| Research Note | | |
|---|---|---|
| **Bidang Ilmu terkait** | : | Software Engineering, Security & Data Analytics |
| **Penyusun** | : | Rita Rijayanti & Doddy Ferdiansyah |
| **Institusi** | : | Universitas Bina Nusantara & Pasundan |
| **Tahun pengerjaan** | : | 2025 - 2026 |

# Analisis Konseptual Verifikasi dan Validasi dalam Software Engineering Security dan Data Analytic: Kajian terhadap Metode, Standar, dan Praktik Pengembangan Perangkat Lunak Modern


[1]Rita Rijayanti
*dept. of Information System, BINUS Online*
Bina Nusantara University
Jakarta, Indonesia
rita.rijayanti@binus.ac.id

[2]Doddy Ferdiansyah
*dept. Informatic Enginering*
Pasundan University
Bandung, Indonesia
doddy.ferdiansyah@unpas.ac.id



**Abstrak:** Verifikasi dan validasi perangkat lunak merupakan dua aktivitas yang dapat dikaitkan dengan dunia software engineering security dan system analysis untuk memastikan perangkat lunak yang dikembangkan memiliki kualitas, keandalan, dan kesesuaian dengan spesifikasi serta mampu memenuhi kebutuhan pengguna. Seiring berkembangnya paradigma pengembangan perangkat lunak dari konvensional menuju Agile, DevOps, dan Continuous Integration/Continuous Deployment (CI/CD), implementasi verification dan validation juga mengalami perubahan yang signifikan, dimana aktivitas nya yang sebelumnya difokuskan pada tahap akhir, saat ini diterapkan secara berkelanjutan sepanjang pengembangan perangkat lunak (software development life cycle - SDLC) untuk meningkatkan kualitas produk dan mempercepat proses pengembangan. Penelitian ini menggunakan metode kajian pustaka dengan menganalisis sumber ilmiah berupa buku teks software engineering, security dan data analytics dengan standar internasional IEEE dan ISO/IEC, serta artikel ilmiah dari jurnal bereputasi. Dimana analisis dilakukan secara deskriptif-analitis untuk merangkum perkembangan konsep, metode, teknik, serta praktik vertifikasi dan validasi dalam pengembangan perangkat lunak modern.

Hasil kajian ini menunjukkan bahwa verifikasi dan validasi perangkat lunak terbukti berperan strategis dalam menjamin kualitas perangkat lunak (software quality) karena dapat mendeteksi kesalahan sejak tahap awal, mengurangi biaya perbaikan, meningkatkan keandalan sistem, serta mendukung pengembangan yang lebih adaptif. Selain itu, static code analysis, automated testing, continuous testing, dan AI-assisted software testing menunjukkan bahwa ruang lingkup verifikasi dan validasi terus berkembang seiring dengan kebutuhan industri perangkat lunak modern. Kajian ini diharapkan menjadi referensi akademik bagi dosen, mahasiswa, dan peneliti dalam memahami perkembangan konsep serta implementasi verifikasi dan validasi.



**Kata kunci:** Software Engineering, Security, Data Analysis, Verifikasi, Validasi, Software Testing, Software Quality, Software Development Life Cycle.

# 1. Pendahuluan

## 1.1 Latar Belakang

Perangkat lunak telah berkembang menjadi komponen utama dalam berbagai sistem komputasi yang digunakan di hampir seluruh sektor kehidupan, mulai dari pemerintahan, pendidikan, kesehatan, industri manufaktur, transportasi, hingga layanan keuangan. Perkembangan teknologi digital yang semakin pesat menyebabkan perangkat lunak tidak lagi hanya berfungsi sebagai alat bantu operasional, tetapi juga menjadi komponen strategis dalam mendukung proses bisnis, pengambilan keputusan, serta penyediaan layanan bagi masyarakat. Kondisi tersebut menuntut perangkat lunak memiliki kualitas tinggi sehingga mampu beroperasi secara andal, aman, efisien, dan sesuai dengan kebutuhan pengguna.

Meningkatnya kompleksitas perangkat lunak membawa konsekuensi berupa semakin besarnya potensi munculnya kesalahan (software defects). Kesalahan dapat terjadi pada berbagai tahapan pengembangan, mulai dari kesalahan dalam mendefinisikan kebutuhan pengguna, perancangan arsitektur, implementasi kode program, integrasi antarkomponen, hingga proses pengujian. Berbagai penelitian menunjukkan bahwa biaya perbaikan kesalahan akan meningkat secara signifikan apabila kesalahan baru ditemukan pada tahap akhir pengembangan atau setelah perangkat lunak digunakan oleh user atau pengguna. Oleh karena itu, para pengembang perangkat lunak berupaya menerapkan berbagai pendekatan untuk mendeteksi kesalahan sedini mungkin sehingga dampaknya terhadap biaya, waktu, dan kualitas produk dapat diminimalkan.

Dalam disiplin software engineering, security dan data analytic pendekatan yang paling banyak digunakan untuk menjamin kualitas perangkat lunak adalah verifikasi dan validasi, dimana kedua aktivitas tersebut merupakan bagian penting dari proses *Software Quality Assurance* yang bertujuan memastikan bahwa perangkat lunak dibangun sesuai spesifikasi teknis sekaligus untuk memastikan kualitas, keandalan, dan kesesuaian kebutuhan dari pengguna. Dimana verifikasi berorientasi pada pemeriksaan terhadap artefak pengembangan, seperti dokumen kebutuhan, desain sistem, dan kode program, sedangkan validasi berfokus pada evaluasi perangkat lunak melalui proses pengujian untuk memastikan bahwa sistem yang dikembangkan telah memenuhi tujuan penggunaan yang diharapkan.

Meskipun sering digunakan secara bersamaan, verifikasi dan validasi memiliki ruang lingkup dan tujuan yang berbeda. verifikasi menjawab pertanyaan Are we building the product right?, sedangkan validasi menjawab pertanyaan Are we building the right product?. Perbedaan tersebut menunjukkan bahwa keberhasilan suatu proyek perangkat lunak tidak hanya ditentukan oleh ketepatan implementasi teknis, tetapi juga oleh kemampuan perangkat lunak dalam menyelesaikan permasalahan pengguna. Dengan demikian, kedua aktivitas tersebut harus dipandang sebagai proses yang saling melengkapi, bukan sebagai aktivitas yang berdiri sendiri.

