repository.ub.ac.id

# EVALUASI BIAYA PENGEMBANGAN SISTEM RESERVASI ONLINE GEDUNG SERBAGUNA PT GHL PENINSULA MENGGUNAKAN METODE ADVANCE USE CASE POINT DAN REVISED USE CASE POINT

## (Studi Kasus: CV Gumcode Indonesia)

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Savira Fahrunisa
NIM: 145150401111032

PROGRAM STUDI SISTEM INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

UNIVERSITAS BRAWIJAYA

# PENGESAHAN

EVALUASI BIAYA PENGEMBANGAN SISTEM RESERVASI ONLINE GEDUNG SERBAGUNA PT GHL PENINSULA MENGGUNAKAN METODE ADVANCE USE CASE POINT DAN REVISED USE CASE POINT (Studi Kasus: CV Gumcode Indonesia)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Savira Fahrunisa
NIM: 145150401111032

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Juni 2018
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Mochamad Chandra Saputra, S.Kom., M.Eng. | Admaja Dwi Herlambang, S.Pd., M.Pd. |
| NIK:2016098601061001 | NIP: 2016098908021001 |

Mengetahui
Ketua Jurusan Sistem Informasi

Dr. Eng., Herman Tolle, S.T, M.T.
NIP: 19740823 200012 1 001

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

# IDENTITAS TIM PENGUJI

Nama : Himawat Aryadita, S.T, M.Sc

NIP/NIK : 19801018200801 1003

Email : himawat[at]ub[dot]ac[dot]id

Nama : Aditya Rachmadi, S.ST., M.TI

NIP/NIK : 201201860421 1001

Email : rachmadi.aditya[at]ub[dot]ac[dot]id

UNIVERSITAS BRAWIJAYA

# PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 29 Juni 2018

Savira Fahrunisa

NIM: 145150401111032

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

# DAFTAR RIWAYAT HIDUP

**Data Pribadi**

| | |
|---|---|
| Nama | : Savira Fahrunisa |
| Tempat/Tanggal Lahir | : Tulungagung, 2 Mei 1996 |
| Jenis Kelamin | : Perempuan |
| Agama | : Islam |
| Alamat | : Dsn. Sukowidodo Barat RT 02 RW 01 No.43, Ds Sukowidodo, Kec. Karangrejo, Kab. Tulungagung, Jawa Timur |
| Telepon HP | : 085851474026 |

**Latar Belakang Pendidikan**

A. Pendidikan Formal

2002-2008 SDN 1 Sukowidodo

2008-2011 SMPN 3 Tulungagung

2011-2014 SMAN 1 Kauman

2014-2018 Sistem Informasi Universitas Brawijaya

**Pengalaman Organisasi**

- Staff Divisi Acara PTIIK Mengajar 2015
- Staff Divisi Internal Kelompok Riset Mahasiswa (K-RISMA) FILKOM Periode 2015-2016
- Asisten Praktikum Pemrograman Lanjut Periode Genap 2015-2016
- Sekretaris 2 Acara English Corner FILKOM 2016
- Sekretaris Eksternal Kelompok Riset Mahasiswa (K-RISMA) FILKOM Periode 2016-2017

**Kemampuan Teknis**

- Sistem operasi (Microsoft Windows XP/Vista/8/10)
- Bahasa Pemrograman (Java, PHP, HTML)
- Software (Netbeans, IBM Data Studio, DB Designer, Android Studio)
- Database (MySQL, DB2)

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

# KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala berkat, rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul "Evaluasi Biaya Pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula Studi Kasus: CV Gumcode Indonesia", sebagai salah satu syarat bagi penulis untuk memperoleh gelar sarjana komputer pada Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam penyusunan dan penulisan skripsi ini, penulis memperoleh bantuan, bimbingan, dukungan, serta motivasi dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terimakasih kepada:

1. Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
2. Herman Tolle, Dr. Eng., S.T, M.T. selaku Ketua Jurusan Sistem Informasi Fakultas Ilmu Komputer Universitas Brawijaya.
3. Suprapto, S.T, M.T. selaku Ketua Prodi Sistem Informasi Fakultas Ilmu Komputer Universitas Brawijaya.
4. Mochamad Chandra Saputra, S.Kom., M.Eng selaku dosen pembimbing satu yang telah meluangkan banyak waktu untuk memberikan ilmu, bimbingan, saran, motivasi dan do'a pada penulis selama penyelesaian skripsi ini.
5. Admaja Dwi Herlambang, S.Pd., M.Pd. selaku dosen pembimbing dua yang telah meluangkan banyak waktu untuk memberikan ilmu, bimbingan, saran, pada penulis selama penyelesaian skripsi ini.
6. Aditya Rachmadi, S.ST., M.TI. selaku dosen pembimbing akademik yang telah meluangkan waktu untuk memberikan bimbingan, saran, ilmu dan motivasi pada penulis selama masa perkuliahan.
7. Kedua orang tua, Bapak Endra Gunawan dan Ibu Indah Sukariana, serta adik, Rosihan Al-Fauzy, dan keluarga besar yang selalu memberikan dukungan baik moril dan materiil, serta do'a selama pengerjaan skripsi ini.
8. Saudara Ardeni Bayu dan Zulfiar Ryanda, selaku Manajer Proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula dan CEO CV Gumcode Indonesia, yang telah mengizinkan penulis melakukan observasi, wawancara dan penilaian lembar penelitian, serta bersedia membantu dalam proses pengumpulan data penelitian.
9. Segenap teman seperjuangan dan diskusi yang menemani penulis berbagi pikiran dan cerita selama di perkuliahan, Aziza Zuhroh, Liliandara, Eki Yusandhi, Diago Ariesandika dan Indra Kurniawan, Dimas Hartanto, Iqbal Taufiq, Jawara Wahyu, Dhimas Alamsyah, Ridho Fadhlurrahman, Zaenal Kurniawan, Masagus Hariadi, Raga Saputra, Teguh Thapsuandji, Ema Agasta, Windha Parwaningsuci, Siti Sarah R, Desy Miladiana, Istania Salma, Cahyaningtyas Sekar, Farah Amalia, Sinta Kurniasari, Larizza N, Aneta Puspita, Hanna Nur Hanifa, dan Via Aprilia, serta teman-teman angkatan 2014 dan teman-teman lainnya yang tidak dapat penulis sebutkan satu persatu atas pengalaman berharga dan bantuan yang telah diberikan kepada penulis selama di perkuliahan.

10. Semua pihak yang tidak bisa penulis sebutkan satu per satu yang telah membantu penulis dalam menyelesaikan skripsi ini.

Segala kritik serta saran yang membangun penulis harapkan untuk perbaikan penelitian selanjutnya. Semoga skripsi ini bermanfaat bagi penulis maupun pembaca.

Malang, 29 Juni 2018

Penulis

savirafahrunisa@gmail.com

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

# ABSTRAK

**Savira Fahrunisa, Evaluasi Biaya Pengembangan Sistem Reservasi Onlinde Gedung Serbaguna PT GHL Peninsula Menggunakan Metode Advance Use Case Point dan Revised Use Case Point**

**Dosen Pembimbing: Mochamad Chandra Saputra S.Kom., M.Eng. dan Admaja Dwi Herlambang, S.Pd., M.Pd.**

Penelitian ini dilakukan di CV Gumcode Indonesia dimana terdapat sebanyak 40% proyek perangkat lunak disana mengalami keterlambatan dari jadwal, termasuk pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang mengalami keterlambatan selama 3 minggu dari jadwal. Hal ini terjadi karena CV Gumcode Indonesia tidak memiliki standar perhitungan biaya, sehingga kesulitan menentukan ukuran perangkat lunak, estimasi waktu dan estimasi biaya. CV Gumcode Indonesia menghitung biaya berdasarkan pengalaman dan teknologi yang digunakan, sehingga hasilnya berupa campuran dugaan dan perhitungan.

Penelitian ini bertujuan untuk mengelompokkan pekerjaan ke dalam *Work Breakdown Structure* (WBS), lalu menghitung estimasi waktu dan biaya dengan menggunakan metode *Advance Use Case Point* dan *Revised Use Case Point* yang menghasilkan ukuran perangkat lunak berupa estimasi *effort* yang nantinya dikonversikan ke dalam estimasi waktu dan biaya, dan menjadwalkan ke dalam *Gantt Chart*. Metodologi penelitian dimulai dari identifikasi masalah, studi pustaka tentang permasalahan dan metode yang akan digunakan, menentukan jenis penelitian, menyusun instrumen penelitian lalu menguji kelayakannya, menentukan responden, mengumpulkan data, mengolah data, menjadwalkan proyek, menganalisis hasil perbandingan, serta menarik kesimpulan dan saran. Pengumpulan data dilakukan dengan teknik wawancara, observasi sistem dan lembar penilaian, kemudian data diolah menjadi *Use Case Diagram* dan *Use Case Scenario* yang merupakan inputan dari kedua metode. Data ini akan menghasilkan ukuran proyek yang nantinya dikonversi ke dalam estimasi waktu dan biaya. Perhitungan *Advance Use Case Point* menghasilkan alokasi waktu selama 185,93 jam dan biaya sebesar Rp 110.558. 733,00, sedangkan *Revised Use Case Point* menghasilkan alokasi waktu selama 140,1 jam dan biaya sebesar Rp 83.303.382,00. Berdasarkan perbandingan hasil kedua metode, *Advance Use Case Point* lebih direkomendasikan karena melibatkan fitur *End-User Development* yang dibutuhkan oleh sistem.

**Kata kunci**: evaluasi, perangkat lunak, estimasi biaya, advance use case point, revised use case point

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

# ABSTRACT

**Savira Fahrunisa, Cost Evaluation of Sistem Reservasi Onlinde Gedung Serbaguna PT GHL Peninsula Development Using Advance Use Case Point and Revised Use Case Point Method**

**Supervisors: Mochamad Chandra Saputra S.Kom., M.Eng. and Admaja Dwi Herlambang, S.Pd., M.Pd.**

*This research was conducted at CV Gumcode Indonesia where there was 40% of software projects in there were delayed from the schedule, included the development of the Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula which was delayed for 3 weeks from schedule. This happened because CV Gumcode Indonesia didn't have any standard of cost calculation, which made it difficult to determine the size of the software, estimated time and cost. CV Gumcode Indonesia calculated the cost based on experience and technology used, so the result was a mix of guesses and calculations.*

*This study aimed to classify the work into the WBS, then calculated the estimated time and cost using the Advance Use Case Point method and the Revised Use Case Point method which produced software size in the form of estimated effort that would be converted into estimated time and cost, and scheduled it into Gantt Chart. Research methodology began with problem identification, literature study on problem and method to be used, determined the type of research, developed research instruments and then tested its feasibility, determined respondents, collected data, processed data, scheduled projects, analyzed comparison results, and drew conclusion and suggestion. Data collection was done by interview, observation ob the software and assessment sheet. Data processed into Use Case Diagram and Use Case Scenario which was the input to both of the methods. This data produced project size that would be converted into estimated time and cost. The Advance Use Case Point calculation produced time allocation for 185.93 hours and total cost Rp 110,558.733.00, while the Revised Use Case Point produced time allocation for 140.1 hours and total cost Rp 83.303.382,00. Based on the comparison of the results of both methods, Advance Use Case Point was recommended because it involved the End-User Development features required by the system.*

***Keyword***: *evaluation, software, cost estimation, advance use case point, revised use case point*

UNIVERSITAS BRAWIJAYA

# DAFTAR ISI


PENGESAHAN ..... 2
PERNYATAAN ORISINALITAS ..... 3
KATA PENGANTAR ..... 5
ABSTRAK ..... 8
ABSTRACT ..... 9
DAFTAR ISI ..... 10
DAFTAR TABEL ..... 14
DAFTAR GAMBAR ..... 18
DAFTAR LAMPIRAN ..... 19
BAB 1 PENDAHULUAN ..... **Error! Bookmark not defined.**
1.1 Latar belakang ..... **Error! Bookmark not defined.**
1.2 Rumusan masalah ..... **Error! Bookmark not defined.**
1.3 Tujuan ..... **Error! Bookmark not defined.**
1.4 Manfaat ..... **Error! Bookmark not defined.**
1.5 Batasan masalah ..... **Error! Bookmark not defined.**
1.6 Sistematika pembahasan ..... **Error! Bookmark not defined.**
BAB 2 LANDASAN KEPUSTAKAAN ..... **Error! Bookmark not defined.**
2.1 Kajian Pustaka ..... **Error! Bookmark not defined.**
2.2 Profil CV Gumcode Indonesia ..... **Error! Bookmark not defined.**
2.2.2 Visi CV Gumcode Indonesia ..... **Error! Bookmark not defined.**
2.2.3 Misi CV Gumcode Indonesia ..... **Error! Bookmark not defined.**
2.3 Manajemen Proyek ..... **Error! Bookmark not defined.**
*2.4 Project Cost Management* ..... **Error! Bookmark not defined.**
*2.5 Project Schedule Management* ..... **Error! Bookmark not defined.**
*2.6 Project Resource Management* ..... **Error! Bookmark not defined.**
*2.7 Project Life Cycle* ..... **Error! Bookmark not defined.**
*2.8 System Development Life Cycle* ..... **Error! Bookmark not defined.**
2.9 *Work Breakdown Structure* (WBS) ..... **Error! Bookmark not defined.**
*2.10 Gantt Chart* ..... **Error! Bookmark not defined.**
*2.11 Use Case* ..... **Error! Bookmark not defined.**

*2.12* Metode *Advance Use Case Point*..... **Error! Bookmark not defined.**

2.12.1 *Unadjusted Actor Weight* (UAW)..**Error! Bookmark not defined.**

2.12.2 *Unadjusted Use Case Weight* (UUCW)**Error! Bookmark not defined.**

2.12.3 *Unadjusted Use Case Point* (UUCP)**Error! Bookmark not defined.**

2.12.4 *Technical Complexity Factor* (TCF) **Error! Bookmark not defined.**

2.12.5 *Environmental Complexity Factor* (ECF)**Error! Bookmark not defined.**

2.12.6 *Use Case Point* (UCP).....**Error! Bookmark not defined.**

*2.12.7 End User Development Technical Complexity Factor*.....**Error! Bookmark not defined.**

*2.12.8 End User Development Environmental Complexity Factor* ..**Error! Bookmark not defined.**

2.12.9 *Advance Use Case Point* (AUCP)....**Error! Bookmark not defined.**

2.12.10 Estimasi *Effort* .....**Error! Bookmark not defined.**

*2.13* Metode *Revised Use Case Point* .....**Error! Bookmark not defined.**

2.13.1 *Unadjusted Actor Weight* (UAW)..**Error! Bookmark not defined.**

2.13.2 *Unadjusted Use Case Weight* (UUCW)**Error! Bookmark not defined.**

2.13.3 *Unadjusted Use Case Point* (UUCP)**Error! Bookmark not defined.**

2.13.4 *Technical Complexity Factor* (TCF) **Error! Bookmark not defined.**

2.13.5 *Environmental Complexity Factor* (ECF)**Error! Bookmark not defined.**

2.13.6 *Revised Use Case Point* (Re-UCP) ..**Error! Bookmark not defined.**

2.13.7 Estimasi *Effort* .....**Error! Bookmark not defined.**

2.14 Distribusi *Effort* .....**Error! Bookmark not defined.**

2.15 Peran dan Standar Gaji Pengembang ....**Error! Bookmark not defined.**

BAB 3 METODOLOGI .....**Error! Bookmark not defined.**

3.1 Diagram Alir Penelitian .....**Error! Bookmark not defined.**

3.2 Identifikasi Masalah.....**Error! Bookmark not defined.**

3.3 Studi Pustaka .....**Error! Bookmark not defined.**

3.4 Penentuan Jenis Penelitian.....**Error! Bookmark not defined.**

3.5 Penyusunan Instrumen Penelitian.....**Error! Bookmark not defined.**

3.6 Pengujian Kelayakan Instrumen Penelitian**Error! Bookmark not defined.**

3.7 Penentuan Responden .....**Error! Bookmark not defined.**

3.8 Pengumpulan Data .....**Error! Bookmark not defined.**

3.8.1 Wawancara .....**Error! Bookmark not defined.**

3.8.2 Observasi.....**Error! Bookmark not defined.**

3.8.3 Lembar Penilaian.....**Error! Bookmark not defined.**

3.9 Pengolahan Data.....**Error! Bookmark not defined.**

*3.9.1* Perhitungan *Advance Use Case Point***Error! Bookmark not defined.**

*3.9.2* Perhitungan *Revised Use Case Point***Error! Bookmark not defined.**

*3.9.3* Konversi *Advance Use Case Point* dan *Revised Use Case Point ke Dalam Biaya, Waktu, dan SDM*.....**Error! Bookmark not defined.**

3.10 Penjadwalan Proyek .....**Error! Bookmark not defined.**

3.11 Analisis Hasil Perbandingan.....**Error! Bookmark not defined.**

3.12 Penarikan Kesimpulan dan Saran .....**Error! Bookmark not defined.**

BAB 4 HASIL DAN ANALISIS .....**Error! Bookmark not defined.**

4.1 Analisis Work Breakdown Structure.....**Error! Bookmark not defined.**

4.2 Analisis Pemodelan *Use Case* .....**Error! Bookmark not defined.**

4.3 Analisis Kondisi Eksisting Waktu Kerja.....**Error! Bookmark not defined.**

4.4 Analisis Perhitungan dengan *Advance Use Case Point***Error! Bookmark not defined.**

4.4.1 Analisis Perhitungan Unadjusted Use Case Point**Error! Bookmark not defined.**

4.4.2 Perhitungan *Complexity Factor*.....**Error! Bookmark not defined.**

4.4.3 Perhitungan *Use Case Point*.....**Error! Bookmark not defined.**

4.4.4 Perhitungan *End-User Development Factor***Error! Bookmark not defined.**

4.4.5 Perhitungan *Advance Use Case Point***Error! Bookmark not defined.**

4.4.6 Konversi *Advance Use Case Point* ke Biaya, Waktu dan SDM**Error! Bookmark not defined.**

4.5 Analisis Perhitungan dengan *Revised Use Case Point***Error! Bookmark not defined.**

4.5.1 Perhitungan *Unadjusted Use Case Point***Error! Bookmark not defined.**

4.5.2 Perhitungan Complexity Factor .....**Error! Bookmark not defined.**

4.5.3 Perhitungan *Revised Use Case Point***Error! Bookmark not defined.**

4.5.4 Konversi Revised Use Case Point ke Biaya, Waktu dan SDM.**Error! Bookmark not defined.**

4.6 Analisis Perhitungan CV Gumcode Indonesia**Error! Bookmark not defined.**

4.7 Analisis Penjadwalan dengan *Gantt Chart***Error! Bookmark not defined.**

4.7.1 *Gantt Chart* pada *Advance Use Case Point***Error! Bookmark not defined.**

4.7.2 *Gantt Chart* pada *Revised Use Case Point***Error! Bookmark not defined.**

BAB 5 PEMBAHASAN.....**Error! Bookmark not defined.**

5.1 Perbandingan Durasi Hasil Perhitungan Metode Advance Use Case Point, Revised Use Case Point dan CV. Gumcode Indonesia.....**Error! Bookmark not defined.**

5.2 Perbandingan Biaya Hasil Perhitungan Metode *Advance Use Case Point, Revised Use Case Point* dan CV. Gumcode Indonesia**Error! Bookmark not defined.**

5.3 Perbandingan Jumlah Pengembang Hasil Perhitungan Metode Advance Use Case Point, Revised Use Case Point dan CV. Gumcode Indonesia **Error! Bookmark not defined.**

BAB 6 Penutup .....**Error! Bookmark not defined.**

6.1 Kesimpulan .....**Error! Bookmark not defined.**

6.2 Saran .....**Error! Bookmark not defined.**

DAFTAR PUSTAKA.....**Error! Bookmark not defined.**

LAMPIRAN A LAPORAN HASIL WAWANCARA.....**Error! Bookmark not defined.**

LAMPIRAN B HASIL WAWANCARA PROSES PMBOK.**Error! Bookmark not defined.**

LAMPIRAN C BUKTI VALIDASI.....**Error! Bookmark not defined.**

LAMPIRAN D LEMBAR OBSERVASI SISTEM .....**Error! Bookmark not defined.**

LAMPIRAN E KEBUTUHAN FUNGSIONAL SISTEM .....**Error! Bookmark not defined.**

LAMPIRAN F IDENTIFIKASI PENGGUNA .....**Error! Bookmark not defined.**

LAMPIRAN G USE CASE SCENARIO.....**Error! Bookmark not defined.**

LAMPIRAN H LEMBAR PENILAIAN ADVANCE USE CASE POINT**Error! Bookmark not defined.**

LAMPIRAN I LEMBAR PENILAIAN REVISED USE CASE POINT**Error! Bookmark not defined.**

LAMPIRAN J GANTT CHART PADA ADVANCE USE CASE POINT**Error! Bookmark not defined.**

LAMPIRAN K GANTT CHART PADA REVISED USE CASE POINT**Error! Bookmark not defined.**




repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

# DAFTAR TABEL

Tabel 2. 1 Kajian Pustaka .....**Error! Bookmark not defined.**

Tabel 2. 2 Kajian Pustaka (Lanjutan) .....**Error! Bookmark not defined.**

Tabel 2. 3 Tipe dan Bobot Aktor *Advance Use Case Point***Error! Bookmark not defined.**

Tabel 2. 4 Tipe dan Bobot *Use Case Advance Use Case Point***Error! Bookmark not defined.**

Tabel 2. 5 *Technical Factors Advance Use Case Point***Error! Bookmark not defined.**

Tabel 2. 6 *Technical Factors Advance Use Case Point* (Lanjutan)**Error! Bookmark not defined.**

Tabel 2. 7 *Environmental Factors Advance Use Case Point***Error! Bookmark not defined.**

Tabel 2. 8 *End User Technical Factors*.....**Error! Bookmark not defined.**

Tabel 2. 9 *End User Environmental Factor*.....**Error! Bookmark not defined.**

Tabel 2. 10 Tipe dan Bobot Aktor *Revised Use Case Point***Error! Bookmark not defined.**

Tabel 2. 11 Tipe dan Bobot *Use Case Revised Use Case Point***Error! Bookmark not defined.**

Tabel 2. 12 *Technical Factors Revised Use Case Point***Error! Bookmark not defined.**

Tabel 2. 13 *Environmental Factors Revised Use Case Point***Error! Bookmark not defined.**

Tabel 2. 14 Distribusi *Effort* Setiap Aktivitas.....**Error! Bookmark not defined.**

Tabel 2. 15 Distribusi *Effort* Setiap Aktivitas (Lanjutan)**Error! Bookmark not defined.**

Tabel 2. 16 Peran Pengembang .....**Error! Bookmark not defined.**

Tabel 2. 17 Standar Gaji *Kelly Service* 2017 .....**Error! Bookmark not defined.**

Tabel 3. 1 Alur Metode *Advance Use Case Point* (AUCP)**Error! Bookmark not defined.**

Tabel 3. 2 Alur Metode *Advance Use Case Point* (AUCP) (Lanjutan).....**Error! Bookmark not defined.**

Tabel 3. 3 Alur metode *Revised Use Case Point*.....**Error! Bookmark not defined.**

Tabel 3. 4 Alur metode *Revised Use Case Point* (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 1 Standar Gaji Per-Jam Setiap Posisi ..... **Error! Bookmark not defined.**

Tabel 4. 2 Klasifikasi Aktor pada Sistem dengan AUCP**Error! Bookmark not defined.**

Tabel 4. 3 Perhitungan *Unadjusted Actor Weight* dengan AUCP**Error! Bookmark not defined.**

Tabel 4. 4 Transaksi *Use Case Login*.....**Error! Bookmark not defined.**

Tabel 4. 5 Transaksi *Use Case Chat* Admin .....**Error! Bookmark not defined.**

Tabel 4. 6 Transaksi *Use Case* Melakukan *Order* Paket Acara**Error! Bookmark not defined.**

Tabel 4. 7 Transaksi *Use Case* Melakukan *Order* Paket Acara (Lanjutan).....**Error! Bookmark not defined.**

Tabel 4. 8 Transaksi *Use Case* Menambah Review ...**Error! Bookmark not defined.**

Tabel 4. 9 Transaksi *Use Case* Melakukan Registrasi Vendor**Error! Bookmark not defined.**

Tabel 4. 10 Transaksi *Use Case* Melihat ringkasan laporan vendor.....**Error! Bookmark not defined.**

Tabel 4. 11 Transaksi *Use Case* Mengelola transaksi**Error! Bookmark not defined.**

Tabel 4. 12 Transaksi *Use Case* Mengelola transaksi (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 13 Transaksi *Use Case* melihat notifikasi ....**Error! Bookmark not defined.**

Tabel 4. 14 Transaksi *Use case* mengubah profil vendor**Error! Bookmark not defined.**

Tabel 4. 15 Transaksi *Use case* mengelola item .....**Error! Bookmark not defined.**

Tabel 4. 16 Transaksi *Use case* mengelola item (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 17 Transaksi *Use case* melihat ringkasan laporan admin**Error! Bookmark not defined.**

Tabel 4. 18 Transaksi *Use case* melakukan chat *customer***Error! Bookmark not defined.**

Tabel 4. 19 Transaksi *Use case* mengelola kategori vendor**Error! Bookmark not defined.**

Tabel 4. 20 Transaksi *Use case* mengelola kategori vendor (Lanjutan).....**Error! Bookmark not defined.**

Tabel 4. 21 Transaksi *Use case* memverifikasi vendor**Error! Bookmark not defined.**

Tabel 4. 22 Transaksi *Use case* mengelola status vendor**Error! Bookmark not defined.**

Tabel 4. 23 Transaksi *Use case* mengubah persentase profit vendor .....**Error! Bookmark not defined.**

Tabel 4. 24 Transaksi *Use case* mengelola jenis acara**Error! Bookmark not defined.**

Tabel 4. 25 Transaksi *Use case* mengelola jenis acara (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 26 Transaksi *Use case* mengelola ruangan..**Error! Bookmark not defined.**

Tabel 4. 27 Transaksi *Use case* mengelola paket acara**Error! Bookmark not defined.**

Tabel 4. 28 Transaksi *Use case* mengubah status review**Error! Bookmark not defined.**

Tabel 4. 29 Transaksi *Use case* mengelola promosi..**Error! Bookmark not defined.**

Tabel 4. 30 Transaksi *Use case* mengelola slider .....**Error! Bookmark not defined.**

Tabel 4. 31 Transaksi *Use case* mengelola slider (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 32 Transaksi *Use case* memperbarui data kantor**Error! Bookmark not defined.**

Tabel 4. 33 Perhitungan *Unadjusted Use Case Weight* dengan AUCP .....**Error! Bookmark not defined.**

Tabel 4. 34 Hasil Perhitungan *Technical Complexity Factor* pada AUCP .....**Error! Bookmark not defined.**

Tabel 4. 35 Hasil Perhitungan *Technical Complexity Factor* pada AUCP (Lanjutan) .....**Error! Bookmark not defined.**

Tabel 4. 36 Hasil Perhitungan *Environmental Complexity Factor* dengan AUCP .....**Error! Bookmark not defined.**

Tabel 4. 37 Hasil Perhitungan *End-User Development* TCF dengan AUCP .....**Error! Bookmark not defined.**

Tabel 4. 38 Hasil Perhitungan *End-User Development* ECF dengan AUCP .....**Error! Bookmark not defined.**

Tabel 4. 39 *Hours of Effort* Setiap Aktivitas dengan AUCP**Error! Bookmark not defined.**

Tabel 4. 40 Alokasi SDM dan Waktu fase *Software Development* dengan AUCP .....**Error! Bookmark not defined.**

Tabel 4. 41 Alokasi SDM dan Waktu fase On-Going Activity dengan AUCP ....**Error! Bookmark not defined.**

Tabel 4. 42 Pembagian SDM dan Waktu menggunakan AUCP**Error! Bookmark not defined.**

Tabel 4. 43 Perhitungan Alokasi Biaya setiap Aktivitas dengan AUCP .....**Error! Bookmark not defined.**

Tabel 4. 44 Perhitungan Alokasi SDM dengan AUCP **Error! Bookmark not defined.**

Tabel 4. 45 Klasifikasi Aktor pada Sistem dengan Re-UCP**Error! Bookmark not defined.**

Tabel 4. 46 Perhitungan *Unadjusted Actor Weight* dengan Re-UCP .....**Error! Bookmark not defined.**

Tabel 4. 47 Transaksi *Use Case Login*.....**Error! Bookmark not defined.**

Tabel 4. 48 Transaksi *Use Case* Chat Admin .....**Error! Bookmark not defined.**

Tabel 4. 49 Transaksi *Use Case* Melakukan *Order* Paket Acara**Error! Bookmark not defined.**

Tabel 4. 50 Transaksi *Use Case* Menambah Review (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 51 Transaksi *Use Case* Menambah Review .**Error! Bookmark not defined.**

Tabel 4. 52 Transaksi *Use Case* Melakukan Registrasi Vendor**Error! Bookmark not defined.**

Tabel 4. 53 Transaksi *Use Case* Melihat ringkasan laporan vendor.....**Error! Bookmark not defined.**

Tabel 4. 54 Transaksi *Use Case* Mengelola transaksi**Error! Bookmark not defined.**

Tabel 4. 55 Transaksi *Use Case* melihat notifikasi ....**Error! Bookmark not defined.**

Tabel 4. 56 Transaksi *Use case* mengubah profil vendor**Error! Bookmark not defined.**

Tabel 4. 57 Transaksi *Use case* mengelola item .....**Error! Bookmark not defined.**

Tabel 4. 58 Transaksi *Use case* melihat ringkasan laporan admin**Error! Bookmark not defined.**

Tabel 4. 59 Transaksi *Use case* melakukan *chat customer***Error! Bookmark not defined.**

Tabel 4. 60 Transaksi *Use case* mengelola kategori vendor**Error! Bookmark not defined.**

Tabel 4. 61 Transaksi *Use case* memverifikasi vendor**Error! Bookmark not defined.**

Tabel 4. 62 Transaksi *Use case* memverifikasi vendor (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 63 Transaksi *Use case* mengelola status vendor**Error! Bookmark not defined.**

Tabel 4. 64 Transaksi *Use case* mengubah persentase profit vendor .....**Error! Bookmark not defined.**

Tabel 4. 65 Transaksi *Use case* mengelola jenis acara**Error! Bookmark not defined.**

Tabel 4. 66 Transaksi *Use case* mengelola ruangan..**Error! Bookmark not defined.**

Tabel 4. 67 Transaksi *Use case* mengelola ruangan (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 68 Transaksi *Use case* mengelola paket acara**Error! Bookmark not defined.**

Tabel 4. 69 Transaksi *Use case* mengelola paket acara (Lanjutan)**Error! Bookmark not defined.**

Tabel 4. 70 Transaksi *Use case* mengubah status review**Error! Bookmark not defined.**

Tabel 4. 71 Transaksi *Use case* mengelola promosi..**Error! Bookmark not defined.**

Tabel 4. 72 Transaksi *Use case* mengelola slider .....**Error! Bookmark not defined.**

Tabel 4. 73 Transaksi *Use case* memperbarui data kantor**Error! Bookmark not defined.**

Tabel 4. 74 Transaksi *Use case* memperbarui data kantor (Lanjutan) .....**Error! Bookmark not defined.**

Tabel 4. 75 Perhitungan *Unadjusted Use Case Weight* dengan Re-UCP .....**Error! Bookmark not defined.**

Tabel 4. 76 Hasil Perhitungan *Technical Complexity Factor* dengan Re-UCP ..**Error! Bookmark not defined.**

Tabel 4. 77 Hasil Perhitungan *Environmental Complexity Factor* pada Re-UCP .....**Error! Bookmark not defined.**

Tabel 4. 78 *Hours of Effort* Setiap Aktivitas dengan Re-UCP**Error! Bookmark not defined.**

Tabel 4. 79 *Hours of Effort* Setiap Aktivitas dengan Re-UCP (Lanjutan).....**Error! Bookmark not defined.**

Tabel 4. 80 Alokasi SDM dan Waktu fase Software Development pada Re-UCP .....**Error! Bookmark not defined.**

Tabel 4. 81 Alokasi SDM dan Waktu fase On-Going Activity pada Re-UCP .....**Error! Bookmark not defined.**

Tabel 4. 82 Pembagian SDM dan Waktu menggunakan Re-UCP**Error! Bookmark not defined.**

Tabel 4. 83 Perhitungan Alokasi Biaya setiap Aktivitas dengan Re-UCP .....**Error! Bookmark not defined.**

Tabel 4. 84 Perhitungan Alokasi Biaya setiap Aktivitas dengan Re-UCP (Lanjutan) .....**Error! Bookmark not defined.**

Tabel 4. 85 Perhitungan Alokasi SDM dengan Re-UCP**Error! Bookmark not defined.**

Tabel 4. 86 Tabulasi Distribusi Biaya, Waktu dan SDM pada AUCP.....**Error! Bookmark not defined.**

Tabel 4. 87 Tabulasi Distribusi Biaya, Waktu dan SDM pada AUCP (Lanjutan) .....**Error! Bookmark not defined.**

Tabel 4. 88 Tabulasi Distribusi Biaya, Waktu dan SDM pada Re-UCP .....**Error! Bookmark not defined.**

Tabel 5. 1 Perbandingan Hasil Perhitungan Waktu ..**Error! Bookmark not defined.**

Tabel 5. 2 Perbandingan Hasil Perhitungan Waktu (Lanjutan)**Error! Bookmark not defined.**

Tabel 5. 3 Perbandingan Hasil Perhitungan Biaya ....**Error! Bookmark not defined.**

Tabel 5. 4 Perbandingan Hasil Perhitungan Jumlah Pengembang**Error! Bookmark not defined.**

Tabel 5. 5 Perbandingan Hasil Perhitungan Jumlah Pengembang (Lanjutan).**Error! Bookmark not defined.**

# DAFTAR GAMBAR

Gambar 2. 1 Project Life Cycle dan System Development Life Cycle .....**Error! Bookmark not defined.**

Gambar 3. 1 Diagram Alir Metodologi Penelitian.....**Error! Bookmark not defined.**

Gambar 3. 2 Aturan Penomoran Pernyataan Kebutuhan**Error! Bookmark not defined.**

Gambar 3. 3 Aturan Penomoran Kebutuhan Fungsional**Error! Bookmark not defined.**

Gambar 4. 1 *Work Breakdown Structure* CV Gumcode Indonesia**Error! Bookmark not defined.**

Gambar 4. 2 *Work Breakdown Structure Advance Use Case Point* dan *Revised Use Case Point*.....**Error! Bookmark not defined.**

Gambar 4. 3 Use Case Diagram Sistem Reservasi Online Gedung Serbaguna **Error! Bookmark not defined.**

# DAFTAR LAMPIRAN

LAMPIRAN A LAPORAN HASIL WAWANCARA.....**Error! Bookmark not defined.**

A.1 Latar Belakang .....**Error! Bookmark not defined.**

A.2 Maksud dan Tujuan .....**Error! Bookmark not defined.**

A.3 Topik Wawancara .....**Error! Bookmark not defined.**

A.4 Waktu dan Tempat Kegiatan.....**Error! Bookmark not defined.**

A.5 Narasumber.....**Error! Bookmark not defined.**

A.6 Hasil Wawancara .....**Error! Bookmark not defined.**

LAMPIRAN B HASIL WAWANCARA PROSES PMBOK.**Error! Bookmark not defined.**

LAMPIRAN C BUKTI VALIDASI.....**Error! Bookmark not defined.**

C.1 Bukti Validasi Wawancara .....**Error! Bookmark not defined.**

*C.2* Bukti Validasi *Use Case Diagram* dan *Use Case Scenario*.....**Error! Bookmark not defined.**

LAMPIRAN D LEMBAR OBSERVASI SISTEM .....**Error! Bookmark not defined.**

LAMPIRAN E KEBUTUHAN FUNGSIONAL SISTEM .....**Error! Bookmark not defined.**

E.1 Pernyataan Kebutuhan Sistem .....**Error! Bookmark not defined.**

E.2 Kebutuhan Fungsional Sistem .....**Error! Bookmark not defined.**

LAMPIRAN F IDENTIFIKASI PENGGUNA .....**Error! Bookmark not defined.**

LAMPIRAN G USE CASE SCENARIO.....**Error! Bookmark not defined.**

LAMPIRAN H LEMBAR PENILAIAN ADVANCE USE CASE POINT**Error! Bookmark not defined.**

LAMPIRAN I LEMBAR PENILAIAN REVISED USE CASE POINT**Error! Bookmark not defined.**

LAMPIRAN J GANTT CHART PADA ADVANCE USE CASE POINT**Error! Bookmark not defined.**

LAMPIRAN K GANTT CHART PADA REVISED USE CASE POINT**Error! Bookmark not defined.**

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

# BAB 1 PENDAHULUAN

## 1.1 Latar belakang

Semakin berkembangnya teknologi informasi, menjadikan sistem informasi turut berkembang cepat pada seluruh unit kerja dalam berbagai jenis organisasi. Banyak proyek sistem informasi dihasilkan di hampir semua bisnis dan organisasi pemerintah. Proyek sistem informasi memiliki kemiripan dengan proyek pada umumnya, meskipun terdapat perbedaan pada beberapa aspek. Sebuah proyek dapat diartikan sebagai usaha sementara yang bertujuan untuk menciptakan produk, layanan atau hasil yang unik (Marchewka, 2003). Setiap proyek memiliki tiga batasan (*triple constraints*) yang menentukan keberhasilan dari proyek tersebut, yaitu harus terpenuhinya seluruh ruang lingkup pekerjaan dengan kualitas yang baik dalam batasan waktu dan batasan biaya yang disepakati (Schwalbe, 2007).

Berdasarkan data dari CHAOS Summary pada tahun 2009, menunjukkan hanya 32% proyek TI yang dikirimkan tepat waktu dan sesuai anggaran. Sebesar 44% termasuk dalam "*challenge*," yang berarti proyek tersebut terlambat, melebihi anggaran, atau tidak memenuhi persyaratan. Selain itu, 24% proyek gagal, dibatalkan, atau tidak pernah digunakan karena adanya permasalahan pada manajemen proyek (Gray & Larson, 2011). Oleh sebab itu, manajemen proyek yang baik diperlukan untuk memenuhi tujuan spesifik, waktu, biaya, dan kualitas proyek, serta harus memfasilitasi keseluruhan proses untuk memenuhi kebutuhan dan harapan orang-orang yang terlibat dalam kegiatan proyek atau terpengaruh oleh proyek tersebut (Marchewka, 2003). Terdapat 3 tahapan utama dalam manajemen proyek, yaitu: tahap perencanaan, tahap penjadwalan dan tahap pengendalian (Prasetya & Lukiastuti, 2009).

Perencanaan menghasilkan waktu penyelesaian proyek dan perhitungan terhadap biaya pelaksanaan proyek yang nantinya menentukan nilai jual proyek, sehingga berpengaruh pada profit yang diterima yang disebut estimasi (Tantra, 2012). Sebagian besar model estimasi biaya menghasilkan perkiraan usaha, yang kemudian dapat dikonversi menjadi durasi dan biaya proyek (Leung & Fan, 2001). Kemudian, penjadwalan berdasarkan *Work Breakdown Structure* (WBS) untuk mengembangkan rencana proyek dan menghubungkan ruang lingkup proyek dengan jadwal dan biaya. Sesaat setelah proyek selesai, manajer proyek dan tim pengembang melakukan pengendalian berupa evaluasi proyek salah satunya dengan *postmortem review*. Salah satu fokus dari *review* ini adalah meninjau ulang lingkup proyek, biaya, jadwal, dan sasaran kualitas (Marchewka, 2003). Manajer proyek dan tim pengembang harus mengidentifikasi apa yang mereka lakukan dengan benar dan apa yang bisa mereka lakukan dengan lebih baik untuk pelajaran di masa mendatang.

CV Gumcode Indonesia merupakan salah satu *startup* di Kota Malang yang bergerak di bidang teknologi informasi dan bisnis. Sejak *startup* ini berdiri pada tahun 2015, CV Gumcode Indonesia telah bekerjasama dengan beberapa mitra,

agensi dan perusahaan milik negara dan mengerjakan berbagai jenis proyek perangkat lunak seperti, *mobile apps android, mobile apps iOS*, halaman *company profile*, aplikasi *web*, toko *online*, mendesain *UI/UX mobile* dan *website*.

Hasil wawancara dengan CEO CV Gumcode Indonesia, diketahui CV Gumcode Indonesia memiliki beberapa kendala dalam proyek pengembangan perangkat lunak yang berkaitan dengan alokasi sumber daya, sehingga sebanyak 40% dari keseluruhan proyek yang dikerjakan mengalami keterlambatan dari jadwal. Dampak dari keterlambatan proyek tersebut mengakibatkan CV Gumcode Indonesia mengalami kerugian, dikarenakan harus menanggung biaya yang berlebih karena melanggar kesepakatan waktu yang disetujui oleh kedua belah pihak pada saat tanda tangan kontrak. Selain itu, terdapat proyek yang dibatalkan karena proses pengerjaan yang terlalu lama sehingga perangkat lunak tersebut tidak memiliki kesempatan untuk digunakan. Permasalahan ini terjadi disebabkan oleh permintaan klien yang berubah-ubah, pemberian data dari klien yang lama sehingga menghambat pengerjaan, serta kurangnya pengetahuan dalam menghitung ukuran perangkat lunak. Ukuran perangkat lunak merupakan aktivitas yang kritis dalam proses pengembangan perangkat lunak, dimana pendekatan sistematis terhadap estimasi ukuran perangkat lunak penting untuk perencanaan dan pengelolaan proyek agar proyek perangkat lunak sukses memenuhi konsep *triple constrains* (Khrisnegowda, 2013).

CV Gumcode Indonesia memiliki beberapa pertimbangan dalam menentukan alokasi biaya perangkat lunak, diantaranya jenis *platform* yang digunakan, fitur yang dibutuhkan, serta seberapa sulit logika yang dibangun. Untuk mengukur tingkat kesulitan fitur dan logika, *software house* ini memperkirakan berdasarkan pengalaman dan teknologi yang digunakan. Manajer proyek menggali pengalaman untuk proyek serupa, mencoba mengingat rentang waktu proyek-proyek ini, dan ukuran tim, kemudian memperkiraan berdasarkan campuran dugaan dan perhitungan. Cara ini subjektif dan memiliki ketidakpastian lebih tinggi karena tidak memiliki standar perhitungan yang pasti dalam mengukur ukuran perangkat lunak. Penjadwalan yang dilakukan juga masih menggunakan *timeline* sederhana sehingga menghasilkan alokasi waktu dan sumber daya manusia yang kurang detail.

Dari permasalahan di atas, dilakukan penelitian dengan menggunakan proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang mengalami keterlambatan dari yang seharusnya 3 minggu menjadi 6 minggu. Metode yang digunakan untuk estimasi biaya, waktu dan sumber daya manusia adalah *Advance Use Case Point (AUCP)* dan metode *Revised Use Case Point (Re-UCP)*. Metode *AUCP* merupakan perluasan dari metode *Use Case Point* yang melibatkan fitur *end user development* dalam meningkatkan *effort* pengembangan sehingga kepuasan pengguna meningkat (Srivastata, Singh, & Abbas, 2017). Terdapat tambahan pada faktor teknis dan lingkungan yang diberikan kepada pengguna untuk pertimbangan dalam menentukan biaya tambahan saat pengembangan perangkat lunak, sedangkan metode Re-UCP merupakan perluasan dari metode *Use Case Point* dan *Extended Use Case Point* yang dapat mengestimasi *effort* perangkat

lunak disesuaikan untuk semua jenis proyek dengan tingkat kerumitan yang bervariasi dan memiliki cakupan skalabilitas yang dapat menangani aktivitas pengembangan perangkat lunak *agile* (Kirmani & Wahid, 2015). Dalam Re-UCP, aktor, *use case*, faktor lingkungan dan faktor teknis lainnya dikategorikan lebih lanjut. Perhitungan matematika menggunakan kedua metode akan menghasilkan estimasi *effort* yang kemudian dikalikan dengan faktor produktivitas 20 *person-hours* sebagaimana direkomendasikan oleh Karner (Kirmani & Wahid, 2015).

Penelitian ini akan memberikan informasi mengenai proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang telah selesai berjalan kepada CV Gumcode Indonesia, dengan menghitung estimasi biaya, waktu dan sumber daya manusia menggunakan metode *Advance Use Case Point* dan *Revised Use Case Point*, dimana hasil perhitungan metode *Advance Use Case Point* dan *Revised Use Case Point* pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula tersebut, akan dibandingkan dengan hasil perhitungan biaya, waktu dan sumber daya yang dibutuhkan selama pengembangan perangkat lunak dari CV Gumcode Indonesia. Penjadwalan pada proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula akan ditampilkan dalam bentuk *Gantt Chart* berdasarkan WBS.

Metode yang digunakan dalam penelitian ini digunakan untuk mengevaluasi alokasi biaya, waktu, dan sumber daya pada proyek pengembangan perangkat lunak yang telah selesai dengan melakukan perbandingan dari hasil perhitungan *Advance Use Case Point* (AUCP) dan *Revised Use Case Point (Re-UCP)*. Oleh karena itu, penelitian ini berjudul "Evaluasi Biaya Pengembangan Sistem Reservasi Online PT GHL Peninsula Menggunakan Metode *Advance Use Case Point* dan *Revised Use Case Point*".

## 1.2 Rumusan masalah

Berdasarkan latar belakang dan identifikasi masalah yang diuraikan di atas, maka rumusan masalah pada penelitian ini sebagai berikut:

1. Bagaimana hasil implementasi metode *Advance Use Case Point* (AUCP) dalam mengestimasi biaya dan waktu yang dibutuhkan pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula?
2. Bagaimana hasil implementasi metode *Revised Use Case Point* (Re-UCP) dalam mengestimasi biaya dan waktu yang dibutuhkan pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula?
3. Bagaimana pembagian lingkup kerja pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan pendekatan *Work Breakdown Structure* (WBS)?
4. Bagaimana penjadwalan pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan *Gantt chart* berdasarkan *Work Breakdown Structure (WBS)* pada *Advance Use Case Point* (AUCP) dan *Revised Use Case Point (Re-UCP)*?
5. Bagaimana hasil perbandingan estimasi biaya, durasi dan sumber daya manusia pada pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP) dan perhitungan CV Gumcode Indonesia?

## 1.3 Tujuan

Berdasarkan rumusan masalah di atas, maka penelitian ini memiliki tujuan sebagai berikut:

1. Mendeskripsikan hasil implementasi metode *Advance Use Case* dalam mengestimasi biaya, waktu dan sumber daya manusia pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.
2. Mendeskripsikan hasil implementasi metode *Revised Use Case Point* dalam mengestimasi biaya, waktu dan sumber daya manusia pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.
3. Mendeskripsikan pembagian lingkup kerja pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan pendekatan *Work Breakdown Structure* (WBS).
4. Mendeskripsikan penjadwalan pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan *Gantt chart* berdasarkan *Work Breakdown Structure* (WBS) pada *Advance Use Case Point* (AUCP) dan *Revised Use Case Point (Re-UCP)*
5. Mendeskripsikan hasil perbandingan evaluasi biaya, durasi dan sumber daya manusia pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan metode *Advance Use Case Point* dan *Revised Use Case Point* dan perhitungan dari perusahaan berdasarkan pengalaman proyek sebelumnya yang saat ini sedang digunakan oleh CV Gumcode Indonesia.

## 1.4 Manfaat

Manfaat yang diperoleh dari penelitian ini sebagai berikut:

1. Memberikan wawasan mengenai pembagian lingkup kerja proyek menggunakan pendekatan *Work Breakdown Structure* (WBS).
2. Memberikan wawasan mengenai pengambilan keputusan yang optimal untuk mengalokasikan biaya, waktu dan sumberdaya dalam manajemen proyek pengembangan perangkat lunak menggunakan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP).
3. Memberikan wawasan mengenai penjadwalan menggunakan Gantt Chart berdasarkan *Work Breakdown Structure* (WBS).
4. Memberikan masukan berupa pertimbangan kepada pihak CV Gumcode Indonesia dalam mengestimasi biaya, waktu dan sumberdaya di masa mendatang dan menjadikan dasar untuk perbaikan.
5. Dapat dijadikan referensi untuk penelitian terkait penentuan biaya pengembangan perangkat lunak di masa mendatang.

## 1.5 Batasan masalah

Untuk mendapatkan hasil yang optimal dan sesuai permasalahan yang ada, maka pada penelitian ini batasan masalah dan asumsi-asumsi yang digunakan sebagai berikut:

1. Data yang dianalisis berasal dari proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.

2. Penjadwalan dilakukan berdasarkan *work breakdown structure* (WBS) dari proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula dan digambarkan dalam bentuk *Gantt Chart*.
3. Metode yang digunakan adalah metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP).
4. Pembuatan *use case story* dan *use case diagram* berdasarkan hasil wawancara, observasi ke sistem dan validasi dengan pihak CV Gumcode Indonesia.
5. Analisis perhitungan waktu, biaya dan sumber daya manusia yang dibutuhkan tidak mempertimbangkan resiko yang mungkin terjadi selama proyek berlangsung.
6. Pada proses *Project Schedule Management*, peneliti mengungkap keberadaan proses *Plan Schedule Management, Define Activities, Sequence Activities, Estimate Activity Duration, Develop Schedule,* dan *Control Schedule* pada CV Gumcode Indonesia melalui wawancara.
7. Pada proses *Project Cost Management*, peneliti mengungkap keberadaan proses *Plan Cost Management, Estimate Cost, Determine Budget,* dan *Control Cost* pada CV Gumcode Indonesia melalui wawancara.
8. Pada proses *Project Resource Management*, peneliti mengungkap keberadaan proses *Plan Resource Management, Estimate Activity Resource, Acquiry Resource, Develop Team, Manage Team,* dan *Control Resource* pada CV Gumcode Indonesia melalui wawancara.

## 1.6 Sistematika pembahasan

Sistematika pembahasan berisi struktur skripsi yang dimulai dari bab pendahuluan sampai bab penutup dan deskripsi singkat dari masing-masing bab. Diharapkan bagian ini dapat membantu pembaca dalam memahami sistematika pembahasan isi dalam skripsi ini.

**BAB I PENDAHULUAN**

Pada bab ini, menjelaskan tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika pembahasan yang digunakan dalam penelitian.

**BAB 2 LANDASAN KEPUSTAKAAN**

Pada bab ini, menjelaskan tentang landasan kepustakaan, metode yang digunakan sebagai referensi ilmiah yang berhubungan dengan estimasi waktu, biaya dan sumber daya manusia menggunakan metode *Advance Use Case Point* dan *Revised Use Case Point,* serta proses-proses pada area pengetahuan PMBOK yang digunakan dalam penelitian.

**BAB 3 METODOLOGI PENELITIAN**

Pada bab ini, menjelaskan tentang langkah-langkah dalam menyelesaikan masalah penelitian yang berupa tahapan untuk menghasilkan estimasi biaya, waktu dan sumber daya menggunakan metode *Advance Use Case Point* dan *Revised Use Case Point*.

