SKRIPSI

SISTEM PEMBANGKIT KODE UJI BERBASIS *BEHAVIOR SPECIFICATION*

Muhammad Dipo Putra Wandara

NPM: 2016730091

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2021

UNDERGRADUATE THESIS

BEHAVIOR SPECIFICATION BASED TEST CODE GENERATION SYSTEM

Muhammad Dipo Putra Wandara

NPM: 2016730091

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2021

LEMBAR PENGESAHAN

SISTEM PEMBANGKIT KODE UJI BERBASIS *BEHAVIOR SPECIFICATION*

Muhammad Dipo Putra Wandara

NPM: 2016730091

Bandung, 3 Februari 2021

Menyetujui,

Pembimbing

Pascal Alfadian, Nugroho, M.Comp.

Ketua Tim Penguji	**Anggota Tim Penguji**
Lionov, Ph.D.	**Elisati Hulu, M.T.**

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

SISTEM PEMBANGKIT KODE UJI BERBASIS *BEHAVIOR SPECIFICATION*

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 3 Februari 2021

Muhammad Dipo Putra Wandara
NPM: 2016730091

ABSTRAK

Pada pengujian perangkat lunak, pemangku kepentingan bisnis/*stakeholder* umumnya sulit berpartisipasi secara langsung ketika pengujian perangkat lunak dilakukan, dikarenakan *stakeholder* tidak memahami cara implementasi perangkat lunak dan cara pengujiannya. Untuk dapat mengatasi masalah tersebut, salah satu cara yang dapat dilakukan ialah membuat sistem pembangkit kode uji berbasis *behavior specification*. Dengan *behavior specification*, *stakeholder* dapat menulis spesifikasi pengujian perangkat lunak menggunakan bahasa yang dimengerti oleh orang pada umumnya, tanpa perlu memahami cara pengujian perangkat lunak.

Penelitian ini membahas *Behavior-Driven Development* (BDD). BDD menggunakan skenario berbasis *behavior specification*. Perangkat lunak sistem pembangkit kasus uji berbasis *behavior specification* menerima masukan berupa skenario berbasis *behavior specification* yang berformat skenario Gherkin. Skenario Gherkin memiliki *keyword* yang dijadikan dasar untuk pembangkitan kode menggunakan *code generation*. Pengujian dilakukan untuk memastikan kode uji yang dihasilkan mampu untuk menguji perangkat lunak sesuai dengan *behavior specification*.

Berdasarkan hasil pengujian yang dilakukan dengan menggunakan masukan skenario Gherkin, pada penelitian ini telah berhasil dibangun perangkat lunak yang mampu membangkitkan kode uji berdasarkan skenario tertentu. Kode uji yang dihasilkan berupa kode *unit testing* dan kode *integration testing*.

Kata-kata kunci: pengujian perangkat lunak, *behavior-driven development*, *behavior specification*, gherkin

ABSTRACT

In software testing, it is generally difficult for business stakeholders/stakeholder to participate directly when software testing is carried out, because stakeholder does not understand how to implement the software and how to test it. To be able to solve this problem, one way that can be done is to create a behavior specification based test code generator system. With behavior specification, stakeholder can write software test specifications using a language that is understandable to the average person, without needing to understand how to test software.

This research discusses Behavior-Driven Development (BDD). BDD uses a behavior specification based scenario. The behavior specification based test case generator system software accepts input in the form of a behavior specification based scenario in the Gherkin scenario format. The Gherkin scenario has keyword which is the basis for code generation using code generation. Testing is done to ensure that the resulting test code is able to test the software according to the behavior specification.

Based on the results of tests carried out using the input of the Gherkin scenario, this research has successfully built software capable of generating test codes based on certain scenarios. The resulting test code is in the form of code unit testing and code integration testing

Keywords: software testing, behavior-driven development, behavior specification, gherkin

Skripsi ini saya persembahkan kepada papah, mamah, dan Nida Yumna. Semoga dengan skripsi ini bisa menjadi hadiah atas keinginan yang belum tersampai.