Perkembangan metodologi pengembangan perangkat lunak turut mengubah cara organisasi menerapkan verifikasi dan validasi. Pada pendekatan tradisional seperti Waterfall, aktivitas verifikasi dan validasi umumnya dilakukan secara berurutan mengikuti tahapan pengembangan. Sebaliknya, dalam pendekatan Agile dan DevOps, verifikasi dan vvalidasi diterapkan secara iteratif dan berkelanjutan melalui integrasi proses pengembangan, pengujian, dan deployment. Pendekatan ini memungkinkan kesalahan dideteksi lebih awal, meningkatkan

kecepatan umpan balik, serta mendukung pengiriman perangkat lunak yang lebih cepat tanpa mengabaikan kualitas.

Selain perubahan metodologi pengembangan, kemajuan teknologi juga memengaruhi implementasi verifikasi dan validassi. Berbagai perangkat otomatisasi seperti static code analysis, automated testing framework, continuous testing, dan AI-assisted software testing telah banyak digunakan untuk meningkatkan efisiensi proses verifikasi dan validasi, dengan pemanfaatan teknologi tersebut menunjukkan proses verifikasi dan validasi telah berkembang menjadi aktivitas yang proaktif, terintegrasi, dan didukung oleh otomatisasi.

Berdasarkan hal tersebut, team researcher menyimpulkan diperlukan sebuah kajian konseptual yang membahas perkembangan verifikasi dan validasi dari perspektif software engineering, security dan data analytics dengan meninjau konsep dasar, metode, standar internasional, serta praktik implementasinya dalam pengembangan perangkat lunak modern. Kajian ini diharapkan dapat memberikan pemahaman yang komprehensif mengenai posisi verifikasi dan validasi dalam menjamin kualitas perangkat lunak sekaligus menjadi referensi akademik bagi pengembangan penelitian di bidang Teknik Informatika dan ilmu terkait.

### 1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam kajian ini adalah sebagai berikut.

1. Bagaimana konsep verifikasi dan validasi dalam disiplin software engineering, security dan data analytics?
2. Apa saja metode dan teknik yang digunakan dalam proses verifikasi dan validasi?
3. Bagaimana implementasi verifikasi dan validasi pada setiap tahapan dalam SDLC?
4. Bagaimana perkembangan penerapan veripikasi dan validasi dalam paradigma Agile, DevOps, dan pengembangan perangkat lunak modern?
5. Apa saja tantangan yang dihadapi dalam implementasi verifikasi dan validasi pada sistem perangkat lunak yang semakin kompleks?

### 1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Mengkaji konsep dasar verifikasi dan validasi dalam konteks software engineering, security dan data analytic,
2. Menganalisis metode dan teknik verifikasi serta validasi yang digunakan dalam SDLC,
3. Menjelaskan posisi verifikasi dan validasi pada setiap tahapan SDLC,
4. Mengidentifikasi perkembangan implementasi verifikasi dan validasi pada pendekatan Agile, DevOps, dan praktik pengembangan perangkat lunak modern, dan
5. Memberikan gambaran mengenai tantangan dan arah pengembangan penelitian terkait verifikasi dan validasi.

### 1.4 Manfaat Penelitian

Kajian ini diharapkan dapat memberikan manfaat sebagai berikut.

- **Manfaat teoritis**, yaitu memperkaya pemahaman mengenai konsep, metode, standar, dan perkembangan verifikasi dan validasi dalam disiplin software engineering, security dan data analytics.

- **Manfaat praktisnya** yaitu memberikan referensi bagi dosen, mahasiswa, peneliti, maupun praktisi perangkat lunak dalam memahami implementasi verifikasi dan validation sebagai bagian dari upaya peningkatan kualitas perangkat lunak.

## 2. Metode Penelitian

Penelitian ini menggunakan pendekatan kajian pustaka yang bertujuan untuk menganalisis secara konseptual perkembangan verifikasi dan validasi dalam konteks software engineering, security dan data analytics. Pendekatan ini dipilih karena memungkinkan researcher melakukan sintesis terhadap berbagai teori, standar, dan hasil penelitian yang telah dipublikasikan, sehingga diperoleh pemahaman yang komprehensif mengenai perkembangan konsep dan praktik Verifikasi dan Validasi.

Sumber data penelitian terdiri atas tiga kelompok utama. Kelompok pertama adalah buku teks yang menjadi rujukan utama dalam bidang software engineering, security dan data analytics yang dikaitkan dengan dan software testing. Kelompok kedua berupa standar internasional yang diterbitkan oleh IEEE, ISO/IEC, dan organisasi profesi yang berkaitan dengan proses pengembangan dan pengujian perangkat lunak. Kelompok ketiga adalah artikel ilmiah yang dipublikasikan pada jurnal dan prosiding bereputasi yang membahas verifikasi, validation, *Software Testing*, *Software Quality Assurance*, serta praktik pengembangan perangkat lunak modern.

Proses kajian dilakukan melalui 4 tahapan. Tahap pertama adalah mengidentifikasi literatur yang relevan dengan ruang lingkup penelitian. Tahap kedua berupa seleksi sumber berdasarkan kredibilitas penerbit, relevansi terhadap topik, serta penggunaan edisi atau standar terbaru yang tersedia. Tahap ketiga adalah analisis konseptual terhadap definisi, metode, teknik, dan praktik verifikasi serta Validation yang dijelaskan dalam berbagai sumber. Tahap terakhir adalah sintesis hasil analisis untuk mengidentifikasi kesamaan, perbedaan, perkembangan, dan kecenderungan implementasi verifikasi dan validasi dalam pengembangan perangkat lunak modern. Dimanaanalisis data dilakukan menggunakan pendekatan deskriptif-analitis. Setiap konsep dibandingkan berdasarkan perspektif beberapa referensi utama sehingga diperoleh pemahaman yang lebih komprehensif mengenai perkembangan verifikasi dan validasi dalam disiplin rekayasa perangkat lunak. Pendekatan ini tidak bertujuan untuk menguji hipotesis, melainkan untuk menghasilkan sintesis konseptual yang dapat menjadi dasar bagi penelitian maupun pengembangan praktik Software Engineering, Security dan Data Analytics.