**BAB 4 HASIL DAN ANALISIS**

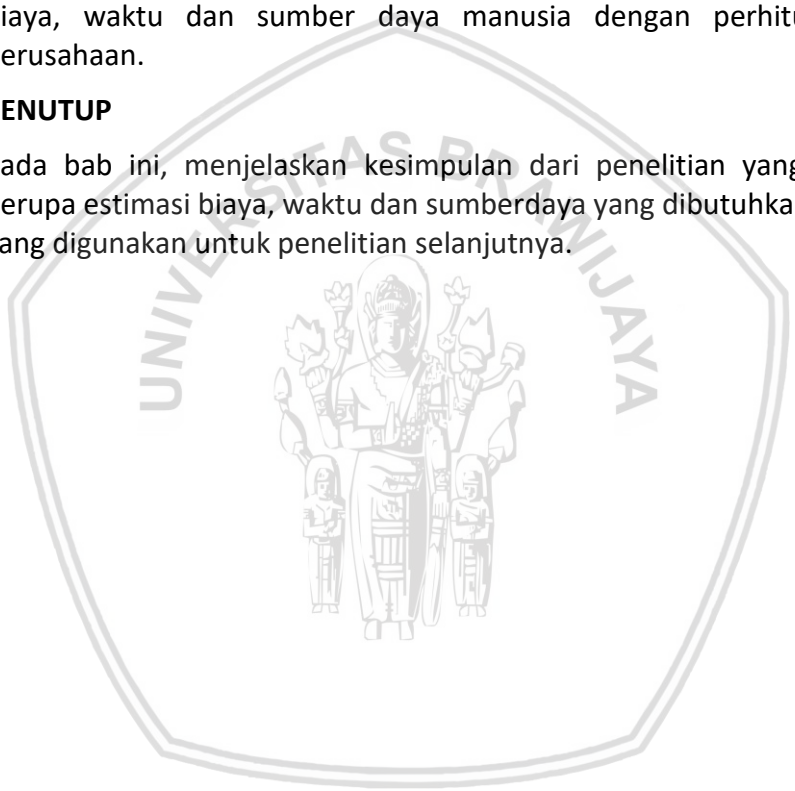
Pada bab ini, menjelaskan tentang hasil pengumpulan data dari penelitian yang dilakukan berupa, wawancara, lembar penilaian yang diisi oleh tim pengembang dan data-data yang dibutuhkan untuk menghasilkan estimasi biaya, waktu dan sumber daya manusia, kemudian melakukan analisis berdasarkan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP).

**BAB 5 PEMBAHASAN**

Pada bab ini, menjelaskan tentang hasil analisis perhitungan menggunakan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP) dan melakukan pembahasan terhadap hasil perbandingan estimasi biaya, waktu dan sumber daya manusia dengan perhitungan dari perusahaan.

**BAB 6 PENUTUP**

Pada bab ini, menjelaskan kesimpulan dari penelitian yang dilakukan berupa estimasi biaya, waktu dan sumberdaya yang dibutuhkan dan saran yang digunakan untuk penelitian selanjutnya.



# BAB 2 LANDASAN KEPUSTAKAAN

## 2.1 Kajian Pustaka

Kajian pustaka yang digunakan dalam penelitian ini adalah beberapa penelitian terdahulu yang dijadikan referensi dan acuan dalam penelitian. Referensi penelitian ini diperoleh dari 3 jurnal internasional sebagai berikut:

**Tabel 2. 1 Kajian Pustaka**

| Metodologi | Hasil |
|---|---|
| Penelitian oleh Mudazir Manzoor Kirmani dan Abdul Wahid di tahun 2015 yang berjudul "*Revised Use Case Point(Re-UCP) Model for Software Effort Estimation*" bertujuan untuk meningkatkan efisiensi estimasi *effort* dalam proyek pengembangan perangkat lunak pada tahap awal pengembangan dengan cara mengusulkan suatu model yang menyempurnakan metode sebelumnya, yaitu *Use Case Point* (UCP) dan *Extended Use Case Point* (e-UCP). Model Re-UCP menggunakan klasifikasi kompleksitas aktor dan *use case* yang berbeda, serta menambahkan parameter pada *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF). Penelitian ini dilakukan pada 14 proyek untuk mengetahui estimasi *effort* dari masing-masing proyek, kemudian hasilnya dibandingkan dengan model UCP dan e-UCP. | Perbandingan pada 14 proyek menunjukkan bahwa Re-UCP telah secara signifikan mengungguli teknik estimasi *effort* model yang ada yaitu, UCP dan e-UCP. Model (Re-UCP) telah menunjukkan perbaikan dalam mengestimasi *effort* untuk proyek pengembangan perangkat lunak dengan deviasi minimum dari *actual effort* bila dibandingkan dengan metode estimasi effort UCP dan e-UCP. |
| Penelitian oleh Archana Srivastava, S. K. Singh, dan Syed Qamar Abbas di tahun 2017 yang berjudul "*Evaluation of Software Project Estimation Methodology: AUCP*" bertujuan untuk meningkatkan kualitas dan kepuasan pengguna akhir dengan menyertakan fitur *End User Development* dalam perangkat lunak untuk keseluruhan estimasi *effort* proyek. Model *Advance Use Case Point* (AUCP) menggunakan lembar penilaian tambahan berupa *End User Development Technical Complexity Factor* (EUD_TCF) dan *End User Development Environmental Complexity Factor* (EUD_ECF) yang ditujukan kepada pengguna akhir yang paling tahu kebutuhan apa yang harus dibangun. Pada tahap awal model Re-UCP dilakukan perhitungan UCP yang nantinya dikalikan dengan EUD_TCF dan EUD_ECF. Penelitian ini dilakukan pada 5 proyek untuk mengetahui besar peningkatan *effort* yang dibutuhkan dalam pengembangan yang menggabungkan fitur EUD. | Perbandingan pada 5 proyek menunjukkan kenaikan rata-rata estimasi effort sekitar 5% -12%. Dalam proyek yang ekstrim mungkin sekitar 20% -33%. Komputasi yang melibatkan pengguna akhir menambah kepuasan kepada pengguna akhir yang mana merupakan tujuan akhir dari sistem Informasi. |
| Penelitian oleh Sangeetha K. dan Prof. Pankaj Dalal berjudul "*Software Sizing with Use Case Point*" bertujuan untuk mengusulkan *Use Case Point Analysis (*UCP*)* sebagai metode untuk memahami dan mengukur ukuran dan produktivitas perangkat lunak, dimana ukuran perangkat lunak (*software size*) penting dalam | Perbandingan pada 10 proyek perangkat lunak menunjukkan bahwa karakteristik *technical* dan *environmental* berpengaruh pada estimasi ukuran yang dihasilkan. Hasil analisis korelasi menunjukkan |

**Tabel 2. 2 Kajian Pustaka (Lanjutan)**

| | |
|---|---|
| menentukan dan mengendalikan biaya, jadwal, kualitas dan produktivitas dalam manajemen proyek perangkat lunak. Dengan menggunakan estimasi UCP, dapat dihasilkan estimasi yang dapat diandalkan pada awal siklus pengembangan. UCP mengklasifikasikan aktor dan *use case*, serta mempertimbangkan *technical complexity factor* dan *environmental complexity factor* dalam menentukan *effort* yang dibutuhkan. | pengaruh karakteristik sistem terhadap estimasi ukuran proyek, dimana karakteristik sistem menunjukkanhubungan yang sangat kuat pada perkiraan ukuran proyek. |

## 2.2 Profil CV Gumcode Indonesia

CV Gumcode Indonesia adalah perusahaan yang bergerak di bidang *IT & Business* dengan beberapa kerjasama dengan partner, instansi serta perusahaan BUMN. CV Gumcode Indonesia mempunyai beberapa produk *Website* dan *Mobile Apps Android*& iOS yang dapat dikembangkan. Selain itu, juga memberikan pelayanan perbaikan dan jasa di berbagai segi proyek.

CV Gumcode Indonesia sebenarnya sudah berdiri sejak tahun 2013 dimulai dengan nama "Artrock". Seiring dengan berjalannya waktu Artrock terus mengalami *rebranding* nama serta perbaikan struktur menjadi "Gum", Gumproject", hingga akhirnya yang sekarang adalah "Gumcode". Pengembang Gumcode sendiri terbagi menjadi beberapa tim dengan spesifikasi dan kemampuan yang berbeda dalam menangani proyek. Saat ini sudah lebih dari 40 proyek terselesaikan, baik *website* maupun *mobile apps*.

Sejak awal Tahun 2015, CV Gumcode Indonesia memulai ekspansi dari proyek-proyek yang lingkupnya kecil hingga projek skala besar sekelas *enterprise*. CV Gumcode Indonesia menargetkan ekspansi ke berbagai daerah diluar kota Malang yaitu Surabaya, Bogor, Jakarta, Batam, Medan dan Balikpapan.

Terdapat 3 proses bisnis yang menjadi fokusCV Gumcode Indonesia, yaitu:

1. Pengembangan produk
2. Kerjasama dengan perusahaan dalam bidang IT
3. Pengerjaan projek

Fokus utama CV Gumcode Indonesia adalah pengembangan produk, dimana ide dan kreativitas dari anggota tim akan dikelola secara maksimal sehingga tidak hanya menciptakan suatu aplikasi, tetapi juga dapat menghasilkan keuntungan bagi CV Gumcode Indonesia. Fokus berikutnya adalah menjadi *partner* IT di beberapa perusahaan dan instansi, dimana projek yang dikerjakan akan terus kami kelola dan kembangkan sesuai kebutuhan. Fokus yang terakhir adalah pengerjaan projek. Baik pengerjaan proyek *website, e-commerce, mobile apps android/ios* atau proyek lainnya.

### 2.2.2 Visi CV Gumcode Indonesia

Memberikan solusi dalam bidang teknologi bagi para pelaku bisnis dan melakukan pengembangan teknologi dalam segala bidang.

### 2.2.3 Misi CV Gumcode Indonesia

1. Berbudi pekerti dalam menjalankan tugas
2. Memegang teguh etos kerja

## 2.3 Manajemen Proyek

Sebuah proyek dapat diartikan sebagai usaha yang kompleks, tidak rutin, yang memiliki batasan berupa anggaran, waktu, sumber daya, dan spesifikasi kinerja yang dirancang untuk memenuhi kebutuhan pelanggan (Gray & Larson, 2011). Proyek melakukan usaha sementara untuk mencapai produk, jasa, atau hasil yang unik (Schwalbe, 2012).

Manajemen proyek berarti menerapkan pengetahuan, keahlian, teknik, dan perangkat dengan tujuan mencapai sasaran tertentu yang dibatasi oleh anggaran dan waktu yang ditentukan. Manajemen proyek dilakukan dalam 3 fase, yaitu (Prasetya & Lukiastuti, 2009):

1) Perencanaan. Pada fase ini, manajer proyek medefinisikan perkerjaan dengan sasaran, target waktu, keahlian khusus untuk tugas-tugas kompleks yang saling berhubungan dan timnya.
2) Penjadwalan. Pada fase ini, manajer proyek mengurutkan dan membagi waktu untuk keseluruhan kegiatan proyek, memutuskan berapa lama waktu yang dibutuhkan untuk setiap kegiatan, serta menghitung berapa banyak orang yang diperlukan untuk setiap tahap.
3) Pengendalian. Pada fase ini, manajer proyek mengawasi anggaran, biaya, kualitas dan sumber daya, dan menggunakan umpan balik untuk melakukan revisi rencana proyek dan mengatur sumber daya.

Manajemen proyek yang buruk menghasilkan kegagalan proyek dimana perangkat lunak mungkin terlambat, menghabiskan biaya lebih dari rencana semula, atau gagal dalam memebuhi harapan pelanggan. Manajemen proyek memiliki kriteria keberhasilan yang berbeda dari satu proyek ke proyek yang lain, tapi bagi sebagian besar proyek, memilikitujuan penting, yaitu: 1) Menyerahkan perangkat lunak tepat waktu sesuai kesepakatan; 2) Mempertahankan keseluruhan biaya tetap pada anggaran; 3) Perangkat lunak memenuhi harapan pelanggan; 4) Menjaga tim pengembang tetap bahagia dan bekerja dengan baik (Sommerville, 2009).

*Project Management Institute* (PMI) membentuk suatu standar PMBOK (*Project Management Body of Knowledge*) yang bertujuan agar dokumen tersebut menjadi panduan utama dalam pelaksanaan proyek. PMBOK® *Guide* mengidentifikasi bagian dari PMBOK yang secara umum diakui sebagai *good practice.* Secara umum diakui karena pengetahuan dan praktik yang ada di dalamnya dapat digunakan oleh sebagian besar proyek sepanjang waktu serta terbukti kegunaan dan nilainya. *Good practice* diartikan sebagai penerapan pengetahuan, keterampilan, alat, dan teknik yang dapat meningkatkan peluang

keberhasilan pada banyak proyek, dimana praktik pada setiap proyek dapat berbeda sesuai proyek yang dikerjakan (PMI, Inc., 2013).

PMBOK® *Guide* hanya menjelaskan proses manajemen proyek. Manajer proyek dan tim pengembang harus secara hati-hati menangani setiap proses. masukan serta keluarannya dan menentukan mana yang sesuai dengan proyek yang sedang mereka kerjakan. PMBOK® *Guide* dapat digunakan sebagai sumber dalam mengelola sebuah proyek sambil mempertimbangkan keseluruhan pendekatan dan metodologi yang harus diikuti untuk proyek tersebut.

PMBOK® Guide memiliki 47 proses dikelompokkan ke dalam 10 area pengetahuan (*Knowledge Area*). Kesepuluh area pengetahuan tersebut antara lain: manajemen ruang lingkup proyek (*scope)*, manajemen biaya proyek (*cost*), manajemen waktu proyek (*time*), manajemen sumber daya manusia proyek (*human resource*), manajemen komunikasi proyek (*communication*), manajemen kualitas proyek (*quality*), manajemen resiko proyek (*risk*), manajemen pemangku kepentingan proyek (*stakeholder*), manajemen pengadaan proyek (*procurement*), dan manajemen integrasi proyek (*integration*) (PMI, Inc., 2013).

## *2.4 Project Cost Management*

*Project Cost Management* terdiri dari proses yang terlibat dalam perencanaan, estimasi, penganggaran, pendanaan, *funding*, pengelolaan, dan pengendalian biaya sehingga proyek dapat diselesaikan sesuai anggaran yang telah disetujui (PMI, Inc., 2013). *Project Cost Management* memiliki beberapa proses diantaranya:

1. *Plan Cost Management*. Proses ini menentukan bagaimana biaya proyek akan diestimasi, dianggarkan, dikelola, dipantau, dan dikendalikan. Proses ini memberikan panduan dan arahan mengenai bagaimana biaya proyek dapat dikelola selama proyek berlangsung.
2. *Estimate Cost*. Proses ini menentukan perkiraan biaya sumber daya yang dibutuhkan untuk menyelesaikan pekerjaan proyek. Estimasi biaya dinyatakan dalam satuan mata uang tertentu (misalnya, dolar, euro, yen, dll.), namun dalam beberapa kasus menggunakan unit pengukuran lainnya, seperti *staff-hours* atau *staff-days.*
3. *Determine Budget*. Proses ini menggabungkan estimasi biaya aktivitas individu atau *work package* untuk menentukan *baseline* biaya dan anggaran proyek mencangkup semua dana untuk melaksanakan proyek. Dasar biaya tersebut dapat dipantau dan dikendalikan selama proyek.
4. *Control Cost*. Proses ini memantau status proyek untuk memperbarui biaya proyek dan mengelola perubahan pada *baseline* biaya, serta mempertahankan *baseline* biaya sepanjang proyek.

## *2.5 Project Schedule Management*

*Project Schedule Management* terdiri dari proses yang diperlukan untuk mengelola penyelesaian proyek secara tepat waktu. Metode penjadwalan, alat

penjadwalan, model penjadwalan dan informasi proyek berupa *Work Breakdown Structure* (WBS), aktivitas, sumber daya, durasi, ketergantungan antar aktivitas (*dependency*), batasan, kalender, *milestone* dan sebagainya digunakan untuk menghasilkan jadwal proyek. Jadwal proyek kemudian dipresentasikan dalam bentuk *activity list, bar chart* atau *network diagram* (PMI, Inc., 2013).

Terdapat beberapa proses pada *Project Schedule Management*, sebagai berikut:

1. *Plan Schedule Management*. Proses ini menetapkan kebijakan, prosedur, dan dokumentasi untuk perencanaan, pengembangan, pengelolaan, pelaksanaan, dan pengendalian jadwal proyek berupa panduan dan arahan mengenai bagaimana jadwal proyek akan dikelola selama proyek berlangsung.
2. *Define Activities*. Proses ini mengidentifikasi dan mendokumentasikan tindakan spesifik yang akan dilakukan untuk menghasilkan *deliverable* proyek.
3. *Sequence Activities.* Proses ini mengidentifikasi dan mendokumentasikan hubungan antar aktivitas proyek. Selain itu, proses ini mendefinisikan urutan logis pekerjaan untuk mendapatkan efisiensi terbesar terhadap semua kendala proyek.
4. *Estimate Activity Duration*. Proses ini memperkirakan jumlah *effort* kerja yang dibutuhkan untuk menyelesaikan aktivitas menggunakan estimasi sumber daya, serta menghasilkan waktu untuk setiap aktivitas yang harus diselesaikan, yang merupakan masukan utama dalam proses *Develop Schedule*.
5. *Develop Schedule*. Proses ini menganalisis urutan kegiatan, jangka waktu, kebutuhan sumber daya, dan kendala jadwal untuk menciptakan model penjadwalan proyek, serta menghasilkan model penjadwalan dengan tanggal yang direncanakan untuk menyelesaikan aktivitas proyek.
6. *Control Schedule*. Proses ini memantau status aktivitas proyek untuk memperbarui kemajuan proyek dan mengelola perubahan pada jadwal untuk mencapai rencana.

## *2.6 Project Resource Management*

*Project Resource Management* terdiri dari proses mengidentifikasi, mengakuisisi, dan mengelola sumber daya yang dibutuhkan agar berhasil menyelesaikan proyek. Tim proyek terdiri atas orang-orang dengan peran dan tanggung jawab yang ditugaskan untuk menyelesaikan proyek. Anggota tim proyek memiliki beragam keahlian dan dapat ditugaskan secara penuh atau paruh waktu, serta ketika proyek berlangsung dapat dilakukan penambahan atau pengurangan (PMI, Inc., 2013).

Terdapat beberapa proses pada *Project Resource Management,* diantaranya:

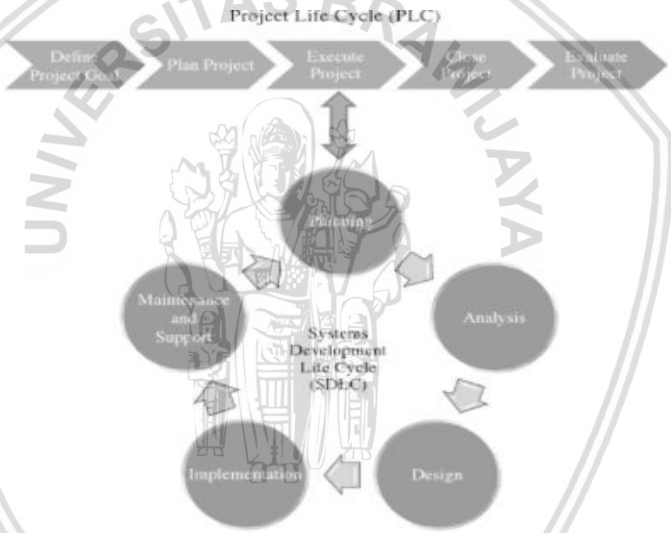
1. *Plan Resource Management*. Proses ini menentukan bagaimana memperkirakan, mengakuisisi, mengelola, dan menggunakan tim dan sumber daya fisik, serta menetapkan pendekatan dan tingkat upaya pengelolaan yang diperlukan untuk mengelola sumber daya proyek berdasarkan jenis dan kompleksitas proyek.
2. *Estimate Activity Resource*. Proses ini mengestimasi sumber daya tim, jenis dan jumlah material, peralatan, dan persediaan yang diperlukan untuk melakukan pekerjaan proyek.
3. *Acquire Resource*. Proses ini menentukan anggota tim, fasilitas, peralatan, bahan, persediaan dan sumber daya lain yang diperlukan untuk menyelesaikan pekerjaan proyek.
4. *Develop Team.* Proses ini bertujuan untuk menghasilkan kerja tim yang lebih baik, meningkatkan interaksi anggota tim, keterampilan dan kompetensi dari masing-masing orang, memotivasi karyawan, mengurangi gesekan pengembang, dan meningkatkan kinerja keseluruhan proyek.
5. *Manage Team.* Proses ini menilai kinerja anggota tim, memberikan umpan balik, menyelesaikan masalah, dan mengelola perubahan tim untuk mengoptimalkan kinerja proyek.
6. *Control Resource*. Proses ini memastikan bahwa sumber daya fisik yang ditugaskan dan dialokasikan untuk proyek, tersedia sesuai rencana, serta memantau pemanfaatan sumber daya yang direncanakan dibandingkan dengan aktual dan mengambil tindakan korektif seperlunya.

## *2.7 Project Life Cycle*

*Project Life Cycle* (PLC) merupakan kumpulan tahap atau fase-fase yang logis yang dihasilkan dari proyek sistem informasi dimana setiap fase harus menghasilkan satu atau lebih *deliverable*. Proyek dipecah menjadi beberapa fase dengan tujuan agar lebih mudah untuk dikelola dan megurangi resiko yang muncul. Fase yang terdapat dalam *Project Life Cycle* (PLC) secara berurutan sebagai berikut: *Define Project Goal, Plan Project, Execute Project Plan, Close Project* dan *Evaluate Project* (Marchewnka, 2003).

1. Pada fase *Define Project Goal*, seluruh tujuan proyek didefinisikan dan berfokus pada upaya memberikan nilai bisnis bagi organisasi. Tujuan yang terdefinisi dengan baik memberikan tim pengembang fokus yang jelas dan mendorong fase selanjutnya dari proyek.
2. Pada fase *Plan Project*, mendefinisikan perencanaan proyek mengenai apa yang akan dilakukan, mengapa melakukan hal tersebut, bagaimana cara melakukannya, siapa yang terlibat, berapa lama waktu yang dibutuhkan, berapa biaya yang dibutuhkan, apa yang dapat menyebabkan kesalahan dan bagaimana cara mengatasinya, bagaimana mengestimasi biaya dan jadwal, mengapa suatu keputusan diambil, dan bagaimana mengetahui keberhasilan.

3. Pada fase *Execute Project Plan*, tim pengembang mengeksekusi perencanaan dimana progress proyek, lingkup kerja, jadwal, biaya dan sumber daya dikelola untuk memastikan proyek mencapai tujuan. Hasilnya adalah produk komplet yang telah selesai diimplementasikan dan dikirimkan ke organisasi. Di dalam fase ini, terdapat *System Development Life Cycle* (SDLC) atau siklus pengembangan sistem.
4. Pada fase *Close Project*, memastikan bahwa semua pekerjaan selesai sesuai rencana dan disetujui oleh tim proyek dan sponsor. Laporan akhir proyek dan presentasi ke klien menutup fase ini disertai dengan dokumentasi semua *deliverable* yang dijanjikan telah selesai sesuai spesifikasi.
5. Pada fase *Evaluate Project*, sistem yang telah diterapkan dievaluasi untuk mengetahui apakah proyek memenuhi tujuan. Tim pengembang mendokumentasikan pengalaman dalam rangka memperoleh pelajaran bagaimana memberlakukan proyek selanjutnya.




**Gambar 2. 1 Project Life Cycle dan System Development Life Cycle**

**Sumber: Marchewka (2003)**

## *2.8 System Development Life Cycle*

*System Development Life Cycle* (SDLC) dapat diartikan sebagai fase berurutan atau tahapan yang ditempuh sistem informasi. SDLC memiliki lima fase dasar, yaitu (Marchewka, 2003):

1. Fase *Planning*. Pada fase ini memastikan tujuan, lingkup kerja, anggaran, jadwal, teknologi, proses pengembangan sistem, metode dan *tool* yang dibutuhkan.
2. Fase *Analysis*. Pada fase ini mengidentifikasi dan mendokumentasikan masalah atau hambatan yang terkait dengan sistem saat ini, serta kebutuhan dan persyaratan khusus untuk sistem yang baru.

3. Fase *Design*. Pada fase ini merancang arsitektur yang melingkupi perancangan jaringan, konfigurasi perangkat keras, basis data, *user interface* dan program aplikasi dengan menggunakan kebutuhan dan model logis "*to be*" sebagai masukan.
4. Fase *Implementation*. Pada fase ini melingkupi pengembangan atau pembangunan sistem, pengujian dan instalasi.
5. Fase *Maintenance* dan *Support*. Pada fase ini melakukan pemeliharaan dan perbaikan pada kesalahan yang ditemukan di sistem, serta menangani perubahan sistem.

## 2.9 *Work Breakdown Structure* (WBS)

*Work Breakdown Structure* (WBS) merupakan pengelompokan pekerjaan yang berorientasi *deliverable* (presentasi atau laporan, rencana, *prototype*, dan sistem aplikasi akhir) yang berkaitan dengan proyek dalam mendeskripsikan lingkup kerja total (Schwalbe, 2012). WBS mendekomposisi atau membagi proyek menjadi komponen yang lebih kecil dan unit kerja yang lebih mudah dikelola disebut *work package* (Marchewka, 2003).

*Work package* yang disusun memenuhi beberapa hal, antara lain: 1) Dapat didefinisikan, agar dapat dipahami dengan mudah oleh partisipan proyek; 2) Dapat dikelola, agar unit kerja dengan tanggungjawab dan wewenang tertentu dapat ditugaskan kepada pihak yang bertanggungjawab; 3) Dapat diestimasi, dapat memperkirakan durasi dari waktu yang dibutuhkan, dan biaya dari sumber daya yang dibutuhkan; 4) Bebas, memiliki ketergantungan minimal dengan elemen lainnya; 5) Terintegrasi, terintegrasi dengan elemen pekerjaan lain; 6) Terukur, dapat digunakan untuk menilai kemajuan proyek; 7) Beradaptasi, penambahan atau penghapusan ruang lingkup kerja yang fleksibel (Rev, 2003).

WBS adalah landasan untuk perencanaan, pelaksanaan, pengendalian, *statusing*, dan pelaporan proyek yang efektif dimana semua pekerjaan yang tercantum dalam WBS harus diidentifikasi, diperkirakan, dijadwalkan, dan dianggarkan. WBS merupakan struktur dan kode yang mengintegrasikan dan menghubungkan ruang lingkup, jadwal, dan biaya (Rev, 2003).

## *2.10 Gantt Chart*

*Gantt chart* merupakan *bar chart* yang merepresentasikan rencana dan *progress* aktual beberapa *task* dan ditampilkan dalam bentuk skala waktu horizontal (Taylor, 2008). *Gantt chart* memberikan format standar dalam menampilkan informasi tentang jadwal proyek dengan mencantumkan daftar aktivitas proyek beserta tanggal mulai dan selesai sesuai kalender. *Gantt chart* untuk proyek perilisan perangkat lunak mengandung *milestone, summary task*, durasi *task* individu, dan panah yang menunjukkan ketergantungan *task* (Schwalbe, 2012). *Milestone* merupakan hasil yang diprediksi dari suatu aktivitas atau rangkaian aktivitas berupa titik di dalam jadwal yang dapat digunakan untuk menilai kemajuan proyek (Sommerville, 2009).

## *2.11 Use Case*

*Use case* mepresentasikan suatu tujuan sistem dan menggambarkannya dalam urutan aktivitas dan interaksi pengguna dalam upaya mencapai tujuan tersebut. *Use case* merupakan teknik yang baik untuk lebih memahami dan mendokumentasikan kebutuhan sistem (Whitten & Bentley, 2007). *Use case* bukanlah fungsi ataupun fitur dan tidak bisa didekomposisi. *Use Case* memiliki nama, deskripsi singkat dan deskripsi detail yang menceritakan bagaimana aktor menggunakan sistem untuk melakukan sesuatu yang mereka anggap penting dan sistem apa yang dibutuhkan untuk memenuhi kebutuhan ini (Bittner, 2002). *Use case* mengidentifikasi aktor yang terlibat di dalam interaksi, dan menamai jenis interaksi tersebut. Beberapa informasi tambahan dicantumkan untuk menjelaskan interaksi yang dilakukan dengan sistem. *Use case* juga mengidentifikasi interaksi antara individu dengan sistem atau pengguna dengan sistem yang lain (Sommerville, 2009).

*Use case diagram* digunakan untuk meringkas apa yang dilakukan oleh sistem dengan menggambarkan aktor dan *use case*. Penjelasan lengkap mengenai apa yang terjadi di dalam *use case* didokumentasikan ke dalam *use case description*. Skenario merupakan kejadian spesifik pada suatu *use case* yang menunjukan apa yang dilakukan sistem ketika suatu *use case* tertentu dijalankan. Skenario merupakan representasi dari *main flow* dan *alternative flow* dari suatu *use case*. Skenario menunjukkan bagaimana *use case* berjalan dari awal sampai akhir. *Use Case Scenario* terdiri atas nama *use case*, *brief description* (deskripsi singkat), *basic flow* (aliran normal), *alternate flow* (aliran alternatif), *sub-flow*, *pre-condition* dan *post condition* (Bittner, 2002).

Transaksi *Use Case* menurut Jacobson et al. (1992), merupakan perjalanan bolak-balik (*roundtrip*) dari pengguna ke sistem, lalu kembali ke pengguna. Namun, sebenarnya tidak ada definisi standar untuk transaksi, berikut beberapa teori yang menjelaskan kriteria transaksi:

1) Perhitungan transaksi meliputi transaksi pada skenario yang berhasil dan *extention* (Nassif et al., 2010).
2) Semua *exception scenario* di dalam use case dianggap sebagai satu transaksi, dengan menganggap bahwa kerangka kerja dirancang bertujuan untuk menangani semua jenis pengecualian (Kashyap et al., 2010).
3) Skenario sukses dan scenario gagal dianggap sama dan dihitung sebagai dua transaksi (Diev, 2006).
4) Transaksi yang terlalu sederhana diabaikan, misalnya memasukkan nama pengguna, kemudian nama tersebut ditampilkan pada layar (Nassif et al., 2010).
5) Transaksi yang melibatkan operasi yang sama dalam *database* dan hanya berbeda instansiasi dihitung sebagai transaksi yang sama, meski ditulis menjadi dua skenario yang berbeda dalam *use case scenario (*Nassif et al., 2010).

## *2.12* Metode *Advance Use Case Point*

Metode *Use Case Points* (UCP) adalah algoritma estimasi *effort* yang diusulkan oleh Gustav Karner (1993). Metode ini digunakan untuk menghasilkan estimasi dari *use case* proyek. Metode UCP menggunakan *use case* proyek, aktor, skenario dan berbagai faktor teknis dan lingkungan, kemudian menghitungnya ke dalam sebuah persamaan. UCP menggunakan model *Use Case* (UC) untuk memperkirakan ukuran dan *effort* dari sistem yang berorientasi objek. *Advance Use Case Point* (AUCP) merupakan modifikasi dari UCP yang memanfaatkan model *End User Development* (EUD) untuk mendapatkan estimasi *effort* (Srivastava, 2017).

*End-User Development* (EUD) dapat didefinisikan sebagai seperangkat metode, teknik, dan alat yang memungkinkan pengguna sistem perangkat lunak menganggap dirinya sebagai pengembang perangkat lunak non-profesional dan memiliki kemampuan untuk membuat, memodifikasi, atau memperluas artefak perangkat lunak (Paterno, 2013). Fitur EUD jika digabungkan dalam perangkat lunak atau situs *web* akan meningkatkan kualitas dan meningkatkan kepuasan pengguna akhir.

### 2.12.1 *Unadjusted Actor Weight* (UAW)

Istilah aktor merujuk pada teori *Use Case* dapat didefinisikan sebagai orang, program lain, perangkat keras, dan sebagainya (Cohn, 2005), atau dapat juga didefinisikan sebagai segala sesuatu yang berinteraksi dengan *use case* (manusia atau sistem eksternal) (Bahrami, 1999). Langkah pertama pada metode *Advance Use Case Point* adalah mengklasifikasikan aktor ke dalam *simple, average* dan *complex*. Aktor diklasifikasikan *simple,* apabila aktor merepresentasikan sistem lain yang berinteraksi dengan sistem melalui API (*Application Programming Interface).* Aktor diklasifikasikan *average,* apabila actor merepresentasikan sistem lain yang berinteraksi dengan sistem melalui protokol seperti TCP/IP, HTTP, atau SOAP dan manusia yang berinteraksi dengan sistem melalui *command line*. Aktor diklasifikasikan *complex,* apabila aktor merepresentasikan manusia yang berinteraksi dengan sistem melalui *Graphical User Interface* (GUI) atau halaman *web* (Sangeetha & Dalal, 2016). Setiap aktor dihitung sekali, meskipun aktor tersebut terlibat dalam banyak *use case*. Aktor yang dipertimbangkan hanya aktor yang terspesialisasi dari aktor yang lebih umum (*generic actor*) atau aktor yang tidak memiliki subtipe (Wazlawick, 2014).

**Tabel 2. 3 Tipe dan Bobot Aktor *Advance Use Case Point***

| Tipe Aktor | Bobot Faktor |
|---|---|
| *Simple* | 1 |
| *Average* | 2 |
| *Complex* | 3 |

### 2.12.2 *Unadjusted Use Case Weight* (UUCW)

Langkah kedua adalah mengklasifikasikan setiap *use case* ke dalam *simple, average* dan *complex* tergantung pada jumlah transaksi dalam deskripsi *use case*,

termasuk skenario sekunder. *Simple use case* memiliki jumlah transaksi kurang dari 3 transaksi, *average use case* memiliki jumlah transaksi antara 4 sampai dengan 7 transaksi, *complex use case* memiliki jumlah transaksi lebih dari 7 transaksi (Sangeetha & Dalal, 2016). Sebuah *use case* dihitung satu kali, dimana hanya proses yang lengkap yang dapat dihitung yang berarti *extention*, varian, dan fragmen lainnya tidak boleh dihitung (Wazlawick, 2014).

**Tabel 2. 4 Tipe dan Bobot *Use Case Advance Use Case Point***

| Kompleksitas *Use Case* | Deskripsi | Bobot |
|---|---|---|
| *Simple* | <= 3 transaksi | 5 |
| *Average* | 4 sampai 7 transaksi | 10 |
| *Complex* | > 7 transaksi | 15 |

### 2.12.3 *Unadjusted Use Case Point* (UUCP)

Langkah ketiga adalah mengkombinasikan *Unadjusted Use Case Weight* (UUCW) dan *Unadjusted Actor Weight* (UAW) yang memberikan ukuran *unadjusted* dari keseluruhan sistem. Ini disebut sebagai *Unadjusted Use Case Points* (UUCP) dan ditentukan oleh Persamaan 2.1.

$$UUCP = UAW + UUCW \quad (2.1)$$

### 2.12.4 *Technical Complexity Factor* (TCF)

*Technical Complexity Factor* (TCF) merupakan salah satu faktor yang diterapkan dalam penentuan perkiraan ukuran perangkat lunak bertujuan untuk mempertimbangkan faktor teknis pada sistem. Dampak *technical complexity* pada *use case* dipresentasikan melalui 13 faktor, seperti ditunjukkan pada Tabel 2.4. Faktor-faktor ini mewakili dampak persyaratan non-fungsional proyek terhadap *effort* untuk menyelesaikan proyek. Setiap faktor dinilai berdasarkan skala 0 sampai dengan 5, dimana 0 berarti faktor tidak relevan dan 5 berarti faktor sangat penting. *Technical Complexity Factor* (TCF) dihitung dengan mengalikan nilai masing-masing faktor (T1-T13) dengan bobotnya dan kemudian menambahkan semua hasilnya untuk mendapatkan jumlah yang disebut *TFactor* (Ribu, 2001). Perhitungan TCF menggunakan Persamaan 2.2.

$$TCF = 0{,}6 + (0{,}01 * TFactor) \quad (2.2)$$

**Tabel 2. 5 *Technical Factors Advance Use Case Point***

| Technical Factor | | Bobot |
|---|---|---|
| T1 | *Distributed System Required* | 2 |
| T2 | *Response Time is Important* | 1 |
| T3 | *End User Efficiency* | 1 |
| T4 | *Complex Internal Processing Required* | 1 |
| T5 | *Reusable Code Must be a focus* | 1 |
| T6 | *Installation Easy* | 0,5 |
| T7 | *Usability* | 0,5 |
| T8 | *Cross-Platform Support* | 2 |
| T9 | *Easy to Change* | 1 |
| T10 | *Highly Concurrent* | 1 |

**Tabel 2. 6 *Technical Factors Advance Use Case Point* (Lanjutan)**

| | | |
|---|---|---|
| T11 | *Custom Security* | 1 |
| T12 | *Dependence on Third-Part Code* | 1 |
| T13 | User Training | 1 |

Keterangan Tabel 2.5, sebagai berikut:

1. Kebutuhan sistem terdistribusi. Pada sistem terdistribusi, setiap unit komputer dapat mengolah data secara otonom atau sendiri-sendiri, serta dapat mengalihkan informasi yang di hasilkan ke pusat (sentral) untuk selanjutnya data tersebut diolah di sentral (Marimin, Tanjung, & Prabowo, 2006). Misalnya, sejumlah *server workstation* dan *host* yang diikat oleh jaringan yang mendukung sejumlah aplikasi, salah satu contohnya, komunikasi antara *client* dan *server*. Komunikasi satu arah biasanya ada pada jaringan *User Datagram Protocol* (UDP), yang mengirimkan informasi satu arah dari *source* ke *destination*, tanpa mengecek apakah *destination* masih ada. Komunikasi dua arah biasanya ada pada *Transport Control Protocol* (TCP), dimana *client* mengirimkan *request* ke *server*, dan *server* menjawab *request*.

2. Pentingnya waktu respon sistem. Menyatakan seberapa pentingnya waktu respon aplikasi untuk pengguna. Waktu respon sistem merupakan bagian dari *performance requirement* pengguna yang menyatakan seberapa baik kemampuan yang dibutuhkan harus dilakukan tanpa memperhatikan bagaimana hal itu dilakukan. Waktu respon menentukan seberapa cepat permintaan harus diproses yang biasanya hasil dari kolaborasi beberapa komponen, seperti prosesor, memori, sistem disk, dan komponen lainnya (Podelko, 2012).

3. Efisiensi pengguna akhir. Menyatakan apakah aplikasi dirancang untuk memungkinkan pengguna hanya melakukan pekerjaan atau untuk meningkatkan efisiensi pengguna. Beberapa hal yang harus diperhatikan diantaranya: adanya *navigation help, online help*, gerakan kursor otomatis, *batch task* yang dikirimkan dari *online transaction*, penggunaan warna dan highlight visual, jumlah *screen*, dukungan *bilingual* dan *multilingual* (Wazlawick, 2014).

4. Kebutuhan kompleksitas proses internal. Menyatakan apakah aplikasi membutuhkan algoritma yang kompleks. Beberapa hal yang harus diperhatikan diantaranya: adanya kontrol yang hati-hati atau aplikasi memerlukan proses yang aman, proses *logic* yang luas, pengolahan matematika yang luas, banyak proses *exception* yang harus ditangani karena transaksi yang tidak lengkap yang perlu diproses ulang (gangguan komunikasi, *missing data value*, atau kegagalan perubahan data), dan pengolahan yang kompleks untuk berbagai kemungkinan *input* dan *output* (Wazlawick, 2014).

5. Fokus pada penggunaan ulang kode. Menyatakan apakah aplikasi dirancang sedemikian sehingga kode dan artefaknya akan dapat digunakan kembali. Kode yang dapat digunakan kembali dikemas secara khusus dan didokumentasikan untuk memudahkan penggunaan ulang. Misalnya, Pengembang Web dan Java memiliki perpustakaan kode dimana komponen dari proyek disimpan. Semua source code disimpan dalam version control

system yang terpisah dari kode yang lain. Para pengembang tidak menggunakan prosedur spesifik penggunaan ulang selain kode normal dan standar dokumentasi (Eriksen, 2004).

6. Kemudahan instalasi. Menyatakan apakah aplikasi membutuhkan intalasi yang mudah, user-friendly, dan apakah perlu mengikutsertakan ahli selama instalasi dan *initial setup* (Wazlawick, 2014). Pada beberapa kasus, perangkat lunak yang di-deploy merupakan pengganti dari perangkat lunak yang sudah ada sebelumnya, sehingga terdapat kebutuhan untuk melakukan migrasi database dari sistem lama ke sistem yang baru. Apabila database yang digunakan berbeda, maka proses migrasi juga melibatkan konversi data (Tantra, 2012).
7. Kemudahan penggunaan. Menyatakan bagaimana kemudahan dalam penggunaan dan dalam mempelajari bagaimana cara menggunakan sistem, serta apakah ada kebutuhan khusus saat mengoperasikan sistem. Beberapa hal yang harus diperhatikan diantaranya: adanya proses memerlukan intervensi operator dalam melakukan inisialisasi, *backup*, dan *recovery*; adanya proses tanpa intervensi operator dalam melakukan inisialisasi, *backup*, dan *recovery*; adanya kebutuhan untuk meminimalkan penyimpanan data di media *offline*, misalnya kaset; dan adanya kebutuhan untuk meminimalkan penggunaan kertas (Wazlawick, 2014).
8. Dukungan *cross-platform*. Menyatakan apakah aplikasi dirancang untuk bekerja di lebih dari satu *platform*. *Cross-platform* merupakan kemampuan perangkat lunak atau aplikasi yang dapat digunakan pada sistem operasi yang berbeda, seperti Windows Phone, Android, iOS, Blackberry, dan *platform* sistem operasi komputer, seperti Windows, Solaris, Amiga, Linux, dan sebagainya (Wahana Komputer, 2013).
9. Kemudahan terhadap perubahan. Menentukan apakah aplikasi akan dibangun sedemikian rupa sehingga mudah untuk dikembangkan lebih lanjut di masa depan. Beberapa hal yang perlu diperhatikan diantaranya: Struktur *report* yang fleksibel untuk menangani *simple queries, medium complexity queries,* dan *high complexity queries* dan kontrol data bisnis yang disimpan dalam tabel yang dikelola oleh pengguna melalui akses online yang perubahannya berlaku hari berikutnya atau segera berlaku (Wazlawick, 2014).
10. Kemampuan konkurensi. Menyatakan apakah aplikasi dirancang untuk mengatasi masalah yang terkait dengan konkurensi seperti, berbagi data dan sumber daya (Wazlawick, 2014). Konkuresi terjadi ketika sebuah transaksi membaca data yang saat itu sedang dimodifikasi oleh transaksi yang lain, atau ketika dua buah transaksi secara stimultan memodifikasi data yang sama (Kleppmann, 2017).
11. Kebutuhan keamanan. Menyatakan apakah aplikasi akan memerlukan desain khusus dan spesifikasi tambahan yang dibutuhkan berkaitan dengan pengamanan akses terhadap sistem atau data. Otentikasi menetapkan identitas satu pihak ke pihak lain. Otentikasi menetapkan identitas pengguna ke beberapa bagian sistem, biasanya dengan menggunakan kata sandi.

Otentikasi menjadi prasyarat untuk kontrol akses. Kontrol akses menentukan apa yang dapat diakses oleh seseorang terhadap sumberdaya. Proses audit mengumpulkan data tentang aktivitas dalam sistem dan menganalisisnya untuk menemukan pelanggaran keamanan atau mendiagnosa penyebabnya (Sandhu & Samarati, 1996).

12. Ketergantungan terhadap kode pihak ketiga. Menyatakan apakah aplikasi menggunakan kode yang sudah dikembangkan, seperti komponen *commercial off-the-shelf* (COTS), *framework* atau *library*. Penggunaan kode *reuse*berkualitas tinggi mengurangi nilai karena berarti usaha pengembangan berkurang. Jika semakin bergantung kode pihak ketiga, maka semakin rendah nilai yang diberikan(Wazlawick, 2014).
13. Pelatihan pengguna. Menyatakan apakah aplikasi mudah digunakan, atau membutuhkan pelatihan yang ekstensif kepada pengguna di masa depan, dan apakah kebutuhan pelatihan pengguna tertentuakan diperhatikan dan aplikasi harus dirancang untuk memfasilitasi pelatihan (Wazlawick, 2014).

### 2.12.5 *Environmental Complexity Factor* (ECF)

*Environmental Complexity Factor (ECF)* merupakan faktor lain yang diterapkan dalam penentuan perkiraan ukuran perangkat lunak yang bertujuan untuk mempertimbangkan kondisi lingkungan dari sistem (Kirmani & Wahid, 2015). *Environmental Factor* digunakan untuk mengukur persyaratan non-fungsional seperti kemudahan penggunaan dan motivasi pemrogram. *Environmental Complexity Factor* (ECF) dihitung dengan mengalikan nilai masing-masing faktor (F1-F8) dengan bobotnya dan menambahkan seluruh hasil untuk mendapatkan jumlah yang disebut *EFactor* (Ribu, 2001). Perhitungan ECF menggunakan Persamaan 2.3.

$$ECF = 1{,}4 + (-0{,}03 * EFactor) \quad (2.3)$$

**Tabel 2. 7 *Environmental Factors Advance Use Case Point***

| Environmental Factor | | Bobot |
|---|---|---|
| F1 | *Familiar with Objectory* | 1,5 |
| F2 | *Stable requirement* | 2 |
| F3 | *Analyst capability* | 0,5 |
| F4 | *Application experience* | 0,5 |
| F5 | *Object-oriented experience* | 1 |
| F6 | *Motivation* | 1 |
| F7 | *Difficult Programming language* | -1 |
| F8 | *Part Time Workers* | -1 |

Keterangan Tabel 2.7, sebagai berikut:

1. Familiaritas dengan proses pengembangan. Faktor ini menilai pengalaman tim terhadap proses pengembangan yang digunakan. Misalnya, tentang metodologi yang diterapkan pada proyek, seperti RUP (*Rational Unified Process*), serta pengetahuan tentang bahasa pemodelan sistem, seperti UML (*Unified Modelling Language*) (Wazlawick, 2014).

2. Kestabilan kebutuhan. Faktor ini mengevaluasi apakah tim telah mampu menjaga kebutuhan tetap stabil di proyek-proyek masa lalu, meminimalkan perubahan selama proyek berlangsung. Faktor ini menilai hanya perubahan perangkat lunak yang disebabkan oleh ketidakmampuan tim untuk mendapatkan kebutuhan yang tepat (Wazlawick, 2014).
3. Kemampuan analisis. Faktor ini menilai pengalaman analis utama untuk memperoleh kebutuhan klien dengan benar dan memodelkannya dengan pemodelan berorientasi objek (Wazlawick, 2014).
4. Pengalaman terhadap aplikasi. Faktor ini menilai keakraban tim dengan area aplikasi atau domain. Misalnya, jika aplikasinya tentang *e-commerce*, apakah tim sudah bekerja dengan sistem di area yang sama (Wazlawick, 2014).
5. Pengalaman pengembangan berorientasi objek. Faktor ini menilai pengalaman dalam melakukan analisis, pemodelan, desain, dan pemrograman berorientasi obyek, terlepas dari area aplikasi dan proses pengembangan yang digunakan (Wazlawick, 2014).
6. Motivasi tim. Faktor ini menilai motivasi tim untuk terlibat dalam tugas yang ditugaskan dan apakah memperlukan adanya pengawasan atau supervisi dalam menjaga produktivitas (Wazlawick, 2014).
7. Tingkat kesulitan bahasa pemrograman. Faktor ini menilai tingkat keakraban pengembang dengan bahasa pemrograman sistem yang akan diimplementasikan (Wazlawick, 2014).
8. Penggunaan pengembang kerja paruh waktu. Faktor ini adalah faktor negatif yang bertentangan dengan enam faktor lingkungan pertama, nilai yang lebih tinggi mewakili lebih banyak waktu pengembangan. Sebuah tim yang memiliki banyak anggota yang terlibat dalam proyek atau aktivitas lain menjadi kurang produktif daripada tim yang berdedikasi (Wazlawick, 2014).

### 2.12.6 *Use Case Point* (UCP)

Total *Use Case Point* (UCP) diperoleh dengan cara mengkalikan *Unadjusted Use Case Points* (UUCP) dengan *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) sesuai Persamaan 2.4.

$$UCP = UUCP * TCF * ECF \quad (2.4)$$

### *2.12.7 End User Development Technical Complexity Factor*

Tambahan *Technical Factor* yang diberikan kepada pengguna akhir berguna untuk kenyamanan pada saat pembangunan proyek. Pemberian fitur *End User Development* (EUD) dalam pengembangan perangkat lunak menimbulkan biaya tambahan dalam pengembangan. *End User Development Technical Complexity Factor (EUD_TCF)* terdiri dari 17 faktor. Setiap faktor memiliki nilai 0 atau 1, dimana 0 berarti fitur tersebut diperlukan dan 1 berarti fitur tersebut tidak diperlukan. *EUD_TCF* dihitung dengan mengalikan nilai masing-masing faktor (T1-T17) dengan bobotnya dan kemudian menambahkan semua hasilnya untuk

mendapatkan jumlah yang disebut *End User Development Technical Factor (EUD_TF)* (Srivastava, 2017) yang dihitung menggunakan Persamaan 2.5.

$$EUD_TCF = 0,6 + (0,01 * EUD_TF) \quad (2.5)$$

**Tabel 2. 8 *End User Technical Factors***

| ***End User Technical Complexity Factor*** | | **Bobot** |
|---|---|---|
| T1 | *Creating throw away codes* | 0,5 |
| T2 | *Creating reusable codes* | 1,2 |
| T3 | *Sharing reusable code* | 1,4 |
| T4 | *Easy and understandable code* | 1 |
| T5 | *Security features in code for more control by end user* | 1,3 |
| T6 | *Authentication features* | 1,12 |
| T7 | *Inbuilt feedback about the correctness* | 1,3 |
| T8 | *Testable code* | 1,2 |
| T9 | *Tools for analyzing by debugging* | 1,4 |
| T10 | *Error detection tools* | 1,2 |
| T11 | *Online help availability* | 1,3 |
| T12 | *Self-efficacy* | 1,11 |
| T13 | *Perceived ease of use* | 1,2 |
| T14 | *Perceived usefulness* | 1 |
| T15 | *Flexible code* | 1,2 |
| T16 | *Scalability features* | 1,25 |
| T17 | *Ease of maintenance* | 1,2 |

Keterangan Tabel 2.8, sebagai berikut:

1. Pembuatan kode *throw-away*. *End-user programmer* seringkali menciptakan Kode *throw-away*. Kode *throw-away* merupakan kode sementara untuk menyelesaikan masalah, kemudian dibuang dan diperbarui dengan kode baru yang lebih rapi dan teroganisir dari sebelumnya, namun dalam banyak kasus kode semacam itu cukup berumur panjang, sehingga dibutuhkan repositori karena kemungkinan nanti pengguna lain menggunakan kembali *script* tersebut untuk tujuan lain (Ko et al., 2011).

2. Pembuatan kode penggunaan ulang. Pengguna akhir dapat menciptakan abstraksi yang dapat digunakan kembali, seperti *reusable code* atau kode yang dapat digunakan kembali, sehingga pada kebanyakan lingkungan *end-user programming* menyediakan cara bagi pengguna untuk membuat abstraksi baru. Salah satu cara untuk memudahkan pengguna akhir menggunakan kembali kode adalah dengan memilihkan abstraksi yang tepat untuk domain masalah pengguna. Pengembang merancang konsep dan tingkat abstraksi yang tepat sesuai dengan konsep pengguna tersebut. Misalnya, *Google Maps API* memberikan kemudahan ke pengguna dalam membubuhkan keterangan pada gambar geografis dengan tipe data khusus (Ko et al., 2011).

3. Berbagi kode penggunaan ulang. Ketika *reusable code* dibuat oleh pengguna akhir, sistem memiliki tantangan untuk mempertahankan repositori yang berisi kontribusi pengguna tersebut. Seorang pengguna menulis *script* sederhana untuk memudahkan beberapa proses bisnis dan kemudian,

pengguna lain menggunakan kembali *script* tersebut untuk tujuan lain. Untuk alasan ini, organisasi menciptakan alat untuk mengotomatisasi dan berbagi pengetahuan dimana *script* di dalam repositori dapat disalin dan diduplikasi untuk mengembangkan *script* baru (Ko et al., 2011).

