KERANGKA KERJA PEMILIHAN *FREE AND OPEN SOURCE SOFTWARE (FOSS)* MENGGUNAKAN METODE *ANALYTICAL HIERARCHY PROCESS*

Ario Santoso dan Indra Budi

Fakultas Ilmu Komputer, Universitas Indonesia, Depok, Indonesia
santoso.ario@gmail.com, indra@cs.ui.ac.id

Abstrak

Banyaknya pilihan *Free Open Source Software* (FOSS) dengan beraneka ragam kualitas menyulitkan kita memilih FOSS yang tepat untuk memenuhi kebutuhan kita. Oleh karena itu, dibutuhkan suatu metode untuk membantu dalam memilih FOSS yang tepat. *Paper* ini bertujuan untuk mengemukakan suatu kerangka kerja (*framework*) yang dapat memberikan tuntunan atau bantuan dalam memilih FOSS. Kerangka kerja yang diajukan ini menggunakan metode *Analytical Hierarchy Process* untuk membantu proses pemilihan solusi.

Kata kunci : *Analytical Hierarchy Process*, *Free and Open Source Software*, Kerangka Kerja Pemilihan FOSS

1. Pendahuluan

FOSS merupakan perangkat lunak yang memberikan kebebasan bagi siapapun untuk menggunakannya, mempelajarinya, memodifikasinya, dan mendistribusikannya tanpa harus membayar apapun [1,2]. Umumnya, FOSS ini dikembangkan oleh suatu komunitas. Penggunaan FOSS memberikan beberapa keuntungan, seperti misalnya membebaskan kita dari ketergantungan terhadap *vendor* [3], penghematan biaya [3], dan memberikan perangkat lunak yang inovatif [4]. Dalam [4], disebutkan "*Innovation happens elsewhere*". Hal tersebut menunjukkan bahwa sehebat apapun organisasi kita, akan selalu ada orang-orang di luar sana yang akan terus berinovasi. Oleh karena itu, semangat pengembangan FOSS yang melibatkan berbagai jenis orang dari berbagai pelosok dunia ini menjanjikan perangkat lunak yang inovatif. Di sisi lain, FOSS juga memiliki kekurangan, diantaranya seperti banyaknya FOSS yang tidak terpublikasikan dengan baik, sehingga kita sulit untuk mencari FOSS tertentu [3], dan juga banyaknya FOSS yang tidak memiliki dokumentasi yang baik [3].

Dewasa ini FOSS tidak hanya bisa muncul sebagai sebuah alternatif tapi juga bisa menjadi solusi untuk memenuhi kebutuhan bisnis. Berbagai macam sektor bisnis mulai menggunakan FOSS. Bagi dunia bisnis, penggunaan FOSS bisa menjadi suatu solusi yang murah (*zero acquisition cost*) namun tetap bisa memberikan kualitas yang baik. Banyaknya pilihan FOSS untuk memenuhi berbagai macam kebutuhan bisnis merupakan suatu nilai tambah bagi FOSS. Kendati demikian, hal ini juga merupakan suatu permasalahan bagi FOSS. Banyaknya pilihan FOSS dengan beraneka ragam kualitas menyulitkan kita untuk memilih FOSS yang tepat. Melihat permasalahan tersebut, *paper* ini ditulis dengan tujuan untuk mengajukan suatu kerangka kerja (*framework*) yang dapat membantu memilih FOSS yang tepat.

Penjelasan dalam *paper* ini dimulai dengan gambaran umum dari kerangka kerja yang diajukan. Selanjutnya, *paper* ini memaparkan detail dari setiap langkah dalam kerangka kerja ini. Terakhir, *paper* ini di tutup dengan kesimpulan dan saran.

2. Gambaran Umum Kerangka Kerja

Memilih FOSS sebagai solusi teknis untuk suatu kebutuhan bisnis mirip dengan memilih *Commercial off-the-shelf* (COTS) *software product*. Menurut [5], langkah yang harus dilakukan dalam memilih COTS adalah:

1. Mengidentifikasi dan melakukan riset untuk mencari produk yang dapat mendukung solusi untuk kebutuhan bisnis kita.
2. Mengumpulkan, mengevaluasi, dan merangking proposal dari *vendor*.
3. Memilih proposal *vendor* terbaik.
4. Menghubungi *vendor* terpilih untuk mendapatkan produknya.