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan rahmat dan karunia-Nya kepada penulis, sehingga penulis dapat menyelesaikan skiripsi dengan judul "Sistem Pembangkit Kode Uji Berbasis *Behavior Specification*". Shalawat dan salam senantiasa tercurah kepada Rasulullah SAW. Penyusunan skripsi ini dimaksudkan untuk memenuhi syarat guna mencapai gelar sarjana di Program studi Teknik Informatika, Fakultas Teknologi Informasi dan Sains, Universitas Parahyangan Bandung.
Penulis menyadari bahawa penulisan ini tidak dapat terselesaikan tanpa dukungan dari berbagai pihak, baik moril maupun materil. Oleh karena itu, penulis ingin mengucapkan ucapan terima kasih kepada :

1. Kedua orang tua, Bapak Irwan Koesdrajat dan Ibu Zamraliani, yang telah memberikan dukungan baik moril serta doa.
2. Bapak Raymond Chandra, M.T. selaku dosen pembimbing yang memberikan dukungan, kritik, saran, dan nasihat yang membangun dalam pelaksanaan penelitian ini maupun terhadap penulisan.
3. Seluruh Bapak/Ibu dosen Teknik Informatika yang telah memberikan pengetahuan dan pelajaran selama penulis berkuliah di Unpar.
4. Nida Yumna yang selalu memberikan dukungan.
5. Teman-teman seperjuangan selama perkuliahan, yaitu : Jaya, Samuel, dan Hashrul.
6. Serta seluruh teman-teman yang telah memberikan semangat dan dukungan yang tidak bisa disebutkan satu persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dan menyakini bahwa kesempunaan hanya milik Allah SWT, oleh karena itu apabila terdapat kesalahan dalam penelitian dan penulisan, penulis mengharapkan kritik dan saran yang dapat membantu penyempurnaan penelitian ini akan penulis terima dengan senang hati. Penulis berharap, semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Bandung, Februari 2021

Penulis

DAFTAR ISI

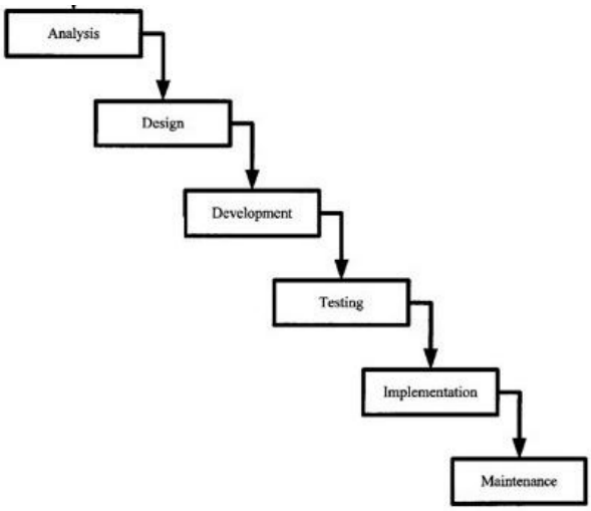
DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pengujian perangkat lunak merupakan salah satu tahapan dari *Software Development Life Cycle* (SDLC). SDLC merupakan metodologi dari pengembangan perangkat lunak, metode ini dibagi menjadi beberapa fase seperti *analysis, design, development, testing, implementation* dan *maintenance*, dapat dilihat pada Gambar 1.1 [8].



Gambar 1.1: *SDLC* [1]

Pengujian perangkat lunak adalah proses untuk mencari kesalahan pada setiap komponen perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek pada komponen dan mengevaluasi fitur-fitur dari perangkat lunak yang sedang dikembangkan [9]. Pengujian perangkat lunak dilakukan sebelum perangkat lunak tersebut digunakan agar memastikan tidak terdapat *error/bug* pada perangkat lunak. Pengujian perangkat lunak membutuhkan sumber daya yang besar, baik itu waktu maupun tenaga kerja, karena itu dibutuhkan pengembang ahli *Software Quality Assurance*(SQA) untuk melakukan pengujian perangkat lunak. SQA bertanggung jawab untuk memastikan bahwa perangkat lunak yang sedang dikembangkan tidak memiliki *error/bug* sehingga dapat digunakan oleh pengguna.

Pada pengujian perangkat lunak terdapat beberapa tahapan, yaitu *unit testing, integration testing, system testing*, dan *acceptance testing. Unit testing* merupakan tahapan awal pada pengujian perangkat lunak. Dalam *unit testing*, penguji menguji setiap unit kode secara individu, dimana unit merupakan bagian terkecil yang dapat diuji dalam suatu perangkat lunak [4]. *Integration testing* merupakan tahapan untuk memverifikasi bahwa setiap modul bekerja sesuai fungsinya dalam struktur dan antarmuka modul [3]. *System testing* merupakan proses verifikasi dan memberikan gambaran eksternal dari sistem dan memungkinkan penguji untuk membuktikan bahwa sistem memenuhi

semua tujuan dan persyaratan sistem [10]. *Acceptance testing* merupakan bentuk pengujian validasi dan dilakukan oleh pengguna dan memungkinkan penguji untuk memverifikasi bahwa sistem sudah memenuhi semua persyaratan pengguna atau logika bisnis [10]. Pada skripsi ini tahapan yang digunakan yaitu *unit testing* dan integration testing.

