Jurnal Ilmiah MEDIA SISFO
Vol. 18, No. 2, Oktober 2024
ISSN: 1978-8126
e-ISSN: 2527-7340

ICONIX Process Pada Arsitektur Zend Framework 3 (Software Effort Estimation) Menggunakan Case Based Reasoning

Maria Rosario Borroek[1], Edrian Hadinata[2]

Sistem Informasi, Fakultas Ilmu Komputer, Universitas Dinamika Bangsa[1], Sistem Informas, Fakultas Teknik dan Komputer, Universitas Harapan Medan[2]
Jl. Jen. Sudirman, Jambi, Indonesia[1], Jl. Imam Bonjol, Medan, Indonesia[2]
Diamar_ros@yahoo.com[1], edrianhadinata@gmail.com[2]

Submitted : 03/10/2024; Reviewed : 10/10/2024; Accepted : 31/10/2024; Published : 31/10/2024

Abstract

Currently, many researchers are focusing on researching various methodologies in software development because by choosing a good methodology, the resulting software will have good quality, be faster and the cost of software development will be more efficient. ICONIX Process is a hybrid method in software engineering development methodology that is formed based on object modeling methodology (Object Modeling Technique), Objectory Method and Booch Method. The use of ICONIX Process which adopts Use Case Driven and with adjustments to the Zend Framework Architecture based on Event-driven Architecture within the scope of the SOLID paradigm is expected to collaborate both on a research scale with software development cases of effort estimation in previous studies. The design process of Software Effort Estimation and Case Based Reasoning using the ICONIX Process method on the Zend Framework 3 Architecture with the aim of minimizing errors in translating diagrams produced in the design process. The calculation of Cosmic Function Point is carried out on the Zend Framework 3 application to be more structured because the calculation process is designed using classes

Keywords : Case Base Reasioning, Iconix Process, Software effort estimation

Abstrak

Saat ini berbagai peneliti banyak yang sedang fokus meneliti berbagai macam metodologi dalam pengembangan *software* karena dengan pemilihan metodologi yang baik maka software yang dihasilkan akan memiliki kualitas yang baik, lebih cepat dan biaya dalam pengembangan software akan menjadi lebih efisien. *ICONIX Process* adalah metode hybrid dalam metodologi pengembangan software engineering yang dibentuk berdasarkan metodologi pemodelan objek (Object Modeling Technique), Metode Objectory dan Metode Booch. Penggunaan ICONIX Process yang mengadopsi Use Case Driven serta dengan penyesuaian terhadap Arsitektur Zend Framework yang berbasis *Event-driven Architecture* dalam lingkup paradigma SOLID diharapkan dapat mengkolaborasikan keduanya pada skala penelitian dengan kasus pengembangan *software effort estimation* pada penelitian sebelumnya. Proses perancangan *Software Effort Estimation* dan *Case Based Reasoning* dengan menggunakan metode ICONIX *Process* pada Arsitektur Zend Framework 3 dengan tujuan agar dapat meminimumkan kesalahan dalam menterjemahkan diagram yang dihasilkan pada proses perancangan. Perhitungan Cosmic Function Point dilakukan pada aplikasi Zend Framework 3 menjadi lebih terstruktur dengan baik dikarenakan proses perhitungan didesain menggunakan *class*.

Kata kunci : case base reasioning, iconix process, software effort estimation

1. Pendahuluan

Pengembangan teknologi *software engineering* saat ini bukan hanya terfokus kepada perkembangan teknologi bahasa pemograman tetapi juga berbagai macam metodologi pengembangan software. Keputusan pemilihan metodologi yang baik dalam pembuatan software juga dapat mempengaruhi *Speed of delivery* yang diperlukan dalam industri pengembangan software selain biaya dan tentunya kualitas produk yang dihasilkan[1].

Metodologi pengembangan *software engineering* dewasa ini telah banyak dikembangkan. Percampuran beragam metodologi telah banyak diteliti oleh akademisi maupun praktisi. [2] mencampurkan antara metode Agile dan metode Scrum menghasilkan metodologi baru yang fleksibel dalam manajemen proyek TI. Ricardo menggabungkan morphological operator dan a linear operator untuk menghasilkan prediksri pengembangan perangkat lunak[3]. Begitu juga dengan ICONIX Process yang telah dibentuk oleh Doug Rosenberg berdasarkan tiga metodologi, yang berbasiskan pemograman berorientasi objek.

ISSN: 1978-8126
e-ISSN: 2527-7340

ICONIX Process mampu membuat gap antara proses desain dan implementasi menjadi semakin kecil. Kesalahan yang sering terjadi dalam proyek pengembangan *software* adalah praktiknya tidak cukup waktu untuk melakukan modeling, proses analisis dan perancangan [4]. Akhirnya terlalu banyak tekanan dari pihak manajemen untuk memproses pengerjaan *software* dalam kode program, terlalu dininya memulai pengerjaan program karena progres pengerjaan sebuah *software* dihitung dari berapa kode program yang telah diselesaikan.

ICONIX Process digerakkan berdasarkan use case seperti RUP tetapi tanpa menambahkan banyak tambahan ketika proses implementasi kode program berlangsung [5]**.** *ICONIX Process* adalah metode hybrid dalam metodologi pengembangan software engineering yang dibentuk berdasarkan metodologi pemodelan objek (Object Modeling Technique), Metode Objectory dan Metode Booch. Metode ini diorisinalkan beberapa tahun sebelum Unified Modeling Language (UML) dan Unified Process (UP) sebagai hasil proses penggabungan dan penyulingan ketiga metode tersebu t [6] [7]. Metode ini bertujuan untuk memutus gap antara proses pengkodean dan hasil detail rancangan, [6]. ICONIX Process sendiri menggunakan sebuah sistem kendali dalam bentuk diagram use cases untuk pengembangan.Dimana use casesakan dimodelkan untuk mengidentifikasi kondisi *sunny day* dan *rainy day* atau kondisi yang umum dan kondisi dalam keadaan tertentu [6]. Tentunya penggunaan metode ini juga dapat diselaraskan dengan penggunaan berbagai macam framework khususnya Zend Framework 3(ZF3) yang hidup diskala *web engineering*.

Bersandarkan pada konsep Event-driven Architecture dengan menganut paradigma prinsip pemograman berorientasi objek SOLID[8], ZF3 mengimplementasikan *listener* dengan nama *event* yang di*manage* oleh Event Manager. Pengimplementasian listener pada ZF3 dengan nama events ini dapat diletakkan dan juga dapat dicabut kembali yang berefek terhadap meningkatnya intensitas keamanan pada framework tersebut, sementara SOLID dalam basis objek oriented programming digunakan untuk membangun *software* yang efisien, *reusable* dan tidak terpisah-pisah yang memiliki ketahanan dan dapat dikelola dalam waktu yang lama[8]. Saat ini ZFyang pada rilis ke-3 telah banyak menghasilkan berbagai macam software enterprice. Dengan opensource framework, ZF3 digunakan untuk pengembangan web secara professional yang bekerja pada PHP7 dengan performa yang terbaik yang dianjurkan dan 4 kali lebih cepat dalam PHP5.

