Makalah Makassar, 25 September 2025

COMPONENT-BASED DEVELOPMENT

Disusun oleh:

KELOMPOK II

NUR AZANI LABADJA (13020230010)
WAHYU MELANIE PRATIWI (13020230015)
MILDAYANTI (13020230084)
AULYA MAHARANI AHMAD (13020230316)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS MUSLIM INDONESIA

MAKASSAR

2025

DAFTAR ISI

BAB I

PENDAHULUAN

A. Latar Belakang

Seiring berjalannya waktu, perangkat lunak semakin kompleks. Integrasi sistem, kebutuhan bisnis yang berkembang, dan harapan pengguna membuat pengembangan perangkat lunak dengan pendekatan tradisional, seperti Model Waterfall, semakin tidak efisien. Kompleksitas ini memerlukan pengembangan perangkat lunak yang lebih modular, fleksibel, dan efisien. Salah satu paradigma yang mencoba mengatasi masalah ini adalah Pengembangan Perangkat Lunak Berbasis Komponen (CBSD).

CBSD memungkinkan pengembang perangkat lunak untuk membangun sistem dengan merakit komponen yang sudah ada sebelumnya, baik yang dikembangkan secara internal atau yang diperoleh dari eksternal (misalnya, Komersial Siap Pakai (COTS)). Konsep ini mirip dengan pembuatan produk dari bagian-bagian yang telah dirakit sebelumnya. Dengan repositori komponen yang distandarisasi, pengembang dapat memprioritaskan arsitektur sistem dan integrasi, dibandingkan dengan menulis setiap baris kode dari awal. Ini tidak hanya mengurangi biaya tetapi juga memperpendek waktu pengembangan dan meningkatkan kualitas perangkat lunak, karena komponen yang tersedia lebih mungkin untuk lulus penilaian kualitas awal.[1]

Namun, meskipun menawarkan banyak manfaat, CBSD juga menghadapi banyak tantangan. Ketersediaan komponen yang memenuhi kebutuhan, masalah kompatibilitas komponen, dan penggunaan ulang merupakan pertimbangan penting. Oleh karena itu, penelitian tentang metrik penggunaan ulang dan model peningkatan kualitas komponen sering dilakukan untuk mendukung implementasi CBSD. [2]

B. Tujuan Penulisan

1. Menguraikan fase dan tahapan pengembangan perangkat lunak berdasarkan komponennya.
2. Menguraikan karakteristik utama CBSD serta bagaimana model ini diterapkan dalam pengembangan perangkat lunak.
3. Menggunakan kekuatan dan kelemahan CBSD sebagai metode pengembangan perangkat lunak.
4. Membandingkan CBSD dengan model pengembangan perangkat lunak lainnya, seperti Waterfall, Prototyping, dan Spiral.
5. Mengidentifikasi beragam alat pendukung yang memfalisitasi penerapan CBSD secara praktis.

Melalui uraian ini, diharapkan pembaca memperoleh pemahaman menyeluruh tentang CBSD, mencakup dimensi teori maupun aplikasi nyata, serta mampu mengevaluasi peluang penggunaannya dalam lingkungan pengembangan perangkat lunak kontemporer.

BAB II

KAJIAN TEORITIS

A. Definisi dan Sejarah

Sebenarnya, pengembangan perangkat lunak berbasis komponen (CBSD) memiliki sejarah yang cukup panjang. Ini semua dimulai dengan ide Douglas McIlroy pada Konferensi Teknik Perangkat Lunak NATO pada tahun 1968. Saat berbicara, dia menekankan fakta bahwa software dapat dibuat dengan cara yang mirip dengan menyusun bagian-bagian elektronik, yaitu dengan membuat modul-modul yang sudah jadi dan kemudian menggabungkannya untuk membuat aplikasi yang lebih kompleks. Pemikiran ini muncul sebagai solusi untuk krisis software yang terjadi pada saat itu, ketika banyak proyek tertunda, biaya meningkat, dan hasilnya seringkali tidak dapat diandalkan.

Walaupun ide McIlroy tampak kreatif, kekurangan teknologi membuatnya sulit diterapkan. Kemunculan pemrograman berorientasi objek (OOP) baru terjadi pada tahun 1980-an. Metode ini memungkinkan penggunaan ulang kode melalui kelas dan objek; namun, itu terlalu berfokus pada detail kecil, yang membuatnya tidak efektif untuk sistem yang lebih besar. Akibatnya, para pengembang dan ahli mulai menyadari pentingnya meningkatkan reuse di tingkat yang lebih luas, yaitu dengan menggunakan komponen software sebagai komponen utama.