Kerangka kerja pemilihan FOSS yang diajukan ini dibuat dengan mencoba mengadaptasi dan melakukan

modifikasi pada langkah-langkah umum yang dilakukan dalam memilih COTS. Secara garis besar kerangka kerja pemilihan FOSS ini terdiri dari 4 tahapan, yaitu:

1. Penyusunan kriteria Evaluasi.
2. Pengidentifikasian/pencarian kandidat solusi.
3. Evaluasi kandidat solusi (dengan metode *Analytical Hierarchy Process*).
4. Penentuan solusi terpilih.

Penjelasan setiap tahapan dipaparkan pada subbab 3.1 - 3.4.

3. Pemaparan Kerangka Kerja

Bab ini menjelaskan detail dari setiap tahapan dalam kerangka kerja pemilihan FOSS ini.

3.1. Penyusunan Kriteria Evaluasi

Tahap penyusunan kriteria evaluasi bertujuan untuk menentukan jenis kriteria pengukuran yang digunakan dalam memilih FOSS. Kriteria evaluasi yang dipilih harus merefleksikan faktor kualitas dari FOSS yang dibutuhkan. Dalam kerangka kerja ini, proses penyusunan kriteria evaluasi dibedakan menjadi dua tahap, yaitu penyusunan kriteria evaluasi umum dan penyusunan kriteria evaluasi khusus. Pembagian dua tahap ini ditujukan untuk membantu proses penyusunan kriteria evaluasi agar tercipta kriteria yang tepat.

3.1.1. Penyusunan Kriteria Evaluasi Umum

Proses penyusunan kriteria evaluasi umum menitikberatkan pada pengidentifikasian faktor kualitas umum (tidak terlalu detil) yang diharapkan dari FOSS yang dicari. Kriteria ini merupakan faktor-faktor umum yang akan dipertimbangkan, ditinjau dan dinilai ketika memilih suatu FOSS. Beberapa kriteria evaluasi umum yang harus diperhatikan diantaranya adalah:

1. Lisensi.
   Kriteria ini menekankan pada jenis lisensi dari software yang dibutuhkan.
2. Kebutuhan dari sisi bisnis (*business need*).
   Kriteria ini menekankan pada kemampuan FOSS dalam memenuhi kebutuhan kita. Hal ini merupakan kriteria yang sangat penting, karena FOSS yang terpilih haruslah dapat memenuhi kebutuhan kita dan menjawab permasalahan yang dihadapi. Oleh karena itu, perlu dipertimbangkan fitur apa saja yang diberikan oleh kandidat FOSS yang dievaluasi, serta sejauh mana fitur-fitur tersebut dapat memenuhi kebutuhan kita.
3. Aspek teknis dari FOSS.
   Kriteria ini menekankan pada hal-hal teknis yang perlu diperhatikan dari FOSS, seperti kondisi teknis dari lingkungan tempat FOSS tersebut diimplementasikan, *bandwidth*, dll.
4. Aspek pengembangan lebih lanjut/ modifikasi.
   Kriteria ini menitikberatkan pada hal-hal yang berkaitan dengan pengembangan dari FOSS tersebut, seandainya ada pengembangan lebih lanjut, karena sejatinya kebutuhan terus berevolusi. Suatu FOSS terkadang tidak sepenuhnya cocok dengan kebutuhan atau lingkungan tempat FOSS tersebut di *deploy* atau terkadang pada suatu waktu FOSS perlu di modifikasi untuk memenuhi kebutuhan yang berubah. Oleh karena itu, terkadang kita perlu melakukan modifikasi atau penyesuaian yang diperlukan [5]. Suatu FOSS yang baik diharapkan memiliki *source code* yang *self-descriptive*, sehingga pengguna dapat melalukan modifikasi yang diperlukan dengan mudah [6].
5. Keamanan.
   Kriteria ini menekankan pada aspek-aspek keamanan dari penggunaan FOSS tersebut. Misalnya FOSS tersebut harus bebas dari ancaman keamanan seperti *snooping*, *spoofing/masquerading*, *repudation of origin*, *denial of receipt*, dll [7].
6. Kualitas *support* dan dokumentasi FOSS
   Kriteria ini menitikberatkan pada segi dokumentasi dan dukungan (*support*) dari FOSS tersebut. Kualitas *support* dan dokumentasi merupakan suatu aspek yang penting untuk diperhatikan dalam memilih FOSS karena sangat menyulitkan jika kita menggunakan suatu FOSS yang tidak memiliki kualitas *support* dan dokumentasi yang baik, padahal terkadang kita harus melakukan modifikasi tertentu.
7. Tingkat stabilitas dan kematangan (*maturity*).
   Kriteria ini menekankan pada tingkat stabilitas (sudah stabil atau belum) dari FOSS tersebut. Tingkat stabilitas dan kematangan merupakan suatu hal yang perlu diperhatikan karena hal ini menunjukkan seberapa tangguhnya FOSS tersebut untuk digunakan. Umumnya, semakin matang dan

stabil suatu FOSS, FOSS tersebut lebih dapat diandalkan.

