

MATERI PEMBEKALAN JUNIOR WEB PROGRAMMER



HARI KE-4

J.620100.019.02

**Menggunakan library atau komponen
pre-existing**



JUNIOR WEB
PROGRAMMER

J.620100.019.02

**Menggunakan Library atau
Komponen Pre-Existing**

6

MENGGUNAKAN LIBRARY ATAU KOMPONEN PRE-EXISTING

Objektif:

1. Melakukan pemilihan unit-unit reuse yang potensial
 2. Melakukan integrasi library atau komponen pre-existing dengan source code yang ada
-

1. UNIT-UNIT REUSE YANG POTENSIAL

Reuse adalah penggunaan kembali kode yang bertujuan untuk menghemat waktu dan sumber daya serta mengurangi redundansi dengan memanfaatkan aset yang telah dibuat dalam beberapa bentuk dalam proses pengembangan produk perangkat lunak. Ide kunci dalam penggunaan kembali adalah bahwa bagian dari program komputer yang ditulis pada satu waktu dapat atau harus digunakan dalam konstruksi program lain yang ditulis di lain waktu.

1.1 Reuse Software

Penggunaan kembali perangkat lunak (Software Reuse) adalah proses menciptakan sistem perangkat lunak dari perangkat lunak yang ada dari pada membangun sistem perangkat lunak dari awal. Ini sederhana namun visi yang kuat diperkenalkan pada tahun 1968. Dengan kata lain *reuse software* adalah integrasi dan penggunaan aset perangkat lunak dari sebuah sistem yang dikembangkan sebelumnya.

Tujuan penggunaan kembali (reuse software) adalah untuk mengurangi biaya, waktu, usaha dan risiko, dan untuk meningkatkan produktivitas, kualitas, kinerja, dan interoperabilitas.

Beberapa jenis reuse software antara lain:

- **Oportunistik Reuse**
Meskipun bersiap-siap untuk memulai sebuah proyek, tim menyadari bahwa ada komponen yang ada bahwa dapat menggunakan kembali.
- **Planning Reuse**
Sebuah tim desain komponen strategis sehingga akan dapat digunakan kembali dalam proyek-proyek masa depan.

Menggunakan kembali oportunistik dapat dikategorikan lebih lanjut:

- **Internal reuse**
Sebuah tim reuses komponen sendiri. Ini mungkin keputusan bisnis, karena tim mungkin ingin mengontrol sebuah komponen penting untuk proyek.
- **External reuse**
Sebuah tim mungkin memilih untuk lisensi pihak ketiga komponen. Lisensi pihak ketiga komponen biaya biasanya tim 1-20 persen dari berapa biaya untuk mengembangkan internal. Tim juga harus memperhatikan waktu yang dibutuhkan untuk mencari, belajar dan mengintegrasikan komponen.

Unit perangkat lunak yang digunakan kembali, mungkin ukuran yang berbeda. Sebagai contoh:

a. **Sistem Aplikasi Reuse**

Seluruh sistem aplikasi dapat digunakan kembali dengan memasukkan, tanpa berubah menjadi sistem lain atau dengan mengkonfigurasi aplikasi untuk pelanggan yang berbeda. Atau, keluarga aplikasi yang memiliki arsitektur yang umum, tetapi yang disesuaikan untuk pelanggan tertentu, dapat dikembangkan.

b. Komponen Reuse

Komponen aplikasi, mulai dari ukuran subsistem ke objek tunggal, dapat digunakan kembali. Sebagai contoh, sistem pencocokan pola yang dikembangkan sebagai bagian dari sistem pengolahan teks dapat digunakan kembali dalam sistem manajemen database.

c. Obyek dan Fungsi Reuse

Komponen perangkat lunak yang melaksanakan fungsi tunggal, seperti fungsi matematika, atau kelas objek dapat digunakan kembali. Bentuk reuse, berdasarkan pada library standar, yang umumnya selama 40 tahun terakhir. Banyak library fungsi dan kelas yang tersedia secara bebas.