Untuk itu penggunaan ICONIX Process yang mengadopsi Use Case Driven serta dengan penyesuaian terhadap Arsitektur Zend Framework yang berbasis *Event-driven Architecture* dalam lingkup paradigma SOLID diharapkan dapat mengkolaborasikan keduanya pada skala penelitian dengan kasus pengembangan *software effort estimation* pada penelitian sebelumnya.

Software effort estimation adalah mengestimasi sejumlah sumber daya yang diperlukan dalam mengembangkan perangkat lunak. Untuk itu software effort estimation merupakan hal yang penting sehingga perlu melihat pengaruh pengukuran perangkat lunak terhadap software effort estimation yang dilakukan dengan teknik pembelajaran mesin [9]. Software Effort Estimation memainkan peranan penting dalam software engineering karena dapat menentukan kelayakan dari proposal penawaran proyek, analisa progress, perencanaan biaya, analisis monitoring proyek dan sebagainya [10].

Kesalahan memperhitungkan sumberdaya, waktu, biaya dan besaran yang selalu berubah pada proses negosiasi akan berdampak buruk pada proses produksi terlebih lagi terhadap penentuan besaran berdasarkan kebutuhan teknologi canggih yang tidak diketahui. Untuk itu akan sulit bagi manajer proyek dalam mengestimasi besaran software dan biaya yang dibutuhkan. Ini adalah ciri-ciri kelemahan *theory domains* dimana metode Case Base Reasoning (CBR) berpotensial sebagai solusinya [11]. CBR dapat menyelesaikan dengan cara menemukan kembali permasalahan yang dinilai mirip dengan kasus sebelumnya kemudian menggunakan kembali solusi yang dimiliki kasus tersebut, dimana CBR merupakan bagian dari Sistem Basis Pengetahuan [12] [13] [14] [15]. Dalam konteks *software effort estimation* sebuah sistem CBR didasari dengan asumsi bahwa setiap proyek software yang sama memiliki jumlah upaya(*effort*) yang sama. Oleh karena itu CBR menggunakan *k*-Nearest Neighbor (*k*-NN) dan menggunakan Euclidean Distance dalam mengecek tingkat kesamaan [16], dimana setiap pengalaman yang ada diingat sebagai kasus didalam basis kasus [17][18]. Algoritma *k*-Nearest Neighbor (*k*-NN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut,[19].

ISSN: 1978-8126
e-ISSN: 2527-7340

2. Metodologi

2.1 Kerangka Dasar Penelitian

Pada penelitian ini dilakukan beberapa tahapan kerja, adalah sebagai berikut:

1. Penelitian Awal
Pada tahapan ini dikumpulkan bahan penelitian dari berbagai sumber pustaka, seperti buku, jurnal (baik cetak maupun online), prosiding, majalah, artikel dan sumber lain yang relevan.

2. Pengumpulan Data
Data yang dikumpulkan pada proses ini adalah fungsional requirement yang merupakan behavioral sistem ketika end user melakukan proses menjalankan software begitu juga hasil akhir penelitian sebelumnya yang menggunakan tahapan dalam metode case based reasoning.

3. Training dan Modeling
Proses training dan modeling adalah proses utama yang terdapat pada inti penelitian ini, proses ini berhubungan dengan *prototype system* dan hybrid arsitektur framework dengan ICONIX Process yang terjadi.

4. Perbaikan dan Evaluasi
Proses ini mengecek apakah *prototype* sistem dari hybrid arsitektur system yang dihybrid telah sesuai dengan yang diinginkan atau tidak, serta mengecek apakah manfaat dan pendekatan antara prototype system sudah dekat dengan code program atau tidak.

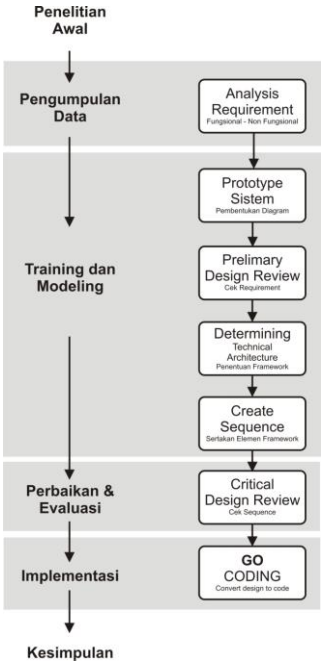
5. Implementasi
Proses implementasi adalah proses convert desain sistem yang ingin dibangun kedalam bahasa pemograman.

6. Kesimpulan
Dari output yang telah dihasilkan maka dapat diambil kesimpulan dengan cara menuliskannya kedalam bentuk laporan dimulai dari kajian teoritis, prosedur metodologi, pengujian serta saran yang dapat diberikan untuk proses penelitian selanjutnya.

2.2 Alur Kerja Penelitian

Berdasarkan penelitian yang telah diuraikan diatas, maka pembuatan diagram kerja digambarkan untuk memudahkan dalam memahami langkah-langkah yang harus dilakukan dalam pembuatan sistem.

.



Gambar 1 Alur Kerja Penelitian

Untuk itu proses perancangan penelitian dan pengerjaan yang akan dilakukan secara lebih detail diuraikan sebagai berikut:

ISSN: 1978-8126
e-ISSN: 2527-7340

1. ***Analisis Requirement***
   Proses ini adalah melakukan analisis kebutuhan dari apa yang dapat dilakukan oleh sistem apakah proses tersebut sudah sesuai dengan tujuan akhir sistem atau tidak.
2. ***Prototype System***
   Proses pembentukan diagram dalam rangka mengurangi gap antara desain dan implementasi metode diuraikan pada bagian ini. Beberapa *prototype* akan dibentuk untuk membuat jarak antara desain dan implementasi semakin dekat.
3. ***Prelimary Design Review***
   Proses ini mereview apakah seluruh *requirement* sudah sesuai dengan semua diagram *prototype system* yang telah dibuat. Setelah proses ini selesai diharapkan tidak terjadi lagi perubahan requirement dilakukan.
4. ***Determining Technical Architecture***
   Pada proses ini dilakukan proses penentuan jenis framework yang akan digunakan. Pada proses ini juga akan mengkaji bagaimana rangka kerja arsitektur framework yang dipilih. Jika terdapat perbedaan dengan desain yang sebelumnya dibuat maka proses penyesuaian dilakukan.
5. ***Create Sequence***
   Pada proses sebelumnya sebuah domain masih berisi data dan belum terdapat keterangan bagaimana proses tersebut dilakukan. Untuk itu pembuatan *sequence* diagram dilakukan untuk mendefenisikan operasi yang dilakukan pada proses tersebut. proses ini juga menyertakan entitas pada arsitektur ZF3.
6. ***Critical Design Review***
   Proses ini adalah proses review apakah proses pembentukan diagram sequence sudah sesuai, dan pastikan apakah semua class dan fungsi sudah memiliki proses atau belum.
7. ***Go Coding***
   Proses ini adalah proses implementasi dari rancangan yang telah dibentuk sebelumnya kedalam bahasa pemograman. Pada proses koding, ada kalanya terdapat hal-hal yang belum ditentukan pada perancangan tetapi baru ditemukan pada proses ini. Langkah yang harus dilakukan adalah mengubah secepatnya desain yang dibuat lalu proses koding dilanjutkan kembali.