Ketujuh kriteria umum yang telah dipaparkan bukanlah suatu kriteria mutlak yang harus diikuti dalam penggunaan kerangka kerja pemilihan FOSS ini. Hal tersebut merupakan aspek-aspek yang umumnya harus di perhatikan. Pemaparan kriteria evaluasi tersebut lebih ditujukan untuk membantu mengidentifikasi faktor kualitas umum yang ingin dicapai. Dalam penggunaan kerangka kerja ini, tidak menutup kemungkinan adanya penambahan atau pengurangan kriteria evaluasi sesuai kebutuhan. Beberapa hal yang dapat ditambahkan seperti *usability, portability, flexibility,* dll [8].

3.1.2. Penyusunan Kriteria Evaluasi Khusus

Dari setiap kriteria evaluasi umum yang telah diidentifikasi, elaborasi komponen-komponen yang lebih spesifik/khusus. Misalnya, kriteria umum "Kebutuhan dari sisi bisnis (*business need*)" dapat dielaborasi lagi menjadi daftar kebutuhan yang harus dipenuhi oleh FOSS yang dicari, seperti misalnya kemampuan melakukan pencatatan transaksi (*logging*). Contoh lain misalnya kriteria umum keamanan (*security*). Kriteria tersebut dapat dielaborasi menjadi ketahanan terhadap *Denial of Service* (DOS) *attack* [7] dan mampu mencegah *snooping* [7].

Letak perbedaan utama kriteria evaluasi umum dan kriteria evaluasi khusus adalah tingkat kespesifikannya. Dalam kriteria evaluasi khusus suatu kriteria bisa sangat spesifik untuk suatu jenis FOSS yang ingin dicari, sedangkan suatu kriteria evaluasi umum tidak hanya terkait pada suatu jenis FOSS yang ingin dicari akan tetapi bisa juga terkait atau bisa digunakan untuk mengevaluasi FOSS jenis lain.

3.1.3. Pendekatan dalam penyusunan kriteria evaluasi

Beberapa pendekatan dapat ditempuh untuk mengelaborasi jenis kriteria evaluasi, misalnya:

1. Mempelajari kebutuhan-kebutuhan yang ada. Dari kebutuhan tersebut, buat daftar kriteria yang harus dipenuhi.
2. Mempelajari contoh-contoh faktor kualitas perangkat lunak, seperti misalnya: *correctness, reliability, maintainability, efficiency, safety,* dll. Beberapa contoh faktor kualitas lain beserta penjelasannya dapat dilihat di [8]. Hal ini dapat membantu menyusun kriteria evaluasi, dikarenakan terkadang kita telah memahami kebutuhan kita, namun kita sulit untuk mengejawantahkannya dalam suatu bentuk daftar kriteria.
3. Mempelajari contoh-contoh dokumen evaluasi suatu perangkat lunak.
4. Berdiskusi dengan pihak-pihak yang cukup berpengalaman dengan konsep FOSS yang ingin dicari.

3.2. Pengidentifikasian/Pencarian Kandidat Solusi

Pada tahap ini dilakukan pencarian kandidat FOSS yang diperkirakan sesuai dengan kebutuhan kita. Tahap pencarian kandidat solusi ini diperlukan jika kita menggunakan kerangka kerja ini sebagai alat untuk memecahkan masalah. Dengan kata lain kita bergerak dari suatu permasalahan, dan berusaha mencari FOSS yang dapat mengatasi masalah tersebut. Jika kita menggunakan kerangka kerja ini hanya untuk menentukan pilihan dari sejumlah kandidat FOSS yang telah ditentukan sebelumnya, maka tahap ini dapat diabaikan.

Dalam mencari kandidat solusi, beberapa cara dapat dilakukan, seperti:

1. Melakukan pencarian di internet dengan menggunakan *search engine* (misal: google, yahoo search, vivisimo).
2. Berkonsultasi dengan penyedia jasa konsultasi Teknologi Informasi (TI).
3. Menghubungi *vendor* penyedia *support* untuk FOSS.
4. Berdiskusi dalam komunitas. Karena sebagian besar FOSS memiliki komunitas pengguna ataupun pengembangnya.