Keuntungan dari software reuse adalah bahwa biaya pengembangan secara keseluruhan dapat dikurangi. Sedikit komponen perangkat lunak harus ditentukan, dirancang, dilaksanakan, dan divalidasi. Pengurangan biaya adalah hanya salah satu keuntungan dari penggunaan kembali. Keuntungan lainnya dari software reuse adalah:

- Peningkatan Ketergantungan;
Software Reuse, yang telah dicoba dan diuji dalam sistem kerja, harus lebih diandalkan daripada perangkat lunak baru. Desain dan implementasi kesalahannya harus ditemukan dan diperbaiki.
- Mengurangi Proses Risiko;
Biaya perangkat lunak yang ada sudah diketahui, sedangkan biaya pengembangan selalu masalah pertimbangan. Ini merupakan faktor penting untuk manajemen proyek karena mengurangi margin of error dalam estimasi biaya proyek. Hal ini terutama berlaku ketika komponen perangkat lunak yang relatif besar seperti subsistem digunakan kembali.

- Keefektifitasan menggunakan Spesialis;
Daripada melakukan pekerjaan yang sama berulang-ulang, spesialis aplikasi dapat mengembangkan perangkat lunak yang dapat digunakan kembali yang merangkum pengetahuan.
- Standar Kepatuhan;
Beberapa standar, seperti standar user interface yang dapat diimplementasikan sebagai satu set komponen dapat digunakan kembali. Misalnya, jika menu dalam user interface yang diimplementasikan dengan menggunakan komponen dapat digunakan kembali, semua aplikasi menyajikan format menu yang sama kepada pengguna. Penggunaan user interface standar dapat meningkatkan ketergantungan karena pengguna akan membuat kesalahan lebih sedikit ketika disajikan dengan antarmuka yang familiar.
- Mempercepat Pengembangan;
Membawa sistem ke pasar cepat mungkin sering lebih penting daripada biaya pembangunan secara keseluruhan. Menggunakan kembali perangkat lunak dapat mempercepat produksi sistem karena kedua pengembangan dan waktu validasi dapat dikurangi.

Ada keuntungan dari software reuse, namun ada biaya dan masalah yang terkait dengan penggunaan kembali juga. Masalah yang timbul antara lain:

- Meningkatkan Biaya Maintenance;
Jika source code atau komponen dari perangkat lunak yang digunakan tidak tersedia, maka biaya maintenance akan meningkat karena unsur-unsur yang digunakan kembali dari sistem dapat menjadi semakin tidak sesuai dengan perubahan sistem.

- Kurangnya Alat Pendukung;
Beberapa perangkat lunak tidak mendukung pengembangan dengan reuse. Mungkin sulit atau tidak mungkin untuk mengintegrasikan alat-alat ini dengan sistem perpustakaan komponen. Proses perangkat lunak diasumsikan oleh alat-alat ini mungkin tidak mengambil reuse ke rekening. Hal ini terutama berlaku untuk alat-alat yang mendukung embedded system engineering, kurang begitu untuk alat pengembangan berorientasi objek.
- Sindrom "Tidak Ditemukan Disini";
Beberapa software engineer memilih untuk menulis ulang komponen karena mereka percaya bahwa mereka dapat memperbaikinya. Hal ini sebagian untuk melakukan dengan kepercayaan dan sebagian hubungannya dengan fakta bahwa menulis software asli dianggap lebih menantang daripada menggunakan kembali software orang lain.
- Membuat, Memelihara, dan Menggunakan Library Komponen;
Mengisi library komponen yang digunakan kembali dan memastikan pengembang software dapat menggunakan library ini bisa menjadi mahal. Proses pengembangan harus disesuaikan untuk memastikan bahwa library tersebut digunakan.
- Menemukan, Memahami, dan Beradaptasi dengan komponen reusable;
Komponen software harus ditemukan di library, dipahami dan, kadangkadang, diadaptasikan untuk bekerja di lingkungan yang baru. Engineering harus cukup yakin untuk menemukan komponen di library sebelumnya termasuk pencarian komponen sebagai bagian dari proses perkembangan normal.