4. Kemudahan kode dimengerti. Pengguna akhir biasanya menciptakan program dengan menggunakan ulang kode yang sudah ada pada suatu program, kemudian beberapa perubahan dilakukan pada kode. Pengguna akhir harus dapat menemukan dan memahami kode sebelum menggunakannya, sehingga kode harus mudah dimengerti (Cypher et al., 2010).
5. Kontrol terhadap fitur keamanan. Aplikasi *web* rentan terhadap eksploitasi, misalnya sistem operasi, *server web, database*, bahasa *script* dan sebagainya. Sangat sulit untuk membangun aplikasi yang cukup aman dan menilai apakah serta kapankah aplikasi aman. Perangkat untuk *end-user development* pada *web* harus memberikan dukungan untuk manajemen keamanan, misalnya adanya validasi *input* pengguna meningkatkan keamanan untuk mencegah serangan *hacker* (Rode, Rosson, & Quinones, 2005).
6. Adanya fitur otentikasi. Pengguna akhir mungkin tidak memiliki banyak kesulitan dalam memahami mekanisme perizinan, namun mungkin tidak dapat menciptakan mekanisme perizinan yang cukup kuat dan aman, sehingga perangkat *end-user development* harus menawarkan pengelolaan izin siapa yang bisa menggunakan aplikasi dan fitur mana yang bisa digunakan sebagai fitur bawaan dan membuatnya dapat dikustomisasi oleh pengembang (Rode, Rosson, & Quinones, 2005).
7. Umpan balik *inbuilt* (*guideline*) terhadap *correctness*. *End-user programmer* biasanya terlalu percaya diri atas kebenaran program mereka dimana faktanya 5% - 23% penilaian yang dilakukan tidak benar. Terdapat beberapa cara untuk memastikan kebenaran suatu program, yaitu melalui verifikasi, pengujian, atau pendekatan lain. Alat untuk mendeteksi kesalahan ini harus memberikan umpan balik langsung tentang nilai yang dihitung dari sebuah program atau lebih mempercayai penilaian negatif pengguna akhir daripada penilaian positif (Ko et al., 2011).
8. Kode dapat diuji. *Testable code* merupakan kode yang dapat dengan mudah diuji. Kode ini dipecah menjadi lapisan-lapisan dimana tiap lapisan memiliki tugasnya masing-masing daripada kode yang diletakkan di dalam satu lokasi (Fisher & Duskis, 2009). Memastikan bahwa kode tersebut dapat diuji artinya memastikan apakah penggabungannya longgar dan diperhitungkan dengan baik, sehingga fleksibel dan mudah digunakan, secara keseluruhan maupun sebagian (Johansen, 2010).
9. Alat analisis *debugging*. *Debugging* merupakan proses menemukan dan menghilangkan kesalahan. Terdapat beberapa pendekatan untuk melakukan *debugging* diantaranya: menganalisis dependensi eksekusi program dan dependensi data, adanya daftar saran perubahan untuk mendapatkan nilai

yang diinginkan, *sharing reasoning*, dan bekerja secara berpasangan (Ko et al., 2011).

10. Alat mendeteksi kesalahan. Pendekatan untuk membantu *end-user programmer* mendeteksi kesalahan dengan cara melakukan pegujian dan memeriksa nilai yang dihitung oleh program apakah sesuai dengan bentuk spesifikasinya. Spesifikasi berfungsi sebagai jawaban untuk menilai kebenaran program. Pengujian menilai kebenaran program berdasarkan kebenaran keluaran program (Ko et al., 2011).
11. Adanya bantuan *online*. Bantuan *online* merupakan padanan elektronik berasal dari buku manual untuk pengguna yang diintegrasikan ke dalam sebuah program. Bantuan *online* terkadang berhubungan dengan situs-situs *web* yang menyediakan hal-hal terbaru dan sumber daya yang lebih lengkap untuk mengatasi permasalahan-permasalahan teknis pada perangkat lunak (Vermaat, 2007).
12. Self-efficacy. *Self-efficacy* merupakan sebuah konstruksi psikologi yang mewakili kepercayaan seseorang terhadap kemampuan mereka untuk mencapai suatu tugas tertentu. Pengembangan perangkat lunak merupakan tugas yang menantang, dimana seseorang dengan tingkat *self-efficacy* rendah cenderung tidak bertahan saat sebuah tugas semakin sulit atau melakukannya dengan cara yang berbeda. *Self-efficacy* pada pria dan wanita berbeda. Wanita memiliki *self-efficacy* yang lebih rendah daripada pria yang menyebabkan adanya hambatan untuk mencapai kesuksesan pada aktivitas *end-user software engineering* baik pria maupun wanita. Desain untuk fitur harus mendukung pengguna akhir menghilangkan hambatan ini, terlepas dari siapa pengguna akhir apakah pria atau wanita (Ko et al., 2011).
13. Persepsi kemudahan penggunaan. Persepsi kemudahan penggunaan menilai sejauh mana seorang pengguna percaya bahwa menggunakan sistem tertentu tidak membutuhkan usaha yang keras atau mengalami kesulitan. Sebuah aplikasi yang dianggap lebih mudah digunakan daripada yang lain akan lebih mudah diterima oleh pengguna (Davis, 1989).
14. Persepsi manfaat. Persepsi manfaat menilai sejauh mana seorang pengguna percaya bahwa menggunakan sistem tertentu akan membantu melakukan pekerjaan dengan lebih baik. Persepsi kebermanfaatan memiliki pengaruh besar terhadap penerimaan pengguna akhir untuk menggunakan teknologi informasi (Davis, 1989).
15. Kode yang fleksibel. Kode yang fleksibel merupakan kode yang dapat ditambahkan oleh pengguna akhir untuk manambah atau mengubah perilakunya. Kode ini memberi pengguna akhir beberapa titik masuk untuk membiarkan pengguna menambahkan kode pengguna sendiri (Robinson, 2007).
16. Adanya fitur skalabilitas. Perancangan sistem mengantisipasi perubahan kebutuhan dan pola penggunaan sumber daya masa depan dengan memastikan bahwa semua sumber daya yang dibutuhkan tersedia sejak awal

atau dengan mendukung penambahan sumber daya secara dinamis saat dibutuhkan (Weinstock & Goodenough, 2006).

17. Kemudahan pemeliharaan. Saat memilih metode perancangan, ada dua aspek yang harus diperhatikan, yaitu dapat menemukan atau melacak lokasi kesalahan dan mendukung modifikasi dan peningkatan sistem, yang dimasukkan ke dalam dokumentasi perancangan yang baik. Adanya dokumentasi memudahkan proses pemeliharaan (Birrell & Ould, 1988).

### *2.12.8 End User Development Environmental Complexity Factor*

*End User Development Environmental Complexity Factor (EUD_ECF)* terdiri dari 8 faktor. Setiap faktor memiliki nilai 0 atau 1, dimana 0 berarti fitur tersebut diperlukan dan 1 berarti fitur tersebut tidak diperlukan. *EUD_ECF* dihitung dengan mengalikan nilai masing-masing faktor (F1-F8) dengan bobotnya dan kemudian menambahkan semua hasilnya untuk mendapatkan jumlah yang disebut *End User Development Environmental Factor (EUD_EF)* (Srivastava, 2017) yang dihitung menggunakan Persamaan 2.6.

$$EUD_ECF = 1{,}4 + (0{,}03 * EUD_EF) \quad (2.6)$$

**Tabel 2. 9 *End User Environmental Factor***

| ***End User Environmental Complexity Factor*** | | **Bobot** |
|---|---|---|
| F1 | *Content level of EUP* | 1,4 |
| F2 | *End User Computing capability* | 0,25 |
| F3 | *Ease of use and feedback* | 1,2 |
| F4 | *Inbuilt system assistance for EUP* | 1,25 |
| F5 | *Training and learning time contrain for end user* | 1,12 |
| F6 | *Reliability of end user code* | 1,2 |
| F7 | *End user storage constraint* | 1,02 |
| F8 | *Risk factors* | 1,12 |

Keterangan Tabel 2.9, sebagai berikut:

1. *Content level End User Programming*. Sebuah sistem harus menawarkan berbagai level modifikasi yang berbeda seiring meningkatnya kompleksitas. Hal ini untuk memastikan bahwa pengguna dapat membuat perubahan kecil dengan mudah, sementara perubahan yang lebih rumit akan melibatkan peningkatan kompleksitas (Lieberman et al., 2006 disitasi dalam Ko et al., 2011, p.11).

2. Kemampuan komputasi pengguna akhir. Keterampilan komputasi pengguna memiliki empat komponen, yaitu kemampuan teknis, pengetahuan tentang bisnis, latar belakang pendidikan dan pengalaman komputasi. Pengguna akhir yang melakukan tugas di dalam lingkungan informasi membutuhkan kemampuan untuk secara efisien mengeksekusi tugas dengan mengaplikasikan sistem ke dalam tugas tersebut (Shen et al., 2007).

3. Kemudahan penggunaan dan umpan balik. Sebagian besar perangkat untuk pengguna akhir kurang memperhatikan kemudahan dalam penggunaan. Pengguna membutuhkan contoh referensi, panduan (*wizard*) yang baik tapi

pendek, server web terintegrasi untuk tujuan pengujian, dan dukungan yang baik selama fase pemasangan aplikasi. Pengguna dapat mengembangkan dan menggunakan aplikasi tanpa bolak-balik beralih antara mode pemrograman dan runtime. Artinya, aplikasi dapat digunakan semaksimal mungkin setelah selesai diprogram, dan ketika batas-batasnya (*boundary*) diuji, lingkungan memberikan umpan balik yang berguna untuk menyarankan langkah selanjutnya yang diambil oleh pengembang (Rode, 2005).

4. Bantuan sistem inbuilt *End-User Programming*. Ada indikasi bahwa dalam mencapai keberhasilan dan kegunaan dengan menyediakan asisten (bantuan) untuk mendorong partisipasi pengguna akhir secara berkelanjutan (Yetim et al., 2012 disitasi dalam Haake et al., 2016). Untuk memberikan informasi kepada pengguna akhir tentang apa yang sedang mereka kerjakan dan fungsi sistem apa yang mereka gunakan, asisten (bantuan) harus menangkap konteks pengguna saat itu dari perspektif sistem dan menyediakan semacam wiki interaktif yang dikombinasikan dengan fitur penyelesaian otomatis yang mendukung pengguna akhir untuk mengartikan kebutuhan mereka (Haake et al., 2016).
5. Batasan waktu pelatihan dan pembelajaran pengguna akhir. Memberikan pelatihan kepada pengguna akhir pada lingkungan komputasi pengguna akhir akan menghasilkan manfaat yang maksimal. Apabila pelatihan tidak menggunakan sumber daya yang cukup dalam memberi pelajaran kepada pengguna, maka memiliki konsekuesnsi berat yang dapat menurunkan poduktivitas dan biaya organisasi lainnya (Srivastava, Singh, & Abbas, 2015).
6. Reliabilitas kode pengguna akhir. Perbedaan utama antara rekayasa perangkat lunak oleh profesional dan rekayasa perangkat lunak oleh pengguna akhir adalah pada sebesar apa perhatian yang diberikan pada masalah kualitas perangkat lunak. Tujuan di balik pemrograman membedakan *end-user programmer* dari aktivitas lain, karena tujuan *end-userprogrammer* menentukan sejauh mana pengguna akhir mempertimbangkan masalah keandalan, penggunaan kembali, dan pemeliharaan, dan sejauh mana mereka terlibat dalam aktivitas yang memperkuat kualitas ini, seperti pengujian, verifikasi, dan *debugging* (Ko et al., 2011).
7. Batasan penyimpanan pengguna akhir. *Storage* atau media penyimpanan merupakan media untuk menyimpan data, intruksi, dan informasi yang digunakan di masa mendatang yang ada pada setiap perangkat lunak. Suatu komputer biasanya menempatkan sistem operasi pada penyimpanan berupa *hardisk* (Irwansyah & Moniaga, 2014).
8. Faktor resiko. Pengguna akhir enggan menggunakan fitur pada sistem baru yang tidak diketahui, karena menganggap fitur tersebut berisiko atau tidak membantu. Menurut Blackwell (2002), seorang pengguna mempertimbangkan empat faktor sebelum mengambil sebuah tindakan, salah satunya adalah persepsi resiko. Program dapat gagal berfungsi, gagal

meberikan manfaat otomatisasi yang diinginkan, atau bahkan mengakibatkan kerusakan. Upaya yang dapat dilakukan untuk menghindari kemungkinan tersebut dilakukan dengan cara pengujian atau *debugging*. Persepsi terhadap resiko juga berpengaruh signifikan pada kesediaan pengguna untuk terlibat dalam rekayasa perangkat lunak (Ko et al., 2011).

### 2.12.9 *Advance Use Case Point* (AUCP)

*Advance Use Case Point* dihitung menggunakan hasil perhitungan dari *Use Case Point, End User Development Technical Complexity Factors* and *End User Development Environmental Complexity Factors* sesuai Persamaan 2.7.

AUCP = UCP * (EUD_TCF * EUD_ECF) (2.7)

### 2.12.10 Estimasi *Effort*

*Effort* per-AUCP dihitung dengan menggunakan 20 *person-hours* per UCP seperti yang disarankan oleh Karner (1993). Persamaan 2.8 digunakan untuk mengubah jumlah AUCP menjadi *person-hours*.

*Effort* = UCP * PH per UCP (2.8)

## *2.13* Metode *Revised Use Case Point*

Metode *Revise Use Case Point* (Re-UCP) mengukur fungsionalitas sistem dengan menghitung semua *use case point* di dalam sistem. Fungsionalitas sistem diperkirakan berdasarkan kumpulan dampak dari faktor yang terkait aktor sistem, perilaku *use case*, dampak lingkungan dan peran *technical complexity factor* pada proyek perangkat lunak. Aktor sistem, *use case*, *environmental factor*dan *technical factor* lainnya dikategorikan lebih lanjut. Tidak seperti UCP dimana aktor atau *use case* dibagi menjadi *simple, average*, atau *complex* dengan beberapa bobot, Re-UCP menggunakan kategorisasi aktor dan *use case* yang berbeda (Kirmani & Wahid, 2015).

### 2.13.1 *Unadjusted Actor Weight* (UAW)

Istilah aktor merujuk pada teori *Use Case* dapat didefinisikan sebagai orang, program lain, perangkat keras, dan sebagainya (Cohn, 2005), atau dapat juga didefinisikan sebagai segala sesuatu yang berinteraksi dengan *use case* (manusia atau sistem eksternal) (Bahrami, 1999). Langkah pertama pada metode *Revised Use Case Point* adalah mengkategorikan aktor ke dalam *simple, average, complex* dan *critical*. Seorang aktor didefinisikan sebagai "*simple*", apabila aktor berinteraksi dengan sistem melalui *Application Programming Interface* (API). Seorang aktor didefinisikan sebagai "*average*", apabila aktor berinteraksi dengan bantuan Interface Interaktif atau *Protocol-Driven*. Seorang aktor didefinisikan sebagai "*complex*", apabila aktor berinteraksi melalui *Graphical User Interface* (GUI), sedangkan seorang aktor didefinisikan sebagai "Kritis", apabila aktor berinteraksi dengan sistem melalui modul yang membutuhkan tindakan *real time* atau kompleksitasnya sangat tinggi (Kirmani& Wahid, 2015). Setiap aktor dihitung sekali, meskipun aktor tersebut terlibat dalam banyak *use case*. Aktor yang

dipertimbangkan hanya aktor yang terspesialisasi dari aktor yang lebih umum (*generic actor*) atau aktor yang tidak memiliki subtipe (Wazlawick, 2014).

**Tabel 2. 10 Tipe dan Bobot Aktor *Revised Use Case Point***

| Tipe Aktor | Bobot |
|---|---|
| *Simple* | 1 |
| *Average* | 2 |
| *Complex* | 3 |
| *Critical* | 4 |

### 2.13.2 *Unadjusted Use Case Weight* (UUCW)

Langkah kedua, *Use case* dalam Re-UCP juga dikategorikan menjadi kategori *simple, average, complex* dan *critical* beserta deskripsi terperinci tentang *use case* dengan pembobotan yang sesuai. Tipe *use case* diklasifikasikan menjadi "*Simple*", apabila jumlah transaksi *use case* kurang dari atau sama dengan 4, menjadi "*Average*", apabila jumlah transaksi *use case* antara 5 dan 8, menjadi "*Complex*", apabila jumlah transaksi *use case* antara 9 dan 15 dan"*Critical*", apabila jumlah transaksi *use case* lebih besar dari 15 (Kirmani & Wahid, 2015).

**Tabel 2. 11 Tipe dan Bobot *Use Case Revised Use Case Point***

| Tipe *Use Case* | Jumlah Transaksi | Bobot |
|---|---|---|
| *Simple* | <= 4 transaksi | 5 |
| *Average* | 5 sampai 8 transaksi | 10 |
| *Complex* | 9 sampai 15 transaksi | 15 |
| *Critical* | > 15 | 20 |

### 2.13.3 *Unadjusted Use Case Point* (UUCP)

Langkah ketiga adalah mengkombinasikan *Unadjusted Use Case Weight* (UUCW) dan the *Unadjusted Actor Weight* (UAW) yang memberikan ukuran *unadjusted* dari keseluruhan sistem. Nilai tersebut disebut sebagai *Unadjusted Use Case Points* (UUCP) dan ditentukan oleh persamaan 2.9.

$$UUCP = UAW + UUCW \quad (2.9)$$

### 2.13.4 *Technical Complexity Factor* (TCF)

*Tehnical Complexity Factor* (TCF) merupakan salah satu faktor yang digunakan untuk mengintegrasikan keunggulan berbagai faktor teknis yang terdaftar dalam sistem pada keseluruhan proses perkembangan dan sekaligus memperkirakan pengaruhnya terhadap keseluruhan proses estimasi *effort* perangkat lunak. Pada metode Re-UCP, jumlah parameter di TCF bertambah dari 13 menjadi 14 dimana parameter skalabilitas dimasukkan sebagai parameter ke-14. Label untuk parameter ke-14 di TCF adalah T14. Tambahan faktor skalabilitas menilai kemampuan untuk menangani beban kerja yang meningkat tanpa menambahkan sumber daya ke sistem (Kirmani & Wahid, 2015). Artinya, sistem terus berjalan secara memadai seiring dengan meningkatnya beban kerja. Misal, mengganti algoritma meningkatkan efisiensi perhitungan dalam arti bahwa beban

kerja yang lebih besar dapat diproses dalam jumlah waktu yang sama (Weinstock & Goodenough, 2006).

Setiap faktor teknikal dinilai berdasarkan skala 0 sampai dengan 5, dimana 0 berarti faktor tidak relevan, 3 berarti rata-rata relevan dan 5 berarti faktor sangat penting. Semua faktor teknis dari T1 sampai T14 dikalikan dengan bobot yang sesuai seperti yang dijelaskan pada Tabel 2.11 dan kemudian menambahkan semua hasilnya untuk mendapatkan jumlah yang disebut *TFactor* (Ribu, 2001). Perhitungan TCF menggunakan Persamaan 2.10.

$$TCF = 0{,}6 + (0{,}01 * TFactor) \quad (2.10)$$

**Tabel 2. 12 *Technical Factors Revised Use Case Point***

| Technical Factor | | Bobot |
|---|---|---|
| T1 | *Distributed System* | 2 |
| T2 | *Response Time is Important* | 1 |
| T3 | *End User Efficiency (online)* | 1 |
| T4 | *Complex Internal Processing Required* | 1 |
| T5 | *Reusable Code Must be a focus* | 1 |
| T6 | *Easy to Install* | 0,5 |
| T7 | *Easy to use* | 0,5 |
| T8 | *Cross-Platform Support* | 2 |
| T9 | *Easy to Change* | 1 |
| T10 | *Highly Concurrent* | 1 |
| T11 | *Custom Security* | 1 |
| T12 | *Dependence on Third-Part Code* | 1 |
| T13 | User Training | 1 |
| T14 | Scalability | 2 |

### 2.13.5 *Environmental Complexity Factor* (ECF)

*Environmental Complexity Factor (ECF)* merupakan faktor lain yang diterapkan dalam penentuan perkiraan ukuran perangkat lunak yang bertujuan untuk mempertimbangkan kondisi lingkungan dari sistem (Kirmani & Wahid, 2015). Pada Re-UCP, jumlah parameter di ECF bertambah dari 8 menjadi 9. Parameter kesembilan yang disertakan adalah metodologi proyek. Faktor ini menilai pengalaman pengembang dalam metodologi proyek yang dipilih untuk pengembangan proyek perangkat lunak.

Label untuk parameter kesembilan dalam ECF adalah E9 dan setiap faktor dinilai berdasarkan skala 0 sampai dengan 5, dimana 0 berarti pengembang tidak berpengalaman, 3 berarti berpengalaman rata-rata, dan 5 berarti pengembang sudah ahli (*expert*). *Environmental Complexity Factor* (ECF) dihitung dengan mengalikan nilai masing-masing faktor (F1-F9) dengan bobotnya dan menambahkan seluruh hasil untuk mendapatkan jumlah yang disebut *EFactor* semua nilai dijumlahkan menghasilkan *Environmental Complexity Factor* (ECF). Perhitungan ECF menggunakan Persamaan 2.11.

$$ECF = 1{,}4 + ( -0{,}03 * EFactor) \quad (2.11)$$

**Tabel 2. 13 *Environmental Factors Revised Use Case Point***

| Environmental Factor | | Bobot |
|---|---|---|
| F1 | *Familiar with Objectory* | 1,5 |
| F2 | *Stable requirement* | 2 |
| F3 | *Analyst capability* | 0,5 |
| F4 | *Application experience* | 0,5 |
| F5 | *Object-oriented experience* | 1 |
| F6 | *Motivation* | 1 |
| F7 | *Difficult Programming language* | -1 |
| F8 | *Part Time Workers* | -1 |
| F9 | *Project Methodology* | 1 |

### 2.13.6 *Revised Use Case Point* (Re-UCP)

Jumlah total semua Re-UCP dihitung dengan mengalikan *Unadjusted Use Case Points* (UUCP), *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) sesuai pada Persamaan 2.12.

Re-UCP = UUCP * TCF * ECF (2.12)

### 2.13.7 Estimasi *Effort*

*Effort* per Re-UCP dihitung dengan menggunakan 20 *person-hours per UCP* seperti yang disarankan oleh Karner (1993). Persamaan 2.13 digunakan untuk mengubah jumlah Re-UCP menjadi *person-hours*.

*Effort* = UCP * PH per UCP (2.13)

## 2.14 Distribusi *Effort*

*Effort* adalah kombinasi dari orang dan waktu yang menunjukkan jumlah waktu kerja produktif sepenuhnya yang dibutuhkan seseorang untuk menyelesaikan pekerjaan tertentu (Jeffery, 2014). Satuan *effort* antara lain: *person-hours, person-days, person-months*, dan pada proyek yang besar digunakan *person-years*. Berdasarkan *guideline* Saleh (2011), *effort* yang didapatkan dibagi menjadi dua kelompok aktivitas, yaitu *Phased Software Development Activity,* dan *Ongoing Life-cycle Activity,* yang dijelaskan sebagai berikut:

1. *Software Development Activities*. Fase ini mencangkup analisis pada kebutuhan pengguna dan spesifikasi, desain, implementasi, *integration testing* dan penerimaan dan *deployment* yang membutuhkan alokasi *effort* sebesar 50% dari keseluruhan *effort*.

   *1) Analysis*, mencangkup di dalamnya aktivitas *requirement* dan *specification*, mendefinisikan kebutuhan fungsional dan non-fungsional, antarmuka antara entitas eksternal dan perangkat lunak yang akan dikembangkan, serta prioritas kebutuhan perangkat lunak.

   *2) Design,* meliputi desain arsitektur tingkat tinggi (identifikasi modul dan antarmuka perangkat lunak), *database*, *interface*, dan desain yang detail.

3) *Implementation,* mentransformasikan perancangan menjadi kode yang dapat dieksekusi.

4) *Integration Testing,* mengintegrasikan modul yang telah diuji secara terpisah ke modul, sistem, dan komponen eksternal, kemudian melakukan rencana pengujian integrasi yang hasilnya perangkat lunak terintegrasi.

5) *Acceptance* dan *Deployment,* dilakukan setelah perangkat lunak disampaikan kepada klien dan diinstall sesuai dengan rencana pemasangan, kemudian *acceptance test plan* dijalankan.

2. *Ongoing Life-Cycle Activity*. Aktivitas di dalamnya mencangkup manajemen proyek, manajemen konfigurasi, dokumentasi, *quality assurance*, serta dukungan dan pelatihan yang membutuhkan alokasi *effort* sebesar 50% dari keseluruhan *effort*.

   1) *Project Management*, PM memantau kemajuan proyek, mengidentifikasi dan menangani resiko, memperbarui jadwal proyek secara berkala dan mengelola sumber daya manusia dan aktivitas manajemen proyek (delegasi dan evaluasi).

   2) *Configuration Management*, mengidentifikasi semua dokumen perangkat lunak, *deliverable*, dan artefak yang akan diproduksi, dimanipulasi, dan dipelihara selama pengembangan dan pemeliharaan perangkat lunak, serta menangani pengelolaan sumber daya perangkat lunak.

   3) *Quality Assurance,* memeriksa *deliverable* setiap fase dan memastikan penggunaannya, kesesuaiannya dengan standar internal dan eksternal.

   4) *Documentation,* dapat berupa dokumen teknis internal yang dibutuhkan untuk aktivitas pemeliharaan perangkat lunak di masa depan dan dokumen lain yang menargetkan pengguna eksternal termasuk *user manual, installation manual,* dan *operation manual*.

   5) *Training* dan *Support,* adanya dukungan teknisi membantu pengembang dalam memecahkan masalah teknis yang timbul saat mengembangkan perangkat lunak.

   6) *Evaluation* dan *Testing,* melakukan aktivitas validasi dan verifikasi sepanjang tahapan pengembangan produk dimana setiap *deliverable* dari setiap fase dievaluasi dan diuji.

**Tabel 2. 14 Distribusi *Effort* Setiap Aktivitas**

| Aktivitas | % *Effort* |
|---|---|
| ***Software Development Activity*** | |
| *Requirement* | 7,5 % |
| *Specification* | 7,5 % |
| *Design* | 10 % |
| *Implementation* | 10 % |
| *Integration Testing* | 7,5 % |
| *Acceptance and Deployment* | 7,5 % |

**Tabel 2. 15 Distribusi *Effort* Setiap Aktivitas (Lanjutan)**

| ***Ongoing Life-Cycle Activity*** | |
|---|---|
| *Project management* | 8,34 % |
| *Configuration management* | 4,16 % |
| *Quality Assurance* | 8,34 % |
| *Documentation* | 4,16 % |
| *Training and Support* | 4,16 % |
| *Evaluation and Testing* | 20,84 % |

**Sumber: Saleh, 2011**

## 2.15 Peran dan Standar Gaji Pengembang

Pada sebuah proyek pengembangan sistem informasi, terdapat beberapa orang yang terlibat dengan tugas-tugas seperti pada Tabel 2.15.

**Tabel 2. 16 Peran Pengembang**

| Peran | Deskripsi |
|---|---|
| *Project Manager* | Bertanggung jawab memimpin jalannya proyek, mendefinisikan lingkup kerja, membentuk tim, mengidentifikasi stakeholder, mengidentifikasi pengambilan keputusan, mengestimasi waktu dan jadwal WBS, mengidentifikasi sumber daya, mengevaluasi kebutuhan proyek, resiko dan rencana kontingensi (Lessard, 2007). |
| *System Analyst* | Bertanggungjawab melakukan analisis terhadap sistem informasi yang akan dibangun, termasuk analisis kebutuhan (*requirement analysis*), melakukan wawancara, studi, pengamatan, analisis kasus, ataupun simulasi kemudian menyerahkannya pada *system designer* (Tantra, 2012). |
| *Programmer/Software Engineer* | Bertanggung jawab membangun perangkat lunak dengan menggunakan bahasa pemrograman terntentu berdasarkan desain sistem yang telah disusun, mengelola basis data, pengujian (*testing*), *debugging* / *troubleshooting* dan pemeliharaan (*maintenance*) (Tantra, 2012). |
| *Tester* | Bertanggung jawab melakukan pegujian sistem untuk menentukan apakah perangkat lunak yang dibangun sesuai dengan desain, menemukan kesalahan seperti *error, bugs,* maupun kesalahan logika dan menentukan tingkatan kualitas perangkat lunak sebelum diluncurkan (Tantra, 2012). |

Perhitungan biaya per aktivitas pelaksanaan proyek menggunakan standar gaji yang dikeluarkan oleh Kelly Service 2017, dikarenakan proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula ini dikerjakan pada tahun 2017 dan menggunakan *range* nilai paling bawah disesuaikan dengan pengalaman yang dimiliki oleh tim pengembang. Tabel 2.16 menjelaskan detail perhitungan gaji yang akan digunakan dalam menghitung biaya per aktivitas.

**Tabel 2. 17 Standar Gaji *Kelly Service* 2017**

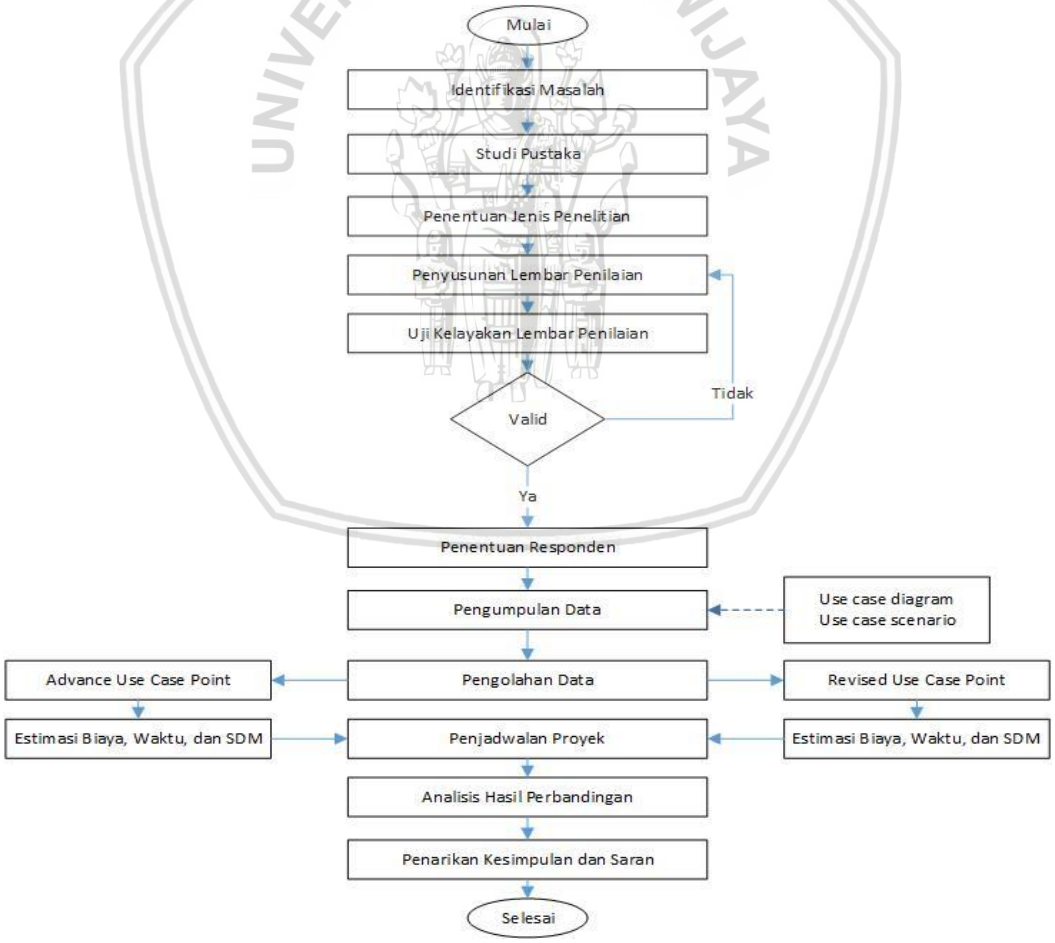
| Posisi dalam Salary Guide | Standar Gaji (per bulan) (Rp) |
|---|---|
| *Project Manager* | Rp 20.000.000,00 |
| *System Analyst* | Rp 7.000.000,00 |
| *Software Engineer* | Rp 5.000.000,00 |
| *Test Analyst* | Rp 8.000.000,00 |
| *Software QA* | Rp 8.000.000,00 |

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

# BAB 3 METODOLOGI

## 3.1 Diagram Alir Penelitian

Diagram alir penelitian menjelaskan tentang metodologi, teknik dan langkah-langkah yang digunakan untuk menyelesaikan masalah penelitian sehingga proses penelitian dapat terarah, terstruktur dan sistematis. Metodologi penelitian yang dilakukan dimulai dari mengidentifikasi masalah, studi pustaka, penentuan jenis penelitian, penyusunan lembar penilaian, lalu hasil penyusunan lembar penilaian diuji validitasnya, jika *valid* maka langkah selanjutnya menentukan responden, pengumpulan data berupa *use case diagram*, *use case scenario* dan lembar penilaian, kemudian langkah pengolahan data, perhitungan estimasi biaya, waktu, dan SDM menggunakan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP), untuk selanjutnya melakukan penjadwalan proyek, analisa hasil perbandingan estimasi biaya, waktu dan SDM dari metode *metode Advance Use Case Point* (AUCP), *Revised Use Case Point* (Re-UCP) dan perhitungan dari perusahaan, dan langkah paling akhir yaitu, menarik kesimpulan serta saran dari penelitian yang dilakukan.




**Gambar 3. 1 Diagram Alir Metodologi Penelitian**

## 3.2 Identifikasi Masalah

Langkah pertama sebelum penelitian adalah menentukan masalah penelitian. Identifikasi masalah bertujuan untuk menemukan permasalahan yang akan diteliti. Masalah dalam penelitian merupakan kesenjangan antara apa yang diharapkan terjadi dengan kenyataan yang terjadi atau antara tujuan dan pecapaian. Masalah penelitian diperoleh dari interaksi langsung dengan pihak CV Gumcode Indonesia dengan tujuan memahami objek penelitian dan permasalahan yang terjadi. Masalah pada penelitian ini bersumber dari hasil teknik pengumpulan data, yaitu wawancara.

Dalam mengidentifikasi masalah, wawancara dilakukan untuk menemukan permasalahan yang akan diteliti dengan cara mengajukan pertanyaan-pertanyaan kepada pihak yang bertugas mengatur jalannya proyek, yaitu kepada Manajer Proyek. Data yang diperoleh dari wawancara, antara lain: bagaimana dan apa saja tahapan proyek pengembangan perangkat lunak, siapa yang bertanggungjawab pada tiap tahap, masalah yang dialami saat proyek pengembangan perangkat lunak, bagaimana cara mengalokasikan biaya, waktu dan SDM, serta gambaran sistem perangkat lunak untuk penelitian.

## 3.3 Studi Pustaka

Studi pustaka bertujuan untuk menambah informasi terhadap masalah yang dipilih, mengembangkan pola penelitian, menghubungkan hasil studi ilmu pengetahuan sebelumnya dan saran untuk penelitian ke depan. Studi pustaka dapat berasal dari berbagai sumber, antara lain: jurnal profesional, laporan, buku-buku, disertasi, sumber eletronik dan penelitian terdahulu yang serupa dengan penelitian yang akan dilakukan. Informasi yang dibutuhkan antara lain, mengenai manajemen proyek, *Project Life Cycle* (PLC), *Software Development Life Cycle* (SDLC), *Work Breakdown Structure* (WBS), *Gantt Chart, Advance Use Case Point,* dan *Revised Use Case Point* sebagai dasar dalam melakukan penelitian ini.

## 3.4 Penentuan Jenis Penelitian

Berdasarkan teknik pengumpulan data yang digunakan, yatu wawancara, observasi dan lembar penilaian, penelitian ini dikategorikan ke dalam jenis penelitian kualitatif. Penelitian kualitatif merupakan penelitian yang menyelidiki fenomena dalam konteks kehidupan nyata tertentu secara mendalam dan mendeskripsikannya dalam bentuk kata-kata atau teks, yang memerlukan instrumen khusus untuk mengukur aspek-aspek dari fenomena tersebut (Recker,2013). Wawancara dan observasi diperlukan untuk mengetahui permasalahan dan gambaran dari sistem yang akan digunakan dalam pembuatan *user case diagram* dan *use case scenario*, dan lembar penilaian nantinya akan dihitung menggunakan persamaan pada metode penelitian yang digunakan, yaitu *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP). Hasil dari pengolahan data tersebut digunakan dalam proses pengendalian berupa evaluasi proyek yang disebut *postmortem review* dimana salah satu fokus dari *review* ini adalah meninjau ulang lingkup proyek, biaya, jadwal, dan sasaran kualitas.

Penelitian ini juga menganalisis beberapa proses pada area pengetahuan PMBOK (*Project Management Body of Knowledge*) yaitu: *Project Cost Management*, *Project Schedule Management*, dan *Project Resource Management* dengan mengungkap keberadaan proses-proses di dalamnya. Standar PMBOK bertujuan menjadi panduan utama dalam pelaksanaan proyek, dimana PMBOK® *Guide* dapat digunakan sebagai sumber dalam mengelola sebuah proyek sambil mempertimbangkan keseluruhan pendekatan dan metodologi yang harus diikuti untuk proyek tersebut (PMI, Inc., 2013). Artinya, manajer proyek dan tim pengembang harus secara hati-hati menangani setiap proses, masukan serta keluarannya dan menentukan mana yang sesuai dengan proyek yang sedang mereka kerjakan. Metode *Advance Use Case Point* dan *Revised Use Case Point* menghasilkan keluaran berupa ukuran perangkat lunak yang dikonversi menjadi durasi dan biaya yang dibutuhkan untuk pengembangan sistem, serta penggunaan *Guideline* Distribusi *Effort* menghasilkan analisis jumlah pengembang. Berdasarkan hal tersebut, area pengetahuan yang akan dideskripsikan pada penelitian ini, antara lain: 1) *Project Cost Management*; 2) *Project Schedule Management*; dan 3) *Project Resource Management*; karena ketiga area pengetahuan ini sesuai dengan keluaran dari metode estimasi biaya perangkat lunak yang digunakan.

## 3.5 Penyusunan Instrumen Penelitian

Pada tahap ini, peneliti menyusun kisi-kisi instrumen penelitian untuk mengumpulkan data. Instrumen penelitian diperlukan untuk mengukur nilai variabel yang diteliti sehingga jumlah instrumen yang digunakan dalam penelitian akan tergantung pada jumlah variable yang diteliti. Dalam penelitian ini, variable yang diteliti adalah *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) untuk metode *Revised Use Case Point* (Re-UCP), dan *Technical Complexity Factor* (TCF), *Environmental Complexity Factor* (ECF) serta *End-user Development Factor* (EUDF) untuk metode *Advance Use Case Point* (AUCP). Untuk memudahkan penyusunan instrumen penelitian, peneliti menentukan indikator yang akan diukur, kemudian peneliti menjabarkan indikator ini ke dalam bentuk butir-butir pertanyaan atau pernyataan. Bentuk instrumen penelitian berupa pertanyaan wawancara dan lembar penilaian. Pertanyaan wawancara digunakan untuk menggali informasi terkait proyek Sistem Reservasi Online PT GHL Peninsula dan proses-proses pada PMBOK (*Project Management Body of Knowledge*) yang dilakukan, sedangkan lembar penelitian untuk mendapatkan data mengenai variable TCF, ECF, dan EUDF.

## 3.6 Pengujian Kelayakan Instrumen Penelitian

Tahap selanjutnya setelah pertanyaan wawancara dan lembar penelitian selesai dibuat adalah peneliti melakukan uji validitas pada kedua intrumen penelitian tersebut. Tujuan menguji validitas pertanyaan wawancara dan lembar penelitian untuk memastikan instrumen tersebut dapat digunakan untuk mengukur apa yang seharusnya diukur.

Uji kelayakan dilakukan oleh dosen ahli sebagai *expert judgment*, yang dalam penelitian ini memohon kesediaan Bapak Mochamad Chandra Saputra, S.Kom., M.Eng dan Bapak Admaja Dwi Herlambang, S.Pd., M.Pd untuk melakukan penilaian terhadap pertanyaan wawancara dan lembar penilaian. Penentuan dosen ahli dalam penelitian ini merupakan hasil pertimbangan bahwa dosen ahli telah berpengalaman dalam proyek pengembangan perangkat lunak dan pernah menjadi pengajar pada mata kuliah Manajemen Proyek Sistem Informasi (MPSI).

Uji validitas lembar penilaian dilakukan sebelum penyebaran lembar penilaian. Penelitian ini menggunakan 2 jenis uji validitas, yaitu:

1. Validitas muka (*face validity*), untuk memastikan pertanyaan atau pernyataan dalam lembar penilaian telah dianggap relevan, masuk akal atau beralasan, tidak ambigu dan jelas yang didasarkan pada intuisi dari ahli pada bidang tersebut.
2. Validitas isi (*content validity*), validitas ini mengutamakan isi yang melibatkan beberapa ahli (*expert*) untuk menilai apakah pertanyaan atau pernyataan pada lembar penilaian telah mempresentasikan semua komponen-komponen variabel penelitian (Swarjana, 2016).

Setiap pernyataan atau pertanyaan tersebut diuji apakah sudah mewakili setiap indikator dan layak untuk digunakan dalam pengambilan data. Apabila lembar penilaian dinilai belum layak, maka dilakukan perbaikan atau penyusunan ulang pertanyaan wawancara dan lembar penilaian. Apabila telah layak, maka akan diserahkan kepada responden untuk proses pengambilan data.

## 3.7 Penentuan Responden

Penentuan responden bertujuan untuk mengetahui kepada siapa instrumen penelitian diajukan. Pada penelitian kualitatif, penentuan responden dapat menggunakan salah satu teknik *non-probability sampling*, yaitu *purposive sampling*. *Purposive sampling* melakukan pengambilan data dengan pertimbangan tertentu seperti, orang tersebut dianggap yang paling mengetahui tentang apa yang peneliti harapkan atau orang itu adalah pemimpin sehingga akan memudahkan peneliti mengeksplorasi objek yang diteliti (Sugiyono, 2011). Berdasarkan pertimbangan informasi tersebut responden penelitian merupakan orang-orang yang terlibat dan mengerti bagaimana proyek pengembangan perangkat lunak berjalan. Pemangku kepentingan (*stakeholder*) dari proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula adalah orang-orang yang terlibat atau terpengaruh oleh kegiatan proyek, termasuk sponsor proyek (*project sponsor)*, tim proyek (*project team)*, pengembang pendukung (*support pengembangf)*, pelanggan (*customer),* pengguna (*user),* pemasok (*supplier)*, dan bahkan penentang proyek. Dari *stakeholder* tersebut, yang bertanggungjawab dalam melaksanakan proyek adalah tim proyek. Tim proyek terdiri atas orang-orang yang memiliki keahlian yang sesuai yang diperlukan dalam pelaksanaan proyek.

Responden penelitian yang digunakan untuk menghimpun data penelitian pada lembar penilaian *Technical Complexity Factor* dan *Environmental Complexity Factor*, yaitu: manajer proyek yang memimpin tim proyek yang diasumsikan manajer proyek mengetahui segala hal tentang proyek dan tim proyek, sedangkan lembar penilaian *End-user Development Factor* (EUDF) dinilai oleh seorang analis sistem dari CV Gumcode Indonesia yang menjadi perantara pertama saat proses *offering* dan penggalian kebutuhan awal dengan klien. Analis sistem awal ini berperan sebagai *end-user* yang mengetahui apa saja yang dibutuhkan oleh sistem dan mengetahui informasi sesuai kesepakatan dengan klien.

## 3.8 Pengumpulan Data

Pengumpulan data penelitian dilakukan untuk memperoleh data-data dan informasi yang berkaitan dengan manajemen proyek yang dijalankan oleh CV Gumcode Indonesia dan Sistem Reservasi Online Gedung Serbaguna PT Peninsula, yang akan dibutuhkan selama proses penelitian. Penelitian ini menggabungkan antara data kualitatif dan data kuantitatif. Data kualitatif merupakan kumpulan informasi deskriptif yang diperoleh dari percakapan atau dalam bentuk naratif berupa kata-kata. Data kuantitatif merupakan data yang berhubungan dengan nilai dalam bentuk angka yang dapat dianalisis atau diolah menggunakan teknik perhitungan matematika atau statisika (Yaumi & Damopolii, 2014). Data kualitatif diperoleh dari transkrip wawancara yang dilakukan di tahap pertama ketika peneliti mengidentifikasi masalah, dan dilanjutkan dengan menggali data tentang gambaran Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang nantinya akan dijadikan acuan dalam proses pembuatan *use case diagram* dan *use case scenario*, serta data terkait proses-proses pada PMBOK yang dilakukan oleh pengembang. Data kuantitatif diperoleh dari lembar penilaian yang diisi oleh manajer proyek dan sistem analis awal yang selanjutnya diolah atau dianalisis menggunakan persamaan perhitungan matematika yang ada pada metode penelitian yang digunakan.

### 3.8.1 Wawancara

Pada penelitian ini, peneliti membutuhkan kesediaan dari manajer proyek yang menangani proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula melalui tanya jawab secara lisan dan bertatap muka. Pada tahap identifikasi masalah, peneliti melakukan teknik wawancara tidak terstruktur yang tidak menggunakan pedoman wawancara secara sistematis dan sering digunakan untuk mendapatkan informasi awal tentang permasalahan pada objek dalam penelitian pendahuluan (Sugiyono, 2011). Wawancara dilakukan dengan mengajukan beberapa pertanyaan untuk menggali masalah yang dialami selama pengerjaan proyek pengembangan perangkat lunak.

Wawancara selanjutnya dilakukan dengan teknik wawancara semi-terstruktur yang menggunakan garis besar pedoman wawancara, namun pelaksanaannya lebih bebas daripada wawancara terstruktur (Sugiyono, 2011). Wawancara ini bertujuan untuk mengetahui gambaran sistem dan analisis kebutuhan fungsional sistem yang nantinya digunakan sebagai acuan dalam

pembuatan *use case diagram* dan *use case scenario*, serta bagaimana cara manajer proyek menentukan biaya pengembangan perangkat lunak dan mengalokasikan waktu dan SDM dalam perencanaan proyek pada CV Gumcode Indonesia. Selain itu, wawancara dilakukan untuk menggali informasi terkait proses-proses PMBOK yang dilakukan oleh CV Gumcode Indonesia saat mengembangkan proyek perangkat lunak.

### 3.8.2 Observasi

Observasi atau pengamatan merupakan salah satu cara yang digunakan untuk mempelajari suatu sistem, dilakukan dengan pengamatan langsung pada suatu kegiatan yang sedang dilakukan (Jogiyanto, 1989). Peneliti melakukan observasi partisipatif pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula dan dokumen *Project Charter*, dimana dalam observasi partisipatif ini, peneliti terlibat dengan kegiatan sehari-hari objek yang diteliti atau sumber data penelitian (Sugiyono, 2011). Peneliti melakukan pengamatan sambil ikut merasakan atau mencoba objek yang diteliti. Peneliti berperan sebagai *user* yang dapat mengakses perangkat lunak sebagai *customer*, vendor maupun admin, sehingga peneliti dapat mencoba langsung fungsi dan fitur di dalam perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Sedangkan observasi pada dokumen *Project Charter* bertujuan untuk memperoleh referensi dalam melengkapi pemodelan *use case diagram*, WBS, dan *timeline* waktu yang dibutuhkan dalam pengembangan perangkat lunak.

### 3.8.3 Lembar Penilaian

Lembar penilaian pada penelitian ini digunakan untuk mengambil data yang diperlukan dalamperhitungan *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) untuk metode *Revised Use Case Point* (Re-UCP), dan *Technical Complexity Factor* (TCF), *Environmental Complexity Factor* (ECF) serta *End-user Development Factor* (EUDF) untuk metode *Advance Use Case Point* (AUCP). Lembar penilaian yang telah teruji kelayakannya kemudian ditujukan kepada manajer proyek perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula dan analis sistem perantara klien. Lembar penilaian TCF dan ECF dinilai oleh manajer proyek, sedangkan lembar penilaian EUD_TCF dan EUD_ECF dinilai oleh analis sistem perantara klien. Selama proses pengisian lembar penilaian, peneliti mendampingi manajer proyek dan perwakilan klien yaitu, analis sistem, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk memastikan jawaban manajer proyek dan analis sistem perantara klien telah sesuai dengan indikator pertanyaan yang ingin diukur.

Peneliti mengolah skor pada setiap lembar penilaian sesuai persamaan dari setiap metode. Pada metode *Revised Use Case Point* (Re-UCP), skor pada setiap indikator dikalikan dengan bobot dari faktor dan dijumlahkan menghasilkan nilai TF untuk lembar penilaian *Technical Complexity Factor* (TCF), dan nilai EF untuk lembar penilaian *Environmental Complexity Factor* (ECF). Pada metode *Advance Use Case Point* (AUCP), skor pada setiap indikator dikalikan dengan bobot dari

faktor dan dijumlahkan menghasilkan nilai TF untuk lembar penilaian *Technical Complexity Factor* (TCF), nilai EF untuk lembar penilaian *Environmental Complexity Factor* (ECF), nilai EUD_TF untuk lembar penilaian *End-User Development Technical Complexity Factor* (EUD_TCF), dan nilai EUD_EF untuk lembar penilaian *End-User Development Environment Complexity Factor* (EUD_TCF).

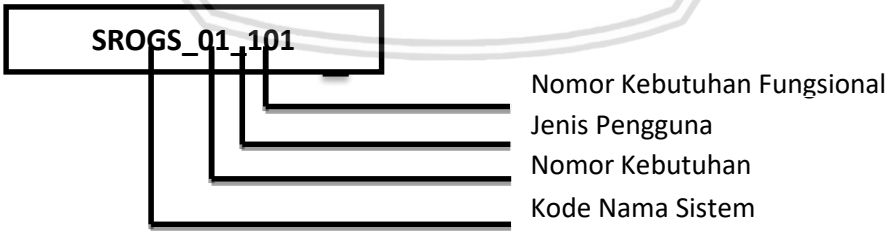
## 3.9 Pengolahan Data

Data-data yang diperoleh dari pengumpulan data, selanjutnya diolah menggunakan teknik perhitungan yang ada di dalam metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP) untuk mengasilkan estimasi biaya, waktu dan SDM yang diperlukan dalam proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Berdasarkan hasil wawancara yang dilakukan kepada Manajer Proyek mengenai proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, didapatkan data kualitatif berupa kebutuhan dari PT GHL Peninsula dan kebutuhan fungsional dari sistem yang dikembangkan untuk memodelkan *use case diagram* dan *use case scenario*, informasi mengenai manajemen proyek dan alokasi waktu dan sumber daya manusia pada penjadwalan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.

**Aturan Penomoran Kebutuhan Sistem**

Penulisan kebutuhan sistem menggunakan aturan penomoran tertentu berupa kode sebagai identitas kebutuhan fungsional sistem. Contoh aturan penomoran yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.2 dan Gambar 3.3.

**Gambar 3. 2 Aturan Penomoran Pernyataan Kebutuhan**




**Gambar 3. 3 Aturan Penomoran Kebutuhan Fungsional**

Pada aturan penomoran pertama terdiri dari dua bagian. Kode kebutuhan dalam aturan penomoran ini adalah RS yang merupakan akronim dari *Requirement Statement*. Nomor kebutuhan dimulai dari satu atau 01. Aturan penomoran pertama ini digunakan untuk pernyataan kebutuhan.

Pada aturan penomoran kedua terdiri dari empat bagian. Kode nama sistem dalam aturan penomoran ini adalah SROGS yang merupakan akronim dari Sistem

Reservasi Online Gedung Serbaguna PT GHL Peninsula. Nomor Kebutuhan mengacu kepada nomor kebutuhan pada aturan penomoran pertama. Jenis Pengguna terdiri atas 4 jenis, yaitu 0 untuk *Guest*, 1 untuk *Customer*, 2 untuk Vendor dan 3 untuk Admin. Nomor kebutuhan fungsional merupakan nomor urut kebutuhan fungsional sistem dimulai dari angka 1. Aturan penomoran ini digunakan untuk mendefinisikan pernyataan kebutuhan dan kebutuhan fungsional sistem yang terlampir pada Lampiran E.