Pada pengujian perangkat lunak, pemangku kepentingan bisnis/*stakeholder* umumnya sulit berpartisipasi secara langsung ketika pengujian perangkat lunak dilakukan, dikarenakan *stakeholder* tidak memahami cara implementasi perangkat lunak dan cara pengujiannya. Untuk dapat mengatasi masalah tersebut, salah satu cara yang dapat dilakukan ialah menggunakan metode *Behavior-Driven Development* (BDD).

Ada beberapa metode untuk mengembangkan perangkat lunak, diantaranya ialah *Test-Driven Development* (TDD) dan *Behavior-Driven Development* (BDD). TDD merupakan praktik rekayasa perangkat lunak yang menginstruksikan pengembang untuk menulis kode baru setelah pengujian dibuat [5]. Tujuan TDD yaitu kode yang bersih, dirancang dengan baik, dapat dengan mudah dirawat dan mengalami kesalahan lebih rendah ketika perangkat lunak diuji [11]. Terlepas dari kelebihannya, terdapat kesulitan dalam mengadopsi dan menggunakan TDD secara efektif. TDD menjadikan pengembang terlalu fokus pada detail, sehingga kehilangan gambaran yang lebih luas tentang tujuan bisnis dan *behavior* dari perangkat lunak yang seharusnya diterapkan [5].

Behavior-Driven Development (BDD) merupakan seperangkat praktik rekayasa perangkat lunak yang dirancang untuk membantu tim membuat dan memberikan perangkat lunak yang berkualitas serta bernilai lebih tinggi. BDD menyediakan bahasa umum, berdasarkan kalimat-kalimat sederhana dan terstruktur yang diekspresikan dalam bahasa Inggris (atau dalam bahasa lainnya) yang memfasilitasi komunikasi antara anggota tim proyek dan *stakeholder*. BDD menempatkan fokus pada *behaviour*/perilaku perangkat lunak daripada strukturnya [5]. Diantara TDD dan BDD, metode BDD dipilih pada skripsi ini dikarenakan BDD berfokus pada *behavior* dari sebuah sistem sedangkan TDD lebih berfokus ke detail sistem sehingga tujuan bisnis yang diinginkan *stakeholder* kurang diperhatikan.

Prinsip inti dari BDD ialah *stakeholder* dan pengembang perangkat lunak merujuk ke sistem yang sama dengan cara yang sama [12]. Berbeda dengan TDD yang berpatokan pada pengujian untuk membangun perangkat lunak yang dikembangkan, BDD berpatokan pada keinginan *stakeholder* untuk mengembangkan perangkat lunak. Untuk mencapai kesepakatan antara pengembang dan *stakeholder*, dibutuhkan bahasa untuk menspesifikasikan *behaviour* sebuah sistem agar dipahami kedua pihak, yang memungkinkan untuk [12]:

- *Stakeholder* menentukan persyaratan dari perspektif bisnis.
- *Stakeholder* melampirkan contoh konkret (skenario) yang menjelaskan *behaviour* sistem.
- Pengembang mengimplementasikan *behaviour* sistem yang diperlukan secara BDD.

Pada pengujian perangkat lunak, *stakeholder* belum tentu memahami cara implementasi perangkat lunak, sehingga *stakeholder* tidak dapat berpartisipasi secara langsung ketika pengujian perangkat lunak dilakukan.

Pengujian dengan basis *behaviour specification* berfokus kepada hal yang *stakeholder* harapkan pada suatu sistem dengan spesifikasi tertentu pada *behavior* sistem. *Stakeholder* dapat membuat spesifikasi tersebut secara konkret dalam bentuk skenario. Pengujian berawal dari skenario yang berisi [12]:

1. *Context/Starting state* : Posisi awal sistem sebelum terjadinya suatu aksi.
2. *Event* : *Task* yang dilakukan oleh pengguna pada sistem.
3. *Outcome* : Hasil yang diharapkan dari sistem.

Skenario tersebut memiliki *keywords* yang fungsinya sudah ditentukan, untuk *context/starting state*, *event* dan *outcome* memiliki *keyword* tersendiri. Setiap *keyword* memiliki makna serta fungsi yang berbeda dan *keyword* tersebut digunakan untuk menjelaskan *behavior* atau cara kerja dari sebuah sistem.

Pengembang menjadikan skenario sebagai dasar cara kerja sistem dan melakukan pengujian berbasis skenario. Pengujian dilakukan untuk memastikan bahwa sistem sudah bekerja sesuai dengan

harapan *stakeholder*. Jika skenario berhasil dilakukan, maka sistem sudah siap untuk digunakan. Dalam pembuatan skenario berbasis *behavior specification*, *stakeholder* dapat berpartisipasi pada pengujian perangkat lunak tanpa perlu memahami kode perangkat lunak.