## 3. Verifikasi dan Validasi

### 3.1 Konsep Dasar Verifikasi dan Validasi

Verifikasi dan validasi merupakan dua aktivitas utama dalam proses *Software Quality Assurance* yang bertujuan untuk memastikan bahwa perangkat lunak dikembangkan sesuai dengan spesifikasi serta mampu memenuhi kebutuhan pengguna. Walaupun kedua istilah tersebut sering digunakan secara bersamaan, keduanya memiliki tujuan, ruang lingkup, dan pendekatan yang berbeda. Verifikasi berorientasi pada evaluasi terhadap artefak pengembangan, seperti dokumen kebutuhan, model desain, maupun kode program, untuk memastikan bahwa setiap hasil pengembangan sesuai dengan spesifikasi yang telah ditetapkan. Sebaliknya, validasi berfokus pada evaluasi perangkat lunak yang telah dapat dijalankan guna

memastikan bahwa sistem benar-benar memenuhi kebutuhan pengguna serta mampu menyelesaikan permasalahan yang menjadi tujuan pengembangannya.

Menurut IEEE Std 1012, verifikasi merupakan proses untuk mengevaluasi apakah suatu produk kerja pada setiap tahapan pengembangan telah memenuhi persyaratan yang ditetapkan pada tahap sebelumnya. Aktivitas verifikasi meliputi review, inspeksi, analisis, asesmen, serta berbagai bentuk evaluasi terhadap artefak pengembangan. Sementara itu, validasi merupakan proses untuk mengevaluasi apakah perangkat lunak yang dihasilkan telah memenuhi kebutuhan operasional pengguna dalam lingkungan penggunaan yang sebenarnya.

Pressman dan Maxim menjelaskan bahwa verifikasi menjawab pertanyaan "*Are we building the product right?",* sedangkan validasi menjawab pertanyaan "*Are we building the right product?* Pernyataan tersebut menunjukkan bahwa kualitas perangkat lunak tidak hanya ditentukan oleh ketepatan implementasi teknis, tetapi juga oleh kemampuan perangkat lunak dalam memenuhi kebutuhan bisnis maupun kebutuhan pengguna akhir. Oleh karena itu, verifikasi dan validasi harus dipandang sebagai proses yang saling melengkapi dalam menjamin kualitas perangkat lunak.

Dalam praktik rekayasa perangkat lunak modern, verifikasi dan validasi tidak lagi diposisikan sebagai aktivitas yang hanya dilakukan pada tahap akhir pengembangan. Perkembangan metodologi Agile, DevOps, serta Continuous Integration/Continuous Deployment (CI/CD) telah mengubah paradigma V&V menjadi aktivitas yang dilakukan secara berkelanjutan sepanjang SDLC. Pendekatan tersebut memungkinkan organisasi mendeteksi kesalahan lebih awal, mengurangi biaya perbaikan, meningkatkan kualitas perangkat lunak, serta mempercepat proses pengiriman produk kepada pengguna.

### 3.2 Perbedaan Verifikasi dan Validasi

Walaupun verifikasi dan Validasi memiliki tujuan yang sama, yaitu meningkatkan kualitas perangkat lunak, kedua aktivitas tersebut memiliki karakteristik yang berbeda. Verifikasi lebih berorientasi pada pemeriksaan proses dan artefak pengembangan, sedangkan validasi lebih menitikberatkan pada evaluasi perangkat lunak yang sudah dapat dijalankan. Perbedaan tersebut menyebabkan metode, teknik, dan waktu pelaksanaannya juga berbeda. Perbedaan tersebut dapat dijelaskan melalui tabel berikut (Tabel 3.1).

**Tabel 3.1 Perbandingan Verifikasi dan Validasi**

| Aspek | Verifikasi | Validasi |
|---|---|---|
| Tujuan | Memastikan produk dibangun sesuai spesifikasi | Memastikan produk memenuhi kebutuhan pengguna |
| Pertanyaan | Are we building the product right? | Are we building the right product? |
| Fokus | Dokumen, desain, model, kode | Produk perangkat lunak |
| Objek | Artefak pengembangan | Sistem yang dapat dijalankan |
| Waktu | Selama proses pengembangan | Setelah sistem dapat dieksekusi |
| Teknik | Review, Walkthrough, Inspection, Code Review, Static Analysis | Unit Test, Integration Test, System Test, UAT |
| Pelaksana | Developer, reviewer, QA | Tester dan pengguna |

| Aspek | Verifikasi | Validasi |
|---|---|---|
| Output | Artefak terValidasi | Software memenuhi kebutuhan pengguna |

Perbedaan tersebut menunjukkan bahwa verifikasi dan validasi bukan merupakan dua proses yang saling menggantikan, melainkan dua aktivitas yang saling melengkapi. Verifikasi bertujuan mencegah munculnya kesalahan sejak tahap awal pengembangan, sedangkan validasi memastikan bahwa perangkat lunak yang telah dikembangkan benar-benar memberikan solusi terhadap kebutuhan pengguna. Kombinasi keduanya menjadi fondasi utama dalam penerapan *Software Quality Assurance* pada berbagai model pengembangan perangkat lunak.

### 3.3 Teknik Verifikasi

Verifikasi merupakan proses sistematis untuk mengevaluasi apakah setiap artefak yang dihasilkan selama proses pengembangan telah memenuhi persyaratan pada tahap sebelumnya. Berbeda dengan validasi yang berorientasi pada pengujian perangkat lunak yang telah dapat dijalankan, verifikasi lebih banyak dilakukan terhadap dokumen, model, desain, maupun kode program sebelum sistem digunakan. IEEE Std 1012 menjelaskan bahwa aktivitas verifikasi mencakup analisis, evaluasi, review, inspeksi, assessment, dan pengujian terhadap produk maupun proses pengembangan. Pendekatan ini bertujuan untuk memastikan bahwa setiap keluaran pada setiap tahapan sesuai dengan spesifikasi yang ditetapkan, sehingga kesalahan dapat dideteksi sedini mungkin. Dimana dalam praktik software engineering, verifikasi tidak hanya bertujuan untuk menemukan kesalahan, tetapi juga untuk menjaga konsistensi proses pengembangan. Organisasi yang menerapkan verifikasi secara sistematis umumnya memiliki dokumentasi yang lebih baik, tingkat kesalahan implementasi yang lebih rendah, serta proses pemeliharaan yang lebih mudah karena setiap perubahan dapat ditelusuri melalui artefak yang telah diverifikasi. Berikut, beberapa teknik verifikasi yang umum digunakan dijelaskan sebagai berikut:

#### 3.3.1 Requirement Review

Requirement Review merupakan proses pemeriksaan terhadap dokumen *Software Requirements Specification*. Aktivitas ini bertujuan untuk memastikan bahwa seluruh kebutuhan telah dituliskan secara jelas, lengkap, konsisten, tidak ambigu, dan dapat diverifikasi. Kesalahan pada tahap kebutuhan merupakan salah satu penyebab utama kegagalan proyek perangkat lunak karena dapat memengaruhi seluruh tahapan pengembangan berikutnya. Pada tahap ini, tim pengembang bersama analis sistem dan pemangku kepentingan melakukan penelaahan terhadap setiap kebutuhan untuk memastikan bahwa seluruh kebutuhan memiliki tujuan yang jelas, tidak saling bertentangan, serta dapat diimplementasikan secara teknis.

#### 3.3.2 Design Review

Setelah kebutuhan disepakati, proses verifikasi dilanjutkan pada tahap perancangan sistem. *Design Review* dilakukan untuk mengevaluasi apakah rancangan perangkat lunak telah memenuhi seluruh kebutuhan fungsional maupun nonfungsional. Artefak yang biasanya ditinjau meliputi:

- Arsitektur perangkat lunak
- Diagram,
- Rancangan basis data,
- Rancangan antarmuka pengguna,
- Desain API, dan
- Spesifikasi modul.

Melalui proses ini, berbagai permasalahan desain dapat ditemukan sebelum memasuki tahap implementasi, sehingga mengurangi risiko perubahan besar pada tahap berikutnya.

### 3.3.3 Code Review

*Code Review* merupakan salah satu teknik verifikasi yang paling banyak digunakan pada pengembangan perangkat lunak modern. Aktivitas ini dilakukan dengan memeriksa kode program secara sistematis untuk mengidentifikasi kesalahan logika, pelanggaran standar pengkodean, duplikasi kode, serta potensi kerentanan keamanan. Berbeda dengan compiler yang hanya memeriksa kesalahan sintaksis, *Code Review* memungkinkan pengembang mengevaluasi kualitas implementasi secara menyeluruh. Proses ini dapat dilakukan secara manual oleh sesama pengembang melalui mekanisme *pull request review* pada sistem pengelolaan kode sumber seperti Git. Berbagai penelitian menunjukkan bahwa *Code Review* mampu meningkatkan kualitas perangkat lunak sekaligus menjadi media berbagi pengetahuan antaranggota tim pengembang.

### 3.3.4 Walkthrough

*Walkthrough* merupakan teknik verifikasi yang dilakukan melalui diskusi informal antara pengembang dengan anggota tim lainnya. Pada teknik ini, pengembang menjelaskan artefak yang telah dibuat, sedangkan peserta lain memberikan masukan mengenai kemungkinan kesalahan maupun alternatif Solusi, yang banyak digunakan karena relatif mudah dilaksanakan, tidak membutuhkan prosedur formal yang kompleks, dan efektif untuk menemukan kesalahan konseptual sejak tahap awal pengembangan.

### 3.3.5 Formal Inspection

*Inspection* merupakan bentuk verifikasi yang lebih formal dibandingkan *Walkthrough*. Teknik ini memiliki prosedur yang terstruktur, melibatkan beberapa reviewer dengan peran tertentu, serta menghasilkan dokumentasi hasil Inspection. Dimana hal ini umumnya diterapkan pada proyek perangkat lunak berskala besar atau sistem kritis karena mampu menemukan berbagai jenis kesalahan sebelum proses implementasi dilakukan. Walaupun membutuhkan waktu lebih lama dibandingkan *Walkthrough*, teknik ini memberikan tingkat kepercayaan yang lebih tinggi terhadap kualitas artefak yang diperiksa.

### 3.3.6 Static Code Analysis

Perkembangan perangkat lunak modern mendorong penggunaan berbagai alat bantu untuk melakukan verifikasi secara otomatis melalui teknik *Static Code Analysis*. Teknik ini menganalisis kode sumber tanpa mengeksekusi program sehingga berbagai potensi kesalahan dapat dideteksi lebih awal.

Beberapa aspek yang dapat dianalisis antara lain:

- Pelanggaran standar pengkodean,
- Kompleksitas kode,
- *Code smell*,
- Potensi *memory leak*,
- Kerentanan keamanan,
- Duplikasi kode, dan
- Bug.

Perangkat seperti SonarQube, PMD, Checkstyle, SpotBugs, ESLint, maupun Pylint banyak digunakan dalam proses *Continuous Integration* untuk melakukan pemeriksaan otomatis setiap kali terjadi perubahan kode. Dimana penggunaan Static Code Analysis tidak menggantikan Code Review, tetapi melengkapinya dengan melakukan pemeriksaan yang objektif dan konsisten. Kombinasi kedua teknik tersebut terbukti mampu meningkatkan kualitas kode sekaligus mengurangi jumlah kesalahan yang ditemukan pada tahap pengujian.

Berdasarkan berbagai teknik tersebut dapat disimpulkan bahwa verifikasi memiliki karakteristik utama berupa evaluasi terhadap artefak pengembangan sebelum perangkat lunak dijalankan. Semakin awal verifikasi dilakukan, semakin kecil biaya yang diperlukan untuk memperbaiki kesalahan. Oleh karena itu, organisasi pengembang perangkat lunak modern menerapkan verifikasi secara berulang pada setiap tahapan SDLC melalui kombinasi review manual dan otomatisasi menggunakan berbagai perangkat lunak pendukung.

## 3.4 Teknik Validasi

### 3.4.1 Unit Testing

*Unit Testing* merupakan pengujian terhadap unit program terkecil, seperti fungsi, prosedur, atau kelas. Tujuan utama *Unit Testing* adalah memastikan bahwa setiap unit bekerja sesuai rancangan sebelum digabungkan dengan komponen lainnya. Framework seperti JUnit, NUnit, PyTest, dan Google Test memungkinkan Unit Testing dilakukan secara otomatis sehingga proses pengujian dapat diulang setiap kali terjadi perubahan kode.