Pada tahun 1990-an, CBSD semakin dikenal karena berbagai alatnya, termasuk middleware dan kerangka kerja yang dibuat khusus untuknya. Teknologi seperti JavaBeans, Enterprise JavaBeans (EJB), dan CORBA (Common Object Request Broker Architecture) sangat penting karena memungkinkan komponen disambungkan ke berbagai platform. Selanjutnya, CBSD berkembang dari konsep teoretis menjadi pendekatan praktis untuk penerapan skala perusahaan [3].

Banyak penelitian juga telah mendefinisikan CBSD. Basha dan Moiz (2012) menyatakan bahwa CBSD adalah pendekatan rekayasa software yang menekankan pembentukan sistem melalui penggabungan elemen-elemen mandiri, didukung oleh perjanjian antarmuka yang jelas, dan mengikuti standar tertentu [4].

Dalam hal ini, "komponen" merujuk pada blok software yang dirancang secara aman dan dapat digunakan pada berbagai proyek dan melakukan tugas tertentu. Satu komponen biasanya terdiri dari beberapa kelas yang saling terkait untuk menjalankan fungsi

tertentu, dan dirancang sehingga dapat terhubung ke yang lain tanpa memahami isi dalamannya.

Lebih lanjut, Koteska dan Velinov (2013) menyatakan bahwa keberhasilan CBSD bergantung pada cara kita dapat mengukur dan mengukur nilai tingkat kemampuan dipakai ulang dari komponen. Mereka memperkenalkan kerangka terintegrasi untuk mengukur fleksibilitas komponen dalam berbagai kondisi. Ini menggarisbawahi bahwa CBSD bukan hanya tentang merakit komponen, tetapi juga memastikan standar kualitas, standar yang konsisten, dan ukuran reusability yang dapat diukur secara akurat[2].

CBSD pada dasarnya adalah model pengembangan software yang berfokus pada penggunaan ulang, kemodularan, dan penyatuan. Aplikasi dibuat dengan menggabungkan komponen yang sudah ada. Studi dan rencana McIlroy baru-baru ini menunjukkan bagaimana CBSD berkembang menjadi komponen penting dari dunia rekayasa software modern.

B. Penerapan Model

sudah banyak aspek pengembangan software modern menggunakan CBSD, termasuk penggunaan framework yang sedang populer, desain arsitektur terbaru, dan teknik khusus yang disesuaikan dengan tuntutan industri tertentu.

Lewat alat-alat seperti JavaBeans, Enterprise JavaBeans (EJB), dan Java Server Pages (JSP), para programmer bisa menyusun aplikasi berbasis komponen di dunia Java. Tiap bagian atau bean ini punya peran spesifik dan bisa dimanfaatkan untuk keperluan beragam. Contohnya, ada komponen Java yang bisa mengatur hubungan dengan database, yang dapat digunakan dalam berbagai proyek tanpa perlu menulis kode dari nol. Itu termasuk dalam kekuatan reusabilitas CBSD [3].

Selain itu, CBSD sangat disukai di ekosistem Microsoft.NET karena framework-nya yang menawarkan komponen dan assembly yang membuat pembuatan aplikasi bisnis lebih mudah. Untuk membangun aplikasi skala besar, pengembang cukup merakit berbagai komponen dengan adanya library dan paket siap pakai. Penelitian menunjukkan bahwa pendekatan ini meningkatkan produktivitas karena organisasi tidak perlu membangun setiap modul dari awal.

Selain itu, CBSD sangat penting untuk arsitektur microservices, yang saat ini menjadi paradigma utama cloud computing. Microservices menganggap setiap layanan sebagai bagian terpisah yang dapat digunakan kembali, digantikan, atau dikembangkan secara terpisah tanpa mempengaruhi keseluruhan sistem. Misalnya, layanan autentikasi

pengguna dapat diintegrasikan secara mandiri ke dalam aplikasi dan diintegrasikan ke berbagai layanan lainnya, seperti pembayaran, notifikasi, dan manajemen profil. Mikroservis memudahkan penambahan atau pengganti layanan baru, yang memenuhi prinsip substitusi CBSD.

Selain itu, domain engineering menggunakan CBSD untuk membangun repositori komponen untuk kebutuhan bidang tertentu. Konsep ini menekankan bahwa setiap bidang, seperti e-commerce, perbankan, kesehatan, atau sistem pendidikan, memiliki kebutuhan berulang yang dapat dipenuhi melalui komponen khusus yang terlibat dalam bidang tersebut. Misalnya, elemen seperti katalog produk, payment gateway, dan shopping cart dapat dibuat sekali dan digunakan kembali di berbagai platform e-commerce dalam industri e-commerce. Dalam bidang kesehatan, komponen seperti sistem pemesanan obat atau rekaman medis elektronik (EMR) dapat digunakan kembali. Menurut penelitian yang dilakukan oleh Basha dan Moiz (2012), domain engineering memiliki peran yang signifikan dalam menurunkan biaya pengembangan perangkat lunak dan meningkatkan konsistensi kualitas antar sistem [4].