Dengan menggunakan teknik *code generation*, *unit testing* dan *integration testing* dapat dibuat. *Code generation* adalah teknik membangun dan menggunakan perangkat lunak untuk menulis perangkat lunak lain. Menggunakan *code generation* memudahkan pembuatan dan pemeliharaan *unit testing* serta *integration testing* [13]. Pembuatan *unit testing* dan *integration testing* dengan *code generation* menggunakan skenario sebagai patokan dasarnya.

Skripsi ini dimaksud untuk membangun sebuah sistem pembangkit kode uji berbasis *behavior specification* yang menghasilkan *unit testing* dan *integration testing*. Sistem pembangkit kode uji dibangun untuk menguji aplikasi berbasis web yang dijadikan sebagai studi kasus. Aplikasi berbasis *web* mempunyai variasi struktur yang beragam tergantung *framework* yang digunakan, jadi pada penelitian ini digunakan aplikasi *web* yang menggunakan *framework* Laravel.

1.2 Rumusan Masalah

Berikut adalah rumusan masalah dari penulisan skripsi :

- Bagaimana cara kerja sistem pembangkit kode uji berbasis *behavior specification*?
- Bagaimana implementasi sistem pembangkit kode uji berbasis *behavior specification*?

1.3 Tujuan

Tujuan dari penulisan skripsi ini adalah sebagai berikut :

- Merancang cara kerja sistem pembangkit kode uji berbasis *behavior specification*.
- Menghasilkan perangkat lunak yang mampu menghasilkan kode uji berbasis *behavior specification*.

1.4 Batasan Masalah

Diperlukan batasan masalah yang jelas mengenai apa yang dibuat dan diselesaikan pada skripsi ini. Adapun batasan-batasan masalah pada skripsi ini ialah sebagai berikut:

- Sistem pembangkit kode uji hanya mampu untuk menguji perangkat lunak berbasis web yang mengikuti standar struktur *framework* Laravel.
- Pada penelitian ini hanya dapat menangani beberapa *keywords* yang sudah ditentukan sebelumnya pada skenario yang digunakan sebagai masukan perangkat lunak sistem pembangkit kode uji berbasis *behavior specification*. Pembatasan dilakukan agar tidak terjadi kesalahan ketika perangkat lunak dijalankan.
- Pengujian hanya dilakukan pada tahapan *unit testing* dan *integration testing*.

1.5 Metodologi

Langkah-langkah yang dilakukan dalam skripsi ini adalah sebagai berikut:

1. Melakukan Studi literatur mengenai pengujian perangkat lunak, *unit testing*, *integration testing*, *code generation*, dan *behaviour specification*.
2. Melakukan eksplorasi perangkat lunak sistem pembangkit kode uji berbasis *behavior specification* sejenis.
3. Menganalisis kebutuhan perangkat lunak sistem pembangkit kode uji berbasis *behavior specification*.
4. Membuat rancangan desain perangkat lunak dan antarmuka perangkat lunak.

5. Membangun sistem yang menerima masukan *behaviour specification*, mengolah, dan mengubah ke bentuk *unit testing* dan *integration testing*.
6. Menguji sistem pembangkit kode uji berbasis *behavior specification*.
7. Melaporkan hasil penelitian dalam bentuk dokumen skripsi.

1.6 Sistematika Pembahasan

- Bab 1. Pendahuluan
  Bab 1 memuat tentang latar belakang masalah, rumusan masalah, tujuan, batasan masalah, dan metodologi untuk sistem pembangkit kode uji berbasis *behavior specification*.
- Bab 2. Landasan Teori
  Bab 2 memuat uraian tentang pengujian perangkat lunak, *tahapan testing*, *test-driven development*, *behavior-driven development*, skenario gherkin, *code generation*, laravel, PHPUnit, dan Dusk.
- BAB 3. Analisis
  Bab 3 memuat analisis masalah, studi kasus, analisis skenario Gherkin, dan analisis kebutuhan perangkat lunak untuk membangun untuk membangun sistem pembangkit kode uji berbasis *behavior specification*.
- BAB 4. Perancangan
  Bab 4 memuat perancangan antarmuka dan perancangan kelas sistem pembangkit kode uji berbasis *behavior specification*.
- BAB 5. Implementasi
  Bab 5 memuat lingkungan implementasi perangkat lunak, implementasi antarmuka, pengujian fungsional, pengujian eksperimental, dan kesimpulan pengujian perangkat lunak sistem pembangkit kode uji berbasis *behavior specification*.
- BAB 6. Kesimpulan dan Saran
  Bab 6 memuat kesimpulan dan saran untuk sistem pembangkit kode uji berbasis *behavior specification*.