Kegiatan yang dilakukan pada saat melakukan perancangan reuse software adalah:

- mendefinisikan antarmuka / spesifikasi
- mendefinisikan pelaksanaan
- pemrograman berorientasi obyek / warisan
- templates generics / templates
- parameter passing modul dekomposisi / parameter kelulusan
- fitur

1.2 Proses Reuse Software

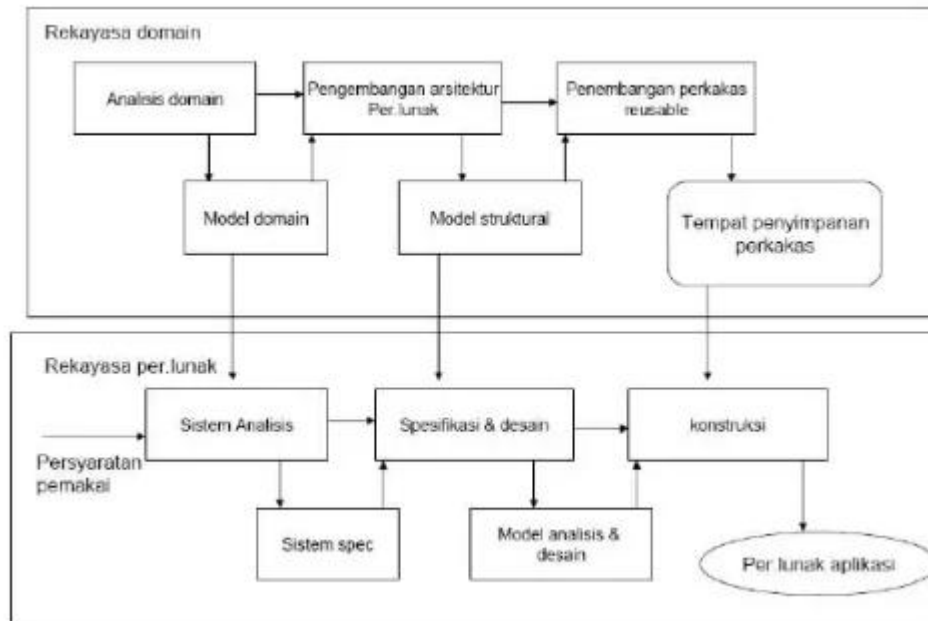
Terdapat 4 hal yang harus diperhatikan saat melakukan reuse software yaitu perkakas reuseable, model proses, fungsi-fungsi karakteristik dan pemodelan struktural dan point struktural, yang dijabarkan sebagai berikut:

1. Perkakas Reuseable;
 - a. Rencana proyek
 - b. Perkiraan biaya
 - c. Arsitektur
 - d. Model dan spesifikasi persyaratan
 - e. Desain
 - f. Kode sumber
 - g. Dokumentasi teknis dan pemakai
 - h. Interface manusia
 - i. Data
 - j. Test case



2. Model Proses;

Model proses tergambar seperti gambar berikut ini.



Gambar 1. Model Proses Reuse Software

Tujuan rekayasa domain adalah mengidentifikasi, mengkonstruksi, meng-katalog dan menyebarkan serangkaian perkakas perangkat lunak yang memiliki aplikabilitas dan perangkat lunak masa depan di dalam suatu domain aplikasi tertentu. Rekayasa domain meliputi 3 aktifitas, yaitu: analisis, konstruksi dan penyebaran.