CBSD digunakan dalam penelitian dan pengembangan sistem kritis serta dalam aplikasi komersial. Misalnya, penggunaan komponen yang telah diuji sebelumnya dalam sistem militer, transportasi, dan komunikasi memberikan jaminan kualitas dan reliabilitas yang lebih tinggi. Repositori komponen yang dikembangkan secara domain-spesifik dapat menjadi aset penting dalam skala besar karena memungkinkan standarisasi antar organisasi dan proyek.

Oleh karena itu, CBSD tidak terbatas pada teknologi tertentu tetapi juga meluas ke berbagai domain aplikasi. Keberhasilan praktiknya menunjukkan bahwa paradigma ini memiliki kemampuan untuk mengatasi kompleksitas perangkat lunak modern, meningkatkan produktivitas, dan meningkatkan kualitas sistem melalui prinsip reuse, modularitas, dan integrasi komponen.

C. Karakteristik Model

Pengembangan Perangkat Lunak Berbasis Komponen (CBSD) memiliki sejumlah ciri khas yang membedakannya secara jelas dari metode pengembangan perangkat lunak lainnya. Ciri yang pertama dan paling utama adalah kemampuannya untuk digunakan kembali atau reusability. Reusability mengacu pada kemampuan sebuah komponen perangkat lunak untuk dipakai di berbagai aplikasi atau proyek tanpa perlu banyak perubahan. Konsep ini menjadi keunggulan penting dari CBSD karena dapat langsung mengurangi waktu yang

dibutuhkan untuk pengembangan serta biaya produksi perangkat lunak. Seperti yang diungkapkan dalam studi, tingkat reusability sebuah komponen sangat dipengaruhi oleh kualitas dokumentasinya, desain antarmuka yang jelas, serta kemandirian fungsional dari komponen tersebut.[3]

Ciri kedua adalah substitutability, yakni kemampuan suatu komponen untuk diganti oleh komponen lain yang memiliki fungsi serupa tanpa mengganggu keseluruhan sistem. Substitutability hanya dapat terwujud jika komponen tersebut memiliki kontrak antarmuka yang jelas dan terstandarisasi. Dengan adanya karakteristik ini, pengembang dapat dengan mudah mengganti komponen yang sudah tua atau tidak efisien dengan versi yang lebih baru, tanpa perlu memodifikasi seluruh sistem. Hal ini juga mendukung keberlangsungan sistem dalam jangka panjang.

Ciri ketiga adalah composability, yaitu kemampuan untuk merakit atau menggabungkan beberapa komponen menjadi satu sistem yang lebih besar dan kompleks. Composability memungkinkan komponen dari berbagai sumber dapat diintegrasikan secara harmonis selama mengikuti standar antarmuka yang ditetapkan. Dengan kata lain, penggabungan sistem dalam CBSD mirip dengan penyusunan blok bangunan, di mana setiap blok memiliki fungsi tertentu yang dapat memberikan kontribusi pada keseluruhan sistem. Hal ini sangat relevan dalam konteks sistem yang besar dan beragam, seperti aplikasi enterprise atau sistem berbasis cloud, yang memerlukan integrasi berbagai layanan secara bersamaan. [3]

Selain itu, extensibility juga menjadi salah satu ciri penting dalam CBSD. Extensibility mengacu pada kemampuan sebuah sistem untuk diperluas dengan menambahkan komponen baru tanpa harus mengubah keseluruhan arsitekturnya. Dengan fitur ini, CBSD mendukung perkembangan sistem seiring dengan bertambahnya kebutuhan bisnis atau perubahan teknologi. Contohnya, ketika sebuah organisasi memerlukan fitur baru, pengembang cukup menambah komponen baru ke dalam sistem yang ada tanpa harus mengubah komponen yang sudah berfungsi dengan baik.

Beberapa literatur juga menyoroti ciri lain yang tidak kalah signifikan, seperti encapsulation dan independence. Encapsulation berarti rincian internal sebuah komponen disembunyikan dari komponen lain, sehingga interaksi hanya dilakukan lewat antarmuka yang telah ditentukan. Independence atau kemandirian menunjukkan bahwa setiap komponen sebaiknya dirancang untuk bisa beroperasi secara mandiri dengan fungsi yang spesifik, sehingga tidak terlampau bergantung pada komponen lainnya. Kedua sifat ini memperkuat

modularitas sistem, sehingga lebih mudah dalam pengelolaan, pengujian, dan pemeliharaan.