3. Hasil dan Pembahasan

Proses perancangan *software* dengan menggunakan metode ICONIX Proses telah banyak dilirik oleh banyak kalangan baik itu di dunia akademisi maupun para praktisioner. ICONIX Proses sendiri menawarkan beragam langkah-langkah yang panjang sebagai upaya mendekatkan hasil rancangan dengan proses pemograman.

Untuk itu pada laporan penelitian ini, perancangan yang dibuat akan menggunakan beberapa jenis diagram UML dan melakukan proses perbaikan kembali jika terdapat *inconsistensi* antara perencanaan yang saat ini dilakukan dengan perencanaan yang telah dilakukan pada bagian yang lain.

3.1 Proses Rancangan

Pada penggunaan metode ICONIX proses awal yang harus dilakukan adalah membuat *functional requirement* dari aplikasi yang akan dibuat. Proses tersebut adalah sebagai berikut:

3.1.1 Functional Requirement

Bentukan *functional requirement* tidak dapat digunakan secara langsung dan pada dasarnya *functional requirement* ini juga bersifat tidak terstruktur. Pemisahan *functional requirement* dan *non functional requirement* juga sebelumnya sebaiknya telah dilakukan.

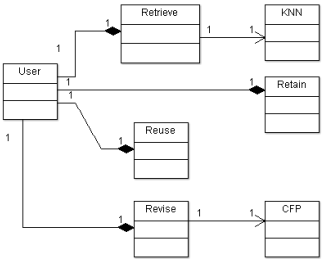
1. Sistem harus dapat memverifikasi user melalui proses login
2. Sistem dapat melakukan proses pencocokan data terhadap dataset COSMIC yang disimpan didalam database (RETRIEVE).
3. Sistem dapat mengurutkan berdasarkan persentase yang tertinggi ke persentase terendah atau sorting persentase kecocokan secara descending-(RETRIEVE).
4. Sistem dapat menampilkan data kecocokan tersebut secara lengkap-(REUSE).
5. Sistem hanya menampilkan data kecocokan berdasarkan kriteria tertentu-(REUSE).
6. Sistem dapat menghitung estimasi projek software yang berdasarkan perhitungan COSMIC Function Point (CFP) - (REVISE).

ISSN: 1978-8126
e-ISSN: 2527-7340

7. Sistem dapat menyimpan perhitungan tersebut ke dataset COSMIC yang tersimpan didatabase-(RETAIN).

3.1.2 Pembuatan Domain Model

Prosedur penggunaan ICONIX *process* juga menggunakan perancangan *class diagram* tetapi kompleksitas *class* tersebut diubah secara berulang pada tahap proses perbaikan kembali. Untuk itu tahap awal yang dilakukan adalah hanya membuat domain model.



Gambar 2 Domain Model Sementara

Domain model difungsikan juga sebagai penyamaan istilah yang dipakai untuk proses selanjutnya. Pada domain model yang dibuat terdapat tujuh domain model yaitu User, Retrieve, Reuse, Revise, Retain, KNN dan CFP. Ini adalah domain model utama yang terdapat pada sistem yang dirancang.Penggunaan ketujuh domain model tersebut adalah :

a. User
Class ini berisi identitas user yang login seperti username, waktu login dan memiliki beberapa kemampuan untuk check session, logout serta aktifitas yang lain.

b. Retrieve
Fungsi dari Retrieve ini digunakan sebagai prototype proses pencocokan antara data yang diterima dari form dengan dataset COSMIC yang terdapat didalam database.

c. Reuse
Class prototype reuse digunakan untuk menampilkan hasil yang diproses sebelumnya oleh prototype class retrieve dan secara detail hasil tersebut juga ditampilkan pada prototype reuse ini.

d. Revise
Untuk class yang memproses perhitungan CFP dan menampilkan hasilnya diproses pada class revise.

e. Retain
Proses penyimpanan dan hasil penyimpanan dilakukan dengan menggunakan class retain.

f. KNN
Class ini adalah class logic dari pada sistem ini. Proses kecocokan data dilakukan pada sistem ini dengan menggunakan K-Nearest Neighbor.

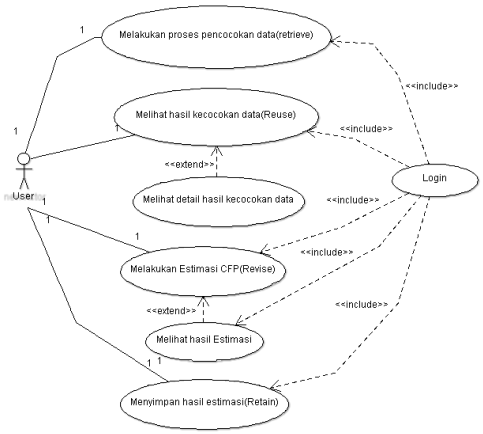
g. CFP
Class logic selanjutnya adalah CFP yang digunakan untuk memproses perhitungan sesuai dengan ketentuan yang terdapat pada COSMIC Functional Size.

Penamaan domain model yang digunakan disesuaikan dengan kapasitas fungsinya sebagai implementasi dari proses penggunaanmetode Case Based Reasoning sehingga lebih mudah dipahami.

3.1.3 Pembuatan Use Case Diagram

Pembuatan *use case diagram* dilakukan agar dapat mendefenisikan *behavioral requirement* berdasarkan *functional requirement*.

ISSN: 1978-8126
e-ISSN: 2527-7340

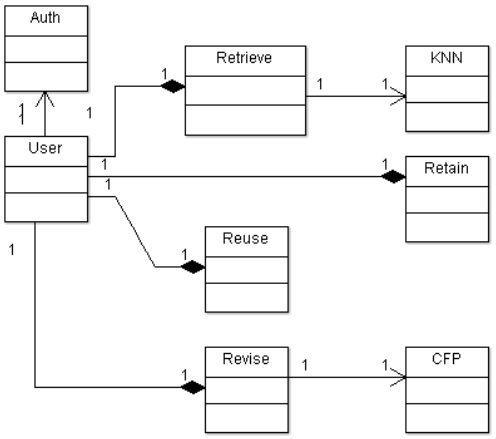


Gambar 3 Use Case Diagram Awal

Dalam pembuatan *use case* nama domain disertakan pada perancangan agar dapat dihubungkan dengan *class* yang dirancang dengan *use case* selain itu dalam ICONIX process penggunaan sebuah *use case* dirancang dengan dua kondisi selain memiliki *sunny-day scenario* sebaliknya juga memiliki *rainy-day scenario* (apa yang terjadi jika sesuatu salah?) atau alternatif lain yang bisa terjadi.