Langkah-langkah proses analisis domain didefinisikan sebagai :

1. Tentukan domain yang akan diselidiki.
 2. Kategorikan item yang akan di-ekstrak dari domain.
 3. Kumpulkan sampel yang presentatif dari aplikasi di dalam domain.
 4. Analisis masing-masing aplikasi di dalam sampel.
 5. Kembangkan sebuah analisis bagi objek.
3. Fungsi-fungsi Karakterisasi;

Serangkaian karakteristik domain untuk suatu perkakas reusable dapat direpresentasikan sebagai $\{D_p\}$ dimana D_{pi} , merepresentasikan karakteristik domain spesifik. Nilai yang diberikan ke D_{pi}

merepresentasikan skala ordinal yang merupakan indikasi mengenai relevansi dari karakteristik untuk perkakas p .

Tipe skala:

- tidak relevan
- relevan dalam situasi yang tidak biasa
- relevan, dapat dimodifikasi meskipun berbeda
- sangat relevan bila perangkat lunak tidak memiliki karakteristik ini reuse tidak efisien
- sangat relevan bila perangkat lunak tidak memiliki karakteristik ini reuse tidak efektif, reuse tidak disetujui.

Bila perangkat lunak baru w akan dibangun didalam domain aplikasi, serangkaian karakteristik domain ditarik untuknya. D_{pi} dan D_{wi} kemudian dibandingkan untuk menentukan apakah perkakas yang ada (p) dapat secara efektif digunakan lagi didalam aplikasi w .

Tabel Karakteristik Domain

Produk	Proses	Personil
Stabilitas persyaratan	Model proses	Motivasi
Per.lunak saat ini	Konformansi proses	Pendidikan
Batasan memori	Lingkungan proyek	Pengalaman - domain aplikasi - proses - platform - bahasa
Ukuran aplikasi	Batasan jadwal	
Kompleksitas interface	Batas anggaran	
Bhs. Pemrograman	produktivitas	
Keamanan/reliabilitas		Produktivitas tim
Lifetime requirement		
Kualitas produk		
Reliabilitas produk		

4. Pemodelan Struktural dan Point Struktural;

Pemodelan struktural adalah suatu pola yang berdasarkan pendekatan rekayasa domain aplikasi yang bekerja dengan asumsi bahwa setiap domain aplikasi memiliki pola pengulangan (fungsi, data, tingkah laku) yang memiliki potensi reuse.

Point struktur memiliki 3 karakteristik :

- a. Point struktur adalah abstraksi yg harus memiliki sejumlah contoh terbatas.
- b. Interface ke point struktur harus sederhana.
- c. Point struktur harus mengimplementasi penyembunyian informasi yang diisikan di dalam point struktur itu sendiri.

1.3 Membangun Komponen Reuse Software

Hal yang perlu diperhatikan saat membangun komponen reuse software antara lain:

1. Analisis dan desain untuk reuse;

Idealnya, model analisis di analisis untuk menentukan elemen-elemen model yang menunjuk ke perkakas reusable yang ada. Kunci untuk desain reuse : (Design For Reuse/DFR)

- Data standar : Domain aplikasi dan struktur data global standar (struktur file atau database) lengkap harus diidentifikasi.
- Protokol interface standar : Tiga tingkat protokol harus dibangun; sifat interface intramodular, desain interface teknis eksternal dan interface manusia mesin.
- Template program : Model struktur dapat berfungsi sebagai template untuk desain arsitektur dari program baru.

2. Model Konstruksi;

Konstruksi dapat dilakukan dengan bahasa pemrograman generasi ketigayang konvensional, bahasa generasi keempat dan generator kode, teknik pemrograman visual dan peranti yang lebih canggih. Serangkaian komponen generik dan dapat diadaptasikan yang disebut kerangka (frame) atau teknologi kerangka.