3.3. Penentuan solusi dengan metode *Analytical Hierarchy Process* (AHP)

Tahap ini merupakan proses penentuan solusi dengan menggunakan metode AHP. AHP merupakan suatu metode untuk memilih solusi terbaik dari beberapa alternatif solusi dengan melibatkan/mempertimbangkan beberapa kriteria pemilihan [1,9,10]. Metode ini membantu pengambil keputusan memilih kandidat solusi yang paling baik dalam memenuhi kriteria yang telah ditentukannya [11]. AHP telah digunakan untuk menyelesaikan berbagai macam permasalahan menentukan solusi yang melibatkan beberapa kandidat solusi serta mempertimbangkan beberapa kriteria pemilihan dalam berbagai jenis bidang. Beberapa contohnya adalah seperti:

1. Dalam bidang teknik (*engineering*) [12]. Dalam bidang teknik industry (*industrial engineering*), AHP digunakan dalam manufaktur yang terintegrasi (*integrated manufacturing*) [13], evalusi dalam keputusan investasi teknologi [14], desain

layout [15], sistem manufaktur yang fleksibel [16], dan juga mengatasi masalah teknis [17].
2. Dalam bidang ekonomi dan perencanaan [18-21].
3. Dalam bidang kesehatan [1,22].

Pemilihan FOSS sebagai solusi untuk memenuhi kebutuhan bisnis merupakan suatu permasalahan menentukan solusi terbaik dari beberapa kandidat solusi yang tersedia dengan mempertimbangkan beberapa kriteria pemilihan. Dalam hal ini, berbagai macam pilihan FOSS menjadi kandidat solusi dan kriteria FOSS yang kita cari menjadi kriteria pemilihan yang dipertimbangkan. Berdasarkan hal tersebut dan juga kegunaan dari AHP, AHP merupakan metode yang cocok untuk membantu menyelesaikan permasalahan ini. Dengan AHP, kita dapat menentukan FOSS yang terbaik dari semua pilihan FOSS berdasarkan kriteria pemilihan yang telah ditentukan.

Dalam kerangka kerja pemilihan FOSS ini, AHP berguna untuk:

1. Menentukan titik berat kriteria evaluasi yang diinginkan.
2. Menentukan preferensi solusi untuk setiap kriteria evaluasi.
3. Memilih urutan (*ranking*) kandidat solusi berdasarkan perhitungan yang dilakukan pada poin ke 1 dan 2.

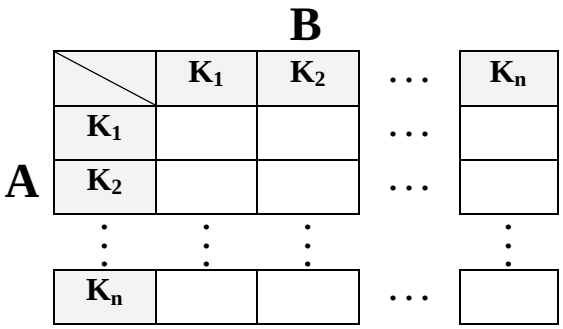
Sub-bab 3.3.1 – 3.3.3 menjelaskan detail penggunaan AHP dalam kerangka kerja ini.

3.3.1. Penentuan Preferensi Kriteria Evaluasi

Dalam tahap ini kita menentukan bobot untuk setiap kriteria evaluasi khusus berdasarkan tingkat preferensi/prioritasnya. Untuk itu kita buat tabel perbandingan kriteria. Misalkan kita memiliki n kriteria (k_1, k_2, ..., k_n), tabel perbandingan kriteria yang disusun akan memiliki ukuran (n+1) x (n+1), dengan kriteria-kriteria tersebut sebagai judul kolom dan baris. Secara garis besar tabel perbandingan tersebut terlihat seperti Gambar 1.

Langkah selanjutnya adalah mengisi setiap sel dalam tabel tersebut dengan nilai hasil perbandingan antar setiap kriteria pada sisi A dengan sisi B. Misalkan kita membandingkan kriteria K_1 dan K_2, aturan penilaian untuk mengisi sel tersebut adalah:

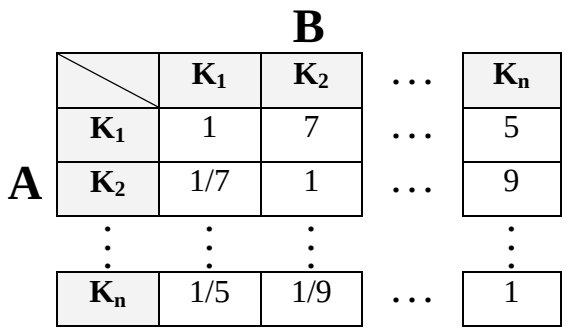
1. Nilai 1/9, kriteria K_2 sangat lebih diinginkan dari kriteria K_1.
2. Nilai 1/8, kriteria K_2 lebih diinginkan dari K_1 hingga sangat lebih diinginkan dari kriteria K_1.