3.1.4 *Requirement Review*

Dalam proses iniperencanaan yang telah dilakukan di*review* kembali apakah masih terdapat kekurangan pada setiap tahapan baik itu pembuatan *functional requirement*, pembuatan domain model maupun *use case diagram*. Pada saat ini masih ditemukan kekurangan pada pembuatan model. Perlulah agar *class User* tidak dibebankan pada proses authentikasi. Sehingga perlu ditambah sebuah class baru yang bernama Auth.



Gambar 4 Perubahan Domain Model 1

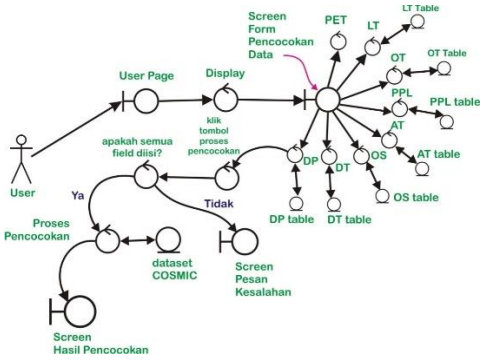
Hasil dari penambahan class Auth terlihat pada gambar 8 dimana *class Auth* adalah *directed association* yang ditanamkan di*classUser*. Walaupun tidak dijalankan secara langsung pada *class* yang lain namun *class User* selalu dijalankan disetiap proses dalam rangka memperoleh identitas user.

3.1.5 *Robussness Analysis*

Dalam perancangan selalu terdapat rentang yang jauh antara implementasi dan proses perancangan. Seorang *architect software engineering* kerap tidak berguna jika rancangannya tidak bisa dimengerti oleh programmer terlebih programmer pemula. Penggunaan *Robusness Analysis digunakan* untuk menjembatani antara analisis dan perancangan. Sehingga gap-gap antara perancangan dan implementasi tidak terlalu jauh dan yang paling penting adalah dapat dipahami secara teknis. Untuk *robustness analysis* pada tiap skenario ditampilkan sebagai berikut :

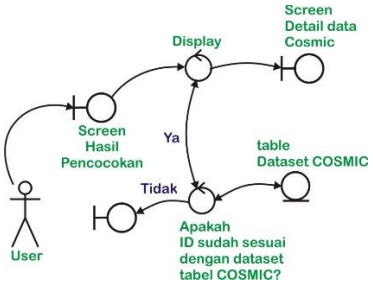
1. Melakukan proses *retrieve* yaitu melakukan pencocokan data pada database hingga proses hasil pencocokan.

ISSN: 1978-8126
e-ISSN: 2527-7340



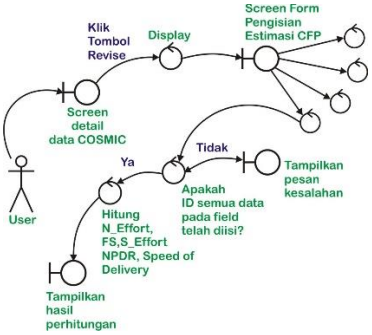
Gambar 5 Robustness Analysis Retrieve

2. Melihat detail hasil kecocokan data



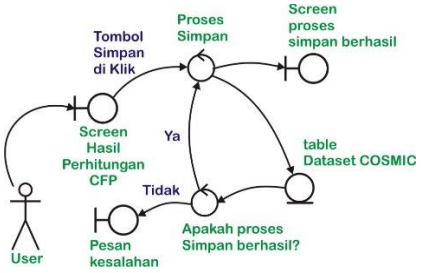
Gambar 6 Robustness Analysis Reuse

3. Melakukan proses estimasi



Gambar 7 Robustness Analysis Revise

4.Menyimpan data estimasi kedalam *database*

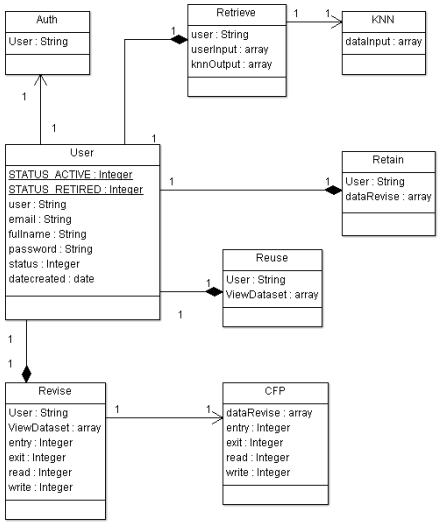


Gambar 8 Robustness Analysis Retain

Proses *robustness analysis* menjelaskan tentang bagaimana skenario digambarkan secara grafik sehingga lebih mudah dipahami oleh programmer. Proses ini juga mengganti grafik *flowchart* dengan cara kerja sistem yang berkerja secara prosedural. Pada tahap ini atribut tiap domain model diisi dengan variabel atau property agar arah dan perubahan rancangan lebih jelas ketika akan diimplementasikan kedalam pola

ISSN: 1978-8126
e-ISSN: 2527-7340

architecturenya. Dalam penelitian ini pola arsitektur framework Zend Framework 3 yang akan digunakan. Perubahan model tersebut adalah sebagai berikut:



Gambar 9 Perubahan Domain Model ke-3

Pada rancangan yang dibentuk, beberapa variabel dibentuk dengan tipe *array* ini dibuat agar variabel inputan,output dan proses tidak terlalu banyak terutama pada hasil tangkapan yang terdapat pada form. Sedangkan pada class user terdapat variabel constant yang bernama STATUS_RETIRED dan STATUS_ACTIVE yang berguna untuk menentukan nilai constant 2 untuk retired dan 1 untuk active dan variabel ini dimanfaatkan jika sistem mendeteksi user dalam keadaan aktif atau ditangguhkan.

3.1.6 Prelimary Design Review

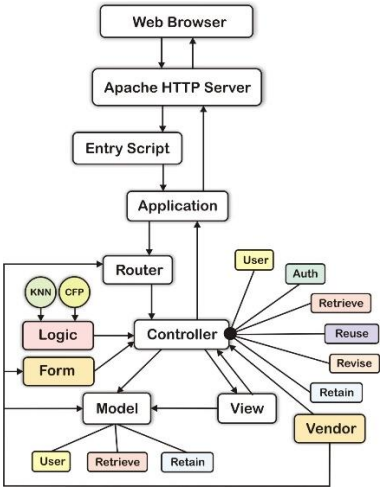
Dalam proses ini developer yang menggunakan metode ICONIX process diharuskan untuk meninjau kembali apakah masih ada revisi yang harus dilakukan pada tahap-tahap lain yang disebabkan oleh perubahan yang telah terjadi. Hal ini adalah langkah terakhir dimana pengguna yang non-technical atau semi teknikal terlibat karena pada proses selanjutnya adalah menentukan teknikal arsitekturnya. Namun bilamana terjadi perubahan maka buatlah dalam sebuah milestone atau rilis untuk versi tambahan.