### *3.9.1* Perhitungan *Advance Use Case Point*

Alur perhitungan metode *Advance Use Case Point* (AUCP) adalah sebagai berikut:

**Tabel 3. 1 Alur Metode *Advance Use Case Point* (AUCP)**

| Tahap | *Input* | Proses | *Output* |
|---|---|---|---|
| 1 | *Use Case Diagram* dan *Use Case Scenario* | 1. Menghitung *Unadjusted Actor Weight* (UAW)<br>2. Menghitung *Unadjusted Use-Case Weight* (UUCW) | Nilai untuk menghitung *Unadjusted Use Case Point* (UUCP) |
| 2. | 1. Nilai *Unadjusted Actor Weight* (UAW)<br>2. Nilai *Unadjusted Use-Case Weight* (UUCW) | Menghitung *Unadjusted Use Case Point* (UUCP) dengan formula UUCP = UAW + UUCW | Nilai *Unadjusted Use Case Point* (UUCP) |
| 3. | Lembar Penilaian *Technical Complexity Factor* (TCF) | Menghitung hasil lembar penilaian *Distributed system required, Response time is important, End-user efficiency, Complex internal processing required, Reusable code must be a focus , Installation easy, Usability, Cross-platform support, Easy to change, Highly concurrent, Custom security, Dependent On third-part code,* dan *user training.* | Nilai *Technical Factor* (TF) |
| 4. | Lembar Penilaian *Environment Complexity Factor* (ECF) | Menghitung hasil lembar penilaian *Familiarity with the objectory, Application experience, Object Oriented experience, Analyst capability, Motivation, Stable requirements, Part time worker,* dan *Difficult programming language.* | Nilai *Environment Factor* (EF) |
| 5. | Nilai *Technical Factor* (TF) dan nilai *Environment Factor* (EF) | Menghitung *Complexity Factor* yaitu: *Technical Complexity Factor* (TCF) dan | Nilai *Complexity Factor* yaitu: TCF dan ECF |

| Tahap | *Input* | Proses | *Output* |
|---|---|---|---|
| | | *Environment Complexity Factor* (ECF) | |
| 6. | Nilai *Unadjusted Use Case Point* (UUCP), *Technical Complexity Factor* (TCF) dan *Environment Complexity Factor* (ECF) | Menghitung nilai *Use Case Point* (UCP) dengan formula UCP = UUCP * TCF * ECF | Nilai *Use Case Point* (UCP) |
| 7. | Lembar Penilaian *End-user Development Technical Complexity Factor* (EUD_TCF) | Menghitung hasil lembar penilaian *Creating throw away codes, Creating reusable codes, Sharing reusable code, Easy & understandable code, Security features in codes for more control by end user, Authentication features, Inbuilt feedback about the correctness, Testable codes, Tools for analyzing by debugging, Error detection tools, Online help availability, Self-efficiency, Perceived ease of use, Perseived usefulness, Flexible codes, Scalability features,* dan *Ease of maintenance*. | Nilai *End User Development Technical Factor* (EUD_TF) |
| 8. | Lembar Penilaian *End-user Development Enviromental Complexity Factor* (EUD_ECF) | Menghitung hasil lembar penilaian *Content level of EUP, End-user computing capability, Easeof use & feedback, Inbuilt system assistance for EUP, Training & learning time constraint for end user, Reliability of end user code, End user storage constraint,* dan *Risk factor*. | Nilai *End User Development Environmental Factor* (EUD_EF) |
| 9. | Nilai EUD_TF dan EUD_EF | Menghitung *EUD Technical Complexity Factor* (EUD_TCF) dan *EUD Environmental Complexity Factor* (EUD_ECF) | Nilai EUD_TCF dan EUD_ECF |
| 10. | Nilai UCP, EUD_TCF dan EUD_ECF | Menghitung nilai *Advance Use Case Point* (AUCP) dengan formula AUCP = UCP * (EUD_TCF * EUD_ECF) | Nilai *Advance Use Case Point* (AUCP) |

**Tabel 3. 2 Alur Metode *Advance Use Case Point* (AUCP) (Lanjutan)**

| Tahap | *Input* | Proses | *Output* |
|---|---|---|---|
| 11. | Nilai *Advance Use Case Point* (AUCP) | Menghitung nilai *hours of effort* | Nilai *hours of effort* |

Keterangan Tabel 3.1 :

1. Melakukan analisis terhadap *use case diagram* dan *use case scenario* untuk mengklasifikasikan aktor dan *use case* berdasarkan kompleksitasnya, kemudian menghitung bobot pada masing-masing aktor dan *use case* untuk menghasilkan nilai *Unadjusted Actor Weight* (UAW) dan *Unadjusted Use Case Weight* (UUCW).
2. Menggunakan hasil perhitungan *Unadjusted Actor Weight* (UAW) dan *Unadjusted Use Case Weight* (UUCW) untuk menghitung nilai *Unadjusted Use Case Point* (UUCP).
3. Mengumpulkan nilai untuk lembar penilaian *Technical Complexity Factor* (TCF) yang terdiri dari 13 faktor teknis. Lembar penilaian TCF dinilai oleh manajer proyek
4. Mengumpulkan nilai untuk lembar penilaian *Environmental Complexity Factor* (ECF) yang terdiri dari 8 faktor lingkungan. Lembar penilaian ECF dinilai oleh manajer proyek.
5. Menghitung nilai bobot pada lembar penilaian *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) dengan cara mengkalikan sesuai bobot masing-masing faktor kemudian menjumlahkannya dan menghitung sesuai persamaan, sehingga diperoleh nilai TCF dan ECF.
6. Nilai *Unadjusted Use Case Point* (UUCP), *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) yang diperoleh sebelumnya digunakan untuk menghitung nilai Use Case Point (UCP) sesuai persamaan.
7. Mengumpulkan nilai untuk lembar penilaian *End-User Development Technical Complexity Factor* (EUD_TCF) yang terdiri dari 17 faktor. Lembar penilaian EUD_TCF dinilai oleh sistem analis awal yang menjadi perantara antara klien dan tim pengembang
8. Mengumpulkan nilai untuk lembar penilaian *End-User Development Enviromental Complexity Factor* (EUD_ECF) yang terdiri dari 8 faktor. Lembar penilaian EUD_ECF dinilai oleh sistem analis awal yang menjadi perantara antara klien dan tim pengembang.
9. Menghitung nilai bobot pada lembar penilaian *End-User Development Technical Complexity Factor* (EUD_TCF) dan *End-User Development Enviromental Complexity Factor* (EUD_ECF) dengan cara mengkalikan sesuai bobot masing-masing faktor kemudian menjumlahkannya dan menghitung sesuai persamaan.
10. Nilai *Use Case Point* (UCP), *End-User Development Technical Complexity Factor* (EUD_TCF) dan *End-User Development Enviromental Complexity Factor*

(EUD_ECF) yang diperoleh sebelumnya digunakan untuk menghitung nilai *Advance Use Case Point* (AUCP) sesuai dengan persamaan.

11. Menghitung nilai *hours of effort* atau total waktu kerja yang dibutuhkan untuk menyelesaikan pengembangan perangkat lunak dengan cara mengkalikan nilai *Advance Use Case Point* (AUCP) dengan 20 *person-hour per-UCP* sesuai Teori Karner (1993).

### *3.9.2* Perhitungan *Revised Use Case Point*

Alur perhitungan metode *Revised Use Case Point* (Re-UCP) dijelaskan sebagai berikut:

**Tabel 3. 3 Alur metode *Revised Use Case Point***

| Tahap | Input | Proses | Output |
|---|---|---|---|
| 1 | *Use Case Diagram* dan *Use Case Scenario* | 1. Menghitung *Unadjusted Actor Weight* (UAW)<br>2. Menghitung *Unadjusted Use-Case Weight* (UUCW) | Nilai untuk menghitung *Unadjusted Use Case Point* (UUCP) |
| 2. | 1. Nilai *Unadjusted Actor Weight* (UAW)<br>2. Nilai *Unadjusted Use-Case Weight* (UUCW) | Menghitung *Unadjusted Use Case Point* (UUCP) dengan formula UUCP = UAW + UUCW | Nilai *Unadjusted Use Case Point* (UUCP) |
| 3. | Lembar Penilaian *Technical Complexity Factor* (TCF) | Menghitung hasil lembar penilaian *Distributed system, Response or throughput performance objectives, End-user efficiency, Complex internal processing, Code must be reusable, Easy to install, Easy to use, Portable, Easy to change, Concurrent, Special security features, Direct access for third parties, Special user training,* dan *Scalability*. | Nilai *Technical Factor* (TF) |
| 4. | Lembar Penilaian *Environment Complexity Factor* (ECF) | Menghitung hasil lembar penilaian *Familiarity with the project, Application experience, OO programming experience, Lead analyst capability, Motivation, Stable requirements, Part time pengembangf, Difficult programming language,* dan *Project methodology.* | Nilai *Environment Factor* (EF) |
| 5. | Nilai *Technical Factor* (TF) dan nilai *Environment Factor* (EF) | Menghitung *Technical Complexity Factor* (TCF) dan *Environment Complexity Factor* (ECF) | Nilai *Technical Complexity Factor* (TCF) dan *Environment Complexity Factor* (ECF) |
| 6. | Nilai *Unadjusted Use Case Point* (UUCP), *Technical Complexity Factor* (TCF) dan | Menghitung nilai *Revised Use Case Point* (Re-UCP) dengan formula Re-UCP = UUCP * TCF * ECF | Nilai *Revised Use Case Point* (Re-UCP) |

**Tabel 3. 4 Alur metode *Revised Use Case Point* (Lanjutan)**

| Tahap | Input | Proses | Output |
|---|---|---|---|
| | *Environment Complexity Factor* (ECF) | | |
| 7. | Nilai *Revised Use Case Point* (Re-UCP) | Menghitung nilai *hours of effort* | Nilai *hours of effort* |

Keterangan Tabel 3.3 :

1. Melakukan analisis terhadap *use case diagram* dan *use case scenario* untuk mengklasifikasikan aktor dan *use case* berdasarkan kompleksitasnya, kemudian menghitung bobot pada masing-masing aktor dan *use case* untuk menghasilkan nilai *Unadjusted Actor Weight* (UAW) dan *Unadjusted Use Case Weight* (UUCW).
2. Menggunakan hasil perhitungan *Unadjusted Actor Weight* (UAW) dan *Unadjusted Use Case Weight* (UUCW) untuk menghitung nilai *Unadjusted Use Case Point* (UUCP).
3. Mengumpulkan nilai untuk lembar penilaian *Technical Complexity Factor* (TCF) yang terdiri dari 14 faktor teknis. Lembar penilaian TCF dinilai oleh manajer proyek.
4. Mengumpulkan nilai untuk lembar penilaian *Environmental Complexity Factor* (ECF) yang terdiri dari 9 faktor lingkungan. Lembar penilaian ECF dinilai oleh manajer proyek.
5. Menghitung nilai bobot pada lembar penilaian *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) dengan cara mengkalikan sesuai bobot masing-masing faktor kemudian menjumlahkannya dan menghitung sesuai persamaan, sehingga diperoleh nilai TCF dan ECF.
6. Nilai *Unadjusted Use Case Point* (UUCP), *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) yang diperoleh sebelumnya digunakan untuk menghitung nilai Use Case Point (UCP) sesuai persamaan.
7. Menghitung nilai *hours of effort* atau total waktu kerja yang dibutuhkan untuk menyelesaikan pengembangan perangkat lunak dengan cara mengkalikan nilai *Revised Use Case Point* (Re-UCP) dengan 20 *person-hour per-UCP* sesuai Teori Karner (1993).

### *3.9.3* Konversi *Advance Use Case Point* dan *Revised Use Case Point ke Dalam Biaya, Waktu, dan SDM*

Konversi hasil perhitungan dari metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point*(Re-UCP) dilakukan untuk menghasilkan nilai estimasi biaya, waktu dan sumber daya manusia yang dibutuhan dalam menyelesaikan pengerjaan proyek pengembangan perangkat lunak. Hasil dari perhitungan kedua metode tersebut masih berupa nilai ukuran perangkat lunak (*software size)*, sehingga dibutuhkan proses konversi ke dalam *hours of effort* untuk selanjutnya dilakukan perhitungan total biaya, waktu pengembangan dan jumlah tenaga kerja.

Pengukuran ukuran perangkat lunak (*software sizing*) merupakan kegiatan penting bertujuan untuk merencanakan dan memantau pengembangan proyek perangkat lunak yang berfokus pada waktu dan anggaran. Ukuran perangkat lunak sangat penting dalam menentukan dan mengendalikan biaya, jadwal, kualitas dan produktivitas dalam manajemen proyek perangkat lunak. Untuk menghasilkan ukuran perangkat lunak diperlukan *software metric* yang memiliki *metric* untuk memprediksi *effort* yang diperlukan untuk implementasi dan pengujian kode. *Software metric* didefinisikan sebagai sebuah angka yang diperoleh dari sebuah proses atau produk dari proyek perangkat lunak.

*Software metric* untuk mengukur ukuran perangkat lunak yang penelitian ini adalah *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP). Kedua metode tersebut menghitung estimasi ukuran perangkat lunak yang selanjutnya dilakukan estimasi *effort*, durasi, biaya, dan sumber daya manusia. Untuk mengkonversi estimasi ukuran perangkat lunak menjadi *effort*, durasi, biaya dan jumlah sumber daya manusia diperlukan persamaan *effort*, tabel distribusi *effort* per fase, tabel distribusi *effort* tiap anggota tim pengembang dan tabel gaji per bulan tiap anggota tim pengembang. Langkah-langkah dalam mengestimasi biaya, waktu dan sumber daya manusia dijelaskan sebagai berikut:

1. Mengestimasi *size software* atau ukuran perangkat lunak.
2. Mengkonversi *size* menjadi *hours of effort* menggunakan persamaan *effort* yang mengkalikan *size* yang diperoleh dengan 20 *person-hours* per UCP sesuai yang disarankan oleh Karner (1993).
3. Mengkonversi *hours of effort* menjadi durasi dengan menggunakan tabel distribusi *effort* per-fase.
4. Dari *effort* yang diperoleh, membuat tabel distribusi *effort* pada setiap anggota tim.
5. Dari *effort* tiap anggota tim, menggunakan tabel gaji pengembang per bulan untuk menghitung nilai estimasi biaya total.

## 3.10 Penjadwalan Proyek

Penjadwalan dilakukan untuk memberikan informasi mengenai jadwal rencana dan kemajuan proyek terkait kinerja sumber daya berupa biaya, rencana durasi proyek, *progress* waktu dan tenaga kerja untuk menyelesaikan proyek. Penjadwalan proyek dapat dilakukan setelah mengetahui estimasi waku dan sumber daya untuk menyelesaikan aktivitas. Untuk melakukan penjadwalan, tahap pertama yang harus dilakukan adalah membuat *Work Breakdown Structure* (WBS). Dalam WBS, peneliti mendekomposisi lingkup proyek dari komponen-komponen utama menjadi satuan-satuan kecil *deliverable* disebut *work package*. WBS membantu peneliti untuk memberikan alokasi waktu untuk menyelesaikan lingkup kerja dan mecapai *milestone* dalam setiap fase. Masing-masing fase yang telah memiliki alokasi waktu kemudian dijadwalkan untuk dilaksanakan secara urut. Peneliti menggunakan alat bantu untuk menyusun jadwal secara rinci, alat bantu tersebut adalah *Gantt chart*. *Gantt chart* dapat menunjukkan urutan

kegiatan, serta berguna untuk melacak dan memantau kemajuan sebuah proyek. *Microsoft Project* digunakan untuk membuat *Gantt chart* dengan menyusun penjadwalan dan pengaturan sumber daya manajemen proyek.

## 3.11 Analisis Hasil Perbandingan

Pada tahap ini, peneliti melakukan analisis hasil perbandingan hasil estimasi biaya, waktu dan sumber daya manusia dari metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP), dengan hasil estimasi dari pihak CV Gumcode Indonesia. Analisis perbandingan hasil bertujuan untuk melihat kesenjangan pada setiap hasil, mengetahui faktor-faktor yang mempengaruhi perbedaan pada setiap hasil, dan dampaknya bagi hasil yang diperoleh, yang nantinya disimpulkan dan digunakan sebagai informasi bagi CV Gumcode Indonesia dalam menentukan metode terbaik untuk estimasi biaya, waktu dan sumber daya manusia yang dibutuhkan di masa mendatang.

Manajer proyek yang mengembangkan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula mengestimasi biaya berdasarkan jumlah fitur, jenis *platform* yang dibutuhkan, dan perkiraan kesulitan fungsi pada sistem dengan mengingat pengalaman pada proyek sejenis sebelumnya, kemudian menentukan jumlah tim pengembang yang dapat mengerjakan proyek dan menentukan waktu dengan menjumlahkan perkiraan waktu pada masing-masing fitur apabila dikerjakan oleh sumber daya manusia, maka kemungkinan dapat selesai dalam waktu berapa lama.

## 3.12 Penarikan Kesimpulan dan Saran

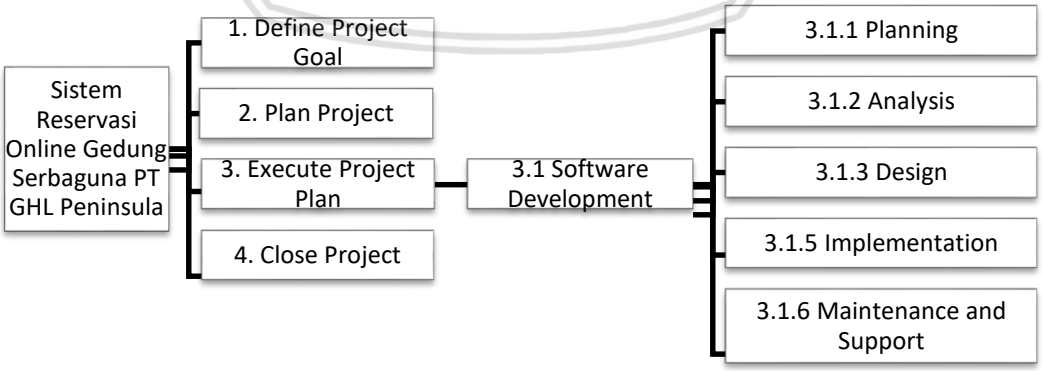
Tahap terakhir pada penelitian ini adalah penarikan kesimpulan dan saran. Kesimpulan diperoleh dari hasil pembahasan yang menjawab rumusan masalah dan memenuhi tujuan penelitian. Sedangkan saran berupa rekomendasi ditujukan kepada pihak *software house* untuk dapat dilakukan tindakan yang terstrutur dan langkah ke depan dalam upaya mencapai sasaran proyek yang lebih baik lagi. Selain itu, saran dapat berguna untuk penelitian selanjutnya.

# BAB 4 HASIL DAN ANALISIS

## 4.1 Analisis Work Breakdown Structure

*Work Breakdown Structure* (WBS) merupakan peta proyek yang digunakan untuk membantu memastikan manajer proyek bahwa seluruh lingkup kerja dan pekerjaan telah teridentifikasi, menetapkannya sebagai dasar kendali, memberikan alokasi waktu dalam menyelesaikan lingkup kerja dan menggapai milestone dalam setiap fase. WBS memfasilitasi evaluasi biaya, waktu, dan kinerja teknis di semua tingkat dalam organisasi selama masa proyek. Analisis WBS dilakukan untuk memberikan informasi kepada manajer proyek dalam mengembangkan WBS agar menghasilkan estimasi biaya, waktu dan sumber daya manusia yang lebih baik dengan cara mengidentifikasi lingkup kerja proyek yang telah digunakan selama masa proyek dan membandingkannya dengan ruang lingkup proyek yang terdapat pada teori Saleh (2011). WBS pada analisis WBS digambarkan dalam bentuk *tree structure view* dimana setiap anak (komponen) ditampilkan sebagai kotak terhubung dengan induk secara hierarki. Representasi WBS model ini sesuai untuk WBS yang memiliki dekomposisi pada setiap variabel yang terdiri dari dua sampai tiga level dan ditujukan untuk pengembangan.

Berdasarkan wawancara dengan manajer proyek yang bertanggungjawab terhadap perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Penisula, peneliti mengidentifikasi lingkup kerja proyek yang sudah berjalan. Gambar 4.1 menampilkan WBS yang sudah berjalan pada proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang terdiri dari empat level. Level pertama adalah nama sistem yang dikembangkan, yaitu Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Level kedua terdiri dari empat fase pada *Project Life Cycle* (PLC), yaitu *Define Project Goal, Plan Project, Execute Project Plan* dan *Close Project*. Level ketiga merupakan siklus pengembangan sistem (SDLC), serta level keempat terdiri dari lima fase, yaitu *Planning, Analysis, Design, Implementation*, serta *Maintenance* dan *Support*.



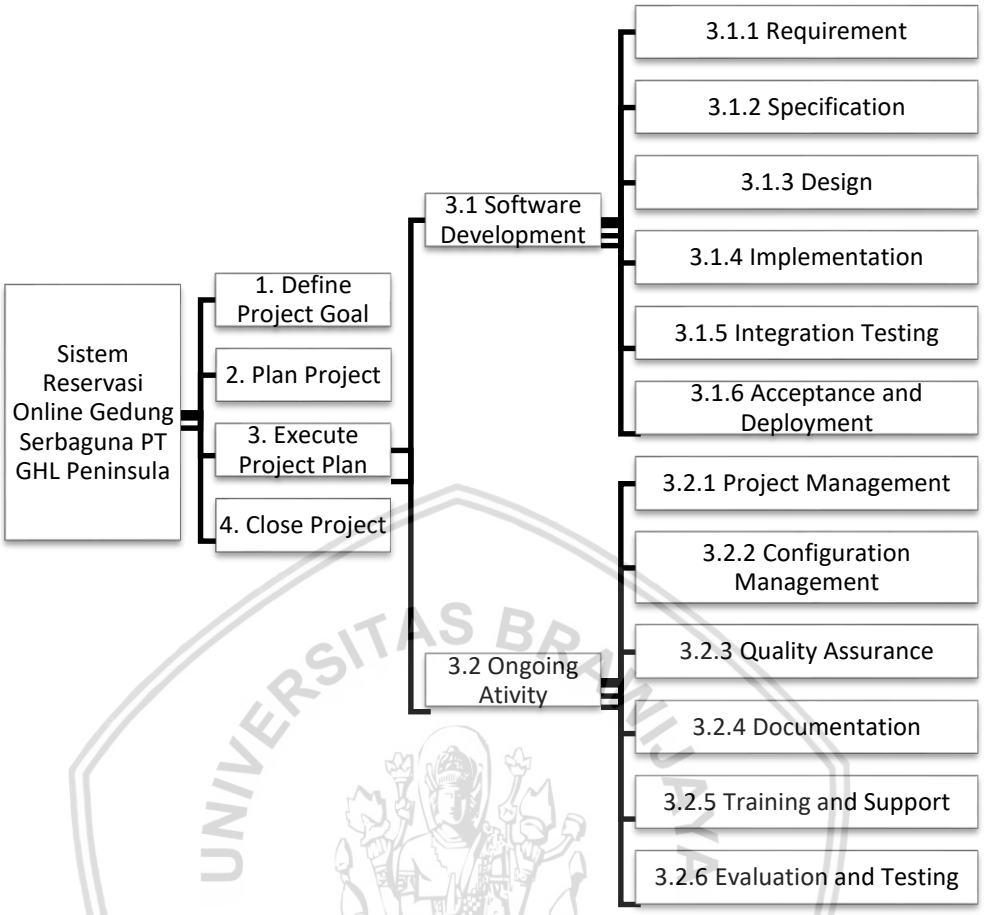

**Gambar 4. 1 *Work Breakdown Structure* CV Gumcode Indonesia**

Pada fase *Define Project Plan*, *marketing* dan analis sistem awal mengidentifikasi apa saja tujuan proyek dan menganalisis kebutuhan sistem yang akan dibangun berdasarkan permintaan klien. Fase *Plan Project*, manajer proyek

dan analis sistem menyusun *project charter*, dokumen SRS (*Software Requirement Specification*), menentukan tanggal mulai dan selesai. Fase *Execute Project Plan*, merupakan fase pengembangan sistem yang terdiri dari: 1) *Planning*, manajer proyek memastikan tujuan, lingkup kerja, anggaran, jadwal pengerjaan, teknologi yang digunakan, dan model pengembangan perangkat lunak; 2) *Analysis*, sistem analis menganalisis kebutuhan sistem, mendekomposisi aktivitas menjadi *task* dan alokasi waktu pada setiap task; 3) *Design*, sistem analis merancang arsitektur, *user interface* dan *database* di awal, kemudian pemrogram *front-end* dan *back-end* melanjutkan rancangan di saat pengembangan atau eksekusi; 4) *Implementation*, pemrogram mengerjakan koding sesuai *task* masing-masing, kemudian melakukan pengujian setelah selesai; 5) *Maintenance* dan *Support*, setelah perangkat lunak diserahterimakan kepada klien, terdapat waktu garansi jika terdapat kesalahan (*error*) atau masalah selama 3 bulan.

Pada WBS perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Penisula yang telah berjalan, tidak sesuai dengan pedoman alokasi *effort* dan biaya yang tepat sesuai yang diusulkan oleh Saleh (2011). Pedoman tersebut memberi manajer proyek informasi yang dapat digunakan untuk memperkirakan keseluruhan *effort* dan anggaran yang diperlukan, dan mengembangkan jadwal proyek secara tepat. Penggunaan pedoman ini akan mendukung perhitungan pada metode *Advance Use Case Point* dan *Revised Use Case Point*.

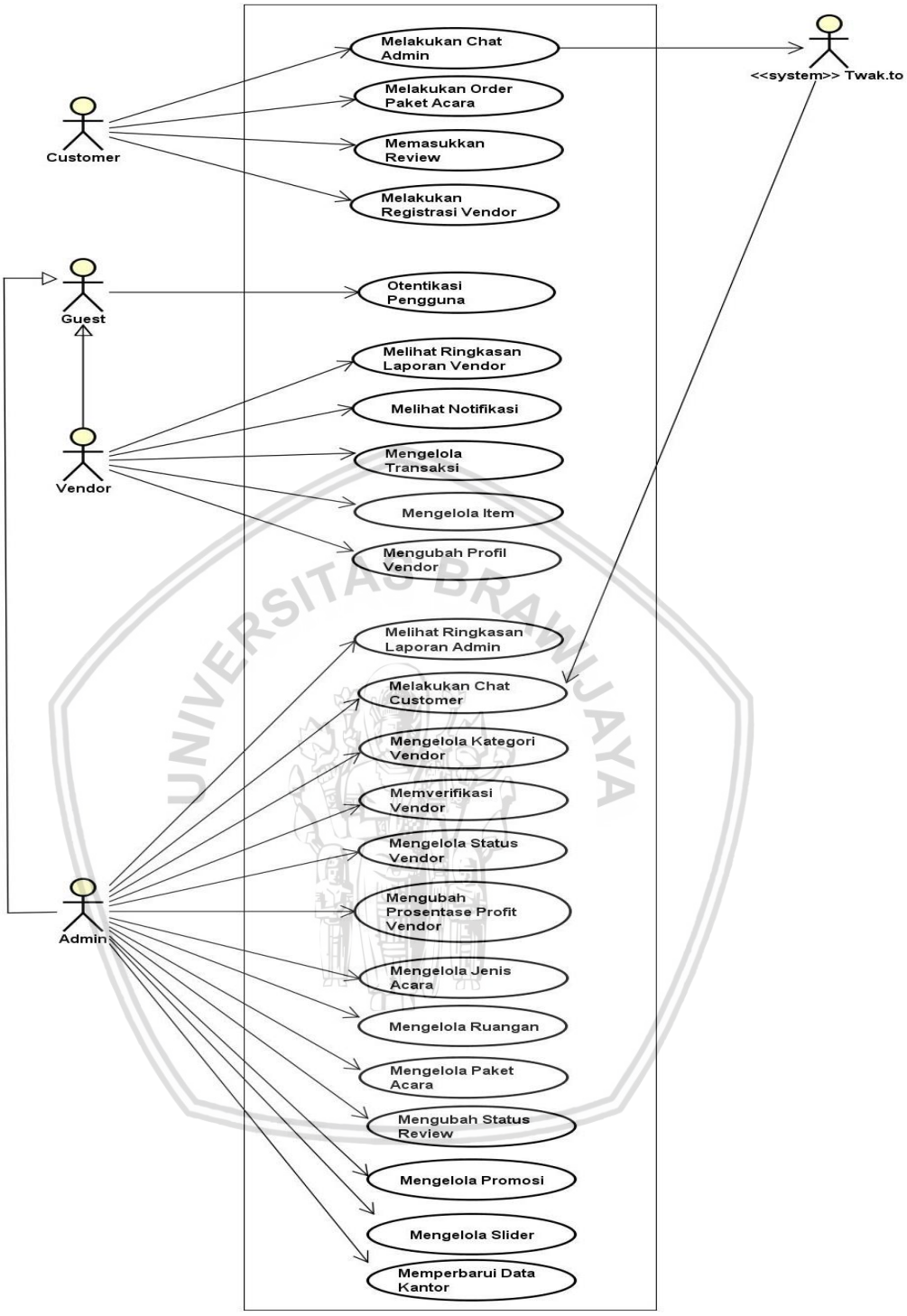
Gambar 4.2 menunjukkan WBS pada metode *Advance Use Case Point* dan *Revised Use Case Point* yang merujuk pada teori Saleh (2011), dan terdiri dari empat level. Level pertama adalah nama sistem yang dikembangkan, yaitu Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Level kedua terdiri dari empat fase pada *Project Life Cycle* (PLC), yaitu *Define Project Goal, Plan Project,Execute Project Plan*dan *Close Project*. Level ketiga terdiri dari tiga fase, yaitu fase *Software Development*, fase *Ongoing Activity*, dan fase *Quality* dan *Testing*. Level keempat, fase *Software Development* didekomposisi menjadi enam fase, yaitu *Requirement, Specification, Design, Implementation*, *Integration Testing*, dan *Acceptance* dan *Deployment*. Fase *Ongoing acticity* didekomposisi menjadi enam fase, yaitu *Project Management, Configuration Management, Documentation*, *Quality Assurance* dan *Evaluation* dan *Testing*, dan *Training* dan *Support*.




**Gambar 4. 2 *Work Breakdown Structure Advance Use Case Point* dan *Revised Use Case Point***

## 4.2 Analisis Pemodelan *Use Case*

Pemodelan *Use Case* bertujuan untuk mempresentasikan tujuan sistem dan menggambarkannya dalam urutan aktivitas dan interaksi pengguna dalam upaya mencapai tujuan tersebut. *Use Case* digambarkan dalam bentuk *use case diagram* beserta penjelasan lengkap (*use case description*) termasuk *use case scenario*. *Use Case Scenario* merupakan realisasi dari *use case* yang memiliki skenario utama (*main flow*) dan skenario alternatif (*alternative flow*). *Use Case Scenario* dibuat dengan tujuan untuk menentukan jumlah transaksi pada setiap *use case* berdasarkan alur sukses dan alur alternatif yang nantinya digunakan dalam menganalisis kompleksitas use case pada metode *Advance Use Case Point* dan *Revised Use Case Point.*




**Gambar 4. 3 Use Case Diagram Sistem Reservasi Online Gedung Serbaguna**

Keterangan Gambar 4.3 :

*Customer* dapat mengakses Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula langsung dan melakukan *chat* dengan admin, order paket acara, memasukkan *review*, dan registrasi sebagai vendor. Ketika melakukan *chat* dengan admin, sistem tawk.to berperan sebagai perantara dalam menyampaikan pesan ke admin. Sistem tawk.to menerima pesan dari *customer* dan

meneruskannya ke admin. Jika ingin masuk sebagai vendor atau admin, pengguna berperan sebagai *guest* dan melakukan otentikasi pengguna. Vendor yang telah berhasil mengakses sistem (*login*) dapat melihat ringkasan laporan vendor, notifikasi, mengelola transaksi, item dan mengubah *profit* vendor. Admin yang telah berhasil mengakses sistem (*login*) dapat melihat ringkasan laporan admin, melakukan *chat* dengan *customer*, mengelola kategori sistem, vendor, status vendor dari aktif ke banned dan sebaliknya, jenis acara, ruangan, paket acara, promosi, slider, mengubah persentase profit vendor, mengubah status review, serta memperbaharui data kantor seperti alamat kantor dan media sosial kantor. Admin membalas *chat* yang masuk dari *customer* melalui sistem tawk.to.

Berdasarkan wawancara yang dilakukan kepada manajer proyek terkait kebutuhan, fungsi serta fitur yang terdapat di dalam sistem dan observasi langsung pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, peneliti menyusun 23 use case, diantaranya: *Use Case* Melakukan *Chat* Admin, *Use Case* Melakukan Order Paket Acara, *Use Case* Menambah *Review*, *Use Case* Melakukan Registrasi Vendor, *Use Case* Otentikasi Pengguna, *Use Case* Melihat Ringkasan Laporan Vendor, *Use Case* Melihat Notifikasi, *Use Case* Mengelola Transaksi, *Use Case* Mengelola Item, *Use Case* Mengubah Profit Vendor, *Use Case* Melihat Ringkasan Laporan Admin, *Use Case* Melakukan *Chat Customer*, *Use Case* Mengelola Kategori Vendor, *Use Case* Memverifikasi Vendor, *Use Case* Mengelola Status Vendor, *Use Case* Mengubah Prosentase Profit Vendor, *Use Case* Mengelola Jenis Acara, *Use Case* Mengelola Ruangan, *Use Case* Mengelola Paket Acara, *Use Case* Mengubah Status Review, *Use Case* Mengelola Promosi, *Use Case* Mengelola Slider, dan *Use Case* Memperbarui Data Kantor.

1) *Use Case* Melakukan Chat Admin, berasal dari pernyataan kebutuhan RS_04 Melakukan komunikasi antara admin dan customer dan memiliki kebutuhan fungsional SROGS_04_102 Melakukan Chat Admin pada Lampiran E. Aktor dalam *use case* ini adalah *customer*. Pada alur utama, customer memilih menu *chat admin*, lalu sistem menampilkan halaman *chat* melalui twak.to. *Customer* mengirimkan pesan pada admin, dan twak.to menyampaikan pesan kepada admin, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, *customer* sudah mengakses sistem dan berada pada halaman utama sistem. Post-kondisi *use case, customer* berhasil berkomunikasi dengan admin.
2) *Use Case* Melakukan *Order* Paket Acara, berasal dari pernyataan kebutuhan RS_01 Menampilkan informasi mengenai Gedung Hijau Lestari, RS_05 Melakukan pemesanan paket acara dan RS_06 Mengirimkan tagihan pembayaran, serta memiliki kebutuhan fungsional SROGS_01_101 Lihat Informasi Paket Acara, SROGS_05_103 Pemesanan Gedung, SROGS_05_104 Cek Ketersediaan Gedung, dan SROGS_06_105 Melakukan Pembayaran pada Lampiran E. Aktor dalam *use case* ini adalah *Customer*. Pada alur utama, *customer* memilih menu *order*, melakukan *order* paket acara atau paket *custom*, kemudian mengecek ketersediaan gedung yang diinginkan dan mengisi *form* pembayaran, lalu sistem mengirimkan tagihan ke email *customer*, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use Case* ini

memiliki *subflow* Melakukan Order Paket Custom, dan alur alternatif untuk data yang tidak lengkap dan gedung yang penuh. Pre-kondisi *use case*, *customer* sudah mengakses sistem dan berada pada halaman utama sistem. Post-kondisi *use case*, *customer* berhasil melakukan pemesanan gedung.

3) *Use Case* Menambah *Review*, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, serta memiliki kebutuhan fungsional SROGS_03_106 Menulis Review pada Lampiran E. Aktor dalam *use case* ini adalah *Customer*. Pada alur utama, *customer* memilih menu *review* dan sistem menampilkan *form review* untuk diisi oleh *customer*. *Customer* mengirimkan *review* dan sistem menyimpan *review*, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, *customer* sudah mengakses sistem, berada pada halaman utama sistem, dan pernah melakukan pemesanan gedung. Post-kondisi *use case*, *customer* berhasil melakukan menambahkan *review*.

4) *Use Case* Melakukan Registrasi Vendor, berasal dari pernyataan kebutuhan RS_07 Melakukan registrasi vendor dan memiliki kebutuhan fungsional SROGS_07_107 Melakukan Registrasi Vendor pada Lampiran E. Aktor dalam *use case* ini adalah *Customer*. Pada alur utama, *customer* memilih menu registrasi vendor dan sistem menampilkan *form* registrasi untuk diisi oleh *customer*. *Customer* mengirimkan *form* dan sistem menyimpan data pendaftaran, untuk lebih jelasnya dapat dilihat pada Lampiran G. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap. Pre-kondisi *use case*, *customer* sudah mengakses sistem dan berada pada halaman utama sistem. Post-kondisi *use case*, *customer* berhasil mendaftarkan diri sebagai vendor.

5) *Use Case* Melihat Ringkasan Laporan Vendor, berasal dari pernyataan kebutuhan RS_07 Melakukan registrasi vendor dan memiliki kebutuhan fungsional SROGS_02_201 Lihat Jadwal Pemesanan, SROGS_02_202 Lihat Profit, SROGS_02_203 Lihat Grafik Transaksi dan SROGS_02_204 Lihat Grafik Profit pada Lampiran E. Aktor dalam *use case* ini adalah Vendor. Pada alur utama, vendor memilih menu *Dashboard* dan sistem menampilkan halaman *dashboard* beserta ringkasan-ringkasan laporan transaksi dan profit vendor, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, vendor berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, vendor berhasil melihat ringkasan laporan vendor.

6) *Use Case* Mengelola Transaksi, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, serta memiliki kebutuhan fungsional SROGS_03_206 Lihat Transaksi, SROGS_03_207 Filter Transaksi, SROGS_03_208 Cetak Laporan Transaksi, dan SROGS_03_209 Cari Transaksi ada Lampiran E. Aktor dalam *use case* ini adalah Vendor. Pada alur utama, *customer* memilih menu transaksi, lalu sistem menampilkan daftar transaksi beserta fungsi-fungsi yang tersedia. Vendor dapat memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use Case* ini memiliki *subflow* Melihat Detail

Transaksi, Memfilter Transaksi, Mencetak Laporan Transaksi dan Mencari Transaksi. Terdapat alur alternatif apabila data tidak tersedia pada periode tertentu dan data transaksi tidak tersedia. Pre-kondisi *use case*, vendorberhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, vendormenampilkan daftar transaksi, serta berhasil melihat detail transaksi, melihat transaksi pada periode tertentu, mencetak laporan transaksi dan mencari transaksi sesuai subflow yang dijalankan.

7) *Use Case* Melihat Notifikasi, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, serta memiliki kebutuhan fungsional SROGS_03_210 Notifikasi Transaksi pada Lampiran E. Aktor dalam *use case* ini adalah Vendor. Pada alur utama, vendor memilih menu transaksi dan sistem menampilkan daftar transaksi yang baru terdaftar, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, vendor berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, vendor berhasil transaksi baru yang masuk.

8) *Use Case* Mengubah Profil Vendor, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, serta memiliki kebutuhan fungsional SROGS_03_205 Edit Profil pada Lampiran E. Aktor dalam *use case* ini adalah Vendor. Pada alur utama, vendor memilih menu *dashboard*, lalu memilih fungsi *edit* profil dan sistem menampilkan *form* untuk mengubah profil. Vendor mengubah profil seperlunya dan menyimpan perubahan. Sistem menyimpan perubahan, untuk lebih jelasnya dapat dilihat pada Lampiran G. Terdapat alur alternatif apabila batal melakukan perubahan. Pre-kondisi *use case*, vendor berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, vendor berhasil mengubah data profil vendor.

9) *Use Case* Mengelola Item, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, serta memiliki kebutuhan fungsional SROGS_03_211 Lihat Item, SROGS_03_212 Tambah Item, SROGS_03_213 Edit Item, dan SROGS_03_214 Hapus Item pada Lampiran E. Aktor dalam *use case* ini adalah Vendor. Pada alur utama, *customer* memilih menu item, lalu sistem menampilkan informasi item beserta fungsi-fungsi yang tersedia. Vendor dapat memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use Case* ini memiliki *subflow* Menambah Item, Mengubah Item, dan Menghapus Item. Terdapat alur alternatif apabila data yang dimasukkan tidak sesuai format. Pre-kondisi *use case*, vendor berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, vendormenampilkan daftar item, serta berhasil menambahkan item, mengubah item dan menghapus item sesuai *subflow* yang dijalankan.

10) *Use Case* Melihat Ringkasan Laporan Admin, berasal dari pernyataan kebutuhan RS_02 Menampilkan informasi mengenai ringkasan laporan terkini,

dan memiliki kebutuhan fungsional SROGS_02_301 Lihat Jumlah Transaksi, SROGS_02_302 Lihat Jumlah Vendor, SROGS_02_303 Lihat Profit, SROGS_02_304 Lihat Grafik Profit, SROGS_02_305 Lihat Grafik Transaksi dan SROGS_02_306 Lihat Jadwal pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu *Dashboard* dan sistem menampilkan halaman *dashboard* beserta ringkasan-ringkasan laporan transaksi, profit perusahaan dan jumlah vendor, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, admin berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, admin berhasil melihat ringkasan laporan admin.

11) *Use Case* Melakukan Chat Customer, berasal dari pernyataan kebutuhan RS_04 Melakukan komunikasi antara admin dan *customer* dan memiliki kebutuhan fungsional SROGS_04_308 Melakukan *Chat Customer* pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu *chat* twak.to, dan melakukan *login* menggunakan email dan *password*. Kemudian admin membalas pesan *customer* yang masuk melalui twak.to, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, admin sudah terdaftar di twak.to, serta berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil berkomunikasi dengan *customer*.

12) *Use Case* Mengelola Kategori Vendor, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, serta memiliki kebutuhan fungsional SROGS_03_324 Tambah kategori vendor, SROGS_03_325 Edit kategori vendor, SROGS_03_326 Hapus kategori vendor dan SROGS_03_327 Cari kategori vendor pada Lampiran E. Aktor dalam *use case* ini adalah admin. Pada alur utama, admin memilih menu Kategori Vendor, lalu sistem menampilkan daftar kategori vendor beserta fungsi-fungsi yang tersedia. Admin dapat memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use Case* ini memiliki *subflow* Menambah Kategori Vendor, Mengubah Kategori Vendor, Menghapus Kategori Vendor, dan Mencari Kategori Vendor. Terdapat alur alternatif apabila data kategori vendor tidak tersedia dan vendor membatalkan perubahan data kategori vendor. Pre-kondisi *use case*, adminberhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case*, adminmenampilkan daftar kategori vendor, serta berhasil menambahkan kategori vendor, mengubah kategori vendor, menghapus kategori vendor dan menemukan kategori vendor yang dicari sesuai *subflow* yang dijalankan.

13) *Use Case* Memverifikasi Vendor, berasal dari pernyataan kebutuhan RS_08 Melakukan verifikasi apabila terdapat vendor baru pada Lampiran E dan memiliki kebutuhan fungsional SROGS_08_310 Verifikasi vendor pada Lampiran F. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu Vendor, dan sistem menampilkan fungsi untuk memverifikasi

vendor. Admin memilih fungsi verifikasi dan sistem menampilkan detail vendor. Admin dapat memilih verifikasi dan memasukkan prosentase keuntungan vendor. Ketika vendor sudah terverifikasi, sistem mengirimkan *password* pada vendor, untuk lebih jelasnya dapat dilihat pada Lampiran G. Terdapat alur alternatif apabila admin menolak untuk memverifikasi vendor. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil memverifikasi vendor.

14) *Use Case* Mengelola Status Vendor, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item, dan memiliki kebutuhan fungsional SROGS_03_329 Ubah status vendor pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu Vendor, dan sistem menampilkan fungsi salah satunya untuk mengubah status vendor. Admin memilih fungsi aktif atau *banned* dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use case* ini memiliki *subflow* Mengubah Status Aktif dan Mengubah Status *Banned*. Terdapat alur alternatif apabila vendor memilih membatalkan perubahan status vendor. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil merubah status vendor sesuai kebutuhan.

15) *Use Case* Mengubah Prosentase Profit Vendor, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item dan memiliki kebutuhan fungsional SROGS_03_319 Ubah presentase profit pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih fitur Aksi Persentase, dan sistem menampilkan form untuk mengubah presentase profit. Admin merubah sesuai kebutuhan dan menyimpannya, lalu sistem menyimpan perubahan, untuk lebih jelasnya dapat dilihat pada Lampiran G. Terdapat alur alternatif apabila vendor memilih membatalkan perubahan prosentase profit vendor. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman Daftar Vendor Aktif. Post-kondisi *use case,* admin berhasil mengubah prosentase keuntungan vendor.

16) *Use Case* Mengelola Jenis Acara, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item dan memiliki kebutuhan fungsional SROGS_03_309 Tambah jenis acara, SROGS_03_310 Edit jenis acara, dan SROGS_03_311 Hapus jenis acara pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu Jenis Acara, dan sistem menampilkan data jenis acara beserta fungsi-fungsi yang tersedia. Admin memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use case* ini memiliki *subflow* Menambah Jenis Acara, Mengubah Jenis Acara, Menghapus Jenis Acara dan Mencari Jenis Acara. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap, data jumlah nominal tidak sesuai format, data jenis acara tidak

tersedia dan membatalkan menghapus jenis acara. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil menampilkan daftar jenis acara, serta berhasil menambahkan jenis acara, mengubah jenis acara, menghapus jenis acara dan menemukan jenis acara yang dicari sesuai *subflow* yang dijalankan.

17) *Use Case* Mengelola Ruangan, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item dan memiliki kebutuhan fungsional SROGS_03_320 Tambah jenis ruangan, SROGS_03_321 Edit ruangan, SROGS_03_322 Hapus ruangan, dan SROGS_03_323 Cari ruangan pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu Ruangan, dan sistem menampilkan data ruangan beserta fungsi-fungsi yang tersedia. Admin memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. Use case ini memiliki *subflow* Menambah Ruangan, Mengubah Ruangan, Menghapus Ruangan dan Mencari Ruangan. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap, data ruangan tidak tersedia dan membatalkan menghapus ruangan. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil menampilkan daftar ruangan, serta berhasil menambahkan ruangan, mengubah ruangan, menghapus ruangan dan menemukan ruangan yang dicari sesuai *subflow* yang dijalankan.

18) *Use Case* Mengelola Paket Acara, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item dan memiliki kebutuhan fungsional SROGS_03_312 Tambah paket acara, SROGS_03_313 Edit paket acara, SROGS_03_314 Hapus paket acara dan SROGS_03_315 Cari paket acara pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu Paket, dan sistem menampilkan data paket acara beserta fungsi-fungsi yang tersedia. Admin memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use case* ini memiliki *subflow* Menambah Paket Acara, Mengubah Paket Acara, Menghapus Paket Acara dan Mencari Paket Acara. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap, data paket acara tidak tersedia dan membatalkan menghapus paket acara. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil menampilkan daftar paket acara, serta berhasil menambahkan paket acara, mengubah data paket acara, menghapus paket acara dan menemukan paket acara yang dicari sesuai *subflow* yang dijalankan.

19) *Use Case* Mengubah Status Review, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item dan memiliki kebutuhan fungsional SROGS_03_328 Lihat review dan SROGS_03_329 Ubah status review pada

Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu *Review*, dan sistem menampilkan data *review* dan fungsi untuk mengubah status *review*. Admin memilih fungsi *Edit* dan mengubah status *review*. Admin memilih fungsi Simpan dan sistem menyimpan perubahan, untuk lebih jelasnya dapat dilihat pada Lampiran G. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil mengubah status *review* untuk ditampilkan atau tidak.

20) *Use Case* Mengelola Promosi, berasal dari pernyataan kebutuhan RS_03 Mengelola paket acara, kategori vendor, data vendor, jenis acara, ruangan, transaksi, *review*, promo dan item dan memiliki kebutuhan fungsional SROGS_03_330 Tambah promo, SROGS_03_331 Edit promo, SROGS_03_332 Hapus promo, dan SROGS_03_333 Cari promo pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu Promosi, dan sistem menampilkan data promosi beserta fungsi-fungsi yang tersedia. Admin memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use case* ini memiliki *subflow* Menambah Promosi, Mengubah Promosi, Menghapus Promosi dan Mencari Promosi. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap, data promosi tidak tersedia dan membatalkan menghapus data promosi. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil menampilkan daftar promosi, serta berhasil menambahkan promosi, mengubah data promosi, menghapus promosi dan menemukan promosi yang dicari sesuai *subflow* yang dijalankan.

21) *Use Case* Mengelola Slider, berasal dari pernyataan kebutuhan RS_09 Mengelola konten yang ada di dalam sistem, dan memiliki kebutuhan fungsional SROGS_09_334 Tambah *slider*, SROGS_09_335 *Edit slider*, SROGS_09_336 Hapus *slider*, dan SROGS_09_337 Cari *slider* pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur utama, admin memilih menu *Slider*, dan sistem menampilkan data *slider* beserta fungsi-fungsi yang tersedia. Admin memilih fungsi yang diinginkan dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use case* ini memiliki *subflow* Menambah *Slider*, Mengubah *Slider*, Menghapus *Slider* dan Mencari *Slider*. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap, data *slider* tidak tersedia dan membatalkan menghapus data slider. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil menampilkan daftar *slider*, serta berhasil menambahkan *slider*, mengubah data *slider*, menghapus *slider* dan menemukan *slider* yang dicari sesuai *subflow* yang dijalankan.

22) *Use Case* Memperbarui Data Perusahaan, berasal dari pernyataan kebutuhan RS_09 Mengelola konten yang ada di dalam sistem, dan memiliki kebutuhan fungsional SROGS_09_338 Edit alamat kantor dan SROGS_09_339 Edit media sosial pada Lampiran E. Aktor dalam *use case* ini adalah Admin. Pada alur

utama, admin memilih menu Footer, dan sistem menampilkan data alamat dan data media sosial perusahaan beserta fungsi untuk mengubah data. Admin memilih fungsi mengubah data alamat atau data media sosial dan sistem menjalankan fungsi terpilih, untuk lebih jelasnya dapat dilihat pada Lampiran G. *Use case* ini memiliki *subflow* Mengubah Alamat dan Mengubah Media Sosial. Terdapat alur alternatif apabila data yang dimasukkan tidak lengkap. Pre-kondisi *use case*, admin sudah berhasil *login* dan berada pada halaman utama sistem. Post-kondisi *use case,* admin berhasil menampilkan alamat dan daftar media soSial perusahaan, serta berhasil mengubah data alamat dan mengubah data media sosial sesuai *subflow* yang dijalankan.

23) *Use Case* Otentikasi Pengguna, berasal dari pernyataan kebutuhan RS_00 Mengelola akses pengguna, dan memiliki kebutuhan fungsional SROGS_00_001 Login pada Lampiran E. Aktor pada *use case* ini adalah *Guest*. Pada alur utama, *Guest* mengakses sistem dan sistem menampilkan *form login*. *Guest* memasukkan email dan *password* dan sistem memeriksa otentikasi *Guest*. Otentikasi berhasil, dan *Guest* masuk sebagai vendor atau admin, untuk lebih jelasnya dapat dilihat pada Lampiran G. terdapat alur alternatif apabila *Guest* memasukkan email dan *password* yang salah. Pre-kondisi *use case*, *Guest* mengakses sistem. Post-kondisi *use case,* guest berhasil masuk ke dalam sistem dan mengakses halaman utama.