Gambar 1. Ilustrasi tabel perbandingan kriteria evaluasi

3. Nilai 1/7, kriteria K_2 lebih diinginkan dari kriteria K_1.
4. Nilai 1/6, kriteria K_2 cukup lebih diinginkan dari K_1 hingga lebih diinginkan dari kriteria K_1.
5. Nilai 1/5, kriteria K_2 cukup lebih diinginkan dari kriteria K_1.
6. Nilai 1/4, kriteria K_2 agak lebih diinginkan dari K_1 hingga cukup lebih diinginkan dari kriteria K_1.
7. Nilai 1/3, kriteria K_2 agak lebih diinginkan dari kriteria K_1.
8. Nilai 1/2, kriteria K_2 sama-sama lebih diinginkan dari kriteria K_1 hingga agak lebih diinginkan dari kriteria K_1.
9. Nilai 1, kedua kriteria sama-sama diinginkan.
10. Nilai 2, kriteria K_1 sama-sama lebih diinginkan dari kriteria K_2 hingga agak lebih diinginkan dari kriteria K_2.
11. Nilai 3, kriteria K_1 agak lebih diinginkan dari kriteria K_2.
12. Nilai 4, kriteria K_1 agak lebih diinginkan dari K_2 hingga cukup lebih diinginkan dari kriteria K_2.
13. Nilai 5, kriteria K_1 cukup lebih diinginkan dari kriteria K_2.
14. Nilai 6, kriteria K_1 cukup lebih diinginkan dari K_2 hingga lebih diinginkan dari kriteria K_2.
15. Nilai 7, kriteria K_1 lebih diinginkan dari kriteria K_2.
16. Nilai 8, kriteria K_1 lebih diinginkan dari K_2 hingga sangat lebih diinginkan dari kriteria K_2.
17. Nilai 9, kriteria K_1 sangat lebih diinginkan dari kriteria K_2.

Gambar 2 mencontohkan pemberian nilai perbandingan antar kriteria. Dalam ilustrasi pada Gambar 2 , misalkan kita membandingkan kriteria K_1



Gambar 2. Contoh pemberian nilai pada tabel perbandingan kriteria evaluasi

dengan K_2, karena menurut kita K_1 lebih penting dari kriteria K_2, kita taruh nilai 7 pada sel (K_1, K_2) dan nilai 1/7 pada sel (K_2, K_1). Dari ilustrasi pengisian tabel perbandingan ini terlihat bahwa matriks tersebut memiliki ciri yaitu nilai setiap sel pada segitiga di atas diagonal saling *invers* dengan nilai sel pada segitiga di bawah diagonal. Ciri yang lainnya adalah setiap sel pada diagonal matriks ini bernilai 1, hal ini dikarenakan setiap kriteria sama pentingnya dengan dirinya sendiri.

Setelah melakukan perbandingan setiap kriteria, kita lakukan normalisasi pada matriks ini dan lakukan penghitungan nilai rata-rata setiap baris untuk memperoleh bobot dari setiap kriteria. Untuk melakukan normalisasi, jumlahkan seluruh isi kolom, lalu bagi semua isi sel dalam kolom dengan nilai hasil penjumlahan tersebut. Berikut adalah contohnya, misalkan kita memiliki matriks hasil penilaian seperti yang ditunjukkan pada Tabel 1. Setelah dilakukan normalisasi kita memperoleh hasil seperti yang ditunjukkan pada Tabel 2.

Tabel 1. Contoh tabel perbandingan kriteria evaluasi

	K_1	K_2	K_3
K_1	1	1/3	7
K_2	3	1	6
K_3	1/7	1/6	1

Tabel 2. Contoh hasil normalisasi tabel perbandingan kriteria evaluasi

	K_1	K_2	K_3
K_1	0.241	0.222	0.5
K_2	0.724	0.667	0.429
K_3	0.034	0.111	0.071

Tabel 3. Contoh tabel bobot kriteria evaluasi

Kriteria	Bobot
K_1	0.321
K_2	0.607
K_3	0.072

3.3.2. Penentuan Preferensi Kandidat Evaluasi

Untuk menentukan preferensi kandidat solusi, kita bandingkan setiap kandidat solusi untuk setiap kriteria evaluasi. Untuk itu kita buat tabel perbandingan kandidat solusi. Tabel tersebut mirip dengan tabel yang dibuat untuk menentukan preferensi kriteria evaluasi, namun judul kolom dan barisnya merupakan nama kandidat solusi. Jika kita memiliki n kriteria evaluasi dan m kandidat solusi, maka kita akan memiliki n buah tabel perbandingan kandidat solusi, yang setiap tabelnya berukuran m x m. Dengan kata lain kita harus membuat tabel perbandingan kandidat solusi sebanyak jumlah kriteria evaluasi.