3.1.7 Penentuan *Technical Architecture*

Pada penelitian ini teknikal arsitektur yang digunakan adalah Zend Framework 3 atau ZF3 yang telah rilis september 2016. Penggunaan ZF3 mengharuskan peneliti untuk mengembangkan program berbasis SOLID disebabkan oleh metodologi dalam penulisan script yang digunakan oleh framework ini. Selain itu framework ini juga menggunakan basis Model-View-Controller atau MVC sehingga jenis script dibagi berdasarkan layer dan kegunaannya.

Penggunaan bentuk MVC mengharuskan peneliti memisahkan fungsi/method yang terdapat pada class. Fungsi yang berhubungan langsung dengan database dikumpulkan kedalam class didalam Model, yang berhubungan dengan template, tampilan dan sebagainya dimasukkan kedalam View sedangkan fungsi yang mengontrol view dan berkomunikasi dengan model dikumpulkan didalam Controller. Selain itu penambahan logika bisnis juga akan dikontrol oleh Controller. Untuk diagram teknikal arsitektur ditampilkan pada gambar 10 sebagai berikut.

ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 10 Teknikal Arsitektur

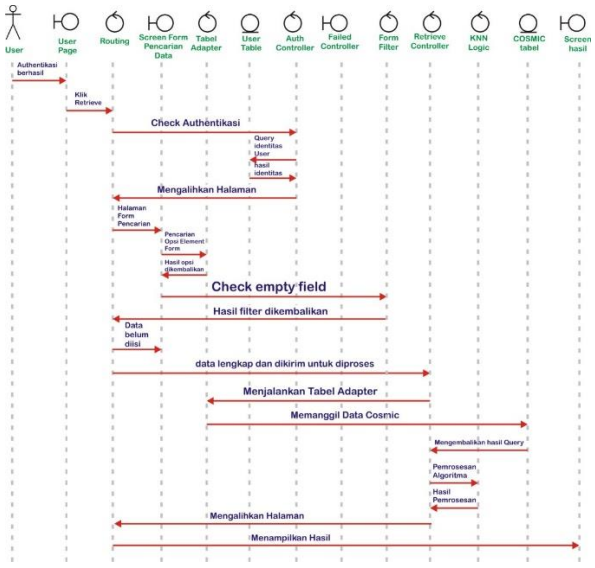
Pada gambar diatas menampilkan pusat pengoperasian tersebar di *controller* sedangkan model dibuat hanya ketika *controller* yang ada bersentuhan dengan database. Sedangkan form dibuat kelas tersendiri sebagai sarana input data ke sistem. Untuk penambahan *class* baru seperti vendor, *class* ini digunakan oleh sistem untuk memanggil plugin tambahan yang telah disediakan oleh ZF3.

3.1.8 Pembuatan *Squence Diagram*

Penggabungan data dan operasi kedalam sebuah entitas adalah fondasi awal sebuah pemograman yang berorientasi objek. Untuk domain model langkah yang telah dilakukan saat ini ialah mengisi domain dengan properti. Langkah selanjutnya adalah mengisi domain model tersebut dengan beragam operasi. *Squence diagram* bermanfaat untuk menemukan operasi yang sesuai dengan hasil *robustness analysis*. Tujuan utamanya adalah mengisi *class* yang ada dengan *behavior* dan tidak menunjukkan langkah-langkah secara detail. Untuk itu pada pembahasan sebelumnya telah diutarakan bahwa penggunaan ZF3 akan menyebabkan terbentuknya class baru seperti *Controller* begitu juga model yang akan membentuk *class* baru seperti UserTabel, User dan seterusnya.

Untuk *squence diagram* yang dibuat ditampilkan berdasarkan skenario dan *functional requirement* serta UML sebagai berikut :

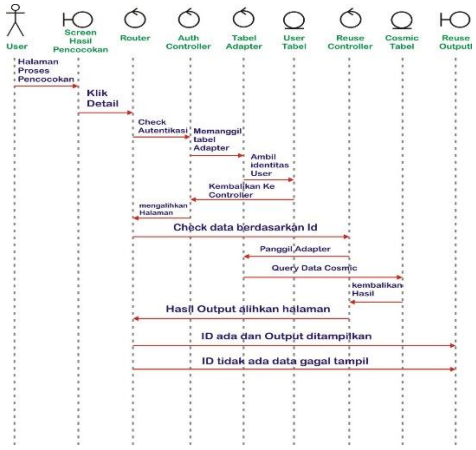
1. Melakukan proses retrieve yaitu melakukan pencocokan data pada database hingga proses hasil pencocokan.



Gambar 11 Squence Diagram Retrieve

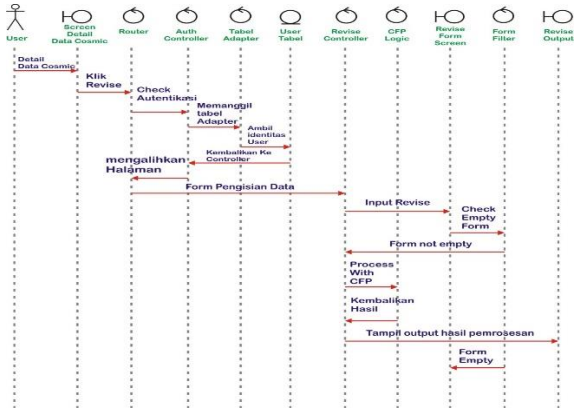
ISSN: 1978-8126
e-ISSN: 2527-7340

2. Detail hasil kecocokan data



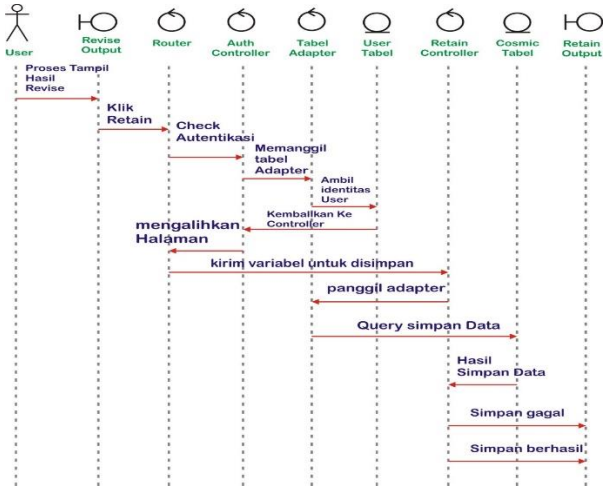
Gambar 12 Squence Diagram Melihat Detail Kecocokan

3. Melakukan Estimasi



Gambar 13 Squence Diagram Proses Revise

4. Menyimpan proses estimasi kedalam tabel.



Gambar 14 Squence Diagram Proses Retain

Setelah proses ini dilakukan, maka proses selanjutnya adalah evaluasi dari proses yang telah dilakukan.