### 3.4.2 Integration Testing

Setelah masing-masing unit berfungsi dengan baik, dilakukan *Integration Testing* untuk memastikan bahwa komunikasi antar modul berlangsung secara benar. Pengujian ini bertujuan mendeteksi kesalahan yang muncul akibat interaksi antar komponen, seperti kesalahan pertukaran data, ketidaksesuaian antarmuka, maupun konflik dependensi.

### 3.4.3 System Testing

*System Testing* mengevaluasi perangkat lunak sebagai satu kesatuan sistem. Seluruh fungsi diuji berdasarkan kebutuhan fungsional maupun nonfungsional yang telah ditetapkan.

Pada tahap ini dilakukan berbagai jenis pengujian, antara lain:

- *Functional Testing*
- *Performance Testing*
- *Load Testing*
- *Stress Testing*
- *Security Testing*
- *Compatibility Testing*

- *Usability Testing*

Melalui pengujian tersebut dapat diketahui apakah perangkat lunak telah memenuhi karakteristik kualitas sebagaimana dijelaskan dalam model kualitas perangkat lunak ISO/IEC 25010.

### 3.4.4 User Acceptance Testing (UAT)

*User Acceptance Testing* (UAT) merupakan tahap akhir validasi sebelum perangkat lunak diimplementasikan. Pengujian dilakukan oleh pengguna atau perwakilan pengguna dengan menggunakan skenario yang mencerminkan proses bisnis sebenarnya.

Tujuan UAT bukan lagi mencari kesalahan teknis, tetapi memastikan bahwa perangkat lunak telah memberikan solusi terhadap kebutuhan operasional organisasi. Keberhasilan UAT menjadi salah satu indikator bahwa perangkat lunak siap digunakan pada lingkungan nyata.

### 3.4.5 Regression Testing

Perubahan perangkat lunak berpotensi menimbulkan kesalahan baru pada fungsi yang sebelumnya telah berjalan dengan baik. Oleh karena itu, setiap perubahan perlu diikuti dengan *Regression Testing* untuk memastikan bahwa modifikasi yang dilakukan tidak memengaruhi fungsi lain dalam sistem.

Pada praktik DevOps, Regression Testing umumnya dijalankan secara otomatis melalui pipeline CI/CD sehingga kualitas perangkat lunak tetap terjaga meskipun perubahan dilakukan secara cepat dan berulang.

Validasi memiliki karakteristik yang berbeda dengan verifikasi karena seluruh aktivitas dilakukan melalui eksekusi perangkat lunak. Meskipun demikian, keberhasilan validasi sangat dipengaruhi oleh kualitas verifikasi yang dilakukan pada tahap sebelumnya. Semakin baik proses verifikasi, semakin kecil kemungkinan ditemukan kesalahan besar pada saat validasi. Oleh karena itu, kedua aktivitas tersebut harus dipandang sebagai satu kesatuan dalam proses penjaminan kualitas perangkat lunak, bukan sebagai dua proses yang berdiri sendiri.

## 3.5 Verifikasi dan Validasi pada Software Development Life Cycle (SDLC)

Verifikasi dan validasi merupakan aktivitas yang diterapkan secara sistematis pada setiap tahapan SDLC. Berbeda dengan pandangan tradisional yang menempatkan proses pengujian hanya pada tahap akhir pengembangan, pendekatan modern menekankan bahwa kualitas perangkat lunak harus dibangun sejak tahap awal melalui integrasi aktivitas verifikasi dan validasi di seluruh siklus pengembangan. Pendekatan ini memungkinkan kesalahan dideteksi sedini mungkin sehingga biaya perbaikan, risiko kegagalan proyek, serta waktu pengembangan dapat diminimalkan.

Pada tahap requirements analysis, verifikasi dilakukan melalui *requirement review* untuk memastikan bahwa kebutuhan pengguna telah dirumuskan secara lengkap, konsisten, tidak ambigu, dan dapat diimplementasikan. Validasi pada tahap ini biasanya dilakukan melalui *requirements validation*, misalnya melalui diskusi, wawancara, prototyping, maupun *use case walkthrough* guna memastikan bahwa kebutuhan yang didokumentasikan benar-benar mencerminkan kebutuhan operasional pengguna.

Memasuki tahap system design, aktivitas verifikasi dilakukan melalui *design review*, evaluasi arsitektur, serta pemeriksaan berbagai model seperti UML, rancangan basis data, dan desain

antarmuka pengguna. Validasi pada tahap ini dapat dilakukan menggunakan prototipe (*prototype validation*) sehingga pengguna dapat memberikan umpan balik terhadap rancangan sebelum proses implementasi dimulai.

Pada tahap implementasi (coding), verifikasi dilakukan melalui *code review*, *walkthrough*, *inspection*, serta *static code analysis*. Berbagai teknik tersebut bertujuan memastikan bahwa kode program telah mengikuti standar pengkodean, memenuhi spesifikasi desain, serta bebas dari berbagai potensi kesalahan maupun kerentanan keamanan. Validasi mulai dilakukan melalui *unit testing* untuk memastikan bahwa setiap fungsi atau modul bekerja sesuai dengan kebutuhan yang telah ditentukan.

Selanjutnya, pada tahap integrasi, validasi dilakukan melalui *integration testing* untuk memastikan bahwa komunikasi antarmodul berjalan dengan baik tanpa terjadi konflik antarmuka maupun pertukaran data. Verifikasi pada tahap ini tetap dilakukan melalui evaluasi konfigurasi, pemeriksaan dependensi, serta analisis kualitas integrasi sistem.

Pada tahap system testing, aktivitas validasi menjadi lebih dominan melalui *system testing*, *performance testing*, *security testing*, *usability testing*, dan berbagai bentuk pengujian lainnya sesuai karakteristik perangkat lunak. Sementara itu, verifikasi tetap dilakukan melalui evaluasi terhadap kelengkapan dokumentasi pengujian, *requirements traceability*, serta kepatuhan terhadap standar pengembangan yang berlaku.