## 4.3 Analisis Kondisi Eksisting Waktu Kerja

Analisis kondisi eksisting waktu kerja bertujuan untuk mengetahui total jam kerja tim proyek dan berapa biaya per-jam yang harus dikeluarkan untuk proses pengembangan tergantung pada peran dari setiap pengembang yang akan digunakan untuk proses analisis perhitungan biaya, waktu dan sumber daya manusia pada metode *Advance Use Case Point* dan *Revised Use Case Point*. Berdasarkan wawancara pada Lampiran A, diketahui bahwa tim proyek pada CV Gumcode Indonesia bekerja selama lima hari dalam seminggu, yaitu Senin sampai dengan Jumat, dengan jam kerja selama 8 jam per-hari, dimulai pukul 08.00 WIB sampai dengan 12.00 WIB, kemudian istirahat, dan dilanjutkan pukul 13.00 WIB sampai dengan 17.00 WIB. Perhitungan standar gaji pada Tabel 4.1, dihitung berdasarkan hari kerja dan jam kerja pada proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula di CV Gumcode Indonesia.

**Tabel 4. 1 Standar Gaji Per-Jam Setiap Posisi**

| Posisi dalam Salary Guide | Standar Gaji (per bulan) (Rp) | Standar Gaji (per jam) (Rp) |
|---|---|---|
| *Project Manager* | Rp 20.000.000,00 | Rp 125.000,00 |
| *System Analyst* | Rp 7.000.000,00 | Rp 43.750,00 |
| *Software Engineer* | Rp 5.000.000,00 | Rp 31.250,00 |
| *Test Analyst* | Rp 8.000.000,00 | Rp 50.000,00 |
| *Software QA* | Rp 8.000.000,00 | Rp 50.000,00 |

Keterangan Tabel 4.1:
1 bulan = 30 hari (4 minggu)

1 minggu = 5 hari (Senin – Jum'at), sehingga 5 * 4 = 20 hari
1 hari = 8 jam (08.00 – 12.00 WIB) dan (13.00 – 17.00 WIB)
Standar gaji per-jam = standar gaji per-bulan/20 hari/8 jam

## 4.4 Analisis Perhitungan dengan *Advance Use Case Point*

Perhitungan estimasi *effort* menggunakan metode *Advance Use Case Point* dan pedoman persebaran *effort* pada masing-masing aktifitas pengembangan perangkat lunak, serta *use case diagram* dan *use case scenario* yang dimodelkan sebelumnya akan digunakan untuk menganalisis perhitungan biaya, waktu dan sumber daya manusia pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Hasil dari analisis perhitungan menggunakan metode *Advance Use Case Point* ini adalah ukuran perangkat lunak, yang kemudian dikonversi ke dalam *hours of effort* untuk menghitung total biaya, waktu dan sumber daya manusia yang dibutuhkan untuk menyelesaikan pengembangan sistem.

### 4.4.1 Analisis Perhitungan Unadjusted Use Case Point

Nilai *Unadjusted Use Case Point* (UUCP) diperoleh dari penjumlahan hasil perhitungan nilai *Unadjusted Actor Weight* (UAW) dan nilai *Unadjusted Use Case Weight* (UUCW). Perhitungan UUCP berdasarkan hasil analisis terhadap *use case* dan *use case scenario* dari perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.

#### *4.4.1.1 Perhitungan Unadjusted Actor Weight*

Analisis perhitungan nilai *Unadjusted Actor Weight* (UAW) bertujuan untuk mengetahui bobot dari masing-masing tipe aktor yang berinteraksi dengan sistem, dimana nilai ini diperoleh dengan cara mengidentifikasi dan mengklasifikasikan tipe aktor yang terlibat dengan sistem. Aktor diklasifikasikan ke dalam *simple, average* dan *complex*. Aktor diklasifikasikan *simple,* apabila aktor merepresentasikan sistem lain yang berinteraksi dengan sistem melalui API (*Application Programming Interface).* Aktor diklasifikasikan *average,* apabila actor merepresentasikan sistem lain yang berinteraksi dengan sistem melalui protokol seperti TCP/IP, HTTP, atau SOAP dan manusia yang berinteraksi dengan sistem melalui *command line*. Aktor diklasifikasikan *complex,* apabila aktor merepresentasikan manusia yang berinteraksi dengan sistem melalui *Graphical User Interface* (GUI) atau halaman *web*.

Berdasarkan identifikasi aktor yang diperoleh dari analisis pada *use case diagram* Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang dilampirkan pada Lampiran F, digunakan empat aktor yang terdiri dari: tiga aktor primer dan satu aktor sekunder. Aktor primer, yaitu *customer*, vendor dan admin, memiliki kebutuhan untuk mengakses sistem melalui halaman web pada Sistem Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, sedangkan aktor sekunder, yaitu tawk.to, merupakan *plug-in* yang berperan sebagai penyedia fasilitas *chat* antara *customer* dan admin, yang berinteraksi dengan sistem melalui API *Javascript*. Oleh karena itu, aktor primer diklasifikasikan menjadi aktor

*complex* dan aktor sekunder diklasifikasikan menjadi aktor *simple*. Penjelasan mengenai peran aktor dan interaksinya dengan sistem ditampilkan pada Tabel 4.2.

**Tabel 4. 2 Klasifikasi Aktor pada Sistem dengan AUCP**

| Nama Aktor | | Deskripsi | Tipe |
|---|---|---|---|
| 1. | *Customer* | Melakukan pemesanan paket acara, chat dengan admin, mengisi *review* dan mendaftar vendor melalui halaman *web*. | *Complex* |
| 2. | Vendor | Melihat ringkasan laporan vendor dan notifikasi, mengelola item dan mengubah profil vendor melalui halaman *web*. | *Complex* |
| 3. | Admin | Melihat ringkasan laporan admin, melakukan *chat* dengan *customer*, mengelola data kategori vendor, data vendor, data status vendor, data prosentase profit vendor, sata jenis acara, data ruangan, data paket acara, data *review*, data promosi, data *slider* dan data informasi kantor melalui halaman *web*. | *Complex* |
| 4. | Tawk.to | Memfasilitasi percakapan langsung antara *customer* dan admin, yang berinteraksi dengan sistem melalui API *Javascript*. | *Simple* |

Hasil klasifikasi aktor dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, kemudian digunakan untuk menghitung nilai total *Unadjusted Actor Weight* dengan mengkalikan jumlah aktor dengan bobot pada masing-masing tipe aktor, yang ditunjukkan pada Tabel 4.3.

**Tabel 4. 3 Perhitungan *Unadjusted Actor Weight* dengan AUCP**

| Tipe | Bobot | Jumlah Aktor | Bobot x Jumlah Aktor |
|---|---|---|---|
| *Simple* | 1 | 1 | 1 |
| *Average* | 2 | 0 | 0 |
| *Complex* | 3 | 3 | 9 |
| | | **Total UAW** | 10 |

#### *4.4.1.2 Perhitungan Unadjusted Use Case Weight*

Analisis perhitungan nilai *Unadjusted Use Case Weight* (UUCW) bertujuan untuk mengetahui bobot kompleksitas dari setiap use case, dimana kompleksitas suatu *use case* ditentukan oleh jumlah transaksi dari *use case scenario*. Bobot kompleksitas tersebut diperoleh dengan cara mengklasifikasikan kompleksitas *use case* dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula ke dalam *simple, average* dan *complex* tergantung pada jumlah transaksi sukses dan alur alternatif dalam *use case description*. *Simple use case* memiliki jumlah transaksi 1 sampai dengan 3 transaksi, *average use case* memiliki jumlah transaksi 4 sampai dengan 7 transaksi, dan *complex use case* memiliki jumlah transaksi lebih dari 7 transaksi.

Berdasarkan *use case scenario* pada Lampiran G, peneliti mengidentifikasi setiap transaksi yang ada pada alur sukses dan alur alternatif, dimana apabila terdapat masing-masing satu transaksi pada skenario sukses dan skenario gagal, transaksi dianggap sama dan dihitung sebagai dua transaksi, serta transaksi yang

melibatkan operasi yang sama dalam *database* dan hanya berbeda instansiasi dihitung sebagai transaksi yang sama, meski ditulis menjadi dua skenario yang berbeda dalam *use case scenario.* Kemudian menjumlahkan jumlah transaksi pada alur sukses dan alur alternatif tersebut dan mengklasifikasikannya berdasarkan aturan pada metode *Advance Use Case Point.* Berikut analisis jumlah transaksi pada setiap *use case* dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang ditampilkan pada Tabel 4.4 sampai Tabel 4.32.

**Tabel 4. 4 Transaksi *Use Case Login***

| |
|---|
| **Transaksi 1:**<br>1. Sistem menampilkan *form login* yang meminta *guest* memasukkan alamat *email* dan *password*.<br>2. *Guest* memasukkan *email* dan *password* dengan benar dan memilih fungsi login.<br>3. Sistem memeriksa otentikasi pengguna dan<br>4. Sistem menampilkan halaman utama sistem.<br>**Transaksi 2:**<br>**Email dan Password Salah**<br>Apabila *guest* tidak memasukkan email atau *password* yang benar, sistem menampilkan pesan peringatan yang menyatakan bahwa email dan *password* salah. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case Login* diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 5 Transaksi *Use Case Chat* Admin**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu *chat* admin.<br>2. Sistem menampilkan halaman *chat* admin melalui twak.to.<br>3. *Customer*melakukan komunikasi kepada admin melalui *twak.to*.<br>4. Twak.to menyampaikan pesan yang dikirimkan oleh *customer* kepada admin.<br>**Transaksi 2:**<br>**Koneksi Terputus**<br>Jika dalam keadaan *offline* maka fungsi *chat* admin akan dialihkan dengan memasukkan nama, email dan pesan yang ingin disampaikan kemudian mensubmit sistem, maka *chat* akan masuk ke halaman admin ketika admin sedang *online.* |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case Chat Admin* diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 6 Transaksi *Use Case* Melakukan *Order* Paket Acara**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu *Order*.<br>2. Sistem menampilkan halaman *order* yang berisi jenis-jenis paket acara yang tersedia.<br>3. *Customer* memilih jenis acara yang diinginkan.<br>4. Sistem menampilkan daftar paket acara dan paket *custom*.<br>5. *Customer* memilih paket acara yang diinginkan<br>Jika *customer* memilih *custom* paket, maka akan menjalankan *subflow* Melakukan Order Paket Custom. |

**Tabel 4. 7 Transaksi *Use Case* Melakukan *Order* Paket Acara (Lanjutan)**

| |
|---|
| 6. Sistem menampilkan detail paket acara, seperti layanan dan fasilitas yang didapatkan, harga *per-pack*, dan minimal pemesanan. Sistem meminta *customer* mengisi *form* untuk mengecek ketersediaan gedung.<br>7. *Customer* memasukkan tanggal dan waktu yang diinginkan.<br>8. Sistem menampilkan ketersediaan gedung sesuai tanggal dan waktu yang dipilih *customer* dan meminta *customer* mengisi *form payment*.<br>9. *Customer* memasukkan data pemesanan secara lengkap, beserta kode promosi apabila ada, lalu memilih fungsi *Submit form payment*.<br>10. Sistem menampilkan detail pembayaran yang harus dibayar oleh *customer* dan pilihan media pembayaran, yaitu melalui *midtrans* atau *direct transfer*.<br>11. *Customer* memilih salah satu media pembayaran.<br>12. Sistem menampilkan *form* konfirmasi penyimpanan order.<br>13. *Customer* melakukan konfirmasi dengan memilih Ya.<br>14. Sistem menyimpan pemesanan dan mengirim email berisi *invoice* ke email customer, dan menampilkan halaman utama system<br>**Transaksi 2:**<br>**Melengkapi Data Pemesanan**<br>Apabila customer tidak melengkapi seluruh data yang dibutuhkan pada *form payment* atau pemesanan, maka sistem menampilkan peringatan untuk melengkapi data dan kembali ke langkah 9 pada *Basic Flow.*<br>**Transaksi 3:**<br>**Gedung Penuh**<br>Apabila dalam sistem tanggal yang dipilih untuk memesan gedung tidak tersedia atau sudah ada yang memesan, maka sistem akan menampilkan 'PENUH' dan *customer* diminta untuk mengganti tanggal lain yang tersedia, kemudian melanjutkan ke langkah 9 pada *Basic Flow*. |
| **Total Transaksi:** 3 |
| **Kategori:** Simple |

*Use Case* Melakukan Order Paket Acara diidentifikasi memiliki satu transaksi pada alur sukses dan dua transaksi pada alur alternatif, sehingga memiliki total tiga transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 8 Transaksi *Use Case* Menambah Review**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu *Review*<br>2. Sistem menampilkan testimoni, jumlah *review* dan *rating* saat ini, serta fungsi Berikan *Rating*.<br>3. *Customer* memilih fungsi Berikan *Rating*.<br>4. Sistem menampilkan form untuk mengisi *review*.<br>5. *Customer* memilih berapa *rating* bintang yang diinginkan dan memasukkan ID pemesanan, nama dan *review* di kolom *review*, lalu memilih fungsi Kirim.<br>6. Sistem menyimpan *review* dan ketika sudah selesai kembali ke halaman utama sistem.<br>**Transaksi 2:**<br>**Melengkapi Data *Review***<br>Apabila *customer* tidak mengisi *form review* secara lengkap, maka sistem menampilkan peringatan untuk meminta *customer* melengkapi seluruh kolom, kemudian melanjutkan ke langkah 6 pada *Basic Flow*. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Menambah *Review* diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 9 Transaksi *Use Case* Melakukan Registrasi Vendor**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu Registrasi Vendor.<br>2. Sistem menampilkan *form* registrasi vendor dan meminta *customer* mengisi data vendor, yaitu nama vendor, email, alamat, kategori vendor, penanggung jawab, acara pilihan, alamat, No KTP, no telefon, foto profil dan foto KTP.<br>3. *Customer* memasukkan data vendor secara lengkap dan memilih fungsi *Submit*.<br>4. Sistem menyimpan data vendor dan menampilkan pesan konfirmasi bahwa *customer* telah berhasil melakukan pendaftaran.<br>**Transaksi 2:**<br>**Melengkapi Data Vendor**<br>Apabila *customer* tidak mengisi *form* pendaftaran vendor secara lengkap, maka sistem tidak dapat menyimpan data vendor dan meminta vendor memasukkan kembali data vendor, kemudian melanjukan ke langkah 4 pada *Basic Flow*. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Melakukan Registrasi Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 10 Transaksi *Use Case* Melihat ringkasan laporan vendor**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu *Dashboard*.<br>2. Sistem menampilkan halaman *dashboard* yang memuat jumlah profit, jumlah transaksi, jumlah item, agenda, grafik transaksi, grafik profit vendor, dan agenda berupa kalender. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melihat Ringkasan Laporan Vendor diidentifikasi hanya memiliki satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 11 Transaksi *Use Case* Mengelola transaksi**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu transaksi.<br>2. Sistem menampilkan daftar transaksi yang masuk dan beberapa fungsi, yaitu *filter* untuk melihat transaksi pada periode waktu tertentu, detail transaksi, dan cetak laporan transaksi.<br>3. Vendor memilih salah satu fungsi.<br>Jika vendor memilih untuk melihat detail transaksi, maka akan menjalankan *subflow* Melihat Detail Transaksi.<br>Jika vendor memilih untuk melihat transaksi pada periode waktu tertentu dan status tertentu, maka akan menjalankan *subflow* Memfilter Transaksi.<br>Jika vendor memilih untuk mencetak laporan transaksi, maka akan menjalankan *subflow* Mencetak Laporan Transaksi.<br>Jika vendor memilih untuk mencari transaksi tertentu, maka akan menjalankan *subflow* Mencari Transaksi.<br>**Transaksi 2:**<br>**Data Periode Tertentu Tidak Tersedia**<br>Pada *subflow* Memfilter Transaksi, apabila tidak ditemukan transaksi pada periode waktu yang ditentukan, maka akan menampilkan keterangan "*No Data Available in Table*" dan kembali pada langkah vendor memasukkan tanggal awal acara, tanggal akhir acara dan status transaksi.<br>**Transaksi 3:** |

**Tabel 4. 12 Transaksi *Use Case* Mengelola transaksi (Lanjutan)**

| |
|---|
| **Data Transaksi Tidak Tersedia**<br>Pada *subflow* Mencari Transaksi, apabila data transaksi yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah vendor memasukkan kata kunci. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Transaksi diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*. Transaksi 2 Data Periode Tertentu Tidak Tersedia dan Transaksi 3 Data Transaksi Tidak Tersedia dihitung satu transaksi karena kedua transaksi ini merupakan operasi yang sama pada database yang sama, meskipun menggunakan instansiasi yang berbeda.

**Tabel 4. 13 Transaksi *Use Case* melihat notifikasi**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih ikon notifikasi di menu Transaksi.<br>2. Sistem menampilkan daftar transaksi yang terdaftar beserta transaksi baru yang masuk. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melihat Notifikasi diidentifikasi memiliki hanya satu transaksi pada alur sukses, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 14 Transaksi *Use case* mengubah profil vendor**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu *Dashboard*.<br>2. Sistem menampilkan halaman *dashboard* yang memuat fungsi untuk mengubah profil.<br>3. Vendor memilih fungsi *Edit* Profil.<br>4. Sistem menampilkan form *Edit* Data<br>5. Vendor mengubah data vendor sesuai kebutuhan dan memilih fungsi *Submit*.<br>6. Sistem menyimpan perubahan dan menampilkan halaman *dashboard*.<br>**Transaksi 2:**<br>**Membatalkan Perubahan Profil**<br>Apabila vendor pada langkah nomor 6, memilih untuk membatalkan penyimpanan perubahan, maka sistem akan menampilkan halaman *Dashboard* dan *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengubah Profil Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple.*

**Tabel 4. 15 Transaksi *Use case* mengelola item**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu Item pada halaman utama<br>2. Sistem menampilkan informasi item dan fungsi yang tersedia untuk vendor. Fungsi tersebut yaitu: menambah item, mengubah item dan menghapus item.<br>3. Vendor memilih salah satu fungsi.<br>Jika vendor memilih fungsi untuk menambah item, maka akan menjalankan *subflow* Menambah Item.<br>Jika vendor memilih fungsi untuk mengubah item, maka akan menjalankan *subflow* Mengubah Item. |

**Tabel 4. 16 Transaksi *Use case* mengelola item (Lanjutan)**

| |
|---|
| Jika vendor memilih fungsi untuk menghapus item, maka akan menjalankan *subflow* Menghapus Item.<br>**Transaksi 2:**<br>**Data Tidak Sesuai Format**<br>Pada langkah memasukkan data item di subflow Menambah Item atau Mengubah Data, apabila vendor tidak memasukkan data pada formulir sesuai dengan format, maka sistem akan menampilkan halaman kosong dan gagal menyimpan, kemudian use case selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Item diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 17 Transaksi *Use case* melihat ringkasan laporan admin**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Dashboard*.<br>2. Sistem menampilkan halaman *dashboard* yang memuat jumlah profit, jumlah transaksi, jumlah vendor, rating dari *review customer*, grafik transaksi, grafik profit, dan agenda berupa kalender. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melihat Ringkasan Laporan Admin diidentifikasi memiliki hanya satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 18 Transaksi *Use case* melakukan chat *customer***

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Chat Admin*<br>2. Sistem menampilkan form login dari twak.to<br>3. Admin memasukkan email dan *password*, kemudian *login* ke dalam twak.to<br>4. Admin membalas pesan *customer* yang masuk melalui twak.to |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melakukan Chat *Customer* diidentifikasi memiliki hanya satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 19 Transaksi *Use case* mengelola kategori vendor**

| |
|---|
| **Transaksi 1:**<br>1. Adminmemilih menu Kategori Vendor<br>2. Sistem menampilkan daftar kategori vendor dan beberapa fungsi, yaitu pencarian, menambah kategori vendor, mengubah kategori vendor dan menghapus kategori vendor.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah kategori vendor, maka akan menjalankan *Subflow* Menambah Kategori vendor.<br>Jika admin memilih fungsi untuk mengubah kategori vendor, maka akan menjalankan *Subflow* Mengubah Kategori vendor.<br>Jika admin memilih fungsi untuk menghapus kategori vendor, maka akan menjalankan *Subflow* Menghapus Kategori vendor.<br>Jika admin memilih fungsi untuk mencari kategori vendor, maka akan menjalankan *Subflow* Mencari Kategori vendor.<br>**Transaksi 2:** |

**Tabel 4. 20 Transaksi *Use case* mengelola kategori vendor (Lanjutan)**

| |
|---|
| **Data Kategori Vendor Tidak Tersedia**<br>Pada *subflow* Mencari Kategori Vendor, apabila data kategori vendor yang dicari tidak terdaftar, maka akan menampilkan keterangan "No Matching Records Found" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 3:**<br>**Membatalkan Perubahan Data**<br>Apabila pada langkah Memilih Simpan data kategori vendor dari subflow Mengubah Kategori Vendor, vendor memilih untuk membatalkan penyimpanan perubahan, maka sistem akan menampilkan daftar kategori vendor dan use case selesai. |
| **Total Transaksi:** 3 |
| **Kategori:** Simple |

*Use Case* Mengelola Kategori Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan dua transaksi pada alur alternatif, sehingga memiliki total tiga transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 21 Transaksi *Use case* memverifikasi vendor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Vendor.<br>2. Sistem menampilkan fungsi, yaitu Verifikasi, Aktif dan *Banned*.<br>3. Admin memilih fungsi Verifikasi.<br>4. Sistem menampilkan daftar vendor yang mendaftar namun belum diverifikasi, dan beberapa fungsi, yaitu Detail untuk melihat detail vendor, verifikasi untuk memverifikasi vendor, dan Tolak untuk menolak vendor bergabung sebagai mitra bisnis.<br>5. Admin memilih fungsi Detail Vendor pada vendor yang ingin diverifikasi.<br>6. Sistem menampilkan informasi detail mengenai vendor.<br>7. Admin memilih fungsi kembali untuk kembali ke daftar vendor dan melakukan verifikasi dan memilih fungsi Verifikasi.<br>8. Sistem menampilkan form konfirmasi yang meminta admin untuk memasukkan persentase keuntungan bagi vendor tersebut.<br>9. Admin memasukkan prosentase keuntungan.<br>10. Admin memilih fungsi Ya, Kirim *Password* untuk menyimpan prosentase dan mengirimkan *password* kepada vendor yang telah disetujui untuk bergabung.<br>11. Sistem menutup form konfirmasi dan menyimpan prosentase lalu mengirimkan email kepada vendor yang disetujui berupa *password* untuk mengakses sistem.<br>12. Sistem menampilkan daftar vendor aktif.<br>**Transaksi 2:**<br>**Menolak Verifikasi Vendor**<br>Apanila admin ingin menolak vendor, maka admin memilih fungsi Tolak, kemudian sistem akan menghapus vendor tersebut dan *Use Case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Memverifikasi Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 22 Transaksi *Use case* mengelola status vendor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Vendor.<br>2. Sistem menampilkan tiga fungsi, yaitu Verifikasi, Aktif dan *Banned*.<br>3. Admin memilih fungsi Aktif atau *Banned*.<br>Jika admin memilih fungsi untuk mengubah status vendor yang aktif, maka menjalankan *Subflow* Mengubah Status Aktif. |

| |
|---|
| Jika admin memilih fungsi untuk mengubah status vendor yang di-*banned*, maka menjalankan *Subflow* Mengubah Status Banned.<br>**Transaksi 2:**<br>**Membatalkan Perubahan Status Vendor**<br>Apabila admin memilih untuk membatalkan penyimpanan perubahan status vendor, maka sistem akan membatalkan perubahan dan menampilkan daftar vendor terdaftar dan *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Status Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 23 Transaksi *Use case* mengubah persentase profit vendor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih fitur Aksi Persentase.<br>2. Sistem menampilkan form untuk mengubah persentase keuntungan vendor.<br>3. Admin mengubah persentase vendor sesuai kebutuhan, lalu memilih fungsi Simpan untuk menyimpan perubahan.<br>4. Sistem menyimpan perubahan dan menampilkan daftar vendor terdaftar yang aktif.<br>**Transaksi 2:**<br>**Membatalkan perubahan persentase profit**<br>Apabilavendor memilih untuk membatalkan penyimpanan perubahan prosentase profit vendor, maka sistem akan membatalkan perubahan dan menampilkan daftar vendor terdaftar dan *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengubah Persentase Profit Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 24 Transaksi *Use case* mengelola jenis acara**

| |
|---|
| **Transaksi 1:**<br>1. Adminmemilih menu Jenis Acara.<br>2. Sistem menampilkan data jenis acara dan beberapa fungsi, yaitu pencarian, menambah jenis acara, mengubah jenis acara, menghapus jenis acara dan mencari jenis acara.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah jenis acara, maka akan menjalankan *Subflow* Menambah Jenis Acara.<br>Jika admin memilih fungsi untuk mengubah jenis acara, maka akan menjalankan *Subflow* Mengubah Jenis Acara.<br>Jika admin memilih fungsi untuk menghapus jenis acara, maka akan menjalankan *Subflow* Menghapus Jenis Acara.<br>Jika admin memilih fungsi untuk mencari jenis acara, maka akan menjalankan *Subflow* Mencari Jenis Acara.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data jenis acara pada *Subflow* Menambah Jenis Acara atau Mengubah Jenis Acara, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan bahwa data tidak tersimpan dan meminta admin untuk melengkapi data.<br>**Transaksi 3:**<br>**Data Jumlah Nominal bukan format angka** |

**Tabel 4. 25 Transaksi *Use case* mengelola jenis acara (Lanjutan)**

| |
|---|
| Pada langkah memasukkan data jenis acara pada *Subflow* Menambah Jenis Acara atau Mengubah Jenis Acara, apabila admin memasukkan jumlah nominal pembayaran awal selain format angka, maka setelah disimpan, jumlah nominal pembayaran awal dirubah menjadi 0.<br>**Transaksi 4:**<br>**Data Jenis Acara Tidak Tersedia**<br>Pada *subflow* Mencari Jenis Acara, apabila data jenis acara yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 5:**<br>**Membatalkan Hapus Jenis Acara**<br>Apabila admin memilih untuk membatalkan hapus jenis acara, maka sistem tidak melakukan penghapusan jenis acara dan use case selesai. |
| **Total Transaksi:** 5 |
| **Kategori:** Average |

*Use Case* Mengelola Jenis Acara diidentifikasi memiliki satu transaksi pada alur sukses dan empat transaksi pada alur alternatif, sehingga memiliki total lima transaksi yang diklasifikasikan dalam kategori *average*

**Tabel 4. 26 Transaksi *Use case* mengelola ruangan**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Ruangan.<br>2. Sistem menampilkan data ruangan dan beberapa fungsi, yaitu pencarian, menambah ruangan, mengubah ruangan, menghapus ruangan dan mencari ruangan.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah ruangan, maka akan menjalankan *Subflow* Menambah Ruangan.<br>Jika admin memilih fungsi untuk mengubah ruangan, maka akan menjalankan *Subflow* Mengubah Ruangan.<br>Jika admin memilih fungsi untuk menghapus ruangan, maka akan menjalankan *Subflow* Menghapus Ruangan.<br>Jika admin memilih fungsi untuk mencari ruangan, maka akan menjalankan *Subflow* Mencari Ruangan.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data ruangan pada *Subflow* Menambah Ruangan atau Mengubah Ruangan, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan bahwa data tidak tersimpan dan meminta admin untuk melengkapi data.<br>**Transaksi 3:**<br>**Data Ruangan Tidak Tersedia**<br>Pada subflow Mencari Ruangan, apabila data ruangan yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 4:**<br>**Membatalkan Hapus Ruangan**<br>Apabila admin memilih untuk membatalkan hapus ruangan, maka sistem tidak melakukan penghapusan ruangan dan use case selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Average |

*Use Case* Mengelola Ruangan diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *average.*

**Tabel 4. 27 Transaksi *Use case* mengelola paket acara**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Paket.<br>2. Sistem menampilkan data paket acara dan beberapa fungsi, yaitu pencarian, menambah paket acara, mengubah paket acara, menghapus paket acara dan mencari paket acara.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah paket acara, maka akan menjalankan *Subflow* Menambah Paket Acara.<br>Jika admin memilih fungsi untuk mengubah paket acara, maka akan menjalankan *Subflow* Mengubah Paket Acara.<br>Jika admin memilih fungsi untuk menghapus paket acara, maka akan menjalankan *Subflow* Menghapus Paket Acara.<br>Jika admin memilih fungsi untuk mencari paket acara, maka akan menjalankan *Subflow* Mencari Paket Acara.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data paket pada *Subflow* Menambah Paket Acara atau Mengubah Paket Acara, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem akan tetap bertahan pada halaman yang sama sampai admin mengisi seluruh kolom.<br>**Transaksi 3:**<br>**Data Paket Acara Tidak Tersedia**<br>Pada *subflow*Mencari Paket Acara, apabila data ruangan yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 4:**<br>**Membatalkan Hapus Paket Acara**<br>Apabila admin memilih untuk membatalkan hapus paket acara, maka sistem tidak melakukan penghapusan paket acara dan *use case* selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Average |

*Use Case* Mengelola Paket Acara diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *average*.

**Tabel 4. 28 Transaksi *Use case* mengubah status review**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Review*.<br>2. Sistem menampilkan data *review* dan fungsi untuk mengubah status *review*.<br>3. Admin memilih fungsi *Edit* pada *review* yang ingin dirubah.<br>4. Sistem menampilkan *form* untuk mengubah status review.<br>5. Admin merubah status *review* sesuai kebutuhan dan menyimpan perubahan dengan memilih fungsi *Save changes.*<br>6. sistem menyimpan perubahan status *review* dan menampilkan daftar *review* beserta yang berhasil dirubah. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Mengubah Status *Review* diidentifikasi memiliki hanya satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 29 Transaksi *Use case* mengelola promosi**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Promosi.<br>2. Sistem menampilkan data promosi dan beberapa fungsi, yaitu pencarian, menambah promosi, mengubah promosi, menghapus promosi dan mencari promosi.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah promosi, maka akan menjalankan *Subflow* Menambah Promosi.<br>Jika admin memilih fungsi untuk mengubah promosi, maka akan menjalankan *Subflow* Mengubah Promosi.<br>Jika admin memilih fungsi untuk menghapus promosi, maka akan menjalankan *Subflow* Menghapus Promosi.<br>Jika admin memilih fungsi untuk mencari promosi, maka akan menjalankan *Subflow* Mencari Promosi.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data promosi pada *Subflow* Menambah Promosi atau Mengubah Promosi, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem akan tetap bertahan pada halaman yang sama dan meminta admin mengisi seluruh kolom.<br>**Transaksi 3:**<br>**Data Promosi Tidak Tersedia**<br>Pada *subflow* **Mencari Promosi**, apabila data promosi yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 4:**<br>**Membatalkan Hapus Promosi**<br>Apabila admin memilih untuk membatalkan hapus promosi, maka sistem tidak melakukan penghapusan promosi dan *use case* selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Average |

*Use Case* Mengelola Promosi diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 30 Transaksi *Use case* mengelola slider**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Slider*.<br>2. Sistem menampilkan data *slider* dan beberapa fungsi, yaitu pencarian, menambah *slider*, mengubah *slider*, menghapus *slider*dan mencari *slider*.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah *slider*, maka akan menjalankan *Subflow* Menambah Slider.<br>Jika admin memilih fungsi untuk mengubah *slider*, maka akan menjalankan *Subflow* Mengubah Slider.<br>Jika admin memilih fungsi untuk menghapus *slider*, maka akan menjalankan *Subflow* Menghapus Slider.<br>Jika admin memilih fungsi untuk mencari *slider*, maka akan menjalankan *Subflow* Mencari Slider.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data slider pada *Subflow* Menambah Slider atau Mengubah Slider, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan bahwa data tidak tersimpan dan meminta admin untuk melengkapi data. |

**Tabel 4. 31 Transaksi *Use case* mengelola slider (Lanjutan)**

| |
|---|
| **Transaksi 3:**<br>**Data Slider Tidak Tersedia**<br>Pada *subflow* Mencari Slider, apabila data slider yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci pada *subflow* Mencari Slider.<br>**Transaksi 4:**<br>**Membatalkan Hapus Slider**<br>Apabila admin memilih untuk membatalkan hapus slider, maka sistem tidak melakukan penghapusan slider dan use case selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Average |

*Use Case* Mengelola Slider diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *average*.

**Tabel 4. 32 Transaksi *Use case* memperbarui data kantor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Footer*.<br>2. Sistem menampilkan data alamat dan data media sosial perusahaan dan fungsi untuk mengubah data alamat dan mengubah data media sosial.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk mengubah alamat perusahaan, maka akan menjalankan *Subflow* Mengubah Alamat.<br>Jika admin memilih fungsi untuk mengubah nama akun media sosial perusahaan, maka akan menjalankan *Subflow* Mengubah Media Sosial.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data alamat pada *Subflow* Mengubah Alamat atau Memasukkan data media sosial pada *Subflow* Mengubah Media Sosial, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan yang meminta admin untuk melengkapi data. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Memperbarui Data Kantor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

Hasil analisis jumlah transaksi dan kategori *use case* pada setiap *use case* dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, kemudian digunakan untuk menghitung nilai total *Unadjusted Use Case Weight* dengan cara mengkalikan jumlah *use case* pada setiap kategori dengan bobot pada masing-masing kategori *use case*, yang ditunjukkan pada Tabel 4.33.

**Tabel 4. 33 Perhitungan *Unadjusted Use Case Weight* dengan AUCP**

| Kategori | Bobot | Jumlah *Use Case* | Bobot x Jumlah Use Case |
|---|---|---|---|
| *Simple* | 5 | 18 | 90 |
| *Average* | 10 | 5 | 50 |
| *Complex* | 15 | 0 | 0 |
| | | **Total UUCW** | 140 |

Nilai *Unadjusted Actor Weight* dan *Unadjusted Use Case Weight* yang telah diketahui, kemudian digunakan untuk meghitung nilai *Unajusted Use Case Point*.

Nilai *Unajusted Use Case Point* diperoleh dari penjumlahan nilai *Unadjusted Actor Weight* dan *Unadjusted Use Case Weight*.

**UUCP = UAW + UUCW**
**= 10 + 140 = 150 UUCP**

Hasil perhitungan *Unadjusted Use Case Point* menghasilkan nilai sejumlah 150 UUCP.

### 4.4.2 Perhitungan *Complexity Factor*

#### *4.4.2.1 Technical Complexity Factor*

*Technical Complexity Factor* (TCF) dibuat untuk mewakili dampak persyaratan non-fungsional proyek terhadap *effort* untuk menyelesaikan proyek. Lembar penilaian untuk mengukur *Technical Complexity Factor* (TCF) pada proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang telah dinilai oleh *expert judgment*, selanjutnya diserahkan kepada manajer proyek yang bertanggungjawab terhadap proyek tersebut. Manajer proyek memberikan nilai pada setiap faktor teknis, dan hasilnya akan dikalikan dengan bobot pada masing-masing faktor. Selama proses pengisian lembar penilaian, peneliti mendampingi manajer proyek, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk memastikan jawaban manajer proyek telah sesuai dengan indikator pertanyaan yang ingin diukur.

Nilai skor dari manajer proyek pada masing-masing faktor dikalikan dengan bobot faktor, kemudian hasil perhitungan setiap faktor dijumlahkan untuk menghasilkan *Tfactor*. *Tfactor* digunakan untuk menghitung TCF dengan rumus: TCF = 0,6+(0,01 * *TFactor*). Analisis hasil perhitungan TCF ini menghasilkan nilai sebesar 0,89. Pada Tabel 4.34 berikut ditampilkan hasil dari perhitungan masing-masing faktor.

**Tabel 4. 34 Hasil Perhitungan *Technical Complexity Factor* pada AUCP**

| ***Technical Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 1. | *Distributed System Required* | 2 | 4 | 8 |
| 2. | *Response Time is Important* | 1 | 0 | 0 |
| 3. | *End User Efficiency* | 1 | 2 | 2 |
| 4. | *Complex Internal Processing Required* | 1 | 3 | 3 |
| 5. | *Reusable Code Must be a focus* | 1 | 3 | 3 |
| 6. | *Installation Easy* | 0,5 | 0 | 0 |
| 7. | *Usability* | 0,5 | 2 | 1 |
| 8. | *Cross-Platform Support* | 2 | 3 | 6 |
| 9. | *Easy to Change* | 1 | 1 | 1 |
| 10. | *Highly Concurrent* | 1 | 1 | 1 |
| 11. | *Custom Security* | 1 | 1 | 1 |
| 12. | *Dependence on Third-Part Code* | 1 | 3 | 3 |

**Tabel 4. 35 Hasil Perhitungan *Technical Complexity Factor* pada AUCP (Lanjutan)**

| ***Technical Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 13. | *User Training* | 1 | 0 | 0 |
| | | | **Total TFactor** | 29 |
| | **Total TCF** | 0,6 +(0,01 * 29) = 0,89 | | |

Penjelasan Nilai Skor pada Tabel 4.34:

4.4.2.1.1.1.1 Faktor *Distributed System Required* bernilai 4, berarti pemrosesan data pada aplikasi didistribusikan dan data ditransfer ke dua arah.

4.4.2.1.1.1.2 Faktor *Response Time is Important* bernilai 0, berarti tidak ada kebutuhan khusus tentang waktu respon yang ditetapkan oleh klien.

4.4.2.1.1.1.3 Faktor *End User Efficiency* bernilai 2, berarti aplikasi membutuhkan pengimplementasian tombol fungsi yang telah ditentukan sebelumnya, menggunakan warna dan *highlight* visual yang tinggi di layar, mengirimkan *batch task* dari transaksi *online*, dan meminimalkan jumlah layar sebagai usaha meningkatkan efisiensi pengguna.

4.4.2.1.1.1.4 Faktor *Complex Internal Processing Required* bernilai 3, berarti aplikasi mempertimbangkan pemrosesan yang aman, menggunakan pengolahan matematika secara luas, dan pengolahan kompleks untuk mengelola berbagai kemungkinan *input* dan *output*.

4.4.2.1.1.1.5 Faktor *Reusable Code must be a focus* bernilai 3, berarti sebesar 10% atau lebih sistem merancang kode dan artefaknya agar dapat digunakan kembali.

4.4.2.1.1.1.6 Faktor *Installation Easy* bernilai 0, berarti klien tidak memberikan pertimbangan khusus dan tidak ada pengaturan khusus yang diperlukan untuk instalasi.

4.4.2.1.1.1.7 Faktor *Usability* bernilai 2, berarti aplikasi mempertimbangkan adanya proses yang efektif untuk inisialisasi, *back-up*, dan pemulihan yang memerlukan intervensi operator, serta aplikasi harus meminimalkan kebutuhan untuk menggunakan kertas.

4.4.2.1.1.1.8 Faktor *Cross-Platform Support* bernilai 3, berarti desain harus mempertimbangkan kebutuhan agar sistem beroperasi di berbagai *platform*, dan aplikasi dirancang untuk beroperasi di lingkungan *hardware* dan *software* heterogen.

4.4.2.1.1.1.9 Faktor *Easy to Change* bernilai 1, berarti sistem dibangun mempertimbangkan kemudahan untuk dikembangkan di masa depan dengan menggunakan struktur *report* yang fleksibel dalam menangani *simple query*.

4.4.2.1.1.1.10 Faktor *Highly Concurrent* bernilai 1, berarti sistem dirancang untuk mengatasi masalah akses data bersamaan yang kadang terjadi.

4.4.2.1.1.1.11 Faktor *Custom Security* bernilai 1, berarti dalam mendesain sistem memperhitungkan mekanisme keamanan.

4.4.2.1.1.1.12 Faktor *Dependence On Third-Party Code* bernilai 3, berarti dalam pengembangan sistem menggunakan kode yang sudah ada yang disesuaikan sesuai kebutuhan dan digunakan pada sebagian kecil aplikasi, yaitu pada bagian *payment gateway*.

4.4.2.1.1.1.13 Faktor *User Training* bernilai 0, berarti tidak ada kebutuhan khusus untuk pelatihan pengguna.

#### *4.4.2.2 Environmental Complexity Factor*

*Environmental Complexity Factor* (ECF) dibuat untuk mewakili dampak persyaratan non-fungsional yang berhubungan dengan lingkungan dimana proyek dikembangkan terhadap *effort* untuk menyelesaikan proyek. Lembar penilaian untuk mengukur *Environmental Complexity Factor* (ECF) pada proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang telah dinilai oleh *expert judgment*, selanjutnya diserahkan kepada manajer proyek yang bertanggungjawab terhadap proyek tersebut. Manajer proyek memberikan nilai pada setiap faktor lingkungan, dan hasilnya akan dikalikan dengan bobot pada masing-masing faktor. Selama proses pengisian lembar penilaian, peneliti mendampingi manajer proyek, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk memastikan jawaban manajer proyek telah sesuai dengan indikator pertanyaan yang ingin diukur.

Nilai skor dari manajer proyek pada masing-masing faktor dikalikan dengan bobot faktor, kemudian hasil perhitungan setiap faktor dijumlahkan untuk menghasilkan *Efactor*. *Efactor* digunakan untuk menghitung ECF dengan rumus: ECF = 1,4+(-0,03 * *EFactor*). Analisis hasil perhitungan ECF ini menghasilkan nilai sebesar 0,8. Pada Tabel 4.36 berikut ditampilkan hasil dari perhitungan masing-masing faktor.

**Tabel 4. 36 Hasil Perhitungan *Environmental Complexity Factor* dengan AUCP**

| ***Environmental Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 1. | *Familiar with Objectory* | 1,5 | 5 | 7,5 |
| 2. | *Stable requirement* | 2 | 2 | 4 |
| 3. | *Analyst capability* | 0,5 | 4 | 2 |
| 4. | *Application experience* | 0,5 | 3 | 1,5 |
| 5. | *Object-oriented experience* | 1 | 4 | 4 |
| 6. | *Motivation* | 1 | 3 | 3 |
| 7. | *Difficult Programming language* | -1 | 0 | 0 |
| 8. | *Part Time Workers* | -1 | 2 | -2 |
| | | | **Total EFactor** | 20 |
| | **Total ECF** | 1,4+(-0,03*20) = 0,8 | | |

Penjelasan Nilai Skor pada Tabel 4.36:

4.4.2.2.1.1.1 Faktor *Familiarity with Objectory* bernilai 5, berarti lebih dari separuh tim telah menggunakan proses pengembangan (metodologi dan bahasa pemodelan sistem) pada proyek ini di banyak proyek.

4.4.2.2.1.1.2 Faktor *Stable Requirement* bernilai 2, berarti terdapat persyaratan yang tidak stabil di masa lalu, klien meminta beberapa perubahan yang disebabkan oleh persyaratan yang tidak lengkap atau tidak benar.

4.4.2.2.1.1.3 Faktor *Analyst Capability* bernilai 4, berarti analis utama memiliki pengalaman lebih dari dua tahun dalam proyek serupa.

4.4.2.2.1.1.4 Faktor *Application Experience* bernilai 3, berarti sebagian besar tim, yaitu 2 orang, memiliki 18 sampai 24 bulan pengalaman dalam proyek di area yang sama.

4.4.2.2.1.1.5 Faktor *Object-oriented experience* bernilai 4, berarti sebagian besar anggota tim memiliki pengalaman lebih dari dua tahun dalam teknik berorientasi objek.

4.4.2.2.1.1.6 Faktor *Motivation* bernilai 3, berarti tim proyek memiliki motivasi, dan biasanya memiliki inisiatif, namun intervensi manajemen masih diperlukan secara tidak penuh untuk menjaga produktivitas.

4.4.2.2.1.1.7 Faktor *Difficult Programming Language* bernilai 0, berarti bahasa pemrograman mudah dan dalam satu minggu dapat dipahami.

4.4.2.2.1.1.8 Faktor *Part Time Worker* bernilai 2, berarti jumlah pengembang paruh waktu dalam tim proyek mencapai 20%, lebih tepatnya satu orang dari empat orang pada proyek ini adalah pengembang paruh waktu.

### 4.4.3 Perhitungan *Use Case Point*

Nilai *Use Case Point* (UCP) diperoleh dari perkalian antara *Unadjusted Use Case Point* (UUCP), *Environmental Complexity Factor* (ECF) dan *Technical Complexity Factor* (TCF).

**UCP = UUCP*TCF*ECF**
**= 150*0,89*0,8**
**= 106,8 UCP**

Perhitungan tersebut menghasilkan nilai *Use Case Point* (UCP) sebesar 106,8 UCP.

### 4.4.4 Perhitungan *End-User Development Factor*

#### *4.4.4.1 End-User Development Technical Complexity Factor*

*End-User Development Technical Complexity Factor* (EUD_TCF) dibuat untuk mewakili dampak persyaratan penambahan fitur *End User Development* (EUD) pada faktor teknis dalam pengembangan perangkat lunak terhadap *effort* untuk menyelesaikan proyek. Lembar penilaian untuk mengukur *End-User Development Technical Complexity Factor* (EUD_ECF) pada proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang telah dinilai oleh *expert judgment*, selanjutnya diserahkan kepada seorang analis sistem dari CV Gumcode Indonesia yang menjadi perantara pertama saat proses *offering* dan penggalian kebutuhan awal dengan klien. Analis sistem ini berperan sebagai *end-user* yang mengetahui apa saja yang dibutuhkan oleh sistem dan mengetahui informasi sesuai kesepakatan dengan klien.

Analis sistem tersebut memberikan nilai pada setiap faktor teknis pada wilayah *end-user development*, dan hasilnya akan dikalikan dengan bobot pada masing-masing faktor. Selama proses pengisian lembar penilaian, peneliti mendampingi analis sistem, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk memastikan jawaban analis sistem telah sesuai dengan indikator pertanyaan yang ingin diukur.

Nilai skor dari analis sistem pada masing-masing faktor dikalikan dengan bobot faktor, kemudian hasil perhitungan setiap faktor dijumlahkan untuk menghasilkan End-*User Development Tfactor* (EUD_TF). EUD_TF digunakan untuk menghitung EUD_TCF dengan rumus: EUD_TCF = 0,6+(0,01 * EUD_TF). Analisis hasil perhitungan EUD_TCF ini menghasilkan nilai sebesar 0,71. Pada Tabel 4.37 berikut ditampilkan hasil dari perhitungan masing-masing faktor.

**Tabel 4. 37 Hasil Perhitungan *End-User Development* TCF dengan AUCP**

| ***End-User Development Technical Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 1. | *Creating throw away codes* | 0,5 | 1 | 0,5 |
| 2. | *Creating reusable codes* | 1,2 | 1 | 1,2 |
| 3. | *Sharing reusable code* | 1,4 | 1 | 1,4 |
| 4. | *Easy and understandable code* | 1 | 1 | 1 |
| 5. | *Security features in code for more control by end user* | 1,3 | 0 | 0 |
| 6. | *Authentication features* | 1,12 | 1 | 1,12 |
| 7. | *Inbuilt feedback about the correctness* | 1,3 | 1 | 1,3 |
| 8. | *Testable code* | 1,2 | 0 | 0 |
| 9. | *Tools for analyzing by debugging* | 1,4 | 0 | 0 |
| 10. | *Error detection tools* | 1,2 | 0 | 0 |
| 11. | *Online help availability* | 1,3 | 0 | 0 |
| 12. | *Self-efficacy* | 1,11 | 1 | 1,11 |
| 13. | *Perceived ease of use* | 1,2 | 1 | 1,2 |
| 14. | *Perceived usefulness* | 1 | 1 | 1 |
| 15. | *Flexible code* | 1,2 | 0 | 0 |
| 16. | *Scalability features* | 1,25 | 0 | 0 |
| 17. | *Ease of maintenance* | 1,2 | 1 | 1.2 |
| | | | **Total EUD_TF** | 11,03 |
| | **Total EUD_TCF** | 0,6 +(0,01 * 11,03) = 0,71 | | |

Penjelasan Nilai Skor pada Tabel 4.37:

4.4.4.1.1.1.1 Faktor *Creating throw away code* bernilai 1, berarti pengguna akhir membutuhkan tempat atau repositori untuk memfasilitasi kode *throw-away* yang mereka ciptakan.

4.4.4.1.1.1.2 Faktor *Creating Reusable Code* bernilai 1, berarti sistem dirancang dan dikembangkan dengan konsep dan tipe abstraksi yang tepat sesuai konsep pengguna akhir.

4.4.4.1.1.1.3 Faktor *Sharing Reusable Code* bernilai 1, berarti sistem mendukung aktivitas berbagi kode kelas, kumpulan fungsi dan prosedur yang dikembangkan pengguna akhir sebagai kode yang dapat digunakan kembali (*reusable code*).

4.4.4.1.1.1.4 Faktor *Easy and Understandable code* bernilai 1, berarti sistem dikembangkan dengan kode yang mudah dimengerti.

4.4.4.1.1.1.5 Faktor *Security Feature in codes for more control by end-user* bernilai 0, berarti sistem tidak mengembangkan fitur yang mendukung manajemen keamanan.

4.4.4.1.1.1.6 Faktor *Authentication Feature* bernilai 1, berarti sistem mengembangkan fitur pengelolaan izin siapa yang dapat menggunakan sistem, fitur apa saja yang

dapat diakses, sebagai fitur bawaan dan dapat dikustomisasi oleh pengembang.

4.4.4.1.1.1.7 Faktor *Inbuilt Feedback about correctness* bernilai 1, berarti sistem menampilkan pesan peringatan atau kesalahan apabila pada saat pengujian kebenaran program mengalami kesalahan.

4.4.4.1.1.1.8 Faktor *Testable Code* bernilai 0, berarti sistem tidak memecah kode ke dalam lapisan-lapisan atau memisahkan berbagai bagian sistem sehingga kode sulit untuk diuji.

4.4.4.1.1.1.9 Faktor *Tools for Analyzing Debugging* bernilai 0, berarti sistem tidak menyediakan alat untuk mendukung debugging seperti menemukan bug atau menganalisa sejarah eksekusi program.

4.4.4.1.1.1.10 Faktor *Error Detection Tools* bernilai 0, berarti sistem tidak menyediakan alat untuk mendeteksi kesalahan pada program yang diciptakan pengguna akhir.

4.4.4.1.1.1.11 Faktor *Online help availability* bernilai 0, berarti sistem tidak mengembangkan padanan elektronik dari sebuah manual untuk pengguna akhir yang biasanya berupa bantuan berbasis web yang menyediakan hal-hal terbaru dan sumber data yang lebih lengkap untuk menanggapi persoalan teknis berkaitan dengan sistem.

4.4.4.1.1.1.12 Faktor *Self-efficacy* bernilai 1, berarti fitur pada sistem didesain untuk mendukung pengguna akhir dalam menyelesaikan sebuah tugas yang sesuai untuk pria ataupun wanita.

4.4.4.1.1.1.13 Faktor *Perceived of Use* bernilai 1, berarti sistem memudahkan pengguna akhir, sehingga tidak akan merasa kesulitan atau perlu berusaha keras dalam menggunakan sistem.

4.4.4.1.1.1.14 Faktor *Perceived usefulness* bernilai 1, berarti sistem dikembangkan dengan memperhatikan manfaat, sehingga pengguna akhir percaya bahwa dengan menggunakan sistem ini akan membantu mereka melakukan pekerjaan mereka dengan lebih baik.

4.4.4.1.1.1.15 Faktor *Flexible Code* bernilai 0, berarti sistem tidak merancang dan mengembangkan kode yang memudahkan pengguna akhir untuk menambah atau mengubah perilaku kode.

4.4.4.1.1.1.16 Faktor *Scalability feature* bernilai 0, berarti sistem tidak dirancang secara khusus untuk menangani perubahan kebutuhan dan pola penggunaan sumber daya di masa depan.

4.4.4.1.1.1.17 Faktor *Ease of maintenance* bernilai 1, berarti memperhatikan aspek pemeliharaan saat perancangan sistem untuk mendukung modifikasi dan peningkatan sistem ke dalam suatu dokumentasi agar memudahkan proses pemeliharaan.

#### *4.4.4.2 End-User Environmental Complexity Factor*

*End-User Development Environmental Complexity Factor* (EUD_ECF) dibuat untuk mewakili dampak persyaratan penambahan fitur *End User Development* (EUD) pada faktor lingkungan dalam pengembangan perangkat lunak terhadap *effort* untuk menyelesaikan proyek. Lembar penilaian untuk mengukur *End-User Development Environmental Complexity Factor* (EUD_ECF) pada proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT

GHL Peninsula yang telah dinilai oleh *expert judgment*, selanjutnya diserahkan kepada seorang analis sistem dari CV Gumcode Indonesia yang menjadi perantara pertama saat proses *offering* dan penggalian kebutuhan awal dengan klien. Analis sistem ini berperan sebagai pengguna yang mengetahui apa saja yang dibutuhkan oleh sistem dan mengetahui informasi sesuai kesepakatan dengan klien.