3. Pengembangan Berbasis Komponen;

Pendekatan konstruksi kadang diacukan ke pengembangan berbasis komponen atau perangkat lunak komponen. Implementasi pengembangan berbasis komponen :

- *Model pertukaran data* : Mekanisme yang memungkinkan pemakai dan aplikasi berinteraksi dan mentransfer data untuk semua komponen reusable.
- *Otomasi* : Berbagai peranti macro dan script harus diimplementasi untuk memfasilitasi interaksi antar komponen reusable.
- *Penyimpanan terstruktur* : Data heterogen (grafis, suara, teks dan numeris) dikumpulkan dan diakses sebagai struktur data tunggal.
- *Model obyek yang mendasari* : Interface Definition Language (IDL)
- *OpenDoc* : Standar dokumen gabungan dan perangkat lunak komponen.
- *OMG/CORBA* : Object Request Broker / ORB memberi pelayanan komponen reusable berkomunikasi dengan komponen lain tanpa melihat lokasi didalam sistem.
- *OLE 2.0 (Object Linking and Embedding)* : Merupakan bagian dari Component Object Model (COM).

1.4 Keuntungan Reuse Software

Keuntungan dari reuse software seperti yang telah dijelaskan sebelumnya disebabkan karena faktor berikut:

- Pengembangan sistematis komponen dapat digunakan kembali.
- Sistematis penggunaan kembali komponen-komponen ini sebagai blok bangunan untuk menciptakan sistem baru.

Sebuah komponen dapat digunakan kembali kode tetapi manfaat yang lebih besar menggunakan kembali datang dari yang lebih luas dan tampilan tingkat tinggi dari apa yang dapat digunakan kembali. Keuntungan utama untuk menggunakan kembali perangkat lunak adalah untuk:

- Perangkat lunak meningkatkan produktivitas
- Mempersingkat waktu pengembangan perangkat lunak
- Meningkatkan interoperabilitas sistem software
- Mengembangkan software dengan lebih sedikit orang
- Pindahkan personel lebih mudah dari proyek ke proyek
- Mengurangi pengembangan perangkat lunak dan biaya pemeliharaan
- Memproduksi lebih banyak perangkat lunak standar
- Menghasilkan perangkat lunak kualitas yang lebih baik dan memberikan keunggulan kompetitif yang kuat

1.5 Pengklasifikasian dan Pengambilan Kembali Komponen

Pengklasifikasian dan pengambilan kembali komponen dapat dilihat dari:

1. Penggambaran Komponen Reuse;

Konsep harus menggambarkan isi komponen yaitu isi dan konteks. Skema klasifikasi komponen perangkat lunak dibagi menjadi tiga kategori:

- *Enumerated Classification*: Komponen digambarkan dengan menentukan struktur hirarki dimana kelas dan sub kelas yang bervariasi dari komponen diterapkan.
- *Faceted Classification*: Sebuah area domain dianalisis dan serangkaian ciri deskriptif dasar diidentifikasi. Ciri-ciri tersebut disebut faceted.
- *Klasifikasi atribut-nilai*: Sama dengan klasifikasi faceted.

2. Lingkungan Reuse;

- Database komponen
- Sistem manajemen pustaka
- Sistem pemanggilan kembali perangkat lunak (OBR)
- Peranti Case

1.6 Dampak Terhadap Ekonomi

Software reuse juga membawa pengaruh dari sisi ekonomi, antara lain pada:

1. Pengaruh Terhadap Kualitas Produktifitas dan Biaya;

- *Kualitas* : Dengan reuse cacat ditemukan dan dieliminasi sebagai hasilnya kualitas komponen meningkat
- *Produktifitas* : Lebih sedikit waktu yang digunakan untuk membuat rencana, model, dokumen, kode dan data yang diperlukan untuk membuat sistem.
- *Biaya* : Penghematan waktu neto untuk reuse diperkirakan dengan memproyeksikan bila proyek dikembangkan sejak awal.