Tahap terakhir meliputi deployment dan maintenance. Sebelum sistem diimplementasikan, validasi dilakukan melalui *UAT* untuk memastikan bahwa perangkat lunak telah memenuhi kebutuhan operasional pengguna. Setelah sistem digunakan, aktivitas verifikasi dan Validasi tetap berlanjut melalui *configuration review*, *change verifikasi*, *regression testing*, serta pemantauan kualitas perangkat lunak selama proses pemeliharaan. Pendekatan ini menunjukkan bahwa verifikasi dan Validasi merupakan proses yang berlangsung sepanjang siklus hidup perangkat lunak, bukan hanya pada tahap akhir.

Untuk memperjelas penerapan kedua aktivitas tersebut pada setiap tahapan SDLC, Tabel 3.2 menyajikan ringkasan hubungan antara tahapan SDLC dengan aktivitas Verifikasi dan Validasi.

**Tabel 3.2 Implementasi Verifikasi dan Validasi pada SDLC**

| Tahap SDLC | Verifikasi | Validasi |
|---|---|---|
| *Requirements Analysis* | *Requirement Review* | *Requirement Validation* |
| *System Design* | *Design Review* | *Prototype Validation* |
| *Implementation* | *Code Review, Static Analysis* | *Unit Testing* |
| *Integration* | *Configuration Review* | *Integration Testing* |
| *System Testing* | *Test Documentation Review* | *System Testing* |
| *Deployment* | *Release Verification* | *User Acceptance Testing* |
| *Maintenance* | *Change Review* | *Regression Testing* |

Berdasarkan uraian tersebut dapat disimpulkan bahwa Verifikasi dan Validasi merupakan aktivitas yang saling melengkapi pada seluruh tahapan SDLC. Verifikasi berfokus pada pemeriksaan setiap artefak yang dihasilkan selama proses pengembangan, sedangkan validasi memastikan bahwa perangkat lunak yang dihasilkan mampu memenuhi kebutuhan pengguna.

Integrasi kedua aktivitas tersebut menjadi landasan utama dalam menghasilkan perangkat lunak yang berkualitas, andal, serta sesuai dengan tujuan pengembangannya.

### 3.5 Perkembangan Verifikasi dan Validasi pada Era Agile dan DevOps

Perkembangan metodologi pengembangan perangkat lunak dari pendekatan tradisional menuju Agile dan DevOps telah mengubah cara organisasi menerapkan verifikasi dan validasi. Pada model pengembangan konvensional, seperti Waterfall, aktivitas Verifikasi dan Validasi umumnya dilakukan secara berurutan mengikuti tahapan SDLC. Verifikasi lebih banyak dilaksanakan pada tahap analisis, desain, dan implementasi, sedangkan Validasi dilakukan setelah perangkat lunak selesai dikembangkan melalui serangkaian proses pengujian. Pendekatan tersebut sesuai untuk proyek yang memiliki kebutuhan relatif stabil, namun kurang adaptif terhadap perubahan kebutuhan pengguna yang terjadi selama proses pengembangan.

Munculnya metodologi Agile membawa perubahan mendasar pada filosofi pengembangan perangkat lunak. Agile menekankan pengembangan secara iteratif dan inkremental dengan siklus pengembangan yang lebih pendek (*iteration* atau *sprint*). Setiap iterasi menghasilkan bagian perangkat lunak yang dapat digunakan, sehingga proses evaluasi terhadap kualitas produk dapat dilakukan lebih cepat. Dalam konteks ini, verifikasi dan validasi tidak lagi dipandang sebagai aktivitas yang dilakukan pada tahap akhir, tetapi menjadi bagian yang terintegrasi dalam setiap siklus pengembangan.

Pada pendekatan Agile, verifikasi dilakukan secara berkelanjutan melalui berbagai aktivitas seperti *requirement refinement*, *design review*, *code review*, *pair programming*, dan *static code analysis*. Seluruh aktivitas tersebut bertujuan memastikan bahwa setiap perubahan yang dilakukan selama sprint tetap sesuai dengan kebutuhan yang telah disepakati serta memenuhi standar kualitas pengembangan perangkat lunak. Pendekatan ini memungkinkan tim menemukan potensi kesalahan sejak tahap awal sehingga risiko akumulasi kesalahan pada akhir proyek dapat diminimalkan.

Sementara itu, validasi dalam Agile dilakukan melalui pengujian yang dilakukan secara berulang pada setiap iterasi. Setiap fitur baru yang dikembangkan akan diuji menggunakan *unit testing*, *integration testing*, dan *acceptance testing* sebelum dinyatakan selesai (*Definition of Done*). Keterlibatan pengguna selama proses pengembangan juga menjadi karakteristik penting dalam validasi berbasis Agile karena umpan balik yang diberikan dapat segera digunakan sebagai dasar perbaikan pada iterasi berikutnya. Dengan demikian, kualitas perangkat lunak tidak hanya dievaluasi berdasarkan kesesuaian terhadap spesifikasi teknis, tetapi juga berdasarkan kemampuan perangkat lunak dalam memenuhi kebutuhan pengguna yang terus berkembang.

Perkembangan selanjutnya ditandai dengan munculnya konsep DevOps yang mengintegrasikan proses *development* dan *operations*. DevOps tidak hanya berfokus pada percepatan proses pengembangan, tetapi juga pada otomatisasi proses pembangunan, pengujian, dan distribusi perangkat lunak. Dalam lingkungan DevOps, verifikasi dan validasi menjadi bagian dari proses otomatis yang berjalan secara berkesinambungan melalui pipeline *Continuous Integration* (CI) dan *Continuous Delivery/Continuous Deployment* (CD).

Pada proses Continuous Integration, setiap perubahan kode yang dikirimkan ke repositori akan secara otomatis memicu serangkaian aktivitas verifikasi. Aktivitas tersebut mencakup kompilasi kode, pemeriksaan standar pengkodean, *static code analysis*, pemeriksaan

kerentanan keamanan, serta eksekusi *unit test*. Apabila salah satu proses tersebut gagal, perubahan kode tidak akan dilanjutkan ke tahap berikutnya. Pendekatan ini memungkinkan kesalahan dideteksi dalam waktu yang sangat singkat sehingga biaya perbaikannya menjadi lebih rendah dibandingkan apabila kesalahan baru ditemukan setelah sistem diimplementasikan.

