyek pengerjaan produk berupa perangkat lunak.

Kerangka kerja scrum bermanfaat untuk memberikan kejelasan tentang apa saja yang perlu dilakukan dan menjadi fokus utama dalam proyek yang dikerjakan oleh tim. Scrum master akan membantu dan memfasilitasi tim developer dengan mengadakan pertemuan rutin setiap hari untuk membahas perkembangan terbaru, mengadakan rapat perencanaan, dan sebagainya.

- Development Team

Development team adalah engineer yang bertugas dalam mengeksekusi sprint backlog.

Adapun dalam Scrum, terdapat beberapa kegiatan, yaitu:

- Sprint Planning

Sprint Planning adalah waktu dimana Scrum Team bersama-sama menentukan tujuan (Sprint Goal) dan apa yang akan dikerjakan dalam 1 sprint (Sprint Backlog). Dalam kegiatan ini, Product Owner bertugas dalam menjelaskan story yang ada pada product backlog, dan Development Team bertugas dalam memecah story menjadi task-task kecil agar jelas lingkup dari story tersebut dan cara implementasinya.

- Sprint Review

Sprint Review merupakan kegiatan yang diadakan di akhir sprint, dimana Development Team akan memaparkan dan mendemokan hasil yang telah mereka capai selama 1 sprint. Product Owner dan stakeholder dapat memberikan feedback dari hasil yang dikerjakan oleh Development Team. Product Owner dan Stakeholder juga berdiskusi untuk memperbaharui product backlog dan menentukan definition of done dari setiap backlog.

- Sprint Retrospective

Sprint Retrospective adalah kegiatan terakhir dalam satu sprint. Sprint Retrospective dilakukan setelah Sprint Review. Jika Sprint Review adalah kegiatan untuk melakukan review terhadap hasil yang telah dikerjakan, Sprint Retrospective adalah review terhadap kinerja dan performa tim selama satu sprint. Format dasar dalam Sprint Retrospective adalah apa yang telah berjalan dengan baik (Done Well), apa yang salah (Went Wrong), apa yang akan ditingkatkan (Commit).

- Daily Stand-Up

Daily Stand-Up adalah kegiatan yang dilakukan setiap hari selama masa sprint. Daily Stand-Up memiliki batas waktu maksimal 15 menit. Dalam kegiatan ini, tim saling memberikan update mengenai progress yang dikerjakan. Format dasar dalam Daily Stand-Up adalah apa yang telah dikerjakan kemarin, kendala apa yang dihadapi, apa yang akan dikerjakan hari ini

Alur kerja dari Scrum adalah:

- Product Owner mendaftarkan story-story yang diurutkan secara prioritas, dinamakan Product Backlog.
- Saat Sprint Planning, tim mengambil sebagian dari product backlog atas (dengan prioritas tinggi), disebut dengan Sprint Backlog, memecahnya menjadi task-task kecil dan menentukan cara mengimplementasikan/mengeksekusinya.
- Tim memiliki waktu tertentu yang dinamakan sprint (2-4 minggu) untuk menyelesaikan Sprint Backlog. Setiap hari tim bertemu untuk menyampaikan progress (Daily Stand-Up).

- Di akhir sprint, tim harus telah menyelesaikan sesuatu yang dapat menambahkan value ke produk. Sprint diakhiri dengan Sprint Review dan Sprint Retrospective.
- Sprint berikutnya mulai, keseluruhan alur kerja dilakukan kembali.