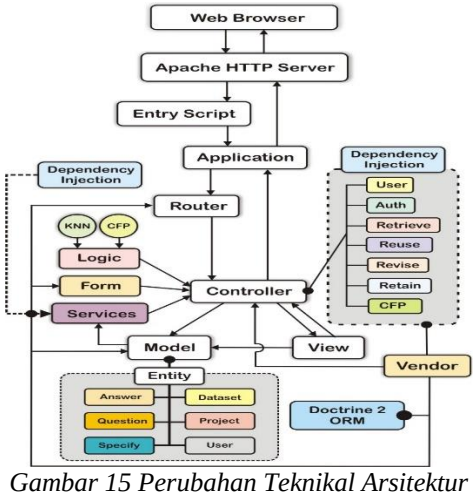
ISSN: 1978-8126
e-ISSN: 2527-7340

3.2 Perubahan Rancangan

Pada penggunaan *table adapter* pada penelitian ini diganti dengan menggunakan vendor yang terdapat pada plugin Zend Framework 3 yaitu *Doctrine* 2 ORM. *Doctrine* 2 ORM adalah *object-relation mapper* (ORM) untuk PHP 5.4+ yang menggunakan prinsip pola pemetaan data dan bermanfaat untuk mempola domain database sehingga menjadi objek yang dapat digunakan dalam pemograman berorientasi objek. Untuk itu perubahan pada *squence diagram* mengganti *Tabel Adapter* dengan *Entity Manager* dan menggunakan *class Doctrine\ORM\Mapping* yang terdapat vendor tetapi tidak merubah proses pada *squence diagram* secara signifikan.

3.2.1 Perubahan Teknikal Arsitektur[1]

Penggunaan *Doctrine* 2 ORM pada arsitektur Zend Framework 3 mengharuskan melakukan perubahan strategi dan sedikit perubahan teknikal arsitektur. Perubahan ini dilakukan lebih cepat agar mengurangi tingkat kesalahan pada proses pembuatan *class diagram* nantinya. Sehingga perubahan tersebut adalah sebagai berikut:



Gambar 15 Perubahan Teknikal Arsitektur

Penggunaan *Dependency Injection* pada teknikal arsitektur ini dilakukan agar program yang di-*inject* lebih sederhana dan memungkinkan lebih bersih dikarenakan tidak ada deklarasi *class* didalam object disetiap fungsi, selain itu *dependency injection* membuat sistem lebih aman dikarenakan class hanya menerima deklarasi *class* melalui injeksi yang terdaftar dirouter. Keamanan fitur diperkuat dengan *dependency injection* yang dilakukan melalui *factory class*. *Dependency Injection* dilakukan pada *Service dan Controller*.

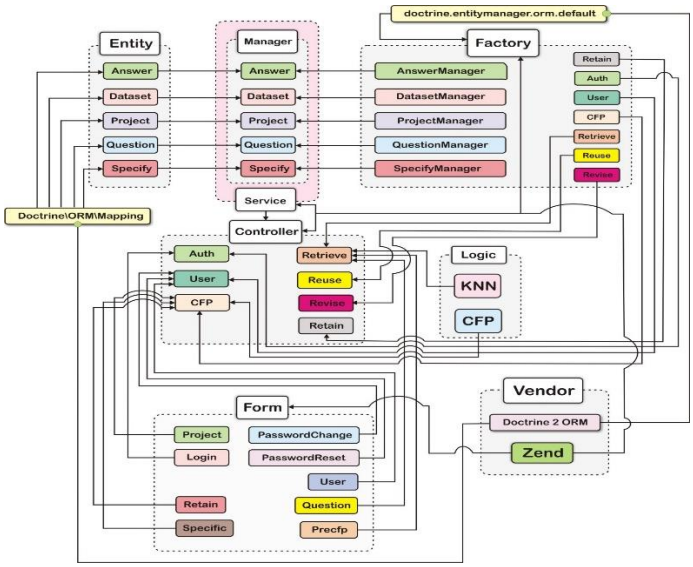
Penggunaan *Doctrine* 2 ORM memungkinkan untuk membentuk domain *database* dalam bentuk sebuah class sehingga dapat dimanfaatkan pada pemograman berorientasi object seperti pada Zend Framework 3 dan membaca setiap anotasi yang dideklarasikan dalam bentuk komentar. Tabel dibentuk menjadi sebuah *class* dan *field* menjadi propertinya sedangkan proses pengambilan setiap baris pada *field* menggunakan method/fungsi `getFieldName()` yang akan me-*return* variabel/properti yang ada pada *class* tersebut dan untuk mengisi nilai dengan method atau fungsi `setFieldName($property)`.

Untuk service, berisikan fitur *class* yang mengatur setiap entitas. Zend Framework 3 meng*injeksi Entity* kedalam service oleh *factory* bersamaan dengan proses *injeksi Entity Manager Doctrine* 2 ORM kedalam *service* dan *service* tersebut ketika akan digunakan juga di injeksi kedalam Controller via factory.

3.2.2 Perubahan Class Diagram

Setelah dilakukan perubahan arsitektur sistem yang dibangun, langkah selanjutnya pada iconix proses adalah mengevaluasi kembali serta merubah class diagram yang telah dibuat sebelumnya. Perubahan tersebut disesuaikan dengan arsitektur sistem yang telah ada. Oleh karena itu maka akan terbentuk beberapa class baru.

ISSN: 1978-8126
e-ISSN: 2527-7340



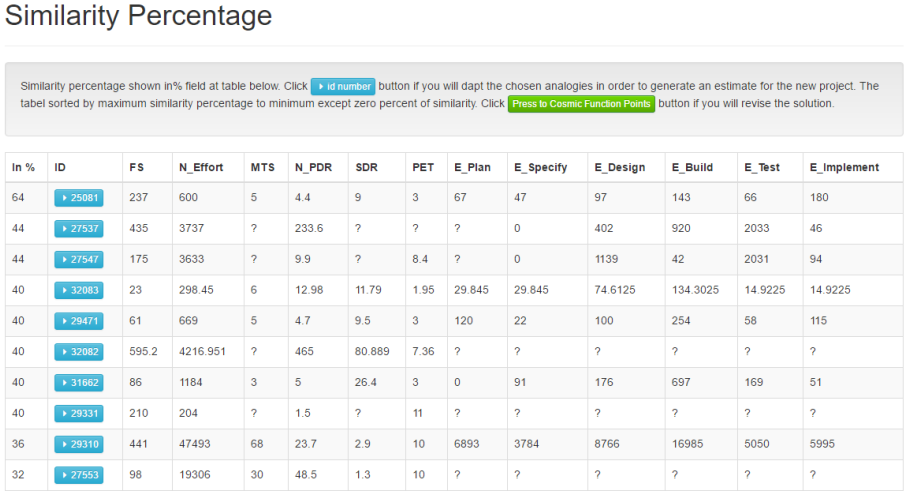
Gambar 16 Perubahan Rancangan Class

Penambahan controller CFP pada diagram atau gambar 16 dilakukan agar user juga dapat melakukan perhitungan besaran proyek tanpa harus melakukan proses pencocokan dengan kasus yang lampau dalam hal ini proses *Case Based Reasoning*. *Class* `ProjectForm` dibuat untuk mengumpulkan data projek yang ingin dihitung besarannya. *Class* `QuestionForm` digunakan untuk mengumpulkan informasi proyek yang ingin dicari kesamaanya dengan data kasus sebelumnya sedangkan *class* `SpecificForm` digunakan untuk memasukkan informasi alur kerja sistem yang terkategori dalam pengukuran besaran *software* menggunakan COSMIC *Function Point*.