Dalam suatu tabel perbandingan kandidat solusi, untuk suatu kriteria evaluasi, kita akan menentukan nilai preferensi suatu kandidat solusi dengan cara melakukan perbandingan antar semua kandidat solusi dalam memenuhi kriteria solusi tersebut. Misalkan pada suatu tabel perbandingan kandidat solusi untuk kriteria evaluasi "kualitas dokumentasi", bandingkan setiap kandidat solusi dan dalam setiap perbandingan tersebut kita tentukan mana yang paling memiliki dokumentasi yang baik diantara setiap dua kandidat yang dibandingkan tersebut dan seberapa lebih baiknya dokumentasi kandidat tersebut. Kriteria penilaian untuk setiap perbandingan sama dengan kriteria penilaian yang digunakan untuk menentukan preferensi kriteria evaluasi.

Setelah dilakukan pembandingan, lakukan normalisasi pada setiap tabel perbandingan kandidat solusi. Cara menormalisasinya sama seperti pada perbandingan kriteria evaluasi. Setelah itu, hitung nilai rata-rata per baris untuk setiap tabel perbandingan kandidat solusi. Nilai rata-rata tersebut merupakan bobot preferensi suatu kandidat solusi pada suatu kriteria evaluasi.

Pada akhir tahap ini, kita peroleh suatu tabel bobot setiap kandidat solusi untuk setiap kriteria evaluasi. Contohnya adalah seperti yang ditunjukkan pada Tabel 4.

Tabel 4. Contoh tabel nilai setiap kandidat solusi untuk setiap kriteria evaluasi

	K_1	K_2	K_3	K_4	…	K_n
S_1					…	
S_2					…	
⋮	⋮	⋮	⋮	⋮		⋮
S_m					…	

Ket:

- S_i = Kandidat solusi ($1 \le i \le m$)
- K_j = Kriteria evaluasi ($1 \le j \le n$).

3.3.3. Penentuan Urutan (Rangking) Kandidat Evaluasi

Untuk menentukan urutan kandidat solusi, langkah yang dilakukan adalah:

1. Mengalikan setiap bobot preferensi kandidat solusi untuk suatu kriteria dengan bobot dari kriteria tersebut.
2. Jumlahkan semua hasil perkalian tersebut. Kita peroleh nilai untuk setiap kandidat solusi.
3. Urutkan nilai kandidat solusi mulai dari nilai tertinggi ke nilai terendah.
4. Terakhir, kita dapatkan urutan kandidat solusi mulai dari kandidat solusi terbaik.

3.4. Penentuan solusi terpilih

Tahap ini merupakan tahap terakhir dari kerangka kerja ini. Dalam ini kita peroleh kandidat solusi terbaik, yaitu kandidat solusi dengan nilai tertinggi berdasarkan langkah sebelumnya.

4. Contoh Penggunaan Kerangka Kerja

Bab ini memberikan contoh penggunaan kerangka kerja ini dalam memilih FOSS. Tujuan contoh ini hanyalah untuk memberikan gambaran bagaimana menggunakan kerangka kerja ini. Dalam contoh ini tidak digunakan produk FOSS yang benar-benar ada. Untuk menyederhanakan contoh, kriteria evaluasi yang dibandingkan tidak akan terlalu banyak. Akan tetapi, dalam kondisi nyata, kriteria pemilihan yang digunakan bisa cukup banyak tergantung kebutuhan kita.

Contoh ini menunjukkan bagaimana kita menggunakan kerangka kerja ini dalam memilih *Integrated Development Environment* (IDE) untuk bahasa pemrograman J. Langkah-langkah pemilihan dijelaskan pada 4.1 – 4.4.

4.1. Penyusunan Kriteria Evaluasi

Dalam tahap penyusunan kriteria evaluasi umum, dihasilkan kriteria evaluasi:

1. Kebutuhan dari sisi bisnis (*business need*).
2. Aspek teknis dari FOSS.
3. Kualitas dokumentasi

Setelah diperoleh kriteria evaluasi umum, kita elaborasi lagi kriteria tersebut untuk memperoleh kriteria evaluasi khusus. Berikut adalah kriteria evaluasi khusus yang diperoleh:

1. Kebutuhan dari sisi bisnis (*business need*) di elaborasi menjadi:
   a. Kemampuan *code completion* (selanjutnya disebut $\mathbf{K_1}$).
   b. Kemampuan manajemen proyek aplikasi yang baik (selanjutnya disebut $\mathbf{K_2}$).
2. Aspek teknis dari FOSS dielaborasi menjadi:
   a. Minimum system requirement yang dibutuhkan (selanjutnya disebut $\mathbf{K_3}$).
3. Kualitas dokumentasi dielaborasi menjadi:
   a. Kualitas dan kemudahan memperoleh dokumentasi penggunaan (selanjutnya disebut $\mathbf{K_4}$).