Analis sistem tersebut memberikan nilai pada setiap faktor lingkungan pada wilayah *end-user development*, dan hasilnya akan dikalikan dengan bobot pada masing-masing faktor. Selama proses pengisian lembar penilaian, peneliti mendampingi sistem analis, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk memastikan jawaban sistem analis telah sesuai dengan indikator pertanyaan yang ingin diukur.

Nilai skor dari analis sistem pada masing-masing faktor dikalikan dengan bobot faktor, kemudian hasil perhitungan setiap faktor dijumlahkan untuk menghasilkan End-*User Development Efactor* (EUD_EF). EUD_EF digunakan untuk menghitung EUD_ECF dengan rumus: EUD_ECF = 1,4+(0,03 * EUD_EF). Analisis hasil perhitungan EUD_TCF ini menghasilkan nilai sebesar 1,47. Pada Tabel 4.38 berikut ditampilkan hasil dari perhitungan masing-masing faktor.

**Tabel 4. 38 Hasil Perhitungan *End-User Development* ECF dengan AUCP**

| ***End-User Development Environmental Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 1. | *Content level of EUP* | 1,4 | 0 | 0 |
| 2. | *End User Computing capability* | 0,25 | 0 | 0 |
| 3. | *Ease of use and feedback* | 1,2 | 0 | 0 |
| 4. | *Inbuilt system assistance for EUP* | 1,25 | 0 | 0 |
| 5. | *Training and learning time contrain for end user* | 1,12 | 0 | 0 |
| 6. | *Reliability of end user code* | 1,2 | 1 | 1,2 |
| 7. | *End user storage constraint* | 1,02 | 0 | 0 |
| 8. | *Risk factors* | 1,12 | 1 | 1,12 |
| | | | **Total EUD_EF** | 2.32 |
| | **Total EUD_ECF** | 1,4 +(0,03 * 2,32) = 1,47 | | |

Penjelasan Nilai Skor pada Tabel 4.38:

4.4.4.2.1.1.1 Faktor *Content level of EUP* bernilai 0, berarti sistem tidak menawarkan berbagai level modifikasi yang berbeda seiring meningkatnya kompleksitas dari perubahan yang dilakukan pengguna akhir.

4.4.4.2.1.1.2 Faktor *End User Computing Capability* bernilai 0, berarti sistem tidak memfasilitasi kemampuan komputasi setiap pengguna akhir yang menganggap semua pengguna akhir memiliki kemampuan untuk secara efisien mengeksekusi tugas menggunakan sistem.

4.4.4.2.1.1.3 Faktor *Ease of use and feedback* bernilai 0, berarti sistem tidak menyediakan referensi atau panduan (*wizard*), atau *server web* yang terintegrasi untuk tujuan pengujian atau dukungan yang baik selama fase pemasangan sistem yang ketika program selesai tidak memberikan umpan balik yang menyarankan langkah selanjutnya yang diambil oleh pengembang.

4.4.4.2.1.1.4 Faktor *Inbuilt System Assistance for EUP* bernilai 0, berarti sistem tidak menyediakan bantuan untuk mendukung pastisipasi pengguna secara berkelanjutan semacam wiki interaktif yang dikombinasikan dengan fitur penyelesaian otomatis.

4.4.4.2.1.1.5 Faktor *Training and learning time constrain for end user* bernilai 0, berarti tidak ada pelatihan kepada pengguna akhir terhadap lingkungan komputasi pengguna akhir.

4.4.4.2.1.1.6 Faktor *Reliability of End User code* bernilai 1, berarti pengguna akhir mempertimbangkan masalah keandalan perangkat lunak, dan aktivitas yang memperkuat kualitas tersebut seperti pengujian atau debugging.

4.4.4.2.1.1.7 Faktor *End User Storage Constraint* bernilai 0, berarti pengguna akhir tidak memiliki kebutuhan khusus terkait penyimpanan dalam pengembangan penguna akhir.

4.4.4.2.1.1.8 Faktor *Risk Factor* bernilai 1, berarti sistem mendukung pengujian untuk mengurangi persepsi resiko sehingga pengguna akhir bersedia untuk terlibat dalam rekayasa perangkat lunak.

### 4.4.5 Perhitungan *Advance Use Case Point*

Nilai *Advance Use Case Point* (AUCP) diperoleh dari perkalian antara *Use Case Point* (UCP), *End-User Development Technical Complexity Factor* (EUD_TCF) dan *End-User Development Environmental Complexity Factor* (EUD_ECF).

**AUCP = UCP * EUD_TCF * EUD_ECF**
**= 106,8 UCP * 0,71 * 1,47**
**= 111,47 UCP**

Perhitungan tersebut menghasilkan nilai *Advance Use Case Point* (AUCP) sebesar 111,47 UCP.

### 4.4.6 Konversi *Advance Use Case Point* ke Biaya, Waktu dan SDM

Hasil perhitungan metode *Advance Use Case Point* (AUCP) dimanfaatkan untuk menghasilkan biaya, waktu dan sumber daya manusia. Peneliti melakukan konversi nilai AUCP ke dalam biaya, waktu dan sumber daya manusia yang dibutuhkan dalam pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Proses konversi membutuhkan persamaan *effort*, tabel distribusi *effort* per fase, tabel distribusi *effort* tiap anggota tim pengembang dan tabel gaji per bulan tiap anggota tim pengembang. Langkah-langkah dalam mengestimasi biaya, waktu dan sumber daya manusia dijelaskan sebagai berikut:

4.4.6.1.1.1.1 Menganalisis nilai *hours of effort* yang dibutuhkan dengan cara mengkalikan nilai *Advance Use Case Point* dengan 20 *person-hours per-use case point*.

***Hours of effort*** **= AUCP * 20**
**= 111,47 UCP * 20**
**= 2.229,4 jam**

Berdasakan perhitungan di atas, dihasilkan nilai *hours of effort* sebesar 2.229,4 *hours of effort* yang berarti proyek pengembangan perangkat lunak

Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula membutuhkan waktu kerja selama 2.229,4 jam.

4.4.6.1.1.1.2 Nilai *hours of effort* kemudian didistribusikan ke dalam setiap fase untuk menghitung waktu, biaya dan alokasi sumber daya manusia. Berdasarkan Pedoman Distribusi *Effort* pada setiap fase, *hours of effort* dibagi ke dalam dua kelompok aktivitas. Distribusi *effort* pada setiap fase dihitung dengan cara mengkalikan nilai persentase *effort* setiap fase dengan total *hours of effort*. Hasil perhitungan ditunjukkan pada Tabel 4.39.

**Tabel 4. 39 *Hours of Effort* Setiap Aktivitas dengan AUCP**

| Kelompok Aktivitas | | *% Effort* | *Effort* |
|---|---|---|---|
| **1** | ***Software Development*** | | |
| A | *Requirement* | 7,5 % | 167,21 |
| B | *Specification* | 7,5 % | 167,21 |
| C | *Design* | 10 % | 222,94 |
| D | *Implementation* | 10 % | 222,94 |
| E | *Integration Testing* | 7,5 % | 167,21 |
| F | *Acceptance and Deployment* | 7,5 % | 167,21 |
| | **Total** | **50 %** | **1.114,7** |
| **2** | ***On-Going Life Cycle Activity*** | | |
| A | *Project management* | 8,34 % | 185,93 |
| B | *Configuration management* | 4,16 % | 92,74 |
| C | *Quality Assurance* | 8,34 % | 185,93 |
| D | *Documentation* | 4,16 % | 92,74 |
| E | *Training and Support* | 4,16 % | 92,74 |
| F | *Evaluation and Testing* | 20,84 % | 464,61 |
| | **Total** | **50 %** | **1.114,7** |

Keterangan:
*%Effort* = Konstanta Mutlak Distribusi *Effort*
*Effort* = *%Effort * hours of effort*

4.4.6.1.1.1.3 Hasil distribusi *effort* per-fase, selanjutnya dikonversi ke dalam jumlah alokasi waktu dan sumber daya manusia. Berdasarkan penelitian Saleh (2011), waktu yang dibutuhkan untuk mengembangkan perangkat lunak (*Software Development*) sama dengan waktu yang dibutuhkan untuk manajemen proyek (*Project Management*), karena manajemen proyek berjalan sepanjang waktu selama perangkat lunak dikembangkan. Oleh karena itu, perhitungan durasi setiap fase pada *Software Development* mengacu pada *effort* pada fase *Project Management*. Pada fase *Requirement* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek, fase *Specification* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek, fase *Design* dialokasikan waktu sebesar 20% dari waktu pada fase manajemen proyek, fase *Implementation* dialokasikan waktu sebesar 20% dari waktu pada fase manajemen proyek, fase *Integration Testing* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek, dan fase *Acceptance and Deployment* dialokasikan waktu sebesar 15% dari waktu pada fase

manajemen proyek. Alokasi jumlah pengembang yang dibutuhkan pada setiap fase dihitung dengan cara membagi *effort* setiap fase dengan waktu setiap fase (Jam). Tabel 4.40 menampilkan perhitungan alokasi waktu dan sumber daya manusia fase pada *Software Development*.

**Tabel 4. 40 Alokasi SDM dan Waktu fase *Software Development* dengan AUCP**

| Aktivitas | | *%Effort* | *Effort* | Persentase | Jam | Jumlah *Person* |
|---|---|---|---|---|---|---|
| A | *Requirement* | 7,5 % | 167,21 | 15 % | 27,89 | 6 |
| B | *Specification* | 7,5 % | 167,21 | 15 % | 27,89 | 6 |
| C | *Design* | 10 % | 222,94 | 20 % | 37,19 | 6 |
| D | *Implementation* | 10 % | 222,94 | 20 % | 37,19 | 6 |
| E | *Integration Testing* | 7,5 % | 167,21 | 15 % | 27,89 | 6 |
| F | *Acceptance and Deployment* | 7,5 % | 167,21 | 15 % | 27,89 | 6 |
| **Total** | | 50 % | 1.114,7 | 100% | 185,93 | |
| ***Project Management*** | | 8,34 % | 185,93 | | 185.93 | |

Keterangan:
*%Effort* = Konstanta Mutlak Distribusi *Effort*
*Effort* = *%Effort * hours of effort*
Persentase = *%Effort/Total Effort**100
Jam = Persentase * total *effort Project Management*
*Person* = *Effort*/Jam

Pada fase-fase di dalam kelompok aktivitas *On-Going Life Cycle Activity*, setiap fase memiliki alokasi waktu yang sama seperti alokasi *hours of effort*, alokasi sumber daya manusia pada setiap fase masing-masing berjumlah 1 orang kecuali pada fase *Evaluation and Testing*. Hal ini terjadi karena persentase alokasi *effort* pada *Evaluation and Testing* lebih besar daripada fase *Project Management* yang berarti fase *Evaluation and Testing* memiliki jangka waktu lebih lama daripada jangka waktu pada fase *Project Management,* sehingga diperlukan cara agar fase ini dapat selesai di waku bersamaan dengan fase *Project Management*. Cara untuk menghitung alokasi sumber daya manusia yang tepat dapat diperoleh dengan membagi effort pada fase *Evaluation and Testing* dengan *effort* pada fase *Project Management.* Hasil pembagian tersebut dibulatkan menghasilkan nilai alokasi sumber daya manusia pada fase *Evaluation and Testing* sejumlah 3 orang dengan alokasi waktu selama 154,87 jam pada setiap orang. Tabel 4.41 menampilkan perhitungan alokasi waktu dan sumber daya manusia fase pada *On-Going Life Cycle Activity*.

**Tabel 4. 41 Alokasi SDM dan Waktu fase On-Going Activity dengan AUCP**

| Aktivitas | | *%Effort* | *Effort* | Jam | Jumlah *Person* |
|---|---|---|---|---|---|
| A | *Project management* | 8,34 % | 185,93 | 185,93 | 1 |
| B | *Configuration management* | 4,16 % | 92,74 | 92,74 | 1 |
| C | *Quality Assurance* | 8,34 % | 185,93 | 185,93 | 1 |
| D | *Documentation* | 4,16 % | 92,74 | 92,74 | 1 |
| E | *Training and Support* | 4,16 % | 92,74 | 92,74 | 1 |
| F | *Evaluation and Testing* | 20,84 % | 464,61 | 154,87 | 3 |

Keterangan:
*%Effort* = Konstanta Mutlak Distribusi *Effort* (Saleh, 2011)
*Effort* = *%Effort * hours of effort*
Persentase = *%Effort/Total Effort**100
*Person* = *Effort*/Jam

Hasil perhitungan alokasi waktu dan sumber daya manusia pada fase *Software Development* dan *On-Going Life Cycle Activity*, selanjutnya ditampilkan ke dalam Tabel 4.42 yang memuat jumlah pengembang dan waktu yang dibutuhkan pada setiap fase.

**Tabel 4. 42 Pembagian SDM dan Waktu menggunakan AUCP**

| *Requirement* | *Specification* | *Design* | *Implementation* | *Integration Testing* | *Acceptance & Deployment* |
|---|---|---|---|---|---|
| 6 pengembang 27,89 jam | 6 pengembang 27,89 jam | 6 pengembang 37,19 jam | 6 pengembang 37,19 jam | 6 pengembang 27,89 jam | 6 pengembang 27,89 jam |
| *Project Management*; 1 pengembang; 185,93 jam | | | | | |
| *Configuration Management* 1 pengembang 92,74 jam, *Quality Assurance* 1 pengembang 185,93 jam, *Documentation* 1 pengembang 92,74 jam, *Training dan support* 1 pengembang 92,74 jam, *Evaluation dan Testing* 3 pengembang 154,87 jam | | | | | |

Berdasarkan Tabel 4.42 di atas, diperoleh waktu yang dibutuhkan dalam pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang mengacu pada waktu untuk fase manajemen proyek, yaitu selama 185,93 jam atau karena jumlah jam kerja selama satu hari adalah 8 jam, maka 185,93 jam sama dengan 24 hari.

4.4.6.1.1.1.4 Melakukan perhitungan alokasi biaya pada setiap fase, lalu kemudian seluruh hasil perhitungan akan dijumlahkan menghasilkan total biaya yang dibutuhkan untuk pengembangan perangkat lunak. Peran pada setiap fase pada *Software Development* dan *On-Going Life Cycle Activity* ditentukan dengan cara menyesuaikan pekerjaan dengan peran pengembang yang mengangani, sesuai deskripsi pekerjaan pada setiap peran pengembang dalam *Indonesia Salary Guide*. Persentase *effort* diperoleh dari konstanta multak Distibusi *Effort* dan *hours of effort* diperoleh dari perkalian antara persentase distribusi *effort* setiap fase dengan keseluruhan *hours of effort*. Gaji per-jam diperoleh berdasarkan pembagian gaji per-bulan dengan hari kerja dan jam kerja tim proyek, setelah itu digunakan untuk menghitung biaya pada setiap fase dengan cara mengkalikan gaji per-jam dengan *hours of effort*

setiap fase. Biaya setiap fase dijumlahkan menghasilkan total biaya yang dibutuhkan untuk mengembangkan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Tabel 4.43 menyajikan hasil perhitungan biaya yang dibutuhkan pada proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.

**Tabel 4. 43 Perhitungan Alokasi Biaya setiap Aktivitas dengan AUCP**

| | Aktivitas | Peran dalam Salary Guide | *%Effort* | *Hours of Effort* | Gaji Per Jam (Rp) | Total (Rp) |
|---|---|---|---|---|---|---|
| 1 | ***Software Development Phased*** | | | | | |
| A | *Requirement* | *System Analyst* | 7,5 % | 167,21 | 43.750 | 7.315.219 |
| B | *Specification* | *System Analyst* | 7,5 % | 167,21 | 43.750 | 7.315.219 |
| C | *Design* | *System Analyst* | 10 % | 222,94 | 43.750 | 9.753.625 |
| D | *Implementation* | *Software Engineer* | 10 % | 222,94 | 31.250 | 6.966.875 |
| E | *Integration Testing* | *Test Analyst* | 7,5 % | 167,21 | 50.000 | 8.360.250 |
| F | *Acceptance and Deployment* | *Software Engineer* | 7,5 % | 167,21 | 31.250 | 5.225.156 |
| **SubTotal** | | | | | | 44.936.344 |
| 2 | ***On-Going Life Cycle Activities*** | | | | | |
| A | *Project management* | *Project Manager* | 8,34 % | 185,93 | 125.000 | 23.241.495 |
| B | *Configuration management* | *Software Engineer* | 4,16 % | 92,74 | 31.250 | 2.898.220 |
| C | *Quality Assurance* | *Software QA* | 8,34 % | 185,93 | 50.000 | 9.296.598 |
| D | *Documentation* | *Software Analyst* | 4,16 % | 92,74 | 43.750 | 4.057.508 |
| E | *Training and Support* | *Software Engineer* | 4,16 % | 92,74 | 31.250 | 2.898.220 |
| F | *Evaluation and Testing* | *Test Analyst* | 20,84 % | 464,61 | 50.000 | 23.230.348 |
| **SubTotal** | | | | | | 65.622.389 |
| **Total** | | | | | | 110.558.733 |

Estimasi total biaya yang dibutuhkan dalam pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula adalah sebesar Rp110.558.733,00. Alokasi sumber daya manusia atau pengembang dihitung berdasarkan peran dan waktu kerja. Apabila aktivitas membutuhkan peran yang sama di waktu yang berbeda, maka aktivitas dilakukan oleh pengembang yang sama, sedangkan apabila aktivitas membutuhkan peran yang sama di waktu yang sama, maka aktivitas dilakukan oleh pengembang yang berbeda.

Hubungan ketergantungan antar aktivitas pada kelompok aktivitas *Software Development*, yaitu aktivitas *Requirement, Specification, Design, Implementation, Integration Testing,* serta *Acceptance* dan *Deployment* dilakukan secara berurutan (sekuensial) dimana suatu aktivitas baru dapat dimulai ketika aktivitas sebelumnya selesai dikerjakan. Pada kelompok aktivitas *On-Going Life Cycle*, aktivitas dilakukan selama proyek berjalan. Hubungan ketergantungan antar aktivitasnya, yaitu *Project Management* dijalankan sejak proyek dimulai, *Configuration Management* dijalankan bersamaan dengan dimulainya aktivitas *Acceptance* dan *Deployment*, *Quality Assurance* dijalankan ketika masing-masing aktivitas *Software Development* berjalan, sehingga ketika *Requirement* mulai dijalankan, maka *Quality Assurance* untuk Requirement juga ikut dimulai, dan seterusnya. *Documentation* berjalan bersamaan dengan dimulainya proses pengembangan. *Training and Support* berjalan bersamaan dengan aktivitas *Acceptance and Deployment*, sedangkan aktivitas *Evaluation and Testing* dijalankan di setiap paket pekerjaan pada *Implementation,* misalnya pada ketika implementasi halaman *customer* mulai dijalankan, maka *Evaluation and Testing* untuk halaman *customer* juga ikut dimulai, dan seterusnya.

Berdasarkan *work breakdown structure* (WBS) dan hubungan ketergantungan antar aktivitas tersebut, diperoleh total alokasi pengembang sejumlah 26 orang. Detail dari alokasi sumber daya manusia atau pengembang yang dibutuhkan dapat dilihat pada Tabel 4.44.

**Tabel 4. 44 Perhitungan Alokasi SDM dengan AUCP**

| Peran dalam *Salary Guide* | Jumlah SDM |
|---|---|
| *Project Manager* | 1 orang |
| *System Analyst* | 7 orang |
| *Software Engineer* | 8 orang |
| *Software OA* | 1 orang |
| *Test Analyst* | 9 orang |
| **Total** | **26 orang** |

## 4.5 Analisis Perhitungan dengan *Revised Use Case Point*

Perhitungan estimasi *effort* menggunakan metode *Revised Use Case Point* dan *guideline* persebaran *effort* pada masing-masing aktifitas pengembangan perangkat lunak, serta *use case diagram* dan *use case scenario* yang dimodelkan sebelumnya akan digunakan untuk menganalisis perhitungan biaya, waktu dan sumber daya manusia pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Hasil dari analisis perhitungan menggunakan metode *Revised Use Case Point* ini adalah ukuran perangkat lunak, yang kemudian dikonversi ke dalam hours of effort untuk menghitung total biaya, waktu dan sumber daya manusia yang dibutuhkan untuk menyelesaikan pengembangan sistem.

### 4.5.1 Perhitungan *Unadjusted Use Case Point*

#### *4.5.1.1 Perhitungan Unadjusted Actor Weight*

Analisis perhitungan nilai *Unadjusted Actor Weight* (UAW) bertujuan untuk mengetahui bobot dari masing-masing tipe aktor yang berinteraksi dengan sistem, dimana nilai ini diperoleh dengan cara mengidentifikasi dan mengklasifikasikan tipe aktor yang terlibat dengan sistem. Aktor diklasifikasikan ke dalam *simple, average*, *complex* dan *critical*. Aktor diklasifikasikan *simple,* apabila aktor berinteraksi dengan sistem melalui API (*Application Programming Interface).* Aktor diklasifikasikan *average,* apabila aktor berinteraksi dengan sistem melalui antarmuka interaktif atau *Protocol-Driven.* Aktor diklasifikasikan *complex,* apabila aktor berinteraksi dengan sistem melalui *Graphical User Interface* (GUI) atau halaman web. Aktor diklasifikasikan *critical*, apabila aktor berinteraksi dengan sistem melalui modul yang membutuhkan aksi *real-time* atau kompleksitasnya sangat tinggi.

Berdasarkan identifikasi aktor yang diperoleh dari analisis pada *use case diagram* Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang dilampirkan pada Lampiran F, digunakan empat aktor yang terdiri dari: tiga aktor primer dan satu aktor sekunder, yaitu aktor primer terdiri dari: *customer*, vendor dan admin, memiliki kebutuhan untuk mengakses sistem melalui halaman web pada Sistem Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, sedangkan aktor sekunder, yaitu tawk.to, merupakan *plug-in* yang berperan sebagai penyedia fasilitas *chat* antara *customer* dan admin, yang berinteraksi dengan sistem melalui API *Javascript*. Oleh karena itu, aktor primer diklasifikasikan menjadi aktor *complex* dan aktor sekunder diklasifikasikan menjadi aktor *simple*. Penjelasan mengenai peran aktor dan interaksinya dengan sistem ditampilkan pada Tabel 4.45.

**Tabel 4. 45 Klasifikasi Aktor pada Sistem dengan Re-UCP**

| Nama Aktor | | Deskripsi | Tipe |
|---|---|---|---|
| 1. | *Customer* | Melakukan pemesanan paket acara, chat dengan admin, mengisi *review* dan mendaftar vendor melalui halaman *web*. | *Complex* |
| 2. | Vendor | Melihat ringkasan laporan vendor dan notifikasi, mengelola item dan mengubah profil vendor melalui halaman *web*. | *Complex* |
| 3. | Admin | Melihat ringkasan laporan admin, melakukan *chat* dengan *customer*, mengelola data kategori vendor, data vendor, data status vendor, data prosentase profit vendor, sata jenis acara, data ruangan, data paket acara, data *review*, data promosi, data *slider* dan data informasi kantor melalui halaman *web*. | *Complex* |
| 4. | Tawk.to | Memfasilitasi percakapan langsung antara *customer* dan admin, yang berinteraksi dengan sistem melalui API *Javascript*. | *Simple* |

Hasil klasifikasi aktor dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, kemudian digunakan untuk menghitung nilai total *Unadjusted*

*Actor Weight* dengan mengkalikan jumlah aktor dengan bobot pada masing-masing tipe aktor, yang ditunjukkan pada Tabel 4.46.

**Tabel 4. 46 Perhitungan *Unadjusted Actor Weight* dengan Re-UCP**

| Tipe | Bobot | Jumlah Aktor | Bobot x Jumlah Aktor |
|---|---|---|---|
| *Simple* | 1 | 1 | 1 |
| *Average* | 2 | 0 | 0 |
| *Complex* | 3 | 3 | 9 |
| *Critical* | 4 | 0 | 0 |
| | | **Total UAW** | 10 |

### *4.5.1.2 Perhitungan Unadjusted Use Case Weight*

Analisis perhitungan nilai *Unadjusted Use Case Weight* (UUCW) bertujuan untuk mengetahui bobot kompleksitas dari setiap *use case*, dimana kompleksitas suatu *use case* ditentukan oleh jumlah transaksi dari *use case*. Bobot kompleksitas dari setiap *use case* diperoleh dengan cara mengklasifikasikan kompleksitas *use case* dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula ke dalam *simple, average*, *complex* dan *critical* tergantung pada jumlah transaksi sukses dan alur alternatif dalam *use case description*. *Simple use case* memiliki jumlah transaksi kurang dari sama dengan 4 transaksi, *average use case* memiliki jumlah transaksi antara 5 sampai dengan 8 transaksi, *complex use case* memiliki jumlah transaksi antara 9 sampai dengan 15 transaksi, dan *critical use case* memiliki jumlah transaksi lebih dari 15 transaksi.

Berdasarkan *use case scenario* pada Lampiran G, peneliti mengidentifikasi setiap transaksi yang ada pada alur sukses dan alur alternatif, dimana apabila terdapat masing-masing satu transaksi pada skenario sukses dan skenario gagal, transaksi dianggap sama dan dihitung sebagai dua transaksi, serta transaksi yang melibatkan operasi yang sama dalam *database* dan hanya berbeda instansiasi dihitung sebagai transaksi yang sama, meski ditulis menjadi dua skenario yang berbeda dalam *use case scenario.* Kemudian menjumlahkan jumlah transaksi pada alur sukses dan alur alternatif tersebut dan mengklasifikasikannya berdasarkan aturan pada metode *Revised Use Case Point.* Berikut analisis jumlah transaksi pada setiap *use case*dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang ditampilkan pada Tabel 4.47 sampai Tabel 4.74.

**Tabel 4. 47 Transaksi *Use Case Login***

| |
|---|
| **Transaksi 1:**<br>1. Sistem menampilkan *form login* yang meminta *guest* memasukkan alamat *email* dan *password*.<br>2. *Guest* memasukkan *email* dan *password* dengan benar dan memilih fungsi login.<br>3. Sistem memeriksa otentikasi pengguna dan<br>4. Sistem menampilkan halaman utama sistem.<br>**Transaksi 2:**<br>**Email dan Password Salah**<br>Apabila *guest* tidak memasukkan email atau *password* yang benar, sistem menampilkan pesan peringatan yang menyatakan bahwa email dan *password* salah. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case Login* diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 48 Transaksi *Use Case* Chat Admin**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu *chat* admin.<br>2. Sistem menampilkan halaman *chat* admin melalui twak.to.<br>3. *Customer* melakukan komunikasi kepada admin melalui *twak.to*.<br>4. Twak.to menyampaikan pesan yang dikirimkan oleh *customer* kepada admin.<br>**Transaksi 2:**<br>**Koneksi Terputus**<br>Jika dalam keadaan *offline* maka fungsi *chat* admin akan dialihkan dengan memasukkan nama, email dan pesan yang ingin disampaikan kemudian mensubmit sistem, maka *chat* akan masuk ke halaman admin ketika admin sedang *online*. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case Chat Admin* diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 49 Transaksi *Use Case* Melakukan *Order* Paket Acara**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu *Order*.<br>2. Sistem menampilkan halaman *order* yang berisi jenis-jenis paket acara yang tersedia.<br>3. *Customer* memilihjenis acara yang diinginkan.<br>4. Sistem menampilkan daftar paket acara dan paket *custom*.<br>5. *Customer* memilih paket acara yang diinginkan<br>6. Jika *customer* memilih *custom* paket, maka akan menjalankan *subflow* Melakukan Order Paket Custom.<br>7. Sistem menampilkan detail paket acara, seperti layanan dan fasilitas yang didapatkan, harga *per-pack*, dan minimal pemesanan. Sistem meminta *customer* mengisi *form* untuk mengecek ketersediaan gedung.<br>8. *Customer* memasukkan tanggal dan waktu yang diinginkan.<br>9. Sistem menampilkan ketersediaan gedung sesuai tanggal dan waktu yang dipilih *customer* dan meminta *customer* mengisi *form payment*.<br>10. *Customer* memasukkan data pemesanan secara lengkap, beserta kode promosi apabila ada, lalu memilih fungsi *Submit form payment*.<br>11. Sistem menampilkan detail pembayaran yang harus dibayar oleh *customer* dan pilihan media pembayaran, yaitu melalui *midtrans* atau *direct transfer*.<br>12. *Customer* memilih salah satu media pembayaran.<br>13. Sistem menampilkan *form* konfirmasi penyimpanan order.<br>14. *Customer* melakukan konfirmasi dengan memilih Ya.<br>15. Sistem menyimpan pemesanan dan mengirim email berisi *invoice* ke *email customer*, dan menampilkan halaman utama sistem<br>**Transaksi 2:**<br>**Melengkapi Data Pemesanan**<br>Apabila *customer* tidak melengkapi seluruh data yang dibutuhkan pada *form payment* atau pemesanan, maka sistem menampilkan peringatan untuk melengkapi data dan kembali ke langkah 9 pada *Basic Flow.*<br>**Transaksi 3:**<br>**Gedung Penuh** |

**Tabel 4. 50 Transaksi *Use Case* Menambah Review (Lanjutan)**

| |
|---|
| Apabila dalam sistem tanggal yang dipilih untuk memesan gedung tidak tersedia atau sudah ada yang memesan, maka sistem akan menampilkan 'PENUH' dan *customer* diminta untuk mengganti tanggal lain yang tersedia, kemudian melanjutkan ke langkah 9 pada *Basic Flow*. |
| **Total Transaksi:** 3 |
| **Kategori:** Simple |

*Use Case* Melakukan Order Paket Acara diidentifikasi memiliki satu transaksi pada alur sukses dan dua transaksi pada alur alternatif, sehingga memiliki total tiga transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 51 Transaksi *Use Case* Menambah Review**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu *Review*<br>2. Sistem menampilkan testimoni, jumlah *review* dan *rating* saat ini, serta fungsi Berikan *Rating*.<br>3. *Customer* memilih fungsi Berikan *Rating*.<br>4. Sistem menampilkan form untuk mengisi *review*.<br>5. *Customer* memilih berapa *rating* bintang yang diinginkan dan memasukkan ID pemesanan, nama dan *review* di kolom *review*, lalu memilih fungsi Kirim.<br>6. Sistem menyimpan*review* dan ketika sudah selesai kembali ke halaman utama sistem.<br>**Transaksi 2:**<br>**Melengkapi Data *Review***<br>Apabila *customer* tidak mengisi *form review* secara lengkap, maka sistem menampilkan peringatan untuk meminta *customer* melengkapi seluruh kolom, kemudian melanjutkan ke langkah 6 pada *Basic Flow*. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Menambah *Review* diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 52 Transaksi *Use Case* Melakukan Registrasi Vendor**

| |
|---|
| **Transaksi 1:**<br>1. *Customer* memilih menu Registrasi Vendor.<br>2. Sistem menampilkan*form*registrasi vendor dan meminta *customer* mengisi data vendor, yaitu nama vendor, email, alamat, kategori vendor, penanggung jawab, acara pilihan, alamat, No KTP, no telefon, foto profil dan foto KTP.<br>3. *Customer* memasukkan data vendor secara lengkap dan memilih fungsi *Submit*.<br>4. Sistem menyimpan data vendor dan menampilkan pesan konfirmasi bahwa *customer* telah berhasil melakukan pendaftaran.<br>**Transaksi 2:**<br>**Melengkapi Data Vendor**<br>Apabila *customer* tidak mengisi *form* pendaftaran vendor secara lengkap, maka sistem tidak dapat menyimpan data vendor dan meminta vendor memasukkan kembali data vendor, kemudian melanjukan ke langkah 4 pada *Basic Flow.* |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Melakukan Registrasi Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 53 Transaksi *Use Case* Melihat ringkasan laporan vendor**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu *Dashboard*.<br>2. Sistem menampilkan halaman *dashboard* yang memuat jumlah profit, jumlah transaksi, jumlah item, agenda, grafik transaksi, grafik profit vendor, dan agenda berupa kalender. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melihat Ringkasan Laporan Vendor diidentifikasi memiliki hanya satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 54 Transaksi *Use Case* Mengelola transaksi**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu transaksi.<br>2. Sistem menampilkan daftar transaksi yang masuk dan beberapa fungsi, yaitu *filter* untuk melihat transaksi pada periode waktu tertentu, detail transaksi, dan cetak laporan transaksi.<br>3. Vendor memilih salah satu fungsi.<br>Jika vendor memilih untuk melihat detail transaksi, maka akan menjalankan *subflow* Melihat Detail Transaksi.<br>Jika vendor memilih untuk melihat transaksi pada periode waktu tertentu dan status tertentu, maka akan menjalankan *subflow* Memfilter Transaksi.<br>Jika vendor memilih untuk mencetak laporan transaksi, maka akan menjalankan *subflow* Mencetak Laporan Transaksi.<br>Jika vendor memilih untuk mencari transaksi tertentu, maka akan menjalankan *subflow* Mencari Transaksi.<br>**Transaksi 2:**<br>**Data Periode Tertentu Tidak Tersedia**<br>Pada *subflow* Memfilter Transaksi, apabila tidak ditemukan transaksi pada periode waktu yang ditentukan, maka akan menampilkan keterangan "*No Data Available in Table*" dan kembali pada langkah vendor memasukkan tanggal awal acara, tanggal akhir acara dan status transaksi.<br>**Transaksi 3:**<br>**Data Transaksi Tidak Tersedia**<br>Pada *subflow* Mencari Transaksi, apabila data transaksi yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah vendor memasukkan kata kunci. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Transaksi diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*. Transaksi 2 Data Periode Tertentu Tidak Tersedia dan Transaksi 3 Data Transaksi Tidak Tersedia dihitung satu transaksi karena kedua transaksi ini merupakan operasi yang sama pada database yang sama, meskipun menggunakan instansiasi yang berbeda.

**Tabel 4. 55 Transaksi *Use Case* melihat notifikasi**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih ikon notifikasi di menu Transaksi.<br>2. Sistem menampilkan daftar transaksi yang terdaftar beserta transaksi baru yang masuk. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melihat Notifikasi diidentifikasi memiliki hanya satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 56 Transaksi *Use case* mengubah profil vendor**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu *Dashboard*.<br>2. Sistem menampilkan halaman *dashboard* yang memuat fungsi untuk mengubah profil.<br>3. Vendor memilih fungsi *Edit* Profil.<br>4. Sistem menampilkan form *Edit* Data<br>5. Vendor mengubah data vendor sesuai kebutuhan dan memilih fungsi *Submit*.<br>6. Sistem menyimpan perubahan dan menampilkan halaman *dashboard*.<br>**Transaksi 2:**<br>**Membatalkan Perubahan Profil**<br>Apabila vendor pada langkah nomor 6, memilih untuk membatalkan penyimpanan perubahan, maka sistem akan menampilkan halaman *Dashboard* dan *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengubah Profil Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 57 Transaksi *Use case* mengelola item**

| |
|---|
| **Transaksi 1:**<br>1. Vendor memilih menu Item pada halaman utama<br>2. Sistem menampilkan informasi item dan fungsi yang tersedia untuk vendor. Fungsi tersebut yaitu: menambah item, mengubah item dan menghapus item.<br>3. Vendor memilih salah satu fungsi.<br>Jika vendor memilih fungsi untuk menambah item, maka akan menjalankan *subflow* Menambah Item.<br>Jika vendor memilih fungsi untuk mengubah item, maka akan menjalankan *subflow* Mengubah Item.<br>Jika vendor memilih fungsi untuk menghapus item, maka akan menjalankan *subflow* Menghapus Item.<br>**Transaksi 2:**<br>**Data Tidak Sesuai Format**<br>Pada langkah memasukkan data item di *subflow* Menambah Item atau Mengubah Data, apabila vendor tidak memasukkan data pada formulir sesuai dengan format, maka sistem akan menampilkan halaman kosong dan gagal menyimpan, kemudian *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Item diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 58 Transaksi *Use case* melihat ringkasan laporan admin**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Dashboard*.<br>2. Sistem menampilkan halaman *dashboard* yang memuat jumlah profit, jumlah transaksi, jumlah vendor, rating dari review customer, grafik transaksi, grafik profit, dan agenda berupa kalender. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melihat Ringkasan Laporan Admin diidentifikasi memiliki satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 59 Transaksi *Use case* melakukan *chat customer***

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Chat Admin*<br>2. Sistem menampilkan form login dari twak.to<br>3. Admin memasukkan email dan *password*, kemudian *login* ke dalam twak.to<br>4. Admin membalas pesan *customer* yang masuk melalui twak.to |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Melakukan Chat Customer diidentifikasi memiliki satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 60 Transaksi *Use case* mengelola kategori vendor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Kategori Vendor<br>2. Sistem menampilkan daftar kategori vendor dan beberapa fungsi, yaitu pencarian, menambah kategori vendor, mengubah kategori vendor dan menghapus kategori vendor.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah kategori vendor, maka akan menjalankan *Subflow* Menambah Kategori vendor.<br>Jika admin memilih fungsi untuk mengubah kategori vendor, maka akan menjalankan *Subflow* Mengubah Kategori vendor.<br>Jika admin memilih fungsi untuk menghapus kategori vendor, maka akan menjalankan *Subflow* Menghapus Kategori vendor.<br>Jika admin memilih fungsi untuk mencari kategori vendor, maka akan menjalankan *Subflow* Mencari Kategori vendor.<br>**Transaksi 2:**<br>**Data Kategori Vendor Tidak Tersedia**<br>Pada *subflow* Mencari Kategori Vendor, apabila data kategori vendor yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 3:**<br>**Membatalkan Perubahan Data**<br>Apabila pada langkah Memilih Simpan data kategori vendor dari *subflow* Mengubah Kategori Vendor, vendor memilih untuk membatalkan penyimpanan perubahan, maka sistem akan menampilkan daftar kategori vendor dan *use case* selesai. |
| **Total Transaksi:** 3 |
| **Kategori:** Simple |

*Use Case* Mengelola Kategori Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan dua transaksi pada alur alternatif, sehingga memiliki total tiga transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 61 Transaksi *Use case* memverifikasi vendor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Vendor.<br>2. Sistem menampilkan fungsi, yaitu Verifikasi, Aktif dan *Banned*.<br>3. Admin memilih fungsi Verifikasi. |

**Tabel 4. 62 Transaksi *Use case* memverifikasi vendor (Lanjutan)**

| |
|---|
| 4. Sistem menampilkan daftar vendor yang mendaftar namun belum diverifikasi, dan beberapa fungsi, yaitu Detail untuk melihat detail vendor, verifikasi untuk memverifikasi vendor, dan Tolak untuk menolak vendor bergabung sebagai mitra bisnis.<br>5. Admin memilih fungsi Detail Vendor pada vendor yang ingin diverifikasi.<br>6. Sistem menampilkan informasi detail mengenai vendor.<br>7. Admin memilih fungsi kembali untuk kembali ke daftar vendor dan melakukan verifikasi dan memilih fungsi Verifikasi.<br>8. Sistem menampilkan form konfirmasi yang meminta admin untuk memasukkan persentase keuntungan bagi vendor tersebut.<br>9. Admin memasukkan prosentase keuntungan.<br>10. Admin memilih fungsi Ya, Kirim *Password* untuk menyimpan prosentase dan mengirimkan *password* kepada vendor yang telah disetujui untuk bergabung.<br>11. Sistem menutup form konfirmasi dan menyimpan prosentase lalu mengirimkan email kepada vendor yang disetujui berupa *password* untuk mengakses sistem.<br>12. Sistem menampilkan daftar vendor aktif.<br>**Transaksi 2:**<br>**Menolak Verifikasi Vendor**<br>Apanila admin ingin menolak vendor, maka admin memilih fungsi Tolak, kemudian sistem akan menghapus vendor tersebut dan *Use Case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Memverifikasi Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 63 Transaksi *Use case* mengelola status vendor**

| |
|---|
| **Transaksi 1**:<br>1. Admin memilih menu Vendor.<br>2. Sistem menampilkan tiga fungsi, yaitu Verifikasi, Aktif dan *Banned*.<br>3. Admin memilih fungsi Aktif atau *Banned*.<br>Jika admin memilih fungsi untuk mengubah status vendor yang aktif, maka menjalankan *Subflow* Mengubah Status Aktif.<br>Jika admin memilih fungsi untuk mengubah status vendor yang di-*banned*, maka menjalankan *Subflow* Mengubah Status Banned.<br>**Transaksi 2:**<br>**Membatalkan Perubahan Status Vendor**<br>Apabila admin memilih untuk membatalkan penyimpanan perubahan status vendor, maka sistem akan membatalkan perubahan dan menampilkan daftar vendor terdaftar dan *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Status Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 64 Transaksi *Use case* mengubah persentase profit vendor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih fitur Aksi Persentase.<br>2. Sistem menampilkan form untuk mengubah persentase keuntungan vendor.<br>3. Admin mengubah persentase vendor sesuai kebutuhan, lalu memilih fungsi Simpan untuk menyimpan perubahan.<br>4. Sistem menyimpan perubahan dan menampilkan daftar vendor terdaftar yang aktif.<br>**Transaksi 2:** |

| |
|---|
| **Membatalkan perubahan persentase profit**<br>Apabila vendor memilih untuk membatalkan penyimpanan perubahan persentase profit vendor, maka sistem akan membatalkan perubahan dan menampilkan daftar vendor terdaftar dan *use case* selesai. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengubah Persentase Profit Vendor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 65 Transaksi *Use case* mengelola jenis acara**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Jenis Acara.<br>2. Sistem menampilkan data jenis acara dan beberapa fungsi, yaitu pencarian, menambah jenis acara, mengubah jenis acara, menghapus jenis acara dan mencari jenis acara.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah jenis acara, maka akan menjalankan *Subflow* Menambah Jenis Acara.<br>Jika admin memilih fungsi untuk mengubah jenis acara, maka akan menjalankan *Subflow* Mengubah Jenis Acara.<br>Jika admin memilih fungsi untuk menghapus jenis acara, maka akan menjalankan *Subflow* Menghapus Jenis Acara.<br>Jika admin memilih fungsi untuk mencari jenis acara, maka akan menjalankan *Subflow* Mencari Jenis Acara.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data jenis acara pada *Subflow* Menambah Jenis Acara atau Mengubah Jenis Acara, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan bahwa data tidak tersimpan dan meminta admin untuk melengkapi data.<br>**Transaksi 3:**<br>**Data Jumlah Nominal bukan format angka**<br>Pada langkah memasukkan data jenis acara pada *Subflow* Menambah Jenis Acara atau Mengubah Jenis Acara, apabila admin memasukkan jumlah nominal pembayaran awal selain format angka, maka setelah disimpan, jumlah nominal pembayaran awal dirubah menjadi 0.<br>**Transaksi 4:**<br>**Data Jenis Acara Tidak Tersedia**<br>Pada *subflow* Mencari Jenis Acara, apabila data jenis acara yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 5:**<br>**Membatalkan Hapus Jenis Acara**<br>Apabila admin memilih untuk membatalkan hapus jenis acara, maka sistem tidak melakukan penghapusan jenis acara dan *use case* selesai. |
| **Total Transaksi:** 5 |
| **Kategori:** Average |

*Use Case* Mengelola Jenis Acara diidentifikasi memiliki satu transaksi pada alur sukses dan empat transaksi pada alur alternatif, sehingga memiliki total lima transaksi yang diklasifikasikan dalam kategori *average*.

**Tabel 4. 66 Transaksi *Use case* mengelola ruangan**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Ruangan. |

**Tabel 4. 67 Transaksi *Use case* mengelola ruangan (Lanjutan)**

| |
|---|
| 2. Sistem menampilkan data ruangan dan beberapa fungsi, yaitu pencarian, menambah ruangan, mengubah ruangan, menghapus ruangan dan mencari ruangan.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah ruangan, maka akan menjalankan *Subflow* Menambah Ruangan.<br>Jika admin memilih fungsi untuk mengubah ruangan, maka akan menjalankan *Subflow* Mengubah Ruangan.<br>Jika admin memilih fungsi untuk menghapus ruangan, maka akan menjalankan *Subflow* Menghapus Ruangan.<br>Jika admin memilih fungsi untuk mencari ruangan, maka akan menjalankan *Subflow* Mencari Ruangan.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data ruangan pada *Subflow* Menambah Ruangan atau Mengubah Ruangan, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan bahwa data tidak tersimpan dan meminta admin untuk melengkapi data.<br>**Transaksi 3:**<br>**Data Ruangan Tidak Tersedia**<br>Pada subflow Mencari Ruangan, apabila data ruangan yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 4:**<br>**Membatalkan Hapus Ruangan**<br>Apabila admin memilih untuk membatalkan hapus ruangan, maka sistem tidak melakukan penghapusan ruangan dan *use case* selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Simple |

*Use Case* Mengelola Ruangan diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 68 Transaksi *Use case* mengelola paket acara**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Paket.<br>2. Sistem menampilkan data paket acara dan beberapa fungsi, yaitu pencarian, menambah paket acara, mengubah paket acara, menghapus paket acara dan mencari paket acara.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah paket acara, maka akan menjalankan *Subflow* Menambah Paket Acara.<br>Jika admin memilih fungsi untuk mengubah paket acara, maka akan menjalankan *Subflow* Mengubah Paket Acara.<br>Jika admin memilih fungsi untuk menghapus paket acara, maka akan menjalankan *Subflow* Menghapus Paket Acara.<br>Jika admin memilih fungsi untuk mencari paket acara, maka akan menjalankan *Subflow* Mencari Paket Acara.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data paket pada *Subflow* Menambah Paket Acara atau Mengubah Paket Acara, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem akan tetap bertahan pada halaman yang sama sampai admin mengisi seluruh kolom.<br>**Transaksi 3:**<br>**Data Paket Acara Tidak Tersedia** |

**Tabel 4. 69 Transaksi *Use case* mengelola paket acara (Lanjutan)**

| |
|---|
| Pada *subflow* Mencari Paket Acara, apabila data ruangan yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 4:**<br>**Membatalkan Hapus Paket Acara**<br>Apabila admin memilih untuk membatalkan hapus paket acara, maka sistem tidak melakukan penghapusan paket acara dan *use case* selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Simple |

*Use Case* Mengelola Paket Acara diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 70 Transaksi *Use case* mengubah status review**

| |
|---|
| **Transaksi 1:**<br>1. Adminmemilih menu *Review*.<br>2. Sistem menampilkan data *review* dan fungsi untuk mengubah status *review*.<br>3. Admin memilih fungsi *Edit* pada *review* yang ingin dirubah.<br>4. Sistem menampilkan *form* untuk mengubah status review.<br>5. Admin merubah status *review* sesuai kebutuhan dan menyimpan perubahan dengan memilih fungsi *Save changes.*<br>6. Sistem menyimpan perubahan status *review* dan menampilkan daftar *review* beserta yang berhasil dirubah. |
| **Total Transaksi:** 1 |
| **Kategori:** Simple |

*Use Case* Mengubah Status Review diidentifikasi memiliki satu transaksi pada alur sukses, sehingga memiliki total satu transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 71 Transaksi *Use case* mengelola promosi**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu Promosi.<br>2. Sistem menampilkan data promosi dan beberapa fungsi, yaitu pencarian, menambah promosi, mengubah promosi, menghapus promosi dan mencari promosi.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah promosi, maka akan menjalankan *Subflow* Menambah Promosi.<br>Jika admin memilih fungsi untuk mengubah promosi, maka akan menjalankan *Subflow* Mengubah Promosi.<br>Jika admin memilih fungsi untuk menghapus promosi, maka akan menjalankan *Subflow* Menghapus Promosi.<br>Jika admin memilih fungsi untuk mencari promosi, maka akan menjalankan *Subflow* Mencari Promosi.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data promosi pada *Subflow* Menambah Promosi atau Mengubah Promosi, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem akan tetap bertahan pada halaman yang sama dan meminta admin mengisi seluruh kolom.<br>**Transaksi 3:**<br>**Data Promosi Tidak Tersedia**<br>Pada *subflow* **Mencari Promosi**, apabila data promosi yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci.<br>**Transaksi 4:**<br>**Membatalkan Hapus Promosi** |

| |
|---|
| Apabila admin memilih untuk membatalkan hapus promosi, maka sistem tidak melakukan penghapusan promosi dan use case selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Simple |

*Use Case* Mengelola Promosi diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 72 Transaksi *Use case* mengelola slider**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Slider*.<br>2. Sistem menampilkan data *slider* dan beberapa fungsi, yaitu pencarian, menambah *slider*, mengubah *slider*, menghapus *slider*dan mencari *slider*.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk menambah *slider*, maka akan menjalankan *Subflow* Menambah Slider.<br>Jika admin memilih fungsi untuk mengubah *slider*, maka akan menjalankan *Subflow* Mengubah Slider.<br>Jika admin memilih fungsi untuk menghapus *slider*, maka akan menjalankan *Subflow* Menghapus Slider.<br>Jika admin memilih fungsi untuk mencari *slider*, maka akan menjalankan *Subflow* Mencari Slider.<br>**Transaksi 2:**<br>**Data Tidak Lengkap**<br>Pada langkah memasukkan data slider pada *Subflow* Menambah Slider atau Mengubah Slider, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan bahwa data tidak tersimpan dan meminta admin untuk melengkapi data.<br>**Transaksi 3:**<br>**Data Slider Tidak Tersedia**<br>Pada *subflow* Mencari Slider, apabila data slider yang dicari tidak terdaftar, maka akan menampilkan keterangan "*No Matching Records Found*" dan kembali pada langkah admin memasukkan kata kunci pada *subflow* Mencari Slider.<br>**Transaksi 4:**<br>**Membatalkan Hapus Slider**<br>Apabila admin memilih untuk membatalkan hapus slider, maka sistem tidak melakukan penghapusan slider dan *use case* selesai. |
| **Total Transaksi:** 4 |
| **Kategori:** Simple |

*Use Case* Mengelola Slider diidentifikasi memiliki satu transaksi pada alur sukses dan tiga transaksi pada alur alternatif, sehingga memiliki total empat transaksi yang diklasifikasikan dalam kategori *simple*.

**Tabel 4. 73 Transaksi *Use case* memperbarui data kantor**

| |
|---|
| **Transaksi 1:**<br>1. Admin memilih menu *Footer*.<br>2. Sistem menampilkan data alamat dan data media sosial perusahaan dan fungsi untuk mengubah data alamat dan mengubah data media sosial.<br>3. Admin memilih salah satu fungsi.<br>Jika admin memilih fungsi untuk mengubah alamat perusahaan, maka akan menjalankan *Subflow* Mengubah Alamat.<br>Jika admin memilih fungsi untuk mengubah nama akun media sosial perusahaan, maka akan menjalankan *Subflow* Mengubah Media Sosial.<br>**Transaksi 2:** |

**Tabel 4. 74 Transaksi *Use case* memperbarui data kantor (Lanjutan)**

| |
|---|
| **Data Tidak Lengkap**<br>Pada langkah memasukkan data alamat pada *Subflow* Mengubah Alamat atau Memasukkan data media sosial pada *Subflow* Mengubah Media Sosial, apabila admin tidak mengisi data yang pada form secara lengkap, maka sistem menampilkan pesan peringatan yang meminta admin untuk melengkapi data. |
| **Total Transaksi:** 2 |
| **Kategori:** Simple |

*Use Case* Mengelola Data Kantor diidentifikasi memiliki satu transaksi pada alur sukses dan satu transaksi pada alur alternatif, sehingga memiliki total dua transaksi yang diklasifikasikan dalam kategori *simple*.