3.3 Tampilan Menu Program

3.3.1 Retrieve

Proses *retrieve* adalah proses awal untuk memulai pencarian data melalui proses estimasi dengan mencari jarak terdekat dengan data yang ada. Proses retrieve dimulai dengan memilih menu CBR. Menu CBR >Question menampilkan form *Question* yang digunakan untuk mengumpulkan kunci pencarian informasi. Setelah semua pertanyaan dijawab pada Retrive Form, tekan tombol Save untuk menyimpan semua jawaban yang telah diisi. Kemudian akan muncul tampilan My Answer list.



In %	ID	FS	N_Effort	MTS	N_PDR	SDR	PET	E_Plan	E_Specify	E_Design	E_Build	E_Test	E_Implement
64	25081	237	600	5	4.4	9	3	67	47	97	143	66	180
44	27537	435	3737	?	233.6	?	?	?	0	402	920	2033	46
44	27547	175	3633	?	9.9	?	8.4	?	0	1139	42	2031	94
40	32083	23	298.45	6	12.98	11.79	1.95	29.845	29.845	74.6125	134.3025	14.9225	14.9225
40	29471	61	669	5	4.7	9.5	3	120	22	100	254	58	115
40	32082	595.2	4216.951	?	465	80.889	7.36	?	?	?	?	?	?
40	31662	86	1184	3	5	26.4	3	0	91	176	697	169	51
40	29331	210	204	?	1.5	?	11	?	?	?	?	?	?
36	29310	441	47493	68	23.7	2.9	10	6893	3784	8766	16985	5050	5995
32	27553	98	19306	30	48.5	1.3	10	?	?	?	?	?	?

Gambar 17 Tampilan hasil proses estimasi berdasarkan kesamaan

Halaman tampilan hasil ini terbagi menjadi dua bagian, yang atas adalah bagian informasi target sedangkan pada bagian *similarity percentage* adalah bagian hasil dari perhitungan kesamaan terhadap dataset COSMIC yang disimpan pada tabel dataset. Kolom In% berisi persentase kesamaan terhadap dataset yang ada. Nilai yang paling besar berada diurutan yang paling atas.Kolom id adalah ID dataset

ISSN: 1978-8126
e-ISSN: 2527-7340

yang tersimpan. Kolom id tersebut dapat diklik karena merupakan tombol yang menuju ke detail data pada proses *Reuse*. FS adalah Functional Size dalam hal ini CFS(Cosmic function points), N_Effort adalah Normalized Effort, MTS (Maximum Team Size), N_PDR (Normalize Productivity Development Rate), SDR (Speed of Development Rate), PET(Project Elapsed Time) jumlah waktu yang digunakan untuk menyelesaikan project, dan selebihnya adalah data persentase pembagian waktu pada live cycle development.

3.3.2 Reuse

Untuk melakukan proses Reuse tekan salah satu tombol yang terdapat pada kolom id. Halaman reuse menampilkan keseluruhan dataset secara lengkap berdasarkan ID yang dipilih. Data ini adalah referensi bagi developer yang ingin menemukan kasus lama yang sesuai dengan project yang akan dikerjakan.

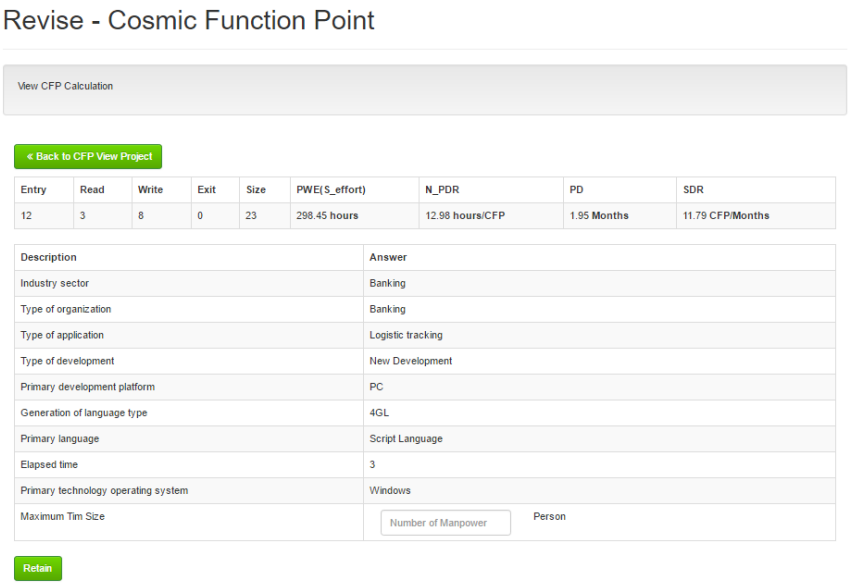
Description	Estimate
ID Dataset	25081
Industri Sector (InS)	Banking
Type of Organisation (OT)	Banking
Type of Application (AT)	Financial transaction process/accounting
Type of Development (DT)	New Development
Primary Development Platform (DP)	PC
Generation of Language Type (LT)	3GL
Primary Programming Language (PPL)	C++
Project Elapsed Time (PET)	3
1st Operating System (OS1)	Windows
Functional Size (FS)	237
Relative Size (RS)	M1
Normalised Work Effort (N_effort)	600
Summary Work Effort (S_effort)	600
Max Team Size (MTS)	5
Normalised Productivity Delivery Rate (N_PDR)	4.4
Speed of Delivery Rate (SDR)	9
Project Elapsed Time (PET)	Java
1st Language (Language1)	67
Work Effort Breakdown (Plan)	47
Work Effort Breakdown (Specify)	97
Work Effort Breakdown (Design)	143
Work Effort Breakdown (Build)	66

Gambar 18 Tampilan Reuse – Adaptation Process

3.3.3 Revise

Untuk melakukan proses revise dapat kembali melakukan proses *GoEstimate* dan menekan tombol *Revice to Cosmic Function Point*. Setelah tombol ditekan maka proses perhitungan *effort* dengan menggunakan ketentuan pada *Cosmic Function Point* dijalankan dan menghasilkan tampilan sebagai berikut.

ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 19 Revise – Tampilan Perhitungan Cosmic Function Point

3.3.4 Retain

Proses *retain* akan terdeteksi oleh sistem jika *project* yang terdaftar berasal dari proses dan menu CBR tetapi jika proses tersebut berasal dari Menu Cosmic > New Project maka proses *retain* dan tombol form tersebut tidak akan muncul. Isikan form *maximum team size*.

Untuk melakukan proses *retain* atau menyimpan data yang telah dihitung kedalam *database* dengan cara menekan tombol *retain*.