4.2. Pencarian Kandidat Solusi

Dalam tahap ini kita lakukan pencarian kandidat solusi. Dalam contoh ini, kita asumsikan kandidat solusi yang didapat adalah IDE A dan IDE B.

4.3. Evaluasi kandidat solusi dengan metode AHP

Dalam tahap ini, kita evaluasi kandidat solusi dengan menggunakan AHP. Proses ini terbagi dalam tiga tahap. Berikut adalah penjelasannya:

1. Menentukan preferensi evaluasi kriteria. Tabel 5 menunjukkan perbandingan kriteria evaluasi. Selanjutnya normalisasi Tabel 5, kita peroleh Tabel 6.

Tabel 5. Tabel perbandingan kriteria evaluasi

	K_1	K_2	K_3	K_4
K_1	1	1/4	1/5	1/7
K_2	4	1	1/3	1/5
K_3	5	3	1	1/3
K_4	7	5	3	1

Tabel 6. Hasil normalisasi tabel perbandingan kriteria evaluasi

	K_1	K_2	K_3	K_4	Rata-rata
K_1	0.059	0.027	0.044	0.085	0.054
K_2	0.235	0.108	0.073	0.119	0.134
K_3	0.294	0.324	0.221	0.199	0.260
K_4	0.412	0.540	0.662	0.597	0.553

2. Menentukan preferensi evaluasi kandidat solusi. Tabel 7 a - d menunjukkan perbandingan kandidat solusi untuk kriteria $K_1 – K_4$. Selanjutnya normalisasi Tabel 7 a - d, kita peroleh Tabel 8 a - d.

Tabel 7. Tabel perbandingan kandidat solusi. a) Untuk Kriteria K_1. b) untuk Kriteria K_2. c) untuk kriteria K_3. d) untuk kriteria K_4.

a)

	A	B
A	1	1/2
B	2	1

b)

	A	B
A	1	3
B	1/3	1

c)

	A	B
A	1	1/5
B	5	1

d)

	A	B
A	1	6
B	1/6	1

Tabel 8. Normalisasi tabel perbandingan kandidat solusi. a) Untuk Kriteria K_1. b) untuk Kriteria K_2. c) untuk kriteria K_3. d) untuk kriteria K_4.

a)

	A	B	Rata-rata
A	0.333	0.333	0.333
B	0.667	0.667	0.667

b)

	A	B	Rata-rata
A	0.750	0.750	0.750
B	0.250	0.250	0.250

c)

	A	B	Rata-rata
A	0.167	0.167	0.167
B	0.833	0.833	0.833

d)

	A	B	Rata-rata
A	0.857	0.857	0.857
B	0.143	0.143	0.143

3. Menentukan urutan/*rangking* kandidat solusi.

   Untuk kandidat IDE A: (0.333 x 0.054) + (0.750 x 0.134) + (0.167 x 0.260) + (0.857 x 0.553) = 0.636.

   Untuk kandidat IDE B: (0.667 x 0.054) + (0.250 x 0.134) + (0.833 x 0.260) + (0.143 x 0.553) = 0.366.

Berdasarkan langkah 4.3, kita peroleh kandidat solusi terbaik adalah IDE A.

5. Kesimpulan dan Saran

Sebuah kerangka kerja untuk memilih FOSS telah diajukan. Kerangka kerja ini memanfaatkan metode *Analytical Hierarcy Process* (AHP), suatu metode pengambilan keputusan yang diharapkan mampu membantu merefleksikan kebutuhan kita dan membantu memilih FOSS yang tepat.

Kerangka kerja ini dapat mengakomodasi keperluan pemecahan suatu masalah ataupun hanya untuk memilih FOSS dari beberapa kandidat FOSS yang telah ditetapkan. Dalam penggunaan untuk memecahkan masalah atau suatu keinginan memenuhi kebutuhan, kita bergerak dari suatu keperluan/ permasalahan dan berusaha mencari FOSS yang dapat mengatasi masalah tersebut.