2. Analisis Biaya Dengan Menggunakan Point Struktur

Idealnya adaptasi, integrasi dan biaya pemeliharaan yang berhubungan pada masing-masing komponen pada suatu pustaka reuse disimpan dari masing-masing contoh penggunaan, lalu dianalisa untuk mengembangkan proyeksi biaya bagi contoh reuse selanjutnya.

$$\text{Usaha keseluruhan} = E_{\text{new}} + E_{\text{adapt}} + E_{\text{int}}$$

Di mana

- **Enew** adalah usaha untuk komponen perangkat lunak baru.
- **Eadapt** adalah usaha untuk menyelesaikan reuse dari tiga point struktur yaitu SP1, SP2, SP3
- **Eint** adalah usaha untuk mengintegrasikan SP1, SP2, SP3

- SP1, SP2 dan SP3 ditentukan dengan mengambil rata-rata data historis.

3. Metrik Reuse

Untuk mengukur manfaat reuse pada sistem berbasis komputer. Reuse pada sistem D diekspresikan sebagai:

$$Rb(S) = [Cnoureuse - Creuse] / Cnoureuse$$

Di mana

- **Cnoureuse** adalah biaya pengembangan tanpa reuse
- **Creuse** adalah biaya pengembangan dengan reuse

Rb(S) diekspresikan sebagai nilai nondimensional dalam range:

$$0 \leq Rb(S) \leq 1$$

Semakin tinggi nilai Rb(S), reuse akan semakin menarik.

Pengukuran reuse dalam sistem OO disebut *reuse leverage* ditentukan sebagai:

$$Rlev = OBJreused / OBJbuilt$$

Di mana:

- **OBJreused** adalah jumlah objek yang digunakan kembali
- **OBJbuilt** adalah jumlah objek yang dibangun
- Bila Rlev bertambah Rb juga bertambah

1.7 Pentingnya Integrasi Reuse Software ke dalam Budaya Perusahaan

Software Reuse adalah strategi penting untuk semua kelompok pengembangan perangkat lunak. Dengan menggunakan kembali kode sementara pindah ke platform generasi berikutnya perusahaan dapat memanfaatkan investasi perangkat lunak yang ada mereka dan mengurangi waktu ke pasar. Namun banyak perusahaan sedang berjuang untuk sepenuhnya menggunakan kembali kode menerapkan seluruh

organisasi mereka. Untuk mencapai efisien dan metodis penggunaan kembali kode, organisasi harus mengintegrasikan tujuan ini ke dalam budaya mereka. Kode Reusing memberikan manfaat terbesar untuk sebuah organisasi jika hal itu dilakukan secara sistematis daripada secara sporadis dan opportunistically. Namun ada banyak hal yang dapat mencegah penggunaan kembali kode sistematis baik teknis dan non-teknis.

1. Software reuse – Masalah Teknis

Di sisi teknis ada banyak perbedaan antara sistem operasi seperti tingkat prioritas tugas yang ditawarkan oleh masing-masing OS yang membuat memodifikasi kode untuk platform yang berbeda membosankan dan rumit. Ini telah membawa perlunya port Cots alat yang secara otomatis akan menjelaskan perbedaan-perbedaan dalam sistem operasi untuk membuat port bekerja lebih cepat dan lebih mudah. Untuk menghindari masalah port sama sekali, organisasi melihat kebutuhan untuk solusi abstraksi untuk melindungi kode mereka terhadap perubahan platform masa depan. Namun mengembangkan sebuah antarmuka abstraksi menggunakan OS asli API tidak akan memberikan portabilitas dan kinerja yang diperlukan dalam aplikasi embedded. Sebaliknya tingkat rendah pendekatan perlu diambil untuk menjamin bahwa sumber daya sistem operasi dasar seperti benang, Semaphore dan mutex akan berperilaku sama di seluruh platform dan kinerja yang tidak terkena dampak. Jika membangun dan memelihara rumah yang di-abstraksi untuk beberapa sistem operasi memerlukan banyak waktu, uang dan sumber daya.