Dari penjelasan di atas, dapat disimpulkan bahwa karakteristik CBSD — termasuk reusability, substitutability, composability, extensibility, encapsulation, dan independence memberikan keunggulan kompetitif yang membedakannya dari metode pengembangan perangkat lunak lainnya. Karakteristik-karakteristik ini tidak hanya menjadikan CBSD lebih efisien dalam pengembangan, tetapi juga lebih adaptif terhadap perubahan kebutuhan pengguna serta perkembangan teknologi.

D. Kelebihan dan Kelemahan

Kelebihan CBSD:

- Penghematan Waktu dan Uang – pengembang tidak perlu mengembangkan semua kode dari awal.
- Kualitas yang Baik – komponen biasanya sudah teruji dan digunakan berkali-kali dalam proyek yang lain.
- Perawatan yang Mudah – sifat modular membuat penggantian atau perbaikan bagian menjadi lebih mudah.
- Kemampuan untuk Berkembang dan Penyesuaian – menambah fitur baru dapat dilakukan dengan menambahkan komponen tambahan dengan mudah.

Kelemahan CBSD:

- Kesesuaian Antar Komponen – sering kali, komponen dari sumber yang berbeda sulit untuk diintegrasikan karena perbedaan standar yang ada.
- Ketersediaan Komponen – tidak semua kebutuhan dalam proyek memiliki komponen yang siap digunakan.
- Biaya Awal yang Tinggi – diperlukan investasi untuk membangun repositori komponen dan pelatihan.
- Masalah Non-Fungsional – seperti isu keamanan, performa, dan keandalan yang terkadang sulit dipastikan pada komponen dari pihak ketiga.[1], [2]

E. Perbedaan dengan metodologi lain

- Model Waterfall: bersifat linier dan tidak mudah beradaptasi dengan perubahan kebutuhan. CBSD lebih adaptif karena memungkinkan modifikasi hanya dengan mengganti elemen.
- Model Prototiping: menekankan pada iterasi cepat untuk mengeksplorasi desain, sementara CBSD lebih fokus pada penggunaan kembali elemen yang telah ada.
- Model Spiral: bertujuan mengelola risiko melalui berbagai iterasi. CBSD lebih menekankan pada modularitas serta arsitektur yang berbasis komponen.
- Pengembangan Agile: memprioritaskan kerja sama tim dan iterasi yang pendek, sedangkan CBSD dapat diterapkan dalam Agile sebagai strategi untuk penggunaan kembali guna mempercepat pengembangan aplikasi.

Dengan demikian, CBSD dapat digabungkan dengan metodologi lain, khususnya Agile, untuk mencapai tingkat efisiensi yang optimal[3]

F. Alat Bantu Model

Ada banyak alat dan teknologi yang berfungsi sebagai pendukung utama dalam proses pembuatan software berbasis komponen (CBSD). Mari kita bahas beberapa contohnya secara ringkas:

- Penyimpanan Komponen (Repositori): Untuk menyimpan dan membagikan komponen kode siap pakai, tempat ini mirip dengan gudang besar. Misalnya, developer Java sering menggunakan Maven Central untuk mendapatkan library dengan cepat, sementara NuGet lebih cocok untuk tim yang bekerja di ekosistem.NET.
- Struktur Pendukung dan Jembatan (Frameworks & Middleware): Ada yang namanya J2EE, .NET, atau CORBA—ini semua bantu hubungkan komponen dari sistem berbeda tanpa ribet. Hasilnya, proses gabung-gabung jadi lebih mudah dan nggak sering error.
- Teknologi Kontainer: Di era sekarang, Docker sama Kubernetes lagi hits banget sebagai langkah maju CBSD. Mereka bikin deployment software di cloud jadi lebih fleksibel, apalagi kalau proyeknya skalanya besar dan berubah-ubah.
- Alat Bantu Rekayasa Perangkat Lunak (CASE Tools): Berbagai software ini membantu dalam tahap pemodelan, evaluasi, dan penyatuannya komponen, membuat proses pengembangan lebih efisien.

- Alat Pengukur Metrik: Sebuah model awal untuk mengukur tingkat kelayakan ulang-pakai (reusability) yang dibuat dalam studi ini, berguna untuk mengecek kualitas komponen sebelum dipakai lebih lanjut [2].