Selain untuk membantu pemilihan FOSS, dengan sedikit modifikasi kerangka kerja ini juga dapat digunakan untuk memilih COTS *software product*. Secara umum perbedaannya terletak pada kriteria evaluasi yang digunakan dan penitikberatan kriteria evaluasi. Langkah secara umum dapat dikatakan akan mirip, namun dapat juga dilakukan sedikit modifikasi.

Dalam penggunaan kerangka kerja ini, disarankan untuk membentuk suatu tim kerja, agar proses pemberian nilai, evaluasi dan pembandingan lebih bersifat objektif dan komprehensif (menyeluruh atau mempertimbangkan segala aspek/sudut pandang). Guna memperluas sudut pandang tim kerja, disarankan untuk meningkatkan pluralitas dalam tim kerja tersebut. Dengan demikian diharapkan proses pemilihan benar-benar menghasilkan pilihan yang tepat.

REFERENSI

[1] T. L. Saaty. “The Analytic Hierarchy Process and Health Care Problems”. Proceeding of International Conference on System Science in Health Care pp 147 – 158. 1980.

[2] J. Feller, B. Fitzgerald, dkk. *Perspective on Free and Open Source Software*. The MIT Press. Massachusetts. 2005.

[3] V. V. Reijswoud dan A. D. Jager. *Free and Open Source Software for development exploring expectations, achievements, and the future*. Polimetrica. Italy. 2008.

[4] R. Goldman dan R. P. Gabriel. *Innovation Happens Elsewhere, Open Source as Business Strategy*. Elsevier. San Fransisco. 2005.

[5] L. D. Bentley dan J. L. Whitten. System *Analysis and Design for the Global Enterprise*. McGraw-Hill. New York. 2007.

[6] R. S. Pressman. *Software Engineering A Practitioner’s Approach*. McGraw-Hill. New York. 2005.

[7] M. Bishop. *Computer Security: Art and Science*. Addison Wesley. Boston. 2002.

[8] D. Galin. *Software Quality Assurance From Theory to Implementation*. Prentice Hall. England. 2004.

[9] E. Turban, J. E. Aronson dan T. Liang. *Decision Support Systems and Intelligent Systems*. Pearson Education. New Jersey. 2005.

[10] T. L. Saaty. "Decision Making with Analytic Hierarchy Process". Int. J. Service of Sciences, vol 1, No. 1, pp 83 – 98, 2008.
[11] B. W. Taylor. *Introduction to Management Science*. 8th edition. Prentice Hall. 2001.
[12] E. Triantaphyllou dan S. H. Mann. "Using the Analytic Hierarchy Process for Decision Making in Engineering Application: Some Challenges". International Journal of Industrial Engineering: Application and Practice, Vol 2, no 1, pp 35-44, 1995.
[13] P. Putrus. "Accounting for Intangibles in Integrated Manufacturing (nonfinancial justification based on the Analytical Hierarchy Process)". Information Strategy, 6, 25-30. 1990.
[14] T. O. Boucher dan E. L. McStravic. "Multi-attribute Evaluation Within a Present Value Framework and its Relation to the Analytic Hierarchy Process". The Engineering Economist, 37, 55-71. 1991.
[15] K. E. Cambron dan G. W. Evans. "Layout Design Using the Analytic Hierarchy Process". Computers & IE, 20, 221 – 229. 1991.
[16] R. N. Wabalickis. "Justification of FMS With the Analytic Hierarchy Process". Journal of Manufacturing Systems, 17, 175 – 182. 1988.
[17] L. Wang dan T. Raz. "Analytic Hierarchy Process Based on Data Flow Problem". Computers & IE, 20, 355 - 365. 1991.
[18] J. R. Emshoff dan T. L. Saaty. "Application of Analytic Hierarchy Process to Long-range Planning Processes". European Journal of Operational Research, Vol 10, No 2, pp 131 – 143. 1982.
[19] T. L. Saaty dan L. G. Vargas. "A Note on Estimating Technological Coefficients by the Analytic Hierarchy Process". Socia Economic Planning Sciences, Vol 13, No. 6, pp 333 – 336. 1979.
[20] T. L. Saaty. "The Sudan Transport Study". Interfaces, Vol 8, No. 1, pp 37 – 57. 1977.
[21] V. Ramanujam dan T. L. Saaty. "Technological choices in the less developed countries: An Analytic Hierarchy Approach". Technological Forecasting and Social Change, Vol 19, No. 1, pp 81 – 98. 1981.
[22] E. J. Lusk. "Analysis of Hospital Capital Decision Alternatives: A Priority Assignment Model". Journal of the Operational Research Society, Vol 30, No 5, pp 439 - 448. 1979.