Para pengembang harus memiliki pengetahuan yang terperinci dari masing-masing sistem operasi dan melakukan banyak pengujian untuk memverifikasi portabilitas platform yang berbeda yang mengakibatkan biaya tinggi. Inilah sebabnya mengapa banyak perusahaan yang

berubah ke arah abstraksi Cots lapisan yang tetap dipertahankan, diuji dan diverifikasi oleh pihak ketiga daripada mengambil fokus jauh dari kompetensi inti organisasi. Common menggunakan API (yang disediakan oleh Cots abstraksi) diseluruh platform juga mengurangi potensi kurva pembelajaran saat mengembangkan sistem operasi baru sehingga membuat lebih mudah untuk menggunakan kembali kode mengadopsi.

Sama seperti menggunakan kembali kode pada sistem operasi yang berbeda memiliki tantangan sendiri, menggunakan kembali kode ketika pindah ke bahasa yang berbeda juga menyajikan kesulitan. Sebagai contoh banyak perusahaan yang sekarang bergerak menjauh dari Ada yang lebih modern bahasa C, karena kurangnya dukungan untuk programmer dan Ada. Organisasi-organisasi ini memanfaatkan konversi bahasa Cots alat untuk konversi otomatis untuk menghindari penulisan ulang.

2. Software Reuse – Masalah Industri

Di sisi non-teknis sementara tingkat atas eksekutif dan lembaga pemerintah mungkin akan melihat manfaat penggunaan kembali kode, ada tujuan kurangnya kesesuaian dengan kelompok rekayasa dan subkontraktor. Berkali-kali kelompok-kelompok ini mempunyai hambatan psikologis untuk menggunakan kembali kode. Mereka mungkin keliru berpikir bahwa menggunakan kembali kode akan menyebabkan bakat-bakat mereka tidak lagi diperlukan. Namun dengan menggunakan kembali kode warisan mereka dengan cepat dan efisien dengan menggunakan kembali kode Cots solusi mereka dapat menyumbangkan bakat untuk proyek-proyek baru dan pengembangan produk, bukannya macet oleh port melelahkan bekerja. Organisasi mungkin juga perlu mengubah kebijakan dan standar produktivitas untuk secara efektif menggunakan kembali kode

mengintegrasikan ke dalam budaya mereka. Alih-alih berfokus pada berapa banyak baris kode baru pengembang berkontribusi mungkin perlu untuk memberikan penghargaan lebih pendek kali untuk penyelesaian proyek. Ini akan memotivasi para pengembang untuk menggunakan alat-alat port Cots sehingga mereka dapat menggunakan kembali sebanyak mungkin lebih cepat untuk memenuhi tenggat waktu sebelumnya. Hal ini akan menyebabkan penyelesaian proyek lebih banyak, lebih banyak produk baru dan akhirnya lebih banyak kesempatan untuk mendapatkan pangsa pasar yang lebih besar dalam organisasi industri.

2. INTEGRASI LIBRARY ATAU KOMPONEN PRE-EXISTING

PHP sudah menjadi bahasa umum yang digunakan untuk web scraping. Biasanya seorang programmer mengambil informasi tertentu dari suatu halaman website yang tidak menyediakan API untuk diakses oleh sistemnya. Dengan menggunakan teknik scraping, seorang programmer dapat mengambil informasi tertentu seperti informasi cuaca dari website BMKG dan menyimpannya di database.

Pada PHP dapat melakukan scraping dengan beberapa library ini di antaranya:

a. Standard Library

PHP memiliki sejumlah library untuk memparsing HTML. Seperti DOM atau libxml. Untuk melakukan HTTP request dapat juga digunakan Url agar dapat mensimulasi proses tersebut.

Contoh:

```
// set post fields
$post = [
    'username' => 'user1',
    'password' => 'passuser1',
    'gender' => 1,
];
$ch = curl_init('http://www.domain.com');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
// execute!
$response = curl_exec($ch);
// close the connection, release resources used
curl_close($ch);
// do anything you want with your response
var_dump($response);
```

b. Guzzle

Guzzle adalah library PHP untuk melakukan HTTP request dengan sangat mudah dan dapat diintegrasikan dengan berbagai macam web service. Guzzle memiliki kelebihan untuk melakukan HTTP request secara asinkron ataupun sinkron, selain itu dia juga mendukung standar PSR-7 dalam penulisan library-nya.