BAB III

PENUTUPAN

A. Kesimpulan

Pengembangan Perangkat Lunak Berbasis Komponen (Component-Based Software Development/CBSD) telah terbukti menjadi salah satu pendekatan utama yang mampu menjawab berbagai tantangan dalam pengembangan perangkat lunak modern.

Paradigma ini menekankan konsep inti seperti reuse (pemanfaatan kembali), modularitas, kemampuan substitusi, komposabilitas, serta ekstensi dalam proses pengembangannya. Dengan demikian, perangkat lunak tidak lagi dipandang sebagai sesuatu yang dibangun sepenuhnya dari nol, melainkan sebagai sistem yang tersusun dari komponen siap pakai. Pendekatan ini memberikan keuntungan berupa efisiensi biaya, percepatan waktu produksi, serta peningkatan kualitas perangkat lunak. Nilai CBSD bahkan melampaui aspek teknis dan memberikan manfaat strategis. Bagi organisasi yang menjunjung tinggi inovasi, kemampuan membangun sistem baru dengan cepat menjadi faktor penting dalam menjaga daya saing.

CBSD menawarkan solusi melalui penggunaan kembali komponen yang telah teruji, sehingga perusahaan dapat lebih fokus pada penciptaan nilai inovatif alih-alih membangun ulang fitur dasar. Selain itu, sifat modular CBSD juga mempermudah pemeliharaan dan pengembangan sistem, karena setiap komponen dapat diperbaiki, diperbarui, atau diganti tanpa harus memengaruhi keseluruhan sistem.

Namun, penerapan CBSD bukan tanpa tantangan. Keberhasilannya sangat bergantung pada ketersediaan komponen berkualitas tinggi, repositori yang lengkap, serta metodologi integrasi yang tepat. Isu-isu seperti kompatibilitas antar-komponen, standardisasi antarmuka, hingga manajemen versi juga merupakan faktor penting yang harus diperhatikan. Penelitian terkini menekankan perlunya pengembangan model pengukuran yang transparan dan dapat digunakan lintas domain, sehingga pengembang dapat menilai kualitas komponen sebelum diintegrasikan ke dalam sistem.[2]

Ke depan, relevansi CBSD diperkirakan akan semakin kuat dengan munculnya komputasi awan, Internet of Things (IoT), dan arsitektur microservices yang menuntut skalabilitas, fleksibilitas, serta kemampuan integrasi tinggi. Konsep containerization dengan alat seperti Docker atau Kubernetes turut memperluas cakupan penerapan CBSD, karena

setiap kontainer dapat diperlakukan sebagai komponen mandiri dalam sistem yang lebih besar.

Selain itu, penelitian masa depan di bidang seleksi komponen otomatis, pemanfaatan kecerdasan buatan untuk rekomendasi reuse, serta rekayasa domain yang menghasilkan repositori komponen spesifik domain akan semakin memperkuat posisi CBSD dalam rekayasa perangkat lunak global. Dengan dukungan perkembangan teknologi ini, CBSD tidak hanya sekadar alternatif, melainkan berpotensi menjadi salah satu paradigma utama yang membentuk masa depan pengembangan perangkat lunak.

Paradigma ini memungkinkan terwujudnya visi jangka panjang: pengembangan perangkat lunak yang lebih cepat, hemat biaya, berkualitas tinggi, dan adaptif terhadap perubahan kebutuhan pengguna serta dinamika industri. Oleh karena itu, CBSD dapat dipandang sebagai fondasi penting dalam transisi menuju generasi baru rekayasa perangkat lunak yang lebih cerdas, modular, dan berkelanjutan.

DAFTAR PUSTAKA

[1] U. Q. Syed, "Component Based Systems: A Novel Methodology in Software Engineering," *Indian J. Sci. Technol.*, vol. 12, no. 02, pp. 1–4, 2019, doi: 10.17485/ijst/2019/v12i2/138330.

[2] B. Koteska and G. Velinov, "Component-based development: A unified model of reusability metrics," *Adv. Intell. Syst. Comput.*, vol. 207 AISC, pp. 335–344, 2013, doi: 10.1007/978-3-642-37169-1_33.

[3] N. Md, J. Basha, G. Ganapathy, and D. M. Moulana, "A Preliminary Exploration on Component Based Software Engineering," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 22, no. 9, p. 143, 2022, [Online]. Available: https://doi.org/10.22937/IJCSNS.2022.22.9.22

[4] N. Md Jubair Basha and S. A. Moiz, "Component based software development: A state of art," *IEEE-International Conf. Adv. Eng. Sci. Manag. ICAESM-2012*, pp. 599–604, 2012.