Validasi dalam lingkungan DevOps juga mengalami transformasi melalui penerapan *Continuous Testing*. Pengujian tidak lagi dilakukan hanya menjelang implementasi sistem, tetapi dilaksanakan secara otomatis pada setiap perubahan perangkat lunak. Berbagai jenis pengujian, seperti *integration testing*, *system testing*, *regression testing*, *performance testing*, hingga *security testing*, dapat diintegrasikan ke dalam pipeline pengembangan sehingga kualitas perangkat lunak dapat dipantau secara berkelanjutan. Pendekatan ini memungkinkan organisasi melakukan rilis perangkat lunak dengan frekuensi yang lebih tinggi tanpa mengurangi tingkat keandalan sistem.

Selain meningkatkan kualitas perangkat lunak, integrasi verifikasi dan validasi dalam Agile dan DevOps memberikan berbagai manfaat strategis. Organisasi dapat memperoleh umpan balik lebih cepat, mempercepat waktu distribusi perangkat lunak (*time-to-market*), mengurangi jumlah *software defects*, meningkatkan kolaborasi antaranggota tim, serta mempermudah proses pemeliharaan sistem. Oleh karena itu, penerapan verifikasi dan validasi secara berkelanjutan telah menjadi salah satu *best practice* dalam pengembangan perangkat lunak modern.

Secara keseluruhan, perkembangan menuju Agile dan DevOps menunjukkan perubahan paradigma verifikasi dan validasi dari aktivitas yang bersifat reaktif menjadi aktivitas yang bersifat proaktif dan berkelanjutan. Fokus utama tidak lagi hanya menemukan kesalahan setelah pengembangan selesai, tetapi mencegah munculnya kesalahan sejak awal melalui integrasi aktivitas verifikasi dan validasi di seluruh siklus pengembangan.

### 3.6 Tantangan dan Arah Perkembangan Verifikasi dan Validasi

Perkembangan teknologi perangkat lunak yang semakin kompleks membawa tantangan baru dalam implementasi verifikasi dan validasi. Arsitektur berbasis microservices, cloud computing, Internet of Things (IoT), serta sistem berbasis *Artificial Intelligence* (AI) menyebabkan proses pengujian tidak lagi terbatas pada pemeriksaan fungsi perangkat lunak, tetapi juga mencakup aspek interoperabilitas, keamanan, skalabilitas, privasi, dan keandalan sistem.

Selain itu, tingginya frekuensi perubahan perangkat lunak pada lingkungan Agile dan DevOps menuntut organisasi menerapkan otomatisasi verifikasi dan validasi melalui *Continuous Integration* dan *Continuous Testing*. Pendekatan tersebut membutuhkan infrastruktur pengujian yang memadai, kemampuan pengelolaan pipeline CI/CD, serta sumber daya manusia yang memiliki kompetensi dalam otomatisasi pengujian.

Pada sistem berbasis AI, tantangan menjadi semakin kompleks karena kualitas perangkat lunak dipengaruhi oleh kualitas data pelatihan, perubahan performa model, bias algoritma, serta aspek explainability. Oleh karena itu, penelitian mengenai verifikasi dan Validasi pada sistem AI saat ini tidak hanya berfokus pada pengujian perangkat lunak, tetapi juga mencakup evaluasi kualitas model, fairness, transparansi, keamanan, dan akuntabilitas.

Ke depan, penerapan AI diperkirakan akan semakin memperkuat proses verifikasi dan validasi melalui pemanfaatan *AI-assisted testing, automatic test generation, predictive defect analysis, autonomous testing*, serta *intelligent quality assurance*. Integrasi teknologi tersebut diharapkan mampu meningkatkan efisiensi pengujian sekaligus mempercepat proses pengembangan perangkat lunak tanpa mengurangi kualitas produk yang dihasilkan.

### 3.7 Standar Verifikasi dan Validasi

Selain berkembang dari sisi metode dan teknik, penerapan verifikasi dan validasi juga didukung oleh berbagai standar internasional yang menjadi acuan dalam proses pengembangan perangkat lunak. Standar-standar tersebut memberikan pedoman mengenai aktivitas, dokumentasi, proses, serta karakteristik kualitas yang harus dipenuhi agar perangkat lunak memiliki tingkat keandalan yang tinggi. Penggunaan standar internasional membantu organisasi menerapkan proses verifikasi dan validasi secara konsisten sekaligus meningkatkan kualitas produk perangkat lunak yang dihasilkan.

Salah satu standar yang secara khusus mengatur proses verifikasi dan validasi adalah IEEE Std 1012-2024. Standar ini menjelaskan ruang lingkup aktivitas verifikasi dan validasi sepanjang siklus hidup sistem, termasuk proses perencanaan, pelaksanaan, dokumentasi, evaluasi hasil, serta pengelolaan risiko. IEEE 1012 menekankan bahwa verifikasi dan validasi merupakan aktivitas yang dilakukan secara independen dan berkesinambungan untuk memastikan kesesuaian produk terhadap spesifikasi maupun kebutuhan pengguna.

Dalam aspek kualitas perangkat lunak, ISO/IEC 25010 menjadi acuan utama dalam mengevaluasi karakteristik kualitas perangkat lunak. Standar ini mendefinisikan delapan karakteristik kualitas, yaitu *functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability,* dan *portability*. Berbagai aktivitas Validasi, khususnya *system testing* dan *acceptance testing*, umumnya mengacu pada karakteristik kualitas tersebut.

Sementara itu, ISO/IEC/IEEE 12207 memberikan kerangka kerja mengenai proses siklus hidup perangkat lunak (*software life cycle processes*). Standar ini menjelaskan bagaimana aktivitas verifikasi dan validasi diintegrasikan ke dalam setiap tahapan pengembangan mulai dari analisis kebutuhan hingga pemeliharaan perangkat lunak. Dengan demikian, standar tersebut menjadi dasar penerapan *Software Quality Assurance* dalam berbagai organisasi pengembang perangkat lunak.

Selain standar internasional tersebut, *International Software Testing Qualifications Board (ISTQB)* menyediakan panduan *best practices* dalam pengujian perangkat lunak. Silabus ISTQB banyak digunakan sebagai referensi dalam penerapan teknik *unit testing, integration testing, system testing,* maupun *acceptance testing* serta menjadi dasar kompetensi bagi praktisi software testing di berbagai negara.