Hasil analisis jumlah transaksi dan kategori *use case* pada setiap *use case* dari Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan metode *Revised Use Case Point*, kemudian digunakan untuk menghitung nilai total *Unadjusted Use Case Weight* dengan cara mengkalikan jumlah *use case* pada setiap kategori dengan bobot pada masing-masing kategori *use case*, yang ditunjukkan pada Tabel 4.75.

**Tabel 4. 75 Perhitungan *Unadjusted Use Case Weight* dengan Re-UCP**

| Kategori | Bobot | Jumlah *Use Case* | Bobot x Jumlah *Use Case* |
|---|---|---|---|
| *Simple* | 5 | 22 | 110 |
| *Average* | 10 | 1 | 10 |
| *Complex* | 15 | 0 | 0 |
| *Critical* | 20 | 0 | 0 |
| | | **Total UUCW** | 120 |

Nilai *Unadjusted Actor Weight* dan *Unadjusted Use Case Weight* yang telah diketahui, kemudian digunakan untuk menghitung nilai *Unajusted Use Case Point*. Nilai *Unajusted Use Case Point* diperoleh dari penjumlahan nilai *Unadjusted Actor Weight* dan *Unadjusted Use Case Weight*.

**UUCP = UAW + UUCW**
**= 10 + 120 = 130 UUCP**

Hasil perhitungan *Unadjusted Use Case Point* menghasilkan nilai sejumlah 130 UUCP.

## 4.5.2 Perhitungan Complexity Factor

### *4.5.2.1 Technical Complexity Factor*

*Technical Complexity Factor* (TCF) dibuat untuk mewakili dampak persyaratan non-fungsional proyek terhadap *effort*untuk menyelesaikan proyek. Lembar penilaian untuk mengukur *Technical Complexity Factor* (TCF) pada proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang telah dinilai oleh *expert judgment*, selanjutnya diserahkan kepada manajer proyek yang bertanggungjawab terhadap proyek tersebut. Manajer proyek memberikan nilai pada setiap faktor teknis, dan hasilnya akan dikalikan dengan bobot pada masing-masing faktor. Selama proses pengisian lembar penilaian, peneliti mendampingi manajer proyek, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk

memastikan jawaban manajer proyek telah sesuai dengan indikator pertanyaan yang ingin diukur.

Nilai skor dari manajer proyek pada masing-masing faktor dikalikan dengan bobot faktor, kemudian hasil perhitungan setiap faktor dijumlahkan untuk menghasilkan *Tfactor*. *Tfactor* digunakan untuk menghitung TCF dengan rumus: TCF = 0,6+(0,01 * *TFactor*). Analisis hasil perhitungan TCF ini menghasilkan nilai sebesar 0,91. Pada Tabel 4.76 berikut ditampilkan hasil dari perhitungan masing-masing faktor.

**Tabel 4. 76 Hasil Perhitungan *Technical Complexity Factor* dengan Re-UCP**

| ***Technical Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 1. | *Distributed System Required* | 2 | 4 | 8 |
| 2. | *Response Time is Important* | 1 | 0 | 0 |
| 3. | *End User Efficiency* | 1 | 2 | 2 |
| 4. | *Complex Internal Processing Required* | 1 | 3 | 3 |
| 5. | *Reusable Code Must be a focus* | 1 | 3 | 3 |
| 6. | *Installation Easy* | 0,5 | 0 | 0 |
| 7. | *Usability* | 0,5 | 2 | 1 |
| 8. | *Cross-Platform Support* | 2 | 3 | 6 |
| 9. | *Easy to Change* | 1 | 1 | 1 |
| 10. | *Highly Concurrent* | 1 | 1 | 1 |
| 11. | *Custom Security* | 1 | 1 | 1 |
| 12. | *Dependence on Third-Part Code* | 1 | 3 | 3 |
| 13. | *User Training* | 1 | 0 | 0 |
| 14. | *Scalability* | 2 | 1 | 2 |
| | | | **Total TFactor** | 31 |
| | **Total TCF** | 0,6 +(0,01 * 31) = 0,91 | | |

Penjelasan Nilai Skor pada Tabel 4.76:

4.5.2.1.1.1.1 Faktor *Distributed System Required* bernilai 4, berarti pemrosesan data pada aplikasi didistribusikan dan data ditransfer ke dua arah.

4.5.2.1.1.1.2 Faktor *Response Time is Important* bernilai 0, berarti tidak ada kebutuhan khusus tentang waktu respon yang ditetapkan oleh klien.

4.5.2.1.1.1.3 Faktor *End User Efficiency* bernilai 2, berarti aplikasi membutuhkan pengimplementasian tombol fungsi yang telah ditentukan sebelumnya, menggunakan warna dan *highlight* visual yang tinggi di layar, mengirimkan *batch task* dari transaksi *online*, dan meminimalkan jumlah layar sebagai usaha meningkatkan efisiensi pengguna.

4.5.2.1.1.1.4 Faktor *Complex Internal Processing Required* bernilai 3, berarti aplikasi mempertimbangkan pemrosesan yang aman, menggunakan pengolahan matematika secara luas, dan pengolahan kompleks untuk mengelola berbagai kemungkinan *input* dan *output*.

4.5.2.1.1.1.5 Faktor *Reusable Code must be a focus* bernilai 3, berarti sebesar 10% atau lebih sistem merancang kode dan artefaknya agar dapat digunakan kembali.

4.5.2.1.1.1.6 Faktor *Installation Easy* bernilai 0, berarti klien tidak memberikan pertimbangan khusus dan tidak ada pengaturan khusus yang diperlukan untuk instalasi.

4.5.2.1.1.1.7 Faktor *Usability* bernilai 2, berarti aplikasi mempertimbangkan adanya proses yang efektif untuk inisialisasi, back-up, dan pemulihan yang memerlukan intervensi operator, serta aplikasi harus meminimalkan kebutuhan untuk menggunakan kertas.

4.5.2.1.1.1.8 Faktor *Cross-Platform Support* bernilai 3, berarti desain harus mempertimbangkan kebutuhan agar sistem beroperasi di berbagai *platform*, dan aplikasi dirancang untuk beroperasi di lingkungan *hardware* dan *software* heterogen.

4.5.2.1.1.1.9 Faktor *Easy to Change* bernilai 1, berarti sistem dibangun mempertimbangkan kemudahan untuk dikembangkan di masa depan dengan menggunakan struktur *report* yang fleksibel dalam menangani *simple query*.

4.5.2.1.1.1.10 Faktor *Highly Concurrent* bernilai 1, berarti sistem dirancang untuk mengatasi masalah akses data bersamaan yang kadang terjadi.

4.5.2.1.1.1.11 Faktor *Custom Security* bernilai 1, berarti dalam mendesain sistem memperhitungkan mekanisme keamanan.

4.5.2.1.1.1.12 Faktor *Dependence On Third-Party Code* bernilai 3, berarti dalam pengembangan sistem menggunakan kode yang sudah ada yang disesuaikan sesuai kebutuhan dan digunakan pada sebagian kecil aplikasi, yaitu pada bagian *payment gateway*.

4.5.2.1.1.1.13 Faktor *User Training* bernilai 0, berarti tidak ada kebutuhan khusus untuk pelatihan pengguna.

4.5.2.1.1.1.14 Faktor *Scalability* bernilai 1, berarti pengembang mempertimbangkan aspek skalabilitas sekedar untuk menangani beban kerja yang mungkin meningkat, namun tidak ada persyaratan khusus dari klien.

#### *4.5.2.2 Environmental Complexity Factor*

*Environmental Complexity Factor* (ECF) dibuat untuk mewakili dampak persyaratan non-fungsional yang berhubungan dengan lingkungan dimana proyek dikembangkan terhadap *effort* untuk menyelesaikan proyek. Lembar penilaian untuk mengukur *Environmental Complexity Factor* (ECF) pada proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang telah dinilai oleh *expert judgment*, selanjutnya diserahkan kepada manajer proyek yang bertanggungjawab terhadap proyek tersebut. Manajer proyek memberikan nilai pada setiap faktor lingkungan, dan hasilnya akan dikalikan dengan bobot pada masing-masing faktor. Selama proses pengisian lembar penilaian, peneliti mendampingi manajer proyek, kemudian melakukan konfirmasi ulang atas setiap jawaban yang diberikan. Hal ini dilakukan untuk memastikan jawaban manajer proyek telah sesuai dengan indikator pertanyaan yang ingin diukur.

Nilai skor dari manajer proyek pada masing-masing faktor dikalikan dengan bobot faktor, kemudian hasil perhitungan setiap faktor dijumlahkan untuk menghasilkan *Efactor*. *Efactor* digunakan untuk menghitung ECF dengan rumus: ECF = 1,4+(-0,03 * *EFactor*). Analisis hasil perhitungan ECF ini menghasilkan nilai sebesar 0,71. Pada Tabel 4.77 berikut ditampilkan hasil dari perhitungan masing-masing faktor.

**Tabel 4. 77 Hasil Perhitungan *Environmental Complexity Factor* pada Re-UCP**

| ***Environmental Complexity Factor*** | | **Bobot** | **Skor (0-5)** | **B x Skor** |
|---|---|---|---|---|
| 1. | *Familiar with Objectory* | 1,5 | 5 | 7,5 |
| 2. | *Stable requirement* | 2 | 2 | 4 |
| 3. | *Analyst capability* | 0,5 | 4 | 2 |
| 4. | *Application experience* | 0,5 | 3 | 1,5 |
| 5. | *Object-oriented experience* | 1 | 4 | 4 |
| 6. | *Motivation* | 1 | 3 | 3 |
| 7. | *Difficult Programming language* | -1 | 0 | 0 |
| 8. | *Part Time Workers* | -1 | 2 | -2 |
| 9. | *Project Methodology* | 1 | 3 | 3 |
| | | | **Total EFactor** | 23 |
| | **Total ECF** | 1,4+(-0,03*23) = 0,71 | | |

Penjelasan Nilai Skor pada Tabel 4.77:

4.5.2.2.1.1.1 Faktor *Familiarity with Objectory* bernilai 5, berarti lebih dari separuh tim telah menggunakan proses pengembangan (metodologi dan bahasa pemodelan sistem) pada proyek ini di banyak proyek.

4.5.2.2.1.1.2 Faktor *Stable Requirement* bernilai 2, berarti terdapat persyaratan yang tidak stabil di masa lalu, klien meminta beberapa perubahan yang disebabkan oleh persyaratan yang tidak lengkap atau tidak benar.

4.5.2.2.1.1.3 Faktor *Analyst Capability* bernilai 4, berarti analis utama memiliki pengalaman lebih dari dua tahun dalam proyek serupa.

4.5.2.2.1.1.4 Faktor *Application Experience* bernilai 3, berarti sebagian besar tim, yaitu 2 orang, memiliki 18 sampai 24 bulan pengalaman dalam proyek di area yang sama.

4.5.2.2.1.1.5 Faktor *Object-oriented experience* bernilai 4, berarti sebagian besar anggota tim memiliki pengalaman lebih dari dua tahun dalam teknik berorientasi objek.

4.5.2.2.1.1.6 Faktor *Motivation* bernilai 3, berarti tim proyek memiliki motivasi, dan biasanya memiliki inisiatif, namun intervensi manajemen masih diperlukan secara tidak penuh untuk menjaga produktivitas.

4.5.2.2.1.1.7 Faktor *Difficult Programming Language* bernilai 0, berarti bahasa pemrograman mudah dan dalam satu minggu dapat dipahami.

4.5.2.2.1.1.8 Faktor *Part Time Worker* bernilai 2, berarti jumlah pengembang paruh waktu dalam tim proyek mencapai 20%, lebih tepatnya satu orang dari empat orang pada proyek ini adalah pengembang paruh waktu.

4.5.2.2.1.1.9 Faktor *Project Methodology* bernilai 3, berarti pengalaman pengembang terhadap metodologi yang digunakan dalam proyek cukup berpengalaman atau berpengalaman sedang.

### 4.5.3 Perhitungan *Revised Use Case Point*

Nilai *Revised Use Case Point* (Re-UCP) diperoleh dari perkalian antara *Unadjusted Use Case Point* (UUCP), *Environmental Complexity Factor* (ECF) dan *Technical Complexity Factor* (TCF).

**Re-UCP = UUCP*TCF*ECF**
**= 130 UUCP * 0,91 * 0,71**
**= 83,99 UCP**

Perhitungan tersebut menghasilkan nilai *Revised Use Case Point* (Re-UCP) sebesar 83,99 UCP.

### 4.5.4 Konversi Revised Use Case Point ke Biaya, Waktu dan SDM

Hasil perhitungan metode *Revised Use Case Point* (Re-UCP) dimanfaatkan untuk menghasilkan biaya, waktu dan sumber daya manusia. Peneliti melakukan konversi nilai Re-UCP ke dalam biaya, waktu dan sumber daya manusia yang dibutuhkan dalam pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Proses konversi membutuhkan persamaan *effort*, tabel distribusi *effort* per fase, tabel distribusi *effort* tiap anggota tim pengembang dan tabel gaji per bulan tiap anggota tim pengembang. Langkah-langkah dalam mengestimasi biaya, waktu dan sumber daya manusia dijelaskan sebagai berikut:

4.5.4.1.1.1.1 Menganalisis nilai *hours of effort* yang dibutuhkan dengan cara mengkalikan nilai *Revised Use Case Point* dengan 20 *person-hours per-use case point*.

***Hours of effort* = Re-UCP * 20**
**= 83,99 UCP * 20**
**=1.679,8 jam**

Berdasarkan perhitungan di atas, dihasilkan nilai *hours of effort* sebesar 1.679,8 *hours of effort* yang berarti proyek pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula membutuhkan waktu kerja selama 1.680,8 jam.

4.5.4.1.1.1.2 Nilai *hours of effort* kemudian didistribusikan ke dalam setiap fase untuk menghitung waktu, biaya dan alokasi sumber daya manusia. Berdasarkan *Guideline* Distribusi *Effort* pada setiap fase, *hours of effort* dibagi ke dalam 2 kelompok aktivitas. Distribusi *effort* pada setiap fase dihitung dengan cara mengkalikan nilai persentase *effort* setiap fase dengan total *hours of effort*. Hasil perhitungan ditunjukkan pada Tabel 4.78.

**Tabel 4. 78 *Hours of Effort* Setiap Aktivitas dengan Re-UCP**

| Kelompok Aktivitas | | *% Effort* | *Effort* |
|---|---|---|---|
| **1** | ***Software Development*** | | |
| A | *Requirement* | 7,5 % | 125,99 |
| B | *Specification* | 7,5 % | 125,99 |
| C | *Design* | 10 % | 167,98 |
| D | *Implementation* | 10 % | 167,98 |
| E | *Integration Testing* | 7,5 % | 125,99 |
| F | *Acceptance and Deployment* | 7,5 % | 125,99 |
| | **Total** | **50 %** | 839,9 |
| **2** | ***On-Going Life Cycle Activity*** | | |
| A | *Project management* | 8,34 % | 140,1 |
| B | *Configuration management* | 4,16 % | 69,88 |

**Tabel 4. 79 *Hours of Effort* Setiap Aktivitas dengan Re-UCP (Lanjutan)**

| Kelompok Aktivitas | | *% Effort* | *Effort* |
|---|---|---|---|
| C | *Quality Assurance* | 8,34 % | 140,1 |
| D | *Documentation* | 4,16 % | 69,88 |
| E | *Training and Support* | 4,16 % | 69,88 |
| F | *Evaluation and Testing* | 20,84 % | 350,07 |
| **Total** | | **50 %** | 839,9 |

Keterangan:
*%Effort* = Konstanta Mutlak Distribusi *Effort*
*Effort* = *%Effort * hours of effort*

4.5.4.1.1.1.3 Hasil distribusi *effort* per-fase, selanjutnya dikonversi ke dalam jumlah alokasi waktu dan sumber daya manusia. Berdasarkan penelitian Saleh (2011), waktu yang dibutuhkan untuk mengembangkan perangkat lunak (*Software Development*) sama dengan waktu yang dibutuhkan untuk manajemen proyek (*Project Management*), karena manajemen proyek berjalan sepanjang waktu selama perangkat lunak dikembangkan. Oleh karena itu, perhitungan durasi setiap fase pada *Software Development* mengacu pada *effort* pada fase *Project Management*. Pada fase *Requirement* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek, fase *Specification* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek, fase *Design* dialokasikan waktu sebesar 20% dari waktu pada fase manajemen proyek, fase *Implementation* dialokasikan waktu sebesar 20% dari waktu pada fase manajemen proyek, fase *Integration Testing* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek, dan fase *Acceptance and Deployment* dialokasikan waktu sebesar 15% dari waktu pada fase manajemen proyek. Alokasi jumlah pengembang yang dibutuhkan pada setiap fase dihitung dengan cara membagi *effort* setiap fase dengan waktu setiap fase (Jam). Tabel 4.80 menampilkan perhitungan alokasi waktu dan sumber daya manusia fase pada *Software Development*.

**Tabel 4. 80 Alokasi SDM dan Waktu fase Software Development pada Re-UCP**

| Aktivitas | | *%Effort* | *Effort* | Persentase | Jam | *Person* |
|---|---|---|---|---|---|---|
| A | *Requirement* | 7,5 % | 125,99 | 15 % | 21,01 | 6 |
| B | *Specification* | 7,5 % | 125,99 | 15 % | 21,01 | 6 |
| C | *Design* | 10 % | 167,98 | 20 % | 28,02 | 6 |
| D | *Implementation* | 10 % | 167,98 | 20 % | 28,02 | 6 |
| E | *Integration Testing* | 7,5 % | 125,99 | 15 % | 21,01 | 6 |
| F | *Acceptance and Deployment* | 7,5 % | 125,99 | 15 % | 21,01 | 6 |
| **Total** | | 50 % | 839,90 | 100 % | 140,1 | |
| **Project Management** | | 8,34 % | 140,1 | | 140,1 | |

Keterangan:
%Effort = Konstanta Mutlak Distribusi Effort (Saleh, 2011)

Effort = %Effort * hours of effort
Persentase = %Effort/Total Effort*100
Jam = Persentase * total effort Project Management
Person = Effort/Jam

Pada fase-fase di dalam kelompok aktivitas *On-Going Life Cycle Activity*, setiap fase memiliki alokasi waktu yang sama seperti alokasi *hours of effort*, alokasi sumber daya manusia pada setiap fase masing-masing berjumlah 1 orang kecuali pada fase *Evaluation and Testing*. Hal ini terjadi karena persentase alokasi *effort* pada *Evaluation and Testing* lebih besar daripada fase *Project Management* yang berarti fase *Evaluation and Testing* memiliki jangka waktu lebih lama daripada jangka waktu pada fase *Project Management,* sehingga diperlukan cara agar fase ini dapat selesai di waku bersamaan dengan fase *Project Management*. Cara untuk menghitung alokasi sumber daya manusia yang tepat dapat diperoleh dengan membagi effort pada fase *Evaluation and Testing* dengan *effort* pada fase *Project Management.* Hasil pembagian tersebut dibulatkan menghasilkan nilai alokasi sumber daya manusia pada fase *Evaluation and Testing* sejumlah 3 orang dengan alokasi waktu selama 116,69 jam pada setiap orang. Tabel 4.81 menampilkan perhitungan alokasi waktu dan sumber daya manusia fase pada On-Going Life Cycle Activity.

**Tabel 4. 81 Alokasi SDM dan Waktu fase On-Going Activity pada Re-UCP**

| | **Aktivitas** | ***%Effort*** | ***Effort*** | **Jam** | ***Person*** |
|---|---|---|---|---|---|
| A | *Project management* | 8,34 % | 140,1 | 140,1 | 1 |
| B | *Configuration management* | 4,16 % | 69,88 | 69,88 | 1 |
| C | *Quality Assurance* | 8,34 % | 140,1 | 140,1 | 1 |
| D | *Documentation* | 4,16 % | 69,88 | 69,88 | 1 |
| E | *Training and Support* | 4,16 % | 69,88 | 69,88 | 1 |
| F | *Evaluation and Testing* | 20,84 % | 350,07 | 116,69 | 3 |

Keterangan:
%Effort = Konstanta Mutlak Distribusi Effort
Effort = %Effort * hours of effort
Persentase = %Effort/Total Effort*100
Person = Effort/Jam

Hasil perhitungan alokasi waktu dan sumber daya manusia pada fase *Software Development* dan *On-Going Life Cycle Activity*, selanjutnya ditampilkan ke dalam Tabel 4.82 yang memuat jumlah pengembang dan waktu yang dibutuhkan pada setiap fase.

**Tabel 4. 82 Pembagian SDM dan Waktu menggunakan Re-UCP**

| ***Requirement*** | ***Specification*** | ***Design*** | ***Implementation*** | ***Integration Testing*** | ***Acceptance & Deployment*** |
|---|---|---|---|---|---|
| 6 pengembang 21,01 jam | 6 pengembang 21,01 jam | 6 pengembang 28,02 jam | 6 pengembang 28,02 jam | 6 pengembang 21.01 jam | 6 pengembang 21,01 jam |
| *Project Management*; 1 pengembang; 140,1 jam | | | | | |
| *Configuration Management* 1 pengembang 69,88 jam, *Quality Assurance* 1 pengembang 140,10 jam, *Documentation* 1 pengembang 69,88 jam, *Training dan support* 1 pengembang 69,88 jam, *Evaluation dan Testing* 3 pengembang 116,69 jam | | | | | |

Berdasarkan Tabel 4.82 di atas, diperoleh waktu yang dibutuhkan dalam pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang mengacu pada waktu untuk fase manajemen proyek, yaitu selama 140,1 jam atau karena jumlah jam kerja selama satu hari kerja adalah 8 jam, maka 140,1 jam sama dengan 18 hari.

4.5.4.1.1.1.4 Melakukan perhitungan alokasi biaya pada setiap fase, lalu kemudian seluruh hasil perhitungan akan dijumlahkan menghasilkan total biaya yang dibutuhkan untuk pengembangan perangkat lunak. Peran pada setiap fase pada *Software Development* dan *On-Going Life Cycle Activity* ditentukan dengan cara menyesuaikan pekerjaan dengan peran pengembang yang mengangani, sesuai deskripsi pekerjaan pada setiap peran pengembang dalam *Indonesia Salary Guide*. Persentase *effort* diperoleh dari konstanta multak Distribusi *Effort* dan *hours of effort* diperoleh dari perkalian antara persentase distribusi *effort* setiap fase dengan keseluruhan *hours of effort*. Gaji per-jam diperoleh berdasarkan pembagian gaji per-bulan dengan hari kerja dan jam kerja tim proyek, setelah itu digunakan untuk menghitung biaya pada setiap fase dengan cara mengkalikan gaji per-jam dengan *hours of effort* setiap fase. Biaya setiap fase dijumlahkan menghasilkan total biaya yang dibutuhkan untuk mengembangkan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Tabel 4.83 menyajikan hasil perhitungan biaya yang dibutuhkan pada proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula.

**Tabel 4. 83 Perhitungan Alokasi Biaya setiap Aktivitas dengan Re-UCP**

| | **Aktivitas** | **Peran dalam Salary Guide** | ***%Effort*** | ***Hours of Effort*** | **Gaji Per Jam (Rp)** | **Biaya Per Aktivitas (Rp)** |
|---|---|---|---|---|---|---|
| 1 | ***Software Development Phased*** | | | | | |
| A | *Requirement* | *System Analyst* | 7,5 % | 125,99 | 43.750 | 5.511.844 |
| B | *Specification* | *System Analyst* | 7,5 % | 125,99 | 43.750 | 5.511.844 |
| C | *Design* | *System Analyst* | 10 % | 167,98 | 43.750 | 7.349.125 |
| D | *Implementation* | *Software Engineer* | 10 % | 167,98 | 31.250 | 5.249.375 |
| E | *Integration Testing* | *Test Analyst* | 7,5 % | 125,99 | 50.000 | 6.299.250 |
| F | *Acceptance and Deployment* | *Software Engineer* | 7,5 % | 125,99 | 31.250 | 3.937.031 |
| **SubTotal** | | | | | | 33.858.469 |
| 2 | ***On-Going Life Cycle Activities*** | | | | | |
| A | *Project management* | *Project Manager* | 8,34 % | 140,10 | 125.000 | 17.511.915 |
| B | *Configuration management* | *Software Engineer* | 4,16 % | 69,88 | 31.250 | 2.183.740 |

**Tabel 4. 84 Perhitungan Alokasi Biaya setiap Aktivitas dengan Re-UCP (Lanjutan)**

| Aktivitas | | Peran dalam Salary Guide | *%Effort* | *Hours of Effort* | Gaji Per Jam (Rp) | Biaya Per Aktivitas (Rp) |
|---|---|---|---|---|---|---|
| C | *Quality Assurance* | *Software QA* | 8,34 % | 140,10 | 50.000 | 7.004.766 |
| D | *Documentation* | *Software Analyst* | 4,16 % | 69,88 | 43.750 | 3.057.236 |
| E | *Training and Support* | *Software Engineer* | 4,16 % | 69,88 | 31.250 | 2.183.740 |
| F | *Evaluation and Testing* | *Test Analyst* | 20,84 % | 350,07 | 50.000 | 17.503.516 |
| **SubTotal** | | | | | | 49.444.913 |
| **Total** | | | | | | 83.303.382 |

Estimasi total biaya yang dibutuhkan dalam pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula adalah sebesar Rp83.303.382,00. Alokasi sumber daya manusia atau pengembang dihitung berdasarkan peran dan waktu kerja. Apabila aktivitas membutuhkan peran yang sama di waktu yang berbeda, maka aktivitas dilakukan oleh pengembang yang sama, sedangkan apabila aktivitas membutuhkan peran yang sama di waktu yang sama, maka aktivitas dilakukan oleh pengembang yang berbeda.

Hubungan ketergantungan antar aktivitas pada kelompok aktivitas *Software Development*, yaitu aktivitas *Requirement, Specification, Design, Implementation, Integration Testing,* serta *Acceptance* dan *Deployment* dilakukan secara berurutan (sekuensial) dimana suatu aktivitas baru dapat dimulai ketika aktivitas sebelumnya selesai dikerjakan. Pada kelompok aktivitas *On-Going Life Cycle*, aktivitas dilakukan selama proyek berjalan. Hubungan ketergantungan antar aktivitasnya, yaitu *Project Management* dijalankan sejak proyek dimulai, *Configuration Management* dijalankan bersamaan dengan dimulainya aktivitas *Acceptance* dan *Deployment*, *Quality Assurance* dijalankan ketika masing-masing aktivitas *Software Development* berjalan, sehingga ketika *Requirement* mulai dijalankan, maka *Quality Assurance* untuk Requirement juga ikut dimulai, dan seterusnya. *Documentation* berjalan bersamaan dengan dimulainya proses pengembangan. *Training and Support* berjalan bersamaan dengan aktivitas *Acceptance and Deployment*, sedangkan aktivitas *Evaluation and Testing* dijalankan di setiap paket pekerjaan pada *Implementation,* misalnya pada ketika implementasi halaman *customer* mulai dijalankan, maka *Evaluation and Testing* untuk halaman *customer* juga ikut dimulai, dan seterusnya.

Berdasarkan *work breakdown structure* (WBS) dan hubungan ketergantungan antar aktivitas, diperoleh total alokasi pengembang sejumlah 26 orang. Pada Tabel 4.89 disajikan analisis alokasi sumber daya manusia yang dibutuhkan.

**Tabel 4. 85 Perhitungan Alokasi SDM dengan Re-UCP**

| Peran dalam Salary Guide | Jumlah |
|---|---|
| *Project Manager* | 1 orang |
| *System Analyst* | 7 orang |
| *Software Engineer* | 8 orang |
| *Software OA* | 1 orang |
| *Test Analyst* | 9 orang |
| **Total** | **26 orang** |

## 4.6 Analisis Perhitungan CV Gumcode Indonesia

Analisis terhadap perhitungan yang digunakan oleh CV Gumcode Indonesia bertujuan untuk mengetahui bagaimana cara menghasilkan perkiraan biaya, waktu dan jumlah anggota tim untuk menyelesaikan pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula. Berdasarkan hasil wawancara pada Lampiran A, manajer proyek menentukan biaya, waktu dan jumlah anggota tim dengan mempertimbangkan jenis platform yang digunakan, fitur yang dibutuhkan, serta seberapa sulit logika yang dibangun, kemudian menggali pengalaman untuk proyek serupa di masa lalu, mencoba mengingat rentang waktu proyek serupa dan ukuran tim, hingga akhirnya perkiraan dihasilkan berdasarkan campuran dugaan dan perhitungan. Gaji untuk anggota tim magang tergantung pada berapa nilai proyek yang akan dikembangkan dan berapa task yang dikerjakan pada setiap fase pengembangan. Alokasi waktu dalam penjadwalan digambarkan dalam bentuk *timeline* sederhana yang kurang detail. Berikut adalah rangkuman hasil wawancara kepada manajer proyek mengenai alokasi biaya, waktu dan sumber daya manusia pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula:

4.6.1.1.1.1.1 Pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula membutuhkan waktu selama 6 minggu, terhitung sejak pertengahan bulan November 2017 sampai dengan Januari awal 2018.

4.6.1.1.1.1.2 Anggota tim pengembang Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula terdiri dari 5 orang, sebagai berikut:

- 1 Orang *Project manager*
- 1 Orang *System Analyst*
- 1 Orang *Programmer web admin*
- 1 Orang *Programmer web vendor*
- 1 Orang *Programmer front-end*

Berikut deskripsi tugas pada masing-masing anggota tim:

- *Project manager*, bertugas memimpin jalannya proyek, mendefinisikan lingkup kerja, membentuk tim, menganalisis biaya dan waktu yang dibutuhkan untuk pengembangan, mengawasi jalannya proyek, dan membuat laporan kemajuan setiap minggu untuk disampaikan kepada management dan tim, serta sebagai perantara antara management dengan tim pengembang.

- *System analyst*, bertugas melakukan analisis terhadap pernagkat lunak yang akan dibangun, termasuk analisis kebutuhan (*requirement analysis*), melakukan wawancara, dan melakukan perancangan terhadap sistem yang akan dibangun.
- *Programmer web admin*, bertugas mengimplementasikan web admin sesuai perancangan yang dilakukan sistem analis.
- *Programmer web vendor*, bertugas mengimplementasikan web vendor sesuai perancangan yang dilakukan sistem analis.
- *Programmer front-end*, bertugas merancang desain antarmuka (front-end) sistem.

4.6.1.1.1.1.3 Biaya yang dibutuhkan untuk pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula adalah sebesar ± Rp40.000.000,00.

## 4.7 Analisis Penjadwalan dengan *Gantt Chart*

Pembuatan *Gantt Chart* harus diawali dengan analisis terhadap penjadwalan. Penjadwalan dilakukan untuk memberikan informasi mengenai jadwal rencana dan kemajuan proyek terkait kinerja sumber daya berupa biaya, rencana durasi proyek, *progress* waktu dan tenaga kerja untuk menyelesaikan proyek. Analisis penjadwalan akan melibatkan pengalokasian waktu pada masing-masing pekerjaan, waktu yang tersedia, serta perkiraan hari-hari libur dan waktu yang hilang, biaya pada masing-masing pekerjaan, jumlah tenaga kerja yang tersedia, untuk kemudian menetapkan batas-batas waktu kapan untuk mulai dan selesai dari masing-masing tugas dan menetapkan *milestone* yang harus dicapai pada setiap fase aktivitas.

### 4.7.1 *Gantt Chart* pada *Advance Use Case Point*

Penjadwalan pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula dibuat berdasarkan *Work Breakdown Structure* (WBS) yang dilengkapi dengan tanggal mulai dan tanggal berakhir setiap task, biaya dan jumlah sumber daya manusia. Penjadwalan digambarkan dalam bentuk *gantt chart* yang memanfaatkan hasil analisis perhitungan biaya, waktu dan sumber daya manusia dari metode *Advance Use Case Point*. Tabel 4.86 menampilkan tabulasi biaya, waktu dan jumlah sumber daya manusia yang digunakan untuk penjadwalan proyek pengembangan perangkat lunak.

**Tabel 4. 86 Tabulasi Distribusi Biaya, Waktu dan SDM pada AUCP**

| Aktivitas | Jumlah Pengembang | Durasi Per Aktivitas (jam) | Biaya Per Aktivitas (Rp) |
|---|---|---|---|
| *Requirement* | 6 | 27,89 | 7.315.219 |
| *Specification* | 6 | 27,89 | 7.315.219 |
| *Design* | 6 | 37,19 | 9.753.625 |
| *Implementation* | 6 | 37,19 | 6.966.875 |
| *Integration Testing* | 6 | 27,89 | 8.360.250 |
| *Acceptance and Deployment* | 6 | 27,89 | 5.225.156 |

**Tabel 4. 87 Tabulasi Distribusi Biaya, Waktu dan SDM pada AUCP (Lanjutan)**

| | | | |
|---|---|---|---|
| *Project management* | 1 | 185,93 | 23.241.495 |
| *Configuration management* | 1 | 92,74 | 2.898.220 |
| *Quality Assurance* | 1 | 185,93 | 9.296.598 |
| *Documentation* | 1 | 92,74 | 4.057.508 |
| *Training and Support* | 1 | 92,74 | 2.898.220 |
| *Evaluation and Testing* | 3 | 154,87 | 23.230.348 |

Pembuatan *Gantt Chart* pada proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan aplikasi Microsoft Project 2013 dimana berisi seluruh aktivitas per-fase beserta keterangan kapan tanggal mulai (kolom *start*), tanggal selesai (kolom *finish*), nama posisi pengembang (kolom *resource)*, jumlah pengembang (kolom *Number of Resource*) dan biaya (kolom *cost*). *Gantt Chart* pada metode *Advance Use Case Point* dilampirkan pada Lampiran J.

### 4.7.2 *Gantt Chart* pada *Revised Use Case Point*

Penjadwalan pengembangan perangkat lunak Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula dibuat berdasarkan *Work Breakdown Structure* (WBS) yang dilengkapi dengan tanggal mulai dan tanggal berakhir setiap task, biaya dan jumlah sumber daya manusia. Penjadwalan digambarkan dalam bentuk *gantt chart* yang memanfaatkan hasil analisis perhitungan biaya, waktu dan sumber daya manusia dari metode *Revised Use Case Point*. Tabel 4.88 menampilkan tabulasi biaya, waktu dan jumlah sumber daya manusia yang digunakan untuk penjadwalan proyek pengembangan perangkat lunak.

**Tabel 4. 88 Tabulasi Distribusi Biaya, Waktu dan SDM pada Re-UCP**

| Aktivitas | Jumlah Pengembang | Durasi Per Aktivitas (jam) | Biaya Per Aktivitas (Rp) |
|---|---|---|---|
| *Requirement* | 6 | 21,01 | 5.511.844 |
| *Specification* | 6 | 21,01 | 5.511.844 |
| *Design* | 6 | 28,02 | 7.349.125 |
| *Implementation* | 6 | 28,02 | 5.249.375 |
| *Integration Testing* | 6 | 21,01 | 6.299.250 |
| *Acceptance and Deployment* | 6 | 21,01 | 3.937.031 |
| *Project management* | 1 | 140,1 | 17.511.915 |
| *Configuration management* | 1 | 69,88 | 2.183.740 |
| *Quality Assurance* | 1 | 140,1 | 7.004.766 |
| *Documentation* | 1 | 69,88 | 3.057.236 |
| *Training and Support* | 1 | 69,88 | 2.183.740 |
| *Evaluation and Testing* | 3 | 116,69 | 17.503.516 |

Pembuatan *Gantt Chart* pada proyek pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan aplikasi Microsoft Project 2013 dimana berisi seluruh aktivitas per-fase beserta keterangan kapan tanggal mulai (kolom *start*), tanggal selesai (kolom *finish*), nama posisi pengembang (kolom *resource)*, jumlah pengembang (kolom *Number of Resource*) dan biaya (kolom *cost*). *Gantt Chart* pada metode *Revised Use Case Point* dilampirkan pada Lampiran K.

# BAB 5 PEMBAHASAN

## 5.1 Perbandingan Durasi Hasil Perhitungan Metode Advance Use Case Point, Revised Use Case Point dan CV. Gumcode Indonesia

Durasi merupakan total waktu yang diperlukan untuk menyelesaikan pekerjaan tertentu, dengan mempertimbangkan waktu tidak produktif, seperti waktu istirahat, liburan dan cuti sakit. Satuan untuk durasi biasanya adalah jam, hari, bulan atau tahun (Trendowicz & Jeffery, 2014). Analisis perbandingan durasi hasil dari perhitungan metode *Advance Use Case Point* (AUCP), *Revised Use Case Point* (Re-UCP) dengan perhitungan dari CV Gumcode Indonesia bertujuan untuk mengetahui faktor-faktor yang mempengaruhi perbedaan pada setiap hasil, dan dampaknya bagi hasil yang diperoleh, yang nantinya disimpulkan dan digunakan sebagai informasi bagi CV Gumcode Indonesia dalam menentukan metode yang lebih baik untuk menghitung estimasi waktu yang dibutuhkan untuk pengembangan perangkat lunak yang dibutuhkan di masa mendatang.

Hasil perhitungan durasi dari CV Gumcode Indonesia merupakan nilai yang ditawarkan dan disepakati oleh kedua belah pihak, yaitu CV Gumcode Indonesia sendiri dan klien, dimana dalam realisasinya mengalami keterlambatan dari jadwal sebenarnya. Pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula seharusnya selesai dalam waktu 18 hari sesuai kontrak kerja, namun kenyataannya selesai dalam waktu 30 hari. Keterlambatan ini menunjukkan bahwa metode estimasi yang digunakan oleh CV Gumcode Indonesia kurang tepat.

Berdasarkan hasil perhitungan, urutan waktu dari cepat ke lambat adalah hasil perhitungan dari metode *Revised Use Case Point* selama 18 hari kerja, metode *Advance Use Case Point* selama 24 hari kerja, dan CV Gumcode Indonesia selama 30 hari kerja. Tabel 5.1 akan menjelaskan tentang faktor-faktor yang mempengaruhi perbedaan hasil perhitungan waktu dan dampaknya pada masing-masing metode.

**Tabel 5. 1 Perbandingan Hasil Perhitungan Waktu**

| ***Advance Use Case Point (AUCP)*** | ***Revised Use Case Point (Re-UCP)*** | **Perhitungan CV Gumcode Indonesia** |
|---|---|---|
| Perhitungan dengan metode *Advance Use Case Point* menghasilkan waktu pengembangan selama 185,93 jam (24 hari). Hasil ini lebih cepat daripada perhitungan CV Gumcode Indonesia, dan lebih lambat daripada hasil metode *Revised Use Case Point*. Beberapa faktor yang menjadi penyebabnya, antara lain: | Perhitungan dengan metode *Revised Use Case Point* menghasilkan waktu pengembangan selama 140,1 jam (18 hari). Hasil ini lebih cepat daripada hasil perhitungan CV Gumcode Indonesia dan metode Advance Use Case Point. Beberapa faktor yang menjadi penyebabnya, antara lain: | Perhitungan CV Gumcode Indonesia menghasilkan waktu pengembangan paling lambat, yaitu 30 hari, dibandingkan hasil dari metode *Advance Use Case Point* dan *Revised Use Case Poin.* Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1. Perhitungan berasal dari pertimbangan jumlah |

**Tabel 5. 2 Perbandingan Hasil Perhitungan Waktu (Lanjutan)**

| *Advance Use Case Point (AUCP)* | *Revised Use Case Point (Re-UCP)* | Perhitungan CV Gumcode Indonesia |
|---|---|---|
| 1. Menggunakan nilai person-hours menurut Karner (1993), yaitu 20 person hours per-UCP.<br>2. Menggunakan Pedoman Distribusi Effort sesuai penelitian Saleh (2011).<br>3. Nilai distribusi *effort* yang lebih besar pada fase menajemen proyek, karena adanya fitur tambahan *End-User Development* pada proses pengembangan.<br>4. Penggunaan *Technical Complexity Factor* sejumlah 13 faktor, dan *Environmental Complexity Factor* sejumlah 8 faktor. | 1. Menggunakan nilai person-hours menurut Karner (1993), yaitu 20 person hours per-UCP.<br>2. Menggunakan Pedoman Distribusi Effort sesuai penelitian Saleh (2011).<br>3. Nilai distribusi *effort* yang lebih kecil pada manajemen proyek, karena adanya pengklasifikasian kompleksitas aktor dan *use case* ke dalam kategori yang berbeda.<br>4. Penggunaan *Technical Complexity Factor* (TCF) sejumlah 14 faktor dan *Environmental Complexity Factor* (ECF) sejumlah 9 faktor. | anggota tim proyek, fitur yang dibutuhkan dan tingkat kesulitan logika, lalu mengingat proyek serupa di masa lalu, dimana beberapa anggota tim memiliki peran ganda pada aktivitas implementasi dan testing.<br>2. Tidak melakukan perubahan pada timeline jadwal apabila terdapat perubahan pada jadwal atau keterlambatan pada suatu aktivitas.<br>3. Pengembang terkadang bekerja lebih dari 8 jam dalam sehari.<br>4. Seringkali dokumentasi perancangan dikerjakan oleh pengembang magang. |

Perhitungan alokasi waktu pada metode *Advance Use Case Point* dan *Revised Use Case Point* menggunakan Pedoman Distribusi *Effort* yang sama, yaitu Pedoman Distribusi *Effort* dari penelitian Saleh (2011). Berdasarkan pedoman tersebut, waktu yang dibutuhkan untuk pengembangan perangkat lunak sama dengan waktu yang didistribusikan pada fase aktivitas manajemen proyek (*Project Management*) dikarenakan aktivitas manajemen proyek berjalan sepanjang waktu selama perangkat lunak dikembangkan. Kedua metode juga menggunakan aturan nilai *person-hours* yang sama untuk menghitung nilai *hours of effort*, yaitu 20 *person-hours* per-UCP milik Karner (1993).

Faktor lain pada metode *Advance Use Case Point* sehingga menghasilkan waktu 24 hari adalah karena adanya fitur tambahan *End-User Development* pada proses pengembangan. Adanya fitur tambahan ini menyebabkan perhitungan nilai *effort* yang dibutuhkan dilakukan dengan cara mengkalikan nilai *Use Case Point* (UCP) dengan *End-User Development Technical Complexity Factor* (EUD_TCF) dan *End-User Development Environmental Factor* (EUD_ECF), dimana nilai UCP sendiri didapatkan dari perkalian nilai *Unadjusted Use Case Point* (UUCP) dengan *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) yang menyebabkan nilai *effort* lebih besar daripada metode *Revised Use Case Point*. Nilai *effort* yang lebih besar tersebut dikalikan nilai *person-hours* menghasilkan nilai *hours of effort*, kemudian nilai *hours of effort* dikalikan dengan persentase distribusi *effort* pada fase manajemen proyek menghasilkan total waktu yang lebih

lama untuk pengembangan perangkat lunak, yaitu 185,93 jam atau 24 hari. Total waktu ini lebih mendekati waktu aktual yang dibutuhkan untuk mengembangkan sistem, yaitu 30 hari. Terdapat perbedaan selama 6 hari yang kemungkinan disebabkan karena adanya perbedaan jumlah alokasi pengembang dan tambahan fitur *End-User Development* yang belum diperhitungkan. Pada Re-UCP memperkerjakan 26 pengembang, sedangkan CV Gumcode Indonesia hanya memperkerjakan 5 orang.

Nilai *effort* yang lebih besar membutuhkan waktu yang lebih lama dan biaya yang lebih tinggi untuk mengembangkan perangkat lunak. Namun, penambahan fitur *End-User Development* ini meningkat kualitas dan kepuasan pengguna akhir karena memfasilitasi pengguna akhir untuk dapat mengembangkan dan menyesuaikan sistem itu sendiri (Srivastava dkk., 2015), sehingga penggunaan metode *Advance Use Case Point* lebih sesuai digunakan apabila terdapat persyaratan atau kebutuhan dari pengguna akhir atau klien yang menginginkan sistem didesain untuk memperbolehkan pengguna akhir melakukan pengembangan atau penyesesuaian terhadap sistem itu sendiri.

Pada metode *Revised Use Case Point* (Re-UCP) menghasilkan waktu 140,1 jam atau 18 hari karena adanya pengklasifikasian kompleksitas aktor dan *use case* dengan range yang lebih luas. Kategori *simple* pada *use case* memiliki jumlah transaksi kurang dari empat, dan *average* memiliki jumlah transaksi lima sampai dengan delapan (Kirmani & Wahid, 2015), sehingga pada penelitian ini menghasilkan nilai *Unadjusted Use Case Weight* (UUCW) yang lebih kecil. Kemudian terdapat faktor teknis tambahan pada *Technical Complexity Factor* (TCF) yang menyebabkan nilai TCF lebih tinggi sehingga menghasilkan nilai Re-UCP yang lebih besar, dan faktor lingkungan tambahan pada *Environmental Complexity Factor* (ECF) yang menyebabkan nilai ECF menjadi lebih tinggi sehingga akan menghasilkan nilai Re-UCP yang semakin kecil. Nilai Re-UCP yang lebih kecil tersebut dikalikan nilai *person-hours* menghasilkan nilai *hours of effort*, kemudian nilai *hours of effort* dikalikan dengan persentase distribusi *effort* pada fase manajemen proyek menghasilkan total waktu yang lebih cepat untuk pengembangan perangkat lunak, yaitu 140,1 jam atau 18 hari. Total waktu ini sama seperti kesepakatan yang disetujui oleh klien dan CV Gumcode Indonesia, namun pada kenyataanya pengembangan membutuhkan waktu 30 hari. Hal ini terjadi kemungkinan karena adanya perbedaan jumlah alokasi pengembang, dimana pada metode Re-UCP memperkerjakan 26 pengembang, sedangkan CV Gumcode Indonesia hanya memperkerjakan 5 pengembang.

Selain beberapa faktor di atas, peneliti melakukan wawancara terkait keberadaan proses-proses yang ada pada *Project Schedule Management* yang dilampirkan pada Lampiran B. *Project Schedule Management* memiliki enam proses di dalamnya, yaitu *Plan Schedule Management, Define Activities, Sequence Activities, Estimate Activity Duration, Develop Schedule* dan *Control Schedule.*

Pada proses *Plan Schedule Management*, terdapat aktivitas menetapkan kebijakan, prosedur, dan dokumentasi untuk perencanaan, pengembangan, pengelolaan, pelaksanaan, dan pengendalian jadwal proyek berupa panduan dan

arahan mengenai bagaimana jadwal proyek akan dikelola selama proyek berlangsung (PMI, Inc,. 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula telah menjalankan proses ini dengan membuat *work breakdown structure* (WBS) beserta durasi aktivitas sebelum memulai proses pengembangan, kemudian menentukan jumlah *pengembangf-days* untuk setiap sumber daya yang dibutuhkan. Namun, durasi aktivitas tersebut masih berupa perkiraan dari manajer proyek dan belum didokumentasikan ke dalam dokumen khusus berisi rencana pengelolaan jadwal, sehingga tidak tersedianya panduan dan arahan tentang bagaimana jadwal proyek akan dikelola di seluruh proyek. Rencana yang tidak dikembangkan dan digunakan dengan benar untuk memandu proyek dapat menyebabkan kemungkinan proyek gagal (Attarzadeh & Ow, 2008).

Pada proses *Define Activity*, terdapat aktivitas mengidentifikasi dan mendokumentasikan tindakan spesifik yang akan dilakukan untuk menghasilkan *deliverable* proyek (PMI, Inc,. 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula telah mendefinisikan ruang lingkup dengan WBS yang berisi rincian dari aktivitas-aktivitas yang dibutuhkan ke dalam *Project Charter*. Di dalam *Project Charter* tersebut telah tercantum kapan tanggal mulai dan akhir proyek pengembangan akan selesai. Namun, tim proyek tidak menetapkan *milestone* pada titik tertentu dan tidak mengubah daftar aktivitas secara periodik selama proyek berjalan. Adanya proses mendefinisikan aktivitas memberikan manfaat untuk memperkirakan, menjadwalkan, melaksanakan, memantau dan mengendalikan pekerjaan proyek (PMI, Inc., 2013), namun adanya perubahan saat proyek berjalan dan tidak adanya *milestone* dan penalti jika mengalami keterlambatan dapat menyebabkan kemungkinan proyek gagal dan biaya yang berlebih, seperti yang terjadi pada proyek Trilogy milik *U.S Federal Bureau of Investigation* (FBI) dimana lebih dari satu tahun terlambat dan melebihi anggaran sebesar $200 juta (Schwalbe, 2014).

Pada proses *Sequence Activity*, terdapat proses mengidentifikasi dan mendokumentasikan hubungan dan ketergantungan antar aktivitas proyek (PMI, Inc., 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula tidak mendokumentasikan hubungan antar aktivitas secara khusus. Beberapa aktivitas dilakukan secara paralel, dan beberapa lainnya dilakukan secara berurutan. Urutan aktivitas direpresentasikan ke dalam bentuk *timeline* sederhana, yang berisi nama aktivitas dan durasi yang dialokasikan ke dalam aktivitas tersebut. Bentuk *timeline* tidak termasuk ke dalam bentuk grafis yang disarankan pada proses *Sequence Avtivity*. Bentuk yang disarankan misalnya, *gantt chart, precedence diagramming method* (PDM), *network diagram*, *critical path analysis*, dsb (Marchewnka, 2003). Tidak adanya hubungan dan ketergantungan aktivitas yang didefinisikan dengan jelas, dapat mengurangi efisiensi terkait kendala proyek (PMI, Inc., 2013) dan tidak dapat menggunakan bentuk penjadwalan yang disarankan, yaitu *network diagram* dan *critical path analysis* (Schwalbe, 2014).

Pada proses *Estimate Activity Duration*, terdapat proses memperkirakan jumlah *effort* kerja yang dibutuhkan untuk menyelesaikan aktivitas dan

memperirakan jumlah sumber daya (PMI, Inc., 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula tidak menggunakan teknik yang disarankan oleh PMI, juga tidak mendokumentasikan cara untuk menghasilkan estimasi, seperti penggunaan dasar estimasi tertentu atau asumsi tertentu. Manajer proyek memperkirakan durasi pekerjaan berdasarkan fitur yang dibutuhkan, tingkat kesulitan logika dan pengalaman untuk proyek serupa di masa lalu, sedangkan setiap proyek pengembangan perangkat lunak bersifat unik (Marchewnka, 2003), sehingga terdapat kemungkinan hasil yang didapatkan tidak akurat. Proses estimasi durasi aktivitas memerlukan estimasi jumlah usaha kerja (*effort*) yang diperlukan untuk menyelesaikan aktivitas dan jumlah sumber daya yang tersedia yang diperkirakan untuk menyelesaikan kegiatan (PMI, Inc., 2013), sehingga manajer proyek dapat menggunakan metode estimasi *effort*, seperti *Advance Use Case Point* dan *Revised Use Case Point* yang dapat mengestimasi *effort* yang diperlukan dalam proyek pengembangan perangkat lunak dengan mengadaptasi beberapa karakteristik atau perilaku proyek (Srivastava dkk., 2015).

Pada proses *Develop Schedule*, terdapat proses menganalisa urutan aktivitas, jangka waktu, kebutuhan sumber daya, dan kendala jadwal untuk menciptakan model penjadwalan proyek (PMI, Inc., 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula telah menjalankan pengurutan aktivitas dan menampilkan dalam bentuk *timeline*, kemudian setiap aktivitas memiliki alokasi waktu. Tim proyek juga memiliki kesepakatan jadwal kapan tanggal mulai dan tanggal selesai, dengan mempertimbangkan hari kerja dan *shift* yang tersedia dalam kalender. Beberapa aktivitas tidak memiliki alokasi waktu khusus seperti, pembuatan dokumen perancangan yang dilakukan oleh pengembang magang, sehingga aktivitas tersebut tidak dimasukkan ke dalam penjadwalan. Adanya pertimbangan-pertimbangan tersebut seharusnya menjadi masukan dalam menyusun *baseline* penjadwalan dan jadwal proyek ke dalam *Gantt Chart*. Tidak dibuatnya *Gantt chart* menyebabkan tampilan informasi jadwal proyek yang direncanakan dan kondisi aktual tidak sesuai format standar, sehingga sulit untuk dibuat dan dimengerti saat memantau kemajuan proyek (Schwalbe, 2014).