Gambar 20 Tampilan Hasil Retain 1

ISSN: 1978-8126
e-ISSN: 2527-7340

N_PDR	12.98
SDR	11.79
PET	1.95
PAS	Planning_Specification_Design_Build_Test_Implement
E_Plan	29.845
E_Specify	29.845
E_Design	74.6125
E_Build	134.3025
E_Test	14.9225
E_Implement	14.9225
Language1	Script Language
OS1	Windows

Gambar 21 Tampilan Hasil Retain 2 (Sambungan)

4. Kesimpulan

Proses perancangan *Software Effort Estimation* dan *Case Based Reasoning* dengan menggunakan metode ICONIX *Process* pada Arsitektur Zend Framework 3 melalui proses bertahap dan melakukan proses evaluasi secara berulang untuk setiap kegiatan agar dapat meminimumkan kesalahan dalam menterjemahkan diagram yang dihasilkan pada proses perancangan. Proses yang terjadi pada arsitektur Zend Framework 3 pada saat implementasi ke kode program khususnya pada mapping *object-object* dalam *database* lebih terbantu disebabkan oleh penggunaan *Doctrine* 2 ORM yang memetakan tabel-tabel dalam sebuah Class. Sehingga dapat dimanfaatkan secara lebih mudah pada pemograman berorientasi objek. Perhitungan Cosmic Function Point dilakukan pada aplikasi Zend Framework 3 menjadi lebih terstruktur dengan baik dikarenakan proses perhitungan didesain menggunakan *class*. Runutan tahap penggunaan ditentukan oleh proses yang terjadi pada metode *Case Based Reasoning.* Pembuatan perhitungan Cosmic Function Point yang terdapat pada menu Cosmic dirasa perlu untuk melakukan perhitungan secara manual tanpa menggunakan proses retain sehingga *user* dapat melakukan perhitungan Cosmic function point secara terpisah dengan proses CBR. Secara keseluruhan aplikasi menggunakan Zend Framework 3(ZF3) dan Doctrine 2 ORM lebih cepat dari pendahulunnya yaitu versi Zend Framework 2. Tanpa Doctrine 2 ORM. Proses pada ZF3 dan Doctrine 2 ORM tidak menggunakan pembuatan *interface* sewaktu memetakan tabel dan tidak menggunakan tabel adapter yang terdefenisi di Module.php sehingga program yang tidak memerlukan database tidak merequest perintah tabel adapter.

Daftar Pustaka

[1] A. Mishra and D. Dubey, "A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios," *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 1, no. 5, pp. 2321–7782, 2013, [Online]. Available: http://www.ijarcsms.com/docs/paper/volume1/issue5/V1I5-0008.pdf

[2] E. S. Hidalgo, "Adapting the scrum framework for agile project management in science: case study of a distributed research initiative," *Heliyon*, vol. 5, no. 3, p. e01447, 2019, doi: 10.1016/j.heliyon.2019.e01447.

[3] R. de A. Araújo, A. L. I. Oliveira, and S. Meira, "A class of hybrid multilayer perceptrons for software development effort estimation problems," *Expert Syst. Appl.*, vol. 90, pp. 1–12, 2017, doi: 10.1016/j.eswa.2017.07.050.

[4] S. Mukaromah, A. B. Putra, and N. C. Wibowo, "Analisis dan Perancangan Sistem Informasi Logbook dengan Iconix Process," *Rekayasa Teknol. Ind. dan Inf. 2017*, pp. 130–135, 2017, [Online]. Available: https://journal.sttnas.ac.id/ReTII/article/download/601/506

[5] A. Ristyawan *et al.*, "Proses Iconix Dalam Analisa Rancangan Aplikasi Informasi," *J. SIMETRIS*, vol. 10, no. 1, pp. 1–14, 2019.

[6] D. Rosenberg, M. Stephens, and M. Collins-cope, *with ICONIX Process*. 2005.

[7] A. A. N. Mostafa, "Five different agile methods," no. April, 2016.

[8] V. Krishna Madasu, T. Venkata Swamy Naidu Venna, and T. Eltaeib, "SOLID Principles in Software Architecture and Introduction to RESM Concept in OOP," *J. Multidiscip. Eng. Sci.*

ISSN: 1978-8126
e-ISSN: 2527-7340

Technol., vol. 2, no. 2, pp. 3159–3199, 2015, [Online]. Available: www.jmest.org

[9] M. R. Borroek, E. Rasywir, and Y. Pratama, "Pengukuran Perangkat Lunak Untuk Effort Estimation Dengan Teknik Pembelajaran Mesin," *J. Media Inform. Budidarma*, vol. 4, no. 2, p. 445, 2020, doi: 10.30865/mib.v4i2.2083.

[10] P. Phannachitta, "On an optimal analogy-based software effort estimation," *Inf. Softw. Technol.*, vol. 125, no. May, p. 106330, 2020, doi: 10.1016/j.infsof.2020.106330.

[11] S. J. Delany, P. Cunningham, and W. Wilke, "The Limits of CBR in Software Project Estimation," *Ger. Work. Case-Based Reason.*, no. November 1999, pp. 1–10, 1998.

[12] S. J. Delany, P. Cunningham, and W. Wilke, "The Limits of CBR in Software Project Estimation," *Ger. Work. Case-Based Reason.*, no. October, pp. 1–10, 1998.

[13] M. R. Borroek, E. Rasywir, Y. Pratama, Fachruddin, and M. Istoningtyas, "Analysis on Knowledge Layer Application for Knowledge Based System," *Proc. 2018 Int. Conf. Electr. Eng. Comput. Sci. ICECOS 2018*, pp. 177–182, 2019, doi: 10.1109/ICECOS.2018.8605262.

[14] K. Banga and S. Takai, "A comparison of case-based reasoning and regression analysis approaches for cost uncertainty modeling," *Proc. ASME Des. Eng. Tech. Conf.*, vol. 6, pp. 213–222, 2010, doi: 10.1115/DETC2010-28232.

[15] A. Smiti and Z. Elouedi, "SCBM: soft case base maintenance method based on competence model," *J. Comput. Sci.*, vol. 25, pp. 221–227, 2018, doi: 10.1016/j.jocs.2017.09.013.

[16] A. Brady, T. Menzies, O. El-Rawas, E. Kocaguneli, and J. W. Keung, "Case-Based Reasoning for Reducing Software Development Effort," *J. Softw. Eng. Appl.*, vol. 03, no. 11, pp. 1005–1014, 2010, doi: 10.4236/jsea.2010.311118.

[17] A. Jadhav and S. K. Shandilya, "Reliable machine learning models for estimating effective software development efforts: A comparative analysis," *J. Eng. Res.*, vol. 11, no. 4, pp. 362–376, 2023, doi: 10.1016/j.jer.2023.100150.

[18] M. Learning, "Encyclopedia of Machine Learning," *Encycl. Mach. Learn.*, 2010, doi: 10.1007/978-0-387-30164-8.

[19] L. Cao and C. Zhang, "Domain driven data mining," *Data Min. Knowl. Discov. Technol.*, vol. 7, no. 4, pp. 196–223, 2008, doi: 10.4018/978-1-59904-960-1.ch009.