Contoh:

```
$client = new GuzzleHttp\Client();
$res = $client->request('GET', 'https://api.github.com/user', [
    'auth' => ['user', 'pass']
]);
echo $res->getStatusCode();
// "200"
```

```

echo $res->getHeader('content-type');
// 'application/json; charset=utf8'
echo $res->getBody();
// {"type":"User"...}
// Send an asynchronous request.
$request      =      new      \GuzzleHttp\Psr7\Request('GET',
'http://httpbin.org');
$promise      =      $client->sendAsync($request)->then(function
($response) {
    echo 'I completed! ' . $response->getBody();
});
$promise->wait();

```

c. Buzz

Buzz adalah library PHP untuk menangani HTTP request yang paling ringan. Selain itu Buzz merupakan library yang dibuat sesederhana mungkin namun mampu menyerupai karakter sebuah web browser.

Contoh:

```

<?php
$browser = new Buzz\Browser();
$response = $browser->get('http://www.google.com');
echo $browser->getLastRequest()."\n";
// $response is an object.
// You can use $response->getContent() or $response->getHeaders()
to get only one part of the response.
echo $response;

```

d. HttpFul

HTTPFul adalah library PHP yang sederhana, mudah dibaca, dan dapat digunakan dengan mudah untuk berbicara kepada HTTP. Mendukung berbagai HTTP Method, custom header, automatic parsing, automatic

payload serialization, basic auth, client side certificate auth, dan memiliki kemampuan untuk membaca request template.

Contoh:

```
// Make a request to the GitHub API with a custom
// header of "X-Trivial-Header: Just as a demo".
$url = "https://api.github.com/users/nategood";
$response = \Httpful\Request::get($url)
    ->expectsJson()
    ->withXTrivialHeader('Just as a demo')
    ->send();
echo "{$response->body->name} joined GitHub on " .
    date('M jS', strtotime($response->body->created_at))
```

e. Requests

Requests adalah library PHP untuk HTTP request yang diperuntukkan manusia. Selain mudah digunakan, bentuk API-nya menyerupai web framework Laravel. Selain itu banyak dukungan yang diberikan seperti browser-style ssl verification, automatic decompression, basic auth, mendukung semua HTTP method, dan lainnya.

Contoh:

```
$headers = array('Accept' => 'application/json');
$options = array('auth' => array('user', 'pass'));
$request = Requests::get('https://api.github.com/gists',
    $headers, $options);
var_dump($request->status_code);
// int(200)
var_dump($request->headers['content-type']);
// string(31) "application/json; charset=utf-8"
var_dump($request->body);
// string(26891) "[...]"
```

f. Goutte

Goutte adalah library PHP untuk proses screen scraping dan web crawling. Goutte menyediakan API yang keren untuk mengekstrak data dari respon HTML atau XML. Untuk menggunakannya kamu memerlukan PHP 5.4 dan Guzzle. Sedangkan untuk instalasinya dapat menggunakan Composer.

Contoh:

```
$goutteClient = new Client();
$guzzleClient = new GuzzleClient(array(
    'timeout' => 60,
));
$goutteClient->setClient($guzzleClient);
// Click on the "Security Advisories" link
$link = $crawler->selectLink('Security Advisories')->link();
$crawler = $client->click($link);
// Get the latest post in this category and display the titles
$crawler->filter('h2 > a')->each(function ($node) {
    print $node->text()."\n";
});
$crawler = $client->request('GET', 'https://github.com/');
$crawler = $client->click($crawler->selectLink('Sign in')->link());
$form = $crawler->selectButton('Sign in')->form();
$crawler = $client->submit($form, array('login' => 'fabpot',
    'password' => 'xxxxxx'));
$crawler->filter('.flash-error')->each(function ($node) {
    print $node->text()."\n";
});
```