Pada proses *Control Schedule*, terdapat proses memantau status aktivitas proyek untuk memperbarui kemajuan proyek dan mengelola perubahan pada jadwal untuk mencapai rencana (PMI, Inc., 2013). Manajer proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula melakukan pemantauan kinerja proyek yang berjalan dengan membandingkan pekerjaan yang seharusnya sudah selesai dengan jadwal yang telah disepakati. Namun, terdapat hari dimana pengembang harus bekerja lebih dari 8 jam sehari apabila target kinerja tidak tercapai yang dampaknya akan menimbulkan biaya tambahan dalam proses pengembangan. Tim proyek juga tidak melakukan pembaruan jadwal apabila terjadi perubahan selama proses pengembangan yang disebabkan oleh keterlambatan suatu aktivitas karena adanya kendala pada sisi teknis. Hal ini menimbulkan kesulitan dalam mempertahankan *baseline* jadwal sepanjang proyek dan memantau kemajuan proyek (PMI, Inc., 2013). *Baseline* proyek atau rencana yang tidak digunakan dengan benar untuk memandu proyek ke depan,

serta pembaruan jadwal yang tidak dilakukan ketika terdapat infomasi tambahan saat proyek berlangsung dapat menyebabkan kemungkinan proyek gagal (Attarzadeh & Ow, 2008).

Berdasarkan analisis perbandingan durasi hasil perhitungan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP), beserta karakteristik dari masing-masing metode, maka metode yang lebih sesuai digunakan untuk menghitung estimasi biaya pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula adalah metode *Advance Use Case Point*, karena dari hasil penilaian pada lembar penilaian *End User Development Technical Complexity Factor* (EUD_TCF) dan *End User Development Environmental Complexity Factor* (EUD_ECF) yang diisi oleh analis sistem menunjukkan Sistem Reservasi Online Gedung Serbaguna memiliki kebutuhan pada wilayah *End User Development*.

Berdasarkan analisis keberadaan proses-proses pada area *Project Schedule Management*, tim proyek sebaiknya memperbaiki dan menjalankan proses-proses yang seharusnya dilakukan dengan lebih baik dan lebih lengkap lagi.

## 5.2 Perbandingan Biaya Hasil Perhitungan Metode *Advance Use Case Point, Revised Use Case Point* dan CV. Gumcode Indonesia

Biaya adalah suatu bentuk pengorbanan atas sumber daya bertujuan untuk mencapai target yang bersifat spesifik atau hasil yang diperoleh sebagai pengganti yang berasal dari pengorbanan tersebut (Widjaya, 2013). Biaya dinyatakan dalam nilai uang dan tercatat di dalam laporan pengeluaran keuangan organisasi. Analisis perbandingan biaya hasil dari perhitungan metode *Advance Use Case Point* (AUCP), *Revised Use Case Point* (Re-UCP) dengan perhitungan dari CV Gumcode Indonesia bertujuan untuk mengetahui faktor-faktor yang mempengaruhi perbedaan pada setiap hasil, dan dampaknya bagi hasil yang diperoleh, yang nantinya disimpulkan dan digunakan sebagai informasi bagi CV Gumcode Indonesia dalam menentukan metode yang lebih baik untuk menghitung estimasi biaya pengembangan perangkat lunak yang dibutuhkan di masa mendatang.

Hasil perhitungan dari CV Gumcode Indonesia merupakan nilai yang ditawarkan dan disepakati oleh kedua belah pihak, yaitu CV Gumcode Indonesia sendiri dan klien, dimana dalam realisasinya mengalami keterlambatan dari jadwal sebenarnya yang secara tidak langsung menyebabkan meningkatnya biaya. Berdasarkan hasil perhitungan, urutan biaya dari rendah ke tinggi adalah hasil perhitungan dari CV Gumcode Indonesia sejumlah ± Rp 40.000.000,00, metode Revised Use Case Point sejumlah Rp 83.303.382,00 dan metode *Advance Use Point* sejumlah Rp 110.558.733,00. Tabel 5.3 akan menjelaskan tentang faktor-faktor yang mempengaruhi perbedaan hasil perhitungan biaya dan dampaknya pada masing-masing metode.

**Tabel 5. 3 Perbandingan Hasil Perhitungan Biaya**

| ***Advance Use Case Point (AUCP)*** | ***Revised Use Case Point (Re-UCP)*** | **Perhitungan CV Gumcode Indonesia** |
|---|---|---|
| Perhitungan dengan metode *Advance Use Case Point* menghasilkan nilai biaya sebesar Rp 110.558.733,00. Hasil ini merupakan nilai tertinggi dibandingkan dengan hasil dari metode *Revised Use Case Point* dan perhitungan CV Gumcode Indonesia. Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1 Penggunaan kelompok aktivitas *Software Development Phased* dan *On-Going Life Cycle Activity* sesuai Guideline Distribusi Effort milik Saleh (2011), menyebabkan tambahan aktivitas yang sebelumnya tidak dilakukan oleh CV Gumcode Indonesia.<br>2 Standar gaji pengembang menggunakan *Indonesia Salary Guide* 2017 milik Kelly Service Inc.<br>3 Jumlah pengembang yang dibutuhkan sejumlah 26 orang dan memiliki peran yang terspesialisasi.<br>4 Nilai *effort* yang lebih tinggi, karena adanya fitur tambahan *End-User Development* pada proses pengembangan.<br>5 Penggunaan *Technical Complexity Factor* sejumlah 13 faktor, dan *Environmental Complexity Factor* sejumlah 8 faktor. | Perhitungan dengan metode *Revised Use Case Point* menghasilkan nilai biaya sebesar Rp 83.303.382,00. Hasil ini berada pada urutan kedua tertinggi setelah hasil metode *Advance Use Case Point*. Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1. Penggunaan kelompok aktivitas *Software Development Phased* dan *On-Going Life Cycle Activity* sesuai Guideline Distribusi Effort milik Saleh (2011), menyebabkan tambahan aktivitas yang sebelumnya tidak dilakukan oleh CV Gumcode Indonesia.<br>2. Standar gaji pengembang menggunakan *Indonesia Salary Guide* 2017 milik Kelly Service Inc.<br>3. Jumlah pengembang yang dibutuhkan sejumlah 26 orang dan memiliki peran yang terspesialisasi.<br>4. Nilai effort yang lebih kecil, karena adanya pengklasifikasian kompleksitas aktor dan *use case* ke dalam kategori yang berbeda.<br>5. Penggunaan *Technical Complexity Factor* (TCF) sejumlah 14 faktor dan *Environmental Complexity Factor* (ECF) sejumlah 9 faktor. | Perhitungan CV Gumcode Indonesia menghasilkan nilai biaya sebesar ± Rp 40.000.000,00. Hasil ini merupakan nilai terendah dibandingkan hasil dari metode *Advance Use Case Point* dan *Revised Use Case Poin.* Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1. Menjalankan aktivitas pada fase *Software Development Life Cycle* (SDCL) tanpa adanya aktivitas *On-Going Life Cycle.*<br>2. Penentuan perkiraan biaya berdasarkan proyek-proyek sejenis di masa lalu.<br>3. Standar gaji pengembang menggunakan ketetapan dari CV Gumcode Indonesia Indoneisa untuk pengembang tetap dan berdasarkan nilai proyek untuk Pengembang paruh waktu.<br>4. Jumlah pengembang yang dibutuhkan sejumlah 5 orang dan memiliki peran ganda. |

Perhitungan alokasi biaya pada metode *Advance Use Case Point* dan *Revised Use Case Point* memiliki kesamaan pada penggunaan kelompok aktivitas, standar gaji yang digunakan dan jumlah pengembang yang dibutuhkan. Kedua metode menggunakan kelompok aktivitas *Software Development Phased* dan *On-Going Life Cycle Activity,* standar gaji pengembang berdasarkan *Indonesia Salary*

*Guide* 2017 milik Kelly Service Inc dan jumlah pengembang yang lebih banyak daripada perhitungan CV Gumcode Indonesia, yaitu sejumlah 26 orang.

Kedua metode menggunakan kelompok aktivitas yang sama, yaitu *Software Development Phased* dan *On-Going Life Cycle* sesuai Pedoman Distribusi *Effort* pada Penelitian Saleh (2011). Dampak dari penggunaan kelompok aktivitas ini adalah meningkatnya *effort*, dan menghasilkan jumlah alokasi pengembang yang dibutuhkan dalam pengembangan. Berdasarkan Lampiran A, CV Gumcode Indonesia sebelumnya hanya menjalankan aktivitas pada setiap fase *Software Development Life Cycle* (SDLC), sehingga ketika menggunakan Pedoman Distribusi *Effort* tersebut, nilai *effort* yang dibutuhkan meningkat. Nilai *effort* yang meningkat menyebabkan waktu pengembangan lebih lama dan biaya lebih tinggi.

Kedua metode membutuhkan jumlah alokasi pengembang yang lebih banyak daripada perhitungan CV Gumcode Indonesia yang berjumlah lima orang, yaitu 26 orang, serta peran pada setiap fase juga telah terspesialisasi tidak seperti pada alokasi CV Gumcode Indonesia yang memiliki peran ganda pada proses pengembangan. Peran yang terspesialisasi ini menyebabkan biaya pada setiap fase berbeda karena menggunakan standar gaji berdasakan *Indonesia Salary Guide 2017* yang dikeluarkan oleh Kelly Service Inc. Standar gaji pada Kelly Service lebih tinggi dibandingkan standar gaji pada CV Gumcode Indonesia, dimana pengembang tetap mendapatkan gaji sesuai ketetapan yang berlaku dan pengembang paruh waktu mendapatkan gaji sesuai nilai proyek, sehingga kondisi ini tidak ideal.

Perhitungan biaya pada metode *Advance Use Case Point* lebih tinggi dibanding hasil dari metode *Revised Use Case Point* dan perhitungan CV Gumcode Indonesia karena melibatkan fitur tambahan *End-User Development* pada proses pengembangan. Metode ini mempertimbangkan faktor teknis dan lingkungan melalui penilaian pada lembar penilaian *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factor* (ECF) dan fitur tambahan *End-User Development* yang menyebabkan penambahan pada nilai *effort* yang dibutuhkan dalam pengembangan. Nilai *effort* yang lebih tinggi, maka akan menyebabkan waktu pengembangan yang lebih lama dan membutuhkan biaya yang lebih tinggi pula.

Pada metode *Revised Use Case Point*, nilai yang didapat lebih tinggi daripada perhitungan CV Gumcode Indonesia, namun lebih rendah daripada perhitungan dengan metode *Advance Use Case Point*, karena adanya perbedaan pada pengklasifikasian kompleksitas aktor dan *use case* dan tambahan faktor pada faktor teknis dan lingkungan. Metode ini menghasilkan perhitungan nilai biaya lebih rendah daripada *Advance Use Case Point* karena nilai *effort* yang juga lebih rendah. Hal ini terjadi karena pada pengklasifikasian aktor dan *use case* memiliki *range* lebih luas. Kategori *simple* pada *use case* memiliki jumlah transaksi kurang dari empat, dan *average* memiliki jumlah transaksi lima sampai dengan delapan (Kirmani & Wahid, 2015), sehingga pada penelitian ini menghasilkan nilai *Unadjusted Use Case Weight* (UUCW) yang lebih kecil, dan nilai *Environmental Complexity Factor* (ECF) yang lebih kecil karena adanya faktor lingkungan

tambahan menyebabkan nilai *effort* lebih kecil daripada metode *Advance Use Case Point*.

Selain beberapa faktor di atas, peneliti melakukan wawancara terkait keberadaan proses-proses pada area pengetahuan *Project Cost Management* yang ada pada *Project Management Body of Knowledge* (PMBOK) yang dilampirkan pada Lampiran B. Area pengetahuan *Project Cost Management* memiliki empat proses di dalamnya, yaitu *Plan Cost Management*, *Estimate Cost*, *Determine Budget* dan *Control Cost*.

Pada proses *Plan Cost Management,* proses ini dilakukan untuk menentukan bagaimana biaya proyek akan diestimasi, dianggarkan, dikelola, dipantau, dan dikendalikan, dimana manfaat utamanya adalah memberikan panduan dan arahan mengenai bagaimana biaya proyek akan dikelola selama proyek berlangsung (PMI, Inc., 2013). Manajer proyek dan tim proyek Sistem Reservasi Gedung Serbaguna PT GHL Peninsula telah menjalankan proses ini di tahap perencanaan sebelum pengembangan dimulai dan mendokumentasikannya. Proses tersebut dilakukan dengan teknik *meeting*, dimana tim proyek mengadakan rapat perencanaan untuk mengembangkan rencana manajemen biaya. Adanya dokumentasi pada fase perencanaan berguna untuk menyediakan pedoman yang baik untuk tim proyek dan dapat menghindari kesalahan yang pernah dilakukan di masa lalu, sehingga estimasi biaya dapat mendekati akurat (Attarzah & Ow, 2008).

Pada proses *Estimate Cost*, terdapat proses untuk pengembangan perkiraan sumber daya biaya yang dibutuhkan untuk menyelesaikan pekerjaan proyek dan dalam prosesnya menggunakan dasar estimasi atau penilaian kuantitatif terntentu dengan mempertimbangkan adanya kontijensi dan cadangan untuk pekerjaan yang tidak direncanakan (PMI, Inc., 2013). Manajer proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula melakukan perhitungan, namun tidak mendokumentasikan dasar estimasi atau asumsi tertentu yang digunakan seperti teknik parametrik, *expert judgment* atau analogi. Perhitungan dilakukan berdasarkan campuran dugaan dan perhitungan. Manajer proyek juga tidak menganggarkan biaya kontijensi atau cadangan jika sewaktu-waktu terdapat pekerjaan yang tidak terduga. Cara yang dilakukan oleh manajer proyek tidak sesuai dengan proses *Estimate Cost* menurut PMBOK. Adanya penggunaan metode estimasi perangkat lunak, seperti *Advance Use Case Point* dan *Revised Use Case Point*, dapat mengurangi kesalahan dalam mengestimasi biaya karena kedua metode tersebut memanfaatkan *Use Case Diagram* pada perancangan berorientasi objek yang disiapkan pada tahap awal desain setelah memahami kebutuhan pengguna, serta kedua metode merupakan perluasan dari metode *Use Case Point* yang merupakan kelompok metode algoritmik dalam mengestimasi biaya (Iqbal, dkk., 2017).

Pada proses *Determined Budget*, terdapat proses menggabungkan estimasi biaya aktivitas individu untuk menentukan *baseline* biaya dan anggaran proyek mencangkup semua dana untuk melaksanakan proyek, termasuk cadangan kontijensi pada *work package* (PMI, Inc., 2013). Manajer proyek dan tim proyek

Sistem Reservasi Gedung Serbaguna PT GHL Peninsula tidak mengembangkan anggaran proyek yang disertai cadangan kontijensi karena merasa proses tersebut tidak diperlukan, namun memiliki prosedur pengendalian formal tertentu apabila terdapat perubahan anggaran. Caranya dengan mendaftar perubahan apa yang dibutuhkan, kemudian melakukan perhitungan ulang dan mengajukan penawaran ulang (*offering*). Berdasarkan aktivitas ini, berarti CV Gumcode Indonesia tidak menjalankan proses *Determine Budget* secara keseluruhan. Tidak adanya alokasi kontijensi mengharuskan penaksir atau manajer proyek mencari dana tambahan apabila biaya melebihi anggaran, dimana hal ini dapat menjadi kemungkinan penyebab begitu banyak proyek melebihi anggaran (Baccarini, 2004).

Pada proses *Control Cost*, terdapat proses memantau status proyek untuk memperbarui biaya proyek dan mengelola perubahan pada *baseline* biaya, serta mempertahankan *baseline* biaya sepanjang proyek (PMI, Inc., 2013). Manajer proyek Sistem Reservasi Gedung Serbaguna PT GHL Peninsula melakukan upaya untuk mencegah perubahan yang tidak disetujui untuk memastikan bahwa pengeluaran biaya tidak melebihi dana yang telah disetujui kecuali masih dalam batas yang diterima atau masih di dalam batasan lingkup kerja (*scope*). Apabila terdapat perubahan kebutuhan terlalu jauh dari lingkup kerja, maka perubahan tersebut akan ditolak. Manajer proyek juga memantau kinerja dan kemajuan proyek pengembangan, namun belum mendokumentasikannya ke dalam suatu dokumen, padahal informasi kinerja seharusnya didokumentasikan ke dalam laporan kinerja. Tidak dilakukannya dokumentasi dapat menyebabkan kesulitan saat evaluasi kinerja dimana biasanya saat dilakukan analisis terhadap kinerja akan ditemukan perubahan pada *baseline* biaya dan jadwal, atau komponen lain dari rencana manajemen proyek (PMI, Inc., 2013), dan estimasi biaya yang tidak direvisi ketika terdapat informasi lain selama proyek berlangsung dapat menyebabkan kemungkinan proyek gagal (Attarzah & Ow, 2008).

Berdasarkan analisis perbandingan biaya hasil perhitungan metode *Advance Use Case Point* dan *Revised Use Case Point,* beserta karakteristik dari masing-masing metode, metode yang lebih sesuai digunakan untuk menghitung estimasi biaya pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula adalah metode *Advance Use Case Point*, karena dari hasil penilaian pada lembar penilaian *End User Development Technical Complexity Factor* (EUD_TCF) dan *End User Development Environmental Complexity Factor* (EUD_ECF) yang diisi oleh analis sistem menunjukkan Sistem Reservasi Online Gedung Serbaguna memiliki kebutuhan pada wilayah *End User Development*.

Berdasarkan analisis keberadaan proses-proses pada area *Project Cost Management*, tim proyek sebaiknya memperbaiki dan menjalankan proses-proses yang seharusnya dilakukan dengan lebih baik dan lebih lengkap lagi.

## 5.3 Perbandingan Jumlah Pengembang Hasil Perhitungan Metode Advance Use Case Point, Revised Use Case Point dan CV. Gumcode Indonesia

*Staffing* mengacu pada jumlah orang yang diperlukan untuk menyelesaikan pekerjaan, tugas atau aktivitas proyek tertentu. Satuan pengembangfing adalah jumlah orang. Besarnya *staffing* berkaitan erat dan bergantung pada waktu dan *effort* yang direncanakan untuk pekerjaan tersebut (Trendowicz & Jeffery, 2014). Analisis perbandingan jumlah pengembang hasil dari perhitungan metode *Advance Use Case Point* (AUCP), *Revised Use Case Point* (Re-UCP) dengan perhitungan dari CV Gumcode Indonesia bertujuan untuk mengetahui faktor-faktor yang mempengaruhi perbedaan pada setiap hasil, dan dampaknya bagi hasil yang diperoleh, yang nantinya disimpulkan dan digunakan sebagai informasi bagi CV Gumcode Indonesia dalam menentukan metode yang lebih baik untuk menghitung estimasi jumlah pengembang pengembangan perangkat lunak yang dibutuhkan di masa mendatang.

Hasil perhitungan dari CV Gumcode Indonesia berdasarkan pengalaman manajer proyek dalam menangani proyek serupa di masa lalu dan mempertimbangkan keterbatasan jumlah pengembang yang tersedia di CV Gumcode Indonesia. Tim proyek Sistem Reservasi Online Gedung Serbaguna PT Peninsula terdiri dari satu manajer proyek, tiga pengembang tetap dan satu pengembang magang. Tabel 5.4 akan menjelaskan tentang faktor-faktor yang mempengaruhi perbedaan hasil perhitungan jumlah pengembang dan dampaknya pada masing-masing metode.

**Tabel 5. 4 Perbandingan Hasil Perhitungan Jumlah Pengembang**

| ***Advance Use Case Point (AUCP)*** | ***Revised Use Case Point (Re-UCP)*** | **Perhitungan CV Gumcode Indonesia** |
|---|---|---|
| Perhitungan dengan metode *Advance Use Case Point* menghasilkan alokasi pengembang sejumlah 26 orang. Hasil ini lebih banyak daripada perhitungan CV Gumcode Indonesia, dan sama seperti hasil metode *Revised Use Case Point*. Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1 Penggunaan kelompok aktivitas *Software Development Phased* dan *On-Going Life Cycle Activity* sesuai Guideline Distribusi Effort milik Saleh (2011), menyebabkan tambahan aktivitas yang sebelumnya tidak | Perhitungan dengan metode *Revised Use Case Point* menghasilkan alokasi pengembang sejumlah 26 orang. Hasil ini lebih banyak daripada hasil perhitungan CV Gumcode Indonesia dan sama seperti metode Advance Use Case Point. Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1. Penggunaan kelompok aktivitas *Software Development Phased* dan *On-Going Life Cycle Activity* sesuai Guideline Distribusi Effort milik Saleh (2011), menyebabkan tambahan aktivitas yang sebelumnya tidak | Perhitungan CV Gumcode Indonesia menghasilkan alokasi pengembang sejumlah 5 orang, hasil yang paling sedikit dibandingkan hasil dari metode *Advance Use Case Point* dan *Revised Use Case Poin.* Beberapa faktor yang menjadi penyebabnya, antara lain:<br>1. *Work breakdown structure* yang digunakan hanyalah aktivitas pada fase *Software Development Life Cycle* (SDLC).<br>2. Beberapa aktivitas dikerjakan oleh |

**Tabel 5. 5 Perbandingan Hasil Perhitungan Jumlah Pengembang (Lanjutan)**

| *Advance Use Case Point (AUCP)* | *Revised Use Case Point (Re-UCP)* | Perhitungan CV Gumcode Indonesia |
|---|---|---|
| dilakukan oleh CV Gumcode Indonesia. | dilakukan oleh CV Gumcode Indonesia. | pengembang yang memiliki peran ganda. |
| 2 Adanya urutan dan ketergantungan antar aktivitas. | 4 Adanya urutan dan ketergantungan antar aktivitas. | |
| 3 Peran pengembang pada setiap aktivitas telah terspesialisasi. | 5 Peran pengembang pada setiap aktivitas telah terspesialisasi. | |

Perhitungan alokasi pengembang pada metode *Advance Use Case Point* dan *Revised Use Case Point* menggunakan Pedoman Distribusi *Effort* yang sama, yaitu Pedoman Distribusi *Effort* dari penelitian Saleh (2011). Berdasarkan pedoman tersebut, waktu yang dibutuhkan untuk pengembangan perangkat lunak sama dengan waktu yang didistribusikan pada fase aktivitas manajemen proyek (*Project Management*), sehingga agar aktivitas-aktivitas pada *Software Phased* selesai bersamaan dengan aktivitas *Project Management*, maka jumlah pengembang yang dialokasikan pada *Software Phased* lebih banyak. Adanya aktivitas *On-Going Life Cycle* juga menambah alokasi pengembang karena sebelumnya CV Gumcode Indonesia tidak menjalankan fase ini. Masing-masing aktivitas pada *Software Phased* dikerjakan oleh 6 orang dan aktivitas *On-Going Life Cycle* dikerjakan oleh 1 orang , kecuali pada aktivitas *Evaluation and Testing* dikerjakan oleh 3 orang. Aktivitas *Evaluation and Testing* memiliki nilai *hours of effort* yang lebih besar daripada aktivitas *Project Management*, sehingga agar dapat selesai bersamaan dengan aktivitas *Project Management* dibutuhkan jumlah pengembang yang lebih banyak. Setiap pekerjaan dilaksanakan oleh satu orang ataupun sebuah tim yang terspesialisasi dari sisi fungsionalitas dan sisi pelaksananya (Tantra, 2013), sehingga tidak ada peran ganda.

Kedua metode juga menerapkan urutan dan ketergantungan antar aktivitas yang sama. Aktivitas pada *Software Phased* dikerjakan secara berurutan mulai dari *Requirement, Specification, Design, Implementation, Integration Testing* dan *Acceptance and Deployment*. Aktivitas pada *On-Going Life Cycle*, terutama *Project Management* dikerjakan selama proses pengembangan berlangsung. Aktivitas *Configuration Management* berjalan bersamaan dengan aktivitas *Acceptance and Deployment*. Aktivitas *Quality Assurance* mengalokasikan *effort* ke masing-masing aktivitas *Software Phased,* sehingga ketika aktivitas *Requirement* mulai dijalankan, maka *Quality Assurance* untuk *Requirement* juga ikut dimulai, dan seterusnya. Aktivitas *Documentation* berjalan bersamaan dengan dimulainya proses pengembangan. Aktitas *Training and Support* berjalan bersamaan dengan aktivitas *Acceptance and Deployment*, sedangkan aktivitas *Evaluation and Testing* mengalokasikan *effort* ke setiap paket pekerjaan pada *Implementation*. Aktivitas *Implementation* memiliki tiga paket pekerjaan, yaitu implementasi halaman *customer*, implementasi halaman vendor dan implementasi halaman admin, sehingga ketika implementasi halaman *customer*

selesai dijalankan, maka *Evaluation and Testing* untuk halaman *customer* juga dimulai, dan seterusnya.

Pada saat pembuatan *Gantt Chart,* setiap aktivitas mencantumkan tanggal mulai dan tangga selesai, serta peran pengembang yang bertanggungjawab. Alokasi pengembang dihitung berdasarkan peran dan waktu kerja. Apabila aktivitas membutuhkan peran yang sama di waktu yang berbeda, maka aktivitas dilakukan oleh pengembang yang sama, sedangkan apabila aktivitas membutuhkan peran yang sama di waktu yang sama, maka aktivitas dilakukan oleh pengembang yang berbeda. Berdasarkan *work breakdown structure* (WBS) dan urutan ketergantungan antar aktivitas, diperoleh total alokasi sumber daya pengembang sejumlah 26 orang.

Selain beberapa faktor di atas, peneliti melakukan wawancara terkait keberadaan proses-proses pada area pengetahuan *Project Resource Management* yang ada pada *Project Management Body of Knowledge* (PMBOK) yang dilampirkan pada Lampiran B. Area pengetahuan *Project Resource Management* memiliki enam proses di dalamnya, yaitu *Plan Resource Management*, *Estimate Activity Resource*, *Acquiry Resource, Develop Team, Manage Team* dan *Control Resource*.

Pada proses *Plan Resource Management*, terdapat proses menentukan bagaimana memperkirakan, mengakuisisi, mengelola, dan menggunakan tim dan sumber daya fisik untuk menetapkan pendekatan dan tingkat upaya pengelolaan yang diperlukan (PMI, Inc., 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula telah menjalankan panduan bagaimana anggota tim proyek dikategorikan, dialokasikan dan dikelola sampai nanti diluncurkan. Proses ini dilakukan sekali di awal. Di awal pengembangan proyek, setiap anggota tim proyek dialokasikan ke bagian-bagian pekerjaan yang tersedia. Namun, tim proyek belum mengembangkan dokumen *Team Charter* karena keterbatasan waktu dan sumber daya manusia. Team charter merupakan dokumen berisi nilai-nilai tim, pedoman komunikasi, perjanjian tim dan pedoman operasi bagi tim (PMI, Inc., 2013). Kurangnya pelatihan dalam proses perencanaan dapat mempengaruhi secara langsung keberhasilan proyek dimana keberhasilan dalam organisasi tidak akan pernah tercapai tanpa personel yang berkualitas dan termotivasi (Belout, 1998).

Pada proses *Estimate Activity Resource*, terdapat proses mengestimasi sumber daya tim, jenis dan jumlah material, peralatan, dan persediaan yang diperlukan untuk melakukan pekerjaan proyek (PMI, Inc., 2013). Tim proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula belum melakukan dokumentasi kebutuhan sumber daya berdasarkan *Work Breakdown Structure* (WBS) ke dalam bentuk *Resource Breakdown Structure*, karena keterbatasan waktu dan sumber daya manusia yang tersedia di CV Gumcode Indonesia. Tim proyek juga tidak mendokumentasikan cara untuk menghasilkan estimasi, seperti penggunaan dasar estimasi tertentu atau asumsi tertentu. Manajer proyek memperkirakan jumlah pengembang pada setiap aktivitas berdasarkan fitur yang dibutuhkan, tingkat kesulitan logika dan pengalaman untuk proyek serupa di masa lalu.

Beberapa pengembang memiliki peran ganda, yaitu seorang *programmer* merangkap sebagai *tester*, dan seorang analis sistem merangkap sebagai *software quality assurance*. Alokasi jumlah pengembang dan pemilihan pengembang yang tepat mempengaruhi keberhasilan proyek (N.R. Kumar, V. Suma & T.R Goplakrisnan N., 2012), sehingga apabila hanya beberapa anggota tim saja yang menguasai pemrograman atau perangkat pengembangan, maka terdapat kemungkinan tidak dapat menyelesaikan pengembangan sistem sesuai jadwal (Attarzah & Ow, 2008).

Pada proses *Acquire Resource*, terdapat proses menentukan anggota tim, fasilitas, peralatan, bahan, persediaan dan sumber daya lain yang diperlukan untuk menyelesaikan pekerjaan proyek (PMI, Inc., 2013). Manajer proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menentukan anggota tim proyek beserta peran dan tanggungjawab atas setiap aktivitas proyek pengembangan perangkat lunak tersebut. Di dalam proses ini, pembuatan kalender sumber daya harus memuat hari kerja, shift, jam mulai dan berakhirnya jam kerja normal, akhir pekan, dan hari libur pada setiap sumber daya, namun tim proyek tidak mengembangkannya. Selain itu, beberapa anggota tim juga bekerja di luar proyek dalam tugas sehari-hari atau melakukan *resource sharing*, sehingga menyebabkan kesulitan dalam menguraikan dan memandu pemilihan sumber daya dan menugaskan sumber daya tersebut ke aktivitas masing-masing (PMI, Inc., 2013).

Pada proses *Develop Team*, terdapat proses meningkatkan kompetensi, interaksi anggota tim, dan lingkungan tim secara keseluruhan untuk meningkatkan kinerja proyek (PMI, Inc., 2013). Tim Proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula telah menggunakan semacam *chat room* untuk saling berkomunikas dan berdiskusi tentang segala sesuatu terkait proyek, yaitu aplikasi Slack. Cara ini termasuk ke dalam teknik *Communication Technology*. Namun, penilaian tentang keefektifan tim proyek masih dilakukan oleh manajer proyek secara informal, bukan oleh tim manajemen proyek khusus. Seharusnya evaluasi kinerja tim dilakukan secara keseluruhan, sehingga tim manajemen proyek dapat mengidentifikasi pelatihan khusus, *coaching, mentoring,* asistensi, atau perubahan yang diperlukan untuk meningkatkan kinerja tim (PMI, Inc., 2013).

Pada proses *Manage Team*, terdapat proses menilai kinerja anggota tim, memberikan umpan balik, menyelesaikan masalah, dan mengelola perubahan tim untuk mengoptimalkan kinerja proyek (PMI, Inc., 2013). Tim Proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula melakukan tindakan perbaikan atau pencegahan apabila terjadi perubahan atau gangguan yang dapat menyebabkan jadwal terlambat atau melampaui anggaran. Tindakan perbaikan dilakukan apabila terdapat kesalahan pada kode program, atau kesalahan teknis yang disebabkan oleh pengguna selama masih dalam batas waktu kesepakatan. Namun, perubahan tim yang mungkin terjadi selama proses pengembangan tidak didokumentasikan ke dalam dokumen perubahan dan tidak mengubah dokumen lain yang mungkin terpengaruh karena adanya perubahan ini, sehingga

kemungkinan akan mempengaruhi perilaku tim dalam mengelola konflik dan menyelesaikan masalah (PMI, Inc., 2013).

Pada proses *Control Resource*, terdapat proses memastikan bahwa sumber daya fisik yang ditugaskan dan dialokasikan untuk proyek, tersedia sesuai rencana, serta memantau pemanfaatan sumber daya yang direncanakan dibandingkan dengan aktual dan mengambil tindakan korektif seperlunya jika dibutuhkan (PMI, Inc., 2013). Manajer proyek Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula memantau pekerjaan proyek yang berjalan dengan melakukan perbandingan antara kebutuhan sumber daya dan alokasi sumber daya. Namun, manajer proyek tidak mendokumentasikan hasil dari pemantauan tersebut dan perubahan yang mungkin tejadi pada saat proses *Manage Team* ke dalam suatu dokumentasi. Misalnya, adanya perubahan pengembang, karena pilihan pribadi maupun peristiwa yang tidak dapat dikendalikan yang dapat mengganggu tim proyek. Gangguan ini dapat menyebabkan keterlambatan jadwal dan berlebihnya anggaran. Manajer proyek perlu melakukan tindakan seperti perubahan *pengembangfing* dengan memindahkan pengembang ke pekerjaan yang berbeda, melakukan *outsourcing* sebagian pekerjaan atau mengganti anggota tim dibutuhkan (PMI, Inc., 2013).

Berdasarkan analisis perbandingan jumlah alokasi pengembang hasil perhitungan metode *Advance Use Case Point* (AUCP) dan *Revised Use Case Point* (Re-UCP), beserta karakteristik dari masing-masing metode, CV Gumcode Indonesia dapat menggunakan *Work Breakdown Structure* (WBS) dan alokasi effort pada masing-masing aktivitas dengan Pedoman Distribusi *Effort* dari penelitian Saleh (2011) karena telah mencangkup seluruh aktivitas pada kondisi ideal proyek pengembangan perangkat lunak.

Berdasarkan analisis keberadaan proses-proses pada area *Project Schedule Management*, tim proyek sebaiknya memperbaiki dan menjalankan proses-proses yang seharusnya dilakukan dengan lebih baik dan lebih lengkap lagi.

# BAB 6 PENUTUP

## 6.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan untuk mengestimasi biaya, waktu dan sumber daya manusia yang dibutuhkan dalam pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula, maka peneliti menarik kesimpulan yang menjawab rumusan masalah pada Bab Pendahuluan, sebagai berikut:

1. Penerapan metode *Advance Use Case Point* untuk menghitung estimasi biaya dan waktu yang dibutuhkan untuk pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula memanfaatkan hasil perhitungan nilai *Unadjusted Actor Weight, Unadjusted Use Case Weight, Unadjusted Use Case Point, Technical Complexity Factor* yang terdiri dari 13 faktor, *Environmental Complexity Factor* yang terdiri dari 8 faktor, *Use Case Point*, serta *End-User Development Feature* yang direpresentasikan ke dalam *End-User Development Technical Complexity Factor* yang terdiri dari 17 faktor dan *End-User Development Environmental Complexity Factor* yang terdiri dari 8 faktor. Nilai-nilai tersebut menghasilkan nilai *Advance Use Case Point*, dan dikonversikan menjadi *hours of effort* dengan cara mengkalikan 20 *person-hours* per-UCP. Perhitungan metode ini menghasilkan estimasi waktu selama 185,93 jam dan biaya sejumlah Rp 110.558.733,00.
2. Penerapan metode *Revised Use Case Point* untuk menghitung estimasi biaya dan waktu yang dibutuhkan untuk pengembangan Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula memanfaatkan hasil perhitungan nilai *Unadjusted Actor Weight, Unadjusted Use Case Weight, Unadjusted Use Case Point, Technical Complexity Factor* yang terdiri dari 14 faktor, dan *Environmental Complexity Factor* yang terdiri dari 9 faktor. Nilai-nilai tersebut menghasilkan nilai *Revised Use Case Point*, dan dikonversikan menjadi *hours of effort* dengan cara mengkalikan 20 *person-hours* per-UCP. Perhitungan metode ini menghasilkan estimasi waktu selama 140,1 jam dan biaya sejumlah Rp 83.303.382,00.
3. Pembagian lingkup kerja pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula menggunakan pendekatan *Work Breakdown Structure* (WBS) pada metode *Advance Use Case Point* dan *Revised Use Case Point* dan keduanya terdiri dari 4 level. Level pertama adalah nama dari sistem yang dikembangkan. Level kedua terdiri dari 4 fase *Project Life Cycle* yang dijalankan, yaitu *Define Project Goal, Plan Project, Execute Project Plan* dan *Close Project*. Level ketiga merupakan sub-fase dari *Execute Project Plan* yang terdiri dari kelompok aktivitas *Software Development* dan *On-Going Life Cycle Activity*. Level keempat merupakan sub-fase dari kemlompok aktivitas *Software Development* yang terdiri dari 6 aktivitas, yaitu *Requirement, Specification, Design, Implementation, Integration and Testing* dan *Acceptance and Deployment*, sedangkan pada fase *On-Going Life Cycle*

*Activity* terdiri dari 6 aktivitas, yaitu *Project Management, Configuration Management, Quality Assurance, Documentation, Training and Support* dan *Evaluation and Testing*.

4. Penjadwalan pada Sistem Reservasi Online Gedung Serbaguna PT GHL Peninsula yang menggunakan *Gantt Chart* berdasarkan *Work Breakdown Structure* (WBS) dan hubungan antar aktivitas pada *Advance Use Case Point* dan *Revised Use Case Point* menghasilkan alokasi sumber daya manusia sejumlah 26 orang dan jumlah pekerjaan pada fase *Software Development* dan *On-Going Life Cycle Activity* sebagai berikut:

   a) Penjadwalan *Gantt Chart* berdasarkan WBS dan hubungan antar aktivitas pada metode *Advance Use Case Point* untuk kelompok aktivitas *Software Development* menghasilkan 6 *work package* yang diuraikan menjadi 13 *task* dan 6 *milestone*, dengan jumlah durasi selama 185,93 jam, biaya sebesar Rp 44.936.344,00 dan jumlah pekerja sebanyak 36 orang. Pada kelompok aktivitas *On-Going Life Cycle Activity* menghasilkan 6 *work package* yang diuraikan menjadi 14 *task* dan 6 *milestone*, dengan jumlah durasi selama 185,93 jam, biaya sebesar Rp 65.622.389,00 dan jumlah pekerja sebanyak 8 orang.

   b) Penjadwalan *Gantt Chart* berdasarkan WBS dan hubungan antar aktivitas pada metode *Revised Use Case Point* untuk kelompok aktivitas *Software Development* menghasilkan 6 *work package* yang diuraikan menjadi 13 *task* dan 6 *milestone*, dengan jumlah durasi selama 140,1 jam, biaya sebesar Rp 33.858.469,00 dan jumlah pekerja sebanyak 36 orang. Pada kelompok aktivitas *On-Going Life Cycle Activity* menghasilkan 6 *work package* yang diuraikan menjadi 14 *task* dan 6 milestone, dengan jumlah durasi selama 140,1 jam, biaya sebesar Rp 49.444.913,00 dan jumlah pekerja sebanyak 8 orang.

5. Perbandingan hasil perhitungan durasi menghasilkan urutan durasi dari cepat ke lambat sebagai berikut:

   1) Perhitungan metode *Revised Use Case Point* selama 140,1 jam.
   2) Perhitungan metode *Advance Use Case Point* selama 185,93 jam.
   3) Perhitungan CV Gumcode Indonesia selama 30 hari atau 240 jam.

   Perbandingan hasil perhitungan biaya menghasilkan urutan biaya dari tinggi ke rendah sebagai berikut:

   1) Perhitungan metode *Revised Use Case Point* sebesar Rp 110.558.733,00.
   2) Perhitungan metode *Advance Use Case Point* sebesar Rp 83.303.382,00.
   3) Perhitungan CV Gumcode Indonesia sebesar ± Rp 40.000.000,00.

   Perhitungan alokasi sumber daya manusia pada metode *Advance Use Case Point* dan *Revised Use Case Point* menghasilkan 26 orang, dimana jumlah ini lebih banyak daripada alokasi oleh CV Gumcode Indonesia yang hanya sebanyak 5 orang.

## 6.2 Saran

Berdasarkan penarikan kesimpulan di atas, penulis merekomendasikan beberapa saran yang dapat digunakan untuk penelitian selanjutnya, yaitu:

- Sebaiknya menggunakan variasi aturan transaksi use case lain yang sesuai dengan kondisi proyek perangkat lunak, sehingga dapat menghasilkan nilai *effort* yang lebih mendekati *effort* yang sebenarnya.
- Sebaiknya menggunakan Pedoman Distribusi Effort berbeda selain dari penelitian Saleh (2011).
- Standar gaji tenaga TI/SI yang digunakan sebaiknya adalah standar gaji yang dikeluarkan oleh Pemerintah Indonesia atau instansi yang bertanggungjawab lain yang mencangkup seluruh daerah di Indonesia, sehingga dapat menghasilkan biaya yang sesuai dengan kemampuan daya beli masyarakat.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

# DAFTAR PUSTAKA


Baccarini, D. 2004. *Estimating Project Cost Contingency Beyond the 10% Syndrome.* Perth: Curtin University. Tersedia di: <https://espace.curtin.edu.au> [Diakses 10 Mei 2018]

Belout, A. 1998. *Effect of Human Resource Management on [roject Effectiveness and Success: toward a new conceptual framwork.* Great Britain: Elsevier Inc. Tersedia di: <https://pdfs.semanticscholar.org> [Diakses 10 Mei 2018]

Birrel, N.D. & Ould, M.A., 1988. *A Practical Handbook for Software Development.* [e-book] Melbourne: Cambridge University Press. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 7 Maret 2018]

Bittner, K. & Spence, I., 2002. *Use Case Modeling.* Boston: Addison Wesley.

Cockburn, A., 2000. *Writing Effective Use Case.* Boston: Addison Wesley.

Cohn, M., 2005. *Estimating with Use Case Point.* [online] Tersedia di: <http://www.mountaingoatsoftware.com/articles/estimating-with-use-case-point> [Diakses 2 April 2018]

Cypher et al., 2010. *No Code Required: Giving User Tools to Transform the Web.* United States of America: Elsevier Inc. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 7 Maret 2018]

Davis, F.D., 1989. *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*. [e-journal] MIS Quarterly, Vol. 13, No. 3 (Sep., 1989), pp. 319-340. Tersedia di: <http://researchgate.net> [Diakses 8 Maret 2018]

Diev, S., 2006. *Software estimation in the maintenance context*, ACM SIGSOFT Software Engineering Notes 31 (2) (2006) 1–8

Ericson, L.W., 2004. *Code Reuse in Object Oriented Software Development.* Course Fall Semester. Norwegian University of Science and Technology. Tersedia di: <http://idi.ntu.no> [Diakses 10 Maret 2018]

Fisher, M., & Duskis, S., 2009. *Spring Persistence: A Running Start.* [e-book] Berkeley: Apress. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 9 Maret 2018]

Haake et al., 2016. *Designing an End-User Participation and Involvement Assistance for Continuous IS Development.* [e-journal] Tersedia di: <http://researchgate.net> [Diakses 8 Maret 2018]

Iqbal, S. Z. et al., 2017. *Comparative Analysis of Common Software Cost Estimation Modelling Technique.* [e-jurnal] Mathematical Modelling and Applications. Vol. 2, No. 3, 2017, pp. 33-39. Tersedia di: <https://www.sciencepublishinggroup.com> [Diakses 10 Mei 2018]

Irwansyah, E., & Moniaga, J.V., 2014. *Pengantar Teknologi Informasi.* [e-book] Tersedia di: Google Books <http://booksgoogle.com> [Diakses 9 Maret 2018]

Jacobson, I., Christerson, M., & Overgaar, G., 1992. O*bject Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley Longman, Inc.

Jeffrey, R., & Trendowicz, A., 2014. *Software Project Effort Estimation: Foundation and Best Practice Guideline.* Switzerland: Springer International Publishing Switzerland 2014. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 15 April 2018]

Johansen, C., 2010. *Test-Driven JavaScript Development.* [e-book] Boston: Addison-Wesley. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 15 Maret 2018]

K., Sangeetha & Dalal, P., 2016. *Software Sizing with Use Case Point.* India: Shrinthji Institute of Technology and Engineering.

Karner, G. 1993. *Resource Estimation for Objectory Projects*. Objective Systems SF AB.

Kashyap, D., Shukla, D. dan Misra, A.K., 2014. *Refining the Use case Classification for Use Case Point Method for Software Effort Estimation*. Association of Computer Electronics and Electrical Engineers. Tersedia di: <https://pdfs.semanticscholar.org> [Diakses 21 Februari 2018].

Kelly Service. 2017. *Indonesia Employment Salary Outlook and Salary Guide*. Indonesia: Kelly Service inc.

Kirmani M. M. & Wahid, A., 2015. *Revised Use Case Point (Re-UCP) Model for Software Effort Estimation.* International Journal of Advanced Computer Science and Application. Tersedia di: <https://semanticscholar.org> [Diakses 19 Januari 2018].

Kleppman, M., 2017. *Designing Data Intensive Application: The Big Ideas Behind Reliable, Scalable, and Maintainable System.* [e-book] Sebastopol: O'Reilly Media Inc. Tersedia di: Google Books <https://books.google.co.id/> [Diakses 10 Maret 2018]

Ko et al., 2011. *The State of the Art in End-User Software Engineering.*ACM Computing Surveys, 43, 3, Article 21. Tersedia melalui: Carnegie Melon University Research Showcase <http://repository.cmu.edu/hcii> [Diakses 15 Maret 2018]

Krisnewgowda, et al., 2013. *Software Sizing and Productivity with Function Point*

Larson, Eric W. et al., 2011. *Project Management the Managerial Process: Fifth Edition.* United States: The McGraw-Hill Companies, Inc.

Lessard, et al., 2007. *Project Management for Engineering Design*. [e-book] USA: Morgan & Claypool Publisher. Tersedia di: Google Books <https://books.google.co.id/> [Diakses 12 Februari 2018]

Leung, et al., 2001. *Software Cost Estimation.* Hongkong: The Hongkong Polytechnic University.

Marchewka, J. 2003. Information Technology Project Management.Holoben; NJ Wiley

Marimin, Tanjung H. & Prabowo H., 2006. Sistem Informasi Manajemen Sumber Daya Manusia. [e-book]. Tersedia di: <https://books.google.co.id/books?id=awC2247vPUkC> [Diakses 5 Mei 2018]

Nassif, A., Capretz ,L.F., & Ho D., 2010. *Enhanching Use Case Points Estimation Method Using Soft Computing Techniques.* Journal of Global Research in Computer Science, [e-journal] 148(3). Tersedia di: <www.gcrs.info> [diakses 1 April 2018].

N.R. Shashi Kumar, V. Suma, & T.R, Gopalakrisnan N., 2012. *Impact Analysis of Allocation Resource by Project Manager on Success of Software Projects.* Bangkok: International Conference on Data Mining and Computer Engineering (ICDMCE'2012). Tersedia di: <https://link.springer.com> [Diakses 10 Mei 2018]

Ow, S.H, & Attarzeh, I. 2008. *Project Management Practice: The Criteria for Success or Failure.* Communication of the IBIMA Volume 1. Tersedia di: <https://ibimapublishing.com/articles/IBIMA/2008/149039/149039.pdf> [Diakses 10 Mei 2018]

Paterno, et al., 2013. *End User Development: Survey of an Emerging Field for Empowering People*. [pdf]. Pisa: Hindawi Publishing Corporation. Tersedia di: <https://scholar.google.co.id> [Diakses 27 Januari 2018]

PMI, Inc., 2013. *A guide to the project management body of knowledge (PMBOK® guide). Fifth edition*. Pennsylvania: Project Management Institute, Inc. Tersedia di: <https://www.pmi.org> [Diakses 12 Maret 2018]

Podelko, A. 2016. *Performance Requirement: The Backbone of the Performance Engineering Process.* Int. CMG Conference, 2012. Tersedia di: <https://scholar.google.co.id> [Diakses 8 Maret 2018]

Prasetya, H. & Lukiastuti, F., 2009. *Manajemen Operasi.* Yogyakarta: Media Pressindo. Tersedia di: Google Books <https://books.google.co.id/books?id=KqMYYO0Vge4C> [Diakses 27 Januari 2018]

Rev, 2003. *Work Breakdown Structure (Office of Engineering and Construction Management ed)*. United State of America.

Ribu, K. 2001. Estimating Object-Oriented Software Projects with *Use case*s. *Master of Science Thesis*. University of Oslo Department of Informatics.

Robinson, D., 2010. *Aspect-Oriented Programming with the E-Verification Language.* [e-book] Burlington: Elsevier Inc. Tersedia di: Google Books <https://books.google.co.id/> [Diakses 17 Maret 2018]

Rode, J., Rosson, M.B., & Quinones, M.A.P., 2005. *End User Development of Web Application.* [pdf]. Netherlands: Kluwer Academic Publisher. Tersedia di: <http://pdfs.semanticscholar.org> [Diakses 15 Maret 2018].

Saleh, K., 2011., *Effort and Cost Allocation in Medium to Large Software Development Projects*. [pdf]. Internasional Journal of Computers(I).

Salvendy, G. 2003. *Handbook of Indutrial Engineering Technology and Operational Management Third Edition*. [e-book]. Tersedia di: <https://books.google.co.id/books?id=sjY3IZ9Unv0C> [Diakses 9 Maret 2018]

Sandhu, R., & Samarati, P., 1996. *Authentication, Access Control, and Audit.* ACM Computing Surveys, Vol. 28, No. 1, March 1996. Tersedia di: <http://semanticscholar.org> [Diakses 9 Maret 2018]

Schwalbe, Kathy. 2015. *An Introduction to Project Management, Fifth Edition.* Minneapolis: Schwalbe Publishing.

Shen et al., 2008. *Design of A Measurement System of EUIC and Its Application.* In: 11th International Conference, CSCWD 2007. Melbourne, Australia, 26-28 April2007. New York: Springer-Verlag Berlin

Sholiq, et al., 2013. *Effort Distribution to Estimate Cost in Small to Medium Software Development Project with Use Case Point.*[pdf]. Surabaya: Institut Teknologi Sepuluh Nopember.

Smith, John M. 2001. *Troubled IT Project Prevention and Turnaround*. London: The Institution of Engineering and Technology. Tersedia di: Google Books <https://books.google.co.id/books?id=ywNDArirgOMC> [Diakses 30 Januari 2018]

Sommerville, I., 2011. Software Engineering (9th edition ed.).[pdf] Addison-Wesley.

Srivastava, A., Singh, S.K. & Abbas, S.Q., 2017. *Evaluation of Project Estimation Methodology: AUCP*. International Journal of Software Engineering and Application Vol.6, No.2. [e-journal]. Tersedia di: <https://www.serialsjournals.com> [Diakses 19 Januari 2018]

Swarjana, I. K. 2016. *Statistik Kesehatan*. Yogyakarta: Penerbit Andi. Tersedia di: Google Books <https://books.google.co.id/books?id=sRcXDQAAQBAJ> [Diakses 12Februari 2018]

Tantra, R. 2012. *Manajemen Proyek Sistem Informasi*. Yogyakarta: Penerbit Andi.

Taylor, J. P. 2008. *Project Schedulling and Cost Control.* United State of America: J. Ross Publishing, Inc. Tersedia di: Google Books <https://books.google.co.id/books?id=AnAeKKFGhTwC> [Diakses 12Februari 2018]

Vermaat, S.C., 2007. *Discovering Computers: Menjelajah Dunia Komputer Fundamental. 3th edition.* [e-book]Jagakarsa: Salemba Infotek. Tersedia di: Google Books <https://books.google.co.id/> [Diakses 18 Februari 2018]

Wahana, K., 2014. *Membangun Aplikasi Mobile Cross Platform dengan Phonegap.* [e-book] Jakarta: PT Elex Media Komputindo. Tersedia di: Google Books <https://books.google.co.id/> [Diakses 10 Maret 2018]

Wazlawick, P.S. 2013. *Object-Oriented Analysis and Design for Information System.* Waltham: Elsevier Inc.

Weinstock, C.B., & Goodenough, J.B., 2006. *On System Scalability.* Tersedia melalui: Carnegie Melon University Research Showcase <http://repository.cmu.edu/> [Diakses 18 Maret 2018]

Whitten, J.L & Bentley L.D., 2007. *System Analysis and Design Methods*. 7th ed. New York: McGraw-Hill Companies, Inc.

Yaumi, M., & Damopolii, M., 2014. *Action Research: Teori, Model, & Aplikasi*. Jakarta: Prenadamedia Group. Tersedia di: Google Books <http:// books?id=X-IuDwAAQBAJ> [Diakses 12Februari 2018]