Secara keseluruhan, keberadaan berbagai standar internasional menunjukkan bahwa verifikasi dan validasi tidak hanya merupakan konsep teoritis, tetapi juga telah menjadi praktik yang terstandarisasi dalam industri perangkat lunak modern. Integrasi antara standar internasional, metode pengembangan, serta otomatisasi pengujian memberikan landasan yang kuat dalam menghasilkan perangkat lunak yang berkualitas tinggi, aman, dan sesuai dengan kebutuhan pengguna.

**Tabel 3.3 Standar Internasional Verifikasi dan Validasi**

| Standar | Fokus | Kontribusi terhadap V&V |
|---|---|---|
| IEEE Std 1012:2024 | Verification & Validation Process | Pedoman aktivitas V&V sepanjang siklus hidup |
| ISO/IEC 25010 | Software Quality Model | Acuan karakteristik kualitas perangkat lunak |
| ISO/IEC/IEEE 12207 | Software Life Cycle Processes | Integrasi V&V pada setiap tahapan SDLC |
| ISTQB CTFL 4.0 | Software Testing | Praktik terbaik teknik pengujian perangkat lunak |

## 4. Kesimpulan

Verifikasi dan Validasi merupakan dua komponen utama dalam pengembangan perangkat lunak yang berperan dalam menjamin kualitas perangkat lunak sepanjang siklus hidup pengembangannya. Meskipun memiliki tujuan yang berbeda, verifikasi dan validasi merupakan proses yang saling melengkapi. Verifikasi berfokus pada pemeriksaan terhadap artefak pengembangan untuk memastikan bahwa perangkat lunak dibangun sesuai dengan spesifikasi yang telah ditetapkan (*building the product right*), sedangkan validasi berorientasi pada evaluasi perangkat lunak untuk memastikan bahwa sistem yang dihasilkan mampu memenuhi kebutuhan dan harapan pengguna (*building the right product*).

Hasil kajian menunjukkan bahwa penerapan verifikasi dan validasi tidak hanya terbatas pada tahap pengujian perangkat lunak, tetapi telah menjadi aktivitas yang terintegrasi pada seluruh tahapan SDLC. Mulai dari analisis kebutuhan, perancangan sistem, implementasi, integrasi, pengujian, hingga pemeliharaan, setiap tahapan memerlukan mekanisme verifikasi maupun Validasi yang sesuai. Pendekatan ini memungkinkan organisasi mendeteksi kesalahan sejak tahap awal pengembangan sehingga biaya perbaikan dapat ditekan dan kualitas perangkat lunak dapat ditingkatkan.

Berbagai teknik verifikasi, seperti *requirement review*, *design review*, *code review*, *walkthrough*, *formal inspection*, dan *static code analysis*, berkontribusi dalam memastikan konsistensi serta kepatuhan terhadap spesifikasi. Sementara itu, teknik validasi, seperti *unit testing*, *integration testing*, *system testing*, *user acceptance testing*, dan *regression testing*, berperan dalam memastikan bahwa perangkat lunak mampu menjalankan fungsi yang diharapkan dalam lingkungan operasional. Kombinasi kedua kelompok teknik tersebut membentuk mekanisme penjaminan kualitas yang komprehensif.

Kajian ini juga menunjukkan bahwa perkembangan metodologi Agile dan DevOps telah mengubah paradigma verifikasi dan validasi dari aktivitas yang bersifat sekuensial menjadi aktivitas yang dilakukan secara iteratif dan berkelanjutan. Integrasi *Continuous Integration* (CI), *Continuous Delivery/Continuous Deployment* (CD), *Continuous Testing*, serta otomatisasi pengujian memungkinkan proses Verifikasi dan Validasi dilakukan secara lebih cepat, konsisten, dan efisien. Pendekatan tersebut memberikan umpan balik yang lebih cepat kepada tim pengembang serta mendukung penyampaian perangkat lunak dengan frekuensi rilis yang lebih tinggi tanpa mengurangi kualitas produk.

Selain itu, perkembangan AI memberikan peluang sekaligus tantangan baru bagi implementasi verifikasi dan validasi. AI dimanfaatkan untuk membantu analisis kualitas kode, menghasilkan kasus uji secara otomatis, mendeteksi anomali, serta meningkatkan efisiensi proses pengujian. Di sisi lain, perangkat lunak berbasis AI memerlukan pendekatan validasi yang lebih luas karena kualitas sistem dipengaruhi oleh model pembelajaran, kualitas data, potensi bias, serta perubahan performa model dari waktu ke waktu. Oleh karena itu, proses verifikasi dan validasi pada sistem berbasis AI perlu mempertimbangkan aspek teknis, kualitas data, transparansi, keamanan, dan akuntabilitas.

Secara keseluruhan, verifikasi dan validasi tetap menjadi fondasi utama dalam pembangunan perangkat lunak yang berkualitas. Seiring meningkatnya kompleksitas sistem dan berkembangnya teknologi pengembangan perangkat lunak, penerapan verifikasi dan validasi yang sistematis, berkelanjutan, dan didukung oleh otomatisasi akan semakin menentukan keberhasilan proyek perangkat lunak. Kajian ini diharapkan dapat menjadi referensi konseptual bagi dosen, mahasiswa, peneliti, maupun praktisi dalam memahami perkembangan teori, metode, dan praktik verifikasi dan validasi pada pengembangan perangkat lunak modern dikaitkan dengan pemahaman software engineering, security, dan system analysis.

**Daftar Pustaka**


[1] Ammann, P., & Offutt, J. (2017). *Introduction to Software Testing* (2nd ed.). Cambridge University Press.

[2] Beck, K., et al. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

[3] Bourque, P., & Fairley, R. E. (Eds.). (2024). *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide V4).* IEEE Computer Society.

[4] International Organization for Standardization. (2011). *ISO/IEC 25010:2011 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models.*

[5] International Organization for Standardization. (2017). *ISO/IEC/IEEE 12207:2017 Systems and Software Engineering—Software Life Cycle Processes.*

[6] International Software Testing Qualifications Board (ISTQB). (2023). *Certified Tester Foundation Level (CTFL) Syllabus Version 4.0.*

[7] Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). John Wiley & Sons.

[8] Pressman, R. S., & Maxim, B. R. (2023). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.

[9] Sommerville, I. (2021). *Software Engineering* (11th ed.). Pearson.

[10] IEEE. (2024). *IEEE Std 1012-2024: IEEE Standard for System, Software, and Hardware Verification and Validation.*