Teknik Penentuan Prioritas Product Backlog pada Metode Scrum Menggunakan Pendekatan Program Dinamis

Addin Nabilal Huda - 13520045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520045@std.stei.itb.ac.id

***Abstrak*—Scrum sebagai salah satu metode pengembangan lunak Agile merupakan metode yang dinilai dapat beradaptasi dengan baik pada perubahan. P*roduct backlog*¸ salah satu artefak penting dalam Scrum, berisi daftar fitur-fitur yang akan dikembangkan yang terurut berdasarkan prioritas. Teknik penentuan prioritas *product backlog* dapat dimodelkan sebagai persoalan *knapsack* dengan setiap fitur memiliki nilai keuntungan dan poin kompleksitas sebagai nilai bobot. Sementara, batasan yang digunakan berupa total poin kompleksitas pada setiap tahapan pengembangan perangkat lunak (*sprint)*. Melalui pendekatan program dinamis, akan dihasilkan *sprint backlog* berisi daftar fitur dengan prioritas tertinggi yang akan diimplementasikan pada suatu *sprint*.**

***Kata Kunci*—Scrum, Agile, *Product Backlog*, Program Dinamis.**

I. PENDAHULUAN

Scrum merupakan salah satu metode pengembangan perangkat lunak yang banyak digunakan oleh perusahaan-perusahaan besar dunia. Bahkan, perusahaan Google, Microsoft, Amazon, dan Adobe menerapkan metode Scrum dalam pengembangan proyeknya. Dalam metode scrum, pengembangan perangkat lunak dibagi menjadi beberapa proses yang disebut sebagai *sprint*.

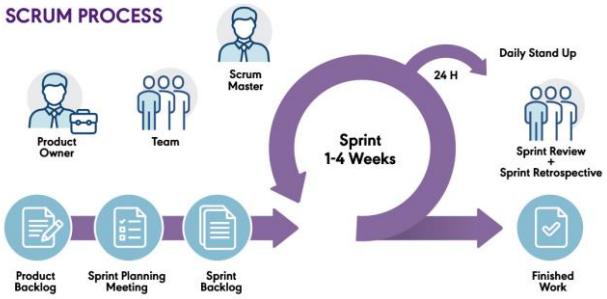
Salah satu hal yang esensial dalam metode Scrum adalah *Product Backlog*, yakni list yang berisi keseluruhan fitur dari produk yang akan dikembangkan. List ini dibuat terurut berdasarkan prioritas sesuai dengan *product roadmap* dan *requirements*. *Product* backlog harus fleksibel dan akan sering berubah karena faktor lain yang mempengaruhi urutan prioritas *product backlog.* Untuk setiap iterasi atau *sprint* dalam metode Scrum, *product owner* harus menentukan fitur-fitur yang akan dimasukkan pada tahap tersebut ke dalam *sprint backlog* berdasarkan prioritas pada *product backlog.* Di sisi lain, *product owner* sebagai orang yang bertanggung jawab pada *product backlog* maupun *sprint backlog* seringkali tidak memiliki cukup waktu untuk mengorganisasi dan mengatur prioritas *backlog backlog*. Untuk itu, teknik penentuan prioritas *product backlog* sangat diperlukan untuk bisa mengorganisasi *product backlog* sesuai dengan visi strategis perusahaan. *Product backlog* yang baik akan menentukan keberhasilan setiap *sprint* dalam mengembangkan produk dengan nilai bisnis tinggi.

Teknik pengurutan prioritas *product backlog* bukanlah sebuah ilmu pasti yang kaku. Terdapat beberapa teknik pengurutan prioritas yang sudah dikembangakan di antaranya Stack Ranking, Kano Model, MoSCoW Method, dan Cost of Delay. Teknik-teknik tersebut memiliki pendekatan yang berbeda dengan perbedaan faktor yang mempengaruhi pengurutan pula.

Makalah ini membahas pengembangan teknik baru dalam penentuan prioritas *product backlog* pada metode Scrum. Pada teknik ini, persoalan penentuan prioritas dimodelkan sebagai persoalan 1/0 (integer) *Knapsack* yang diselesaikan dengan pendekatan program dinamis. Keluaran dari sistem yang dibuat berupa fitur-fitur prioritas dari *product backlog* yang dimuat dalam *sprint backlog*. Dengan teknik ini, fitur-fitur yang dikembangkan pada setiap *sprint* dapat menghasilkan nilai yang optimal dan memberikan kepuasan tinggi untuk pelanggan.

II. LANDASAN TEORI

A. *Metode Scrum*



Gambar 1. Metode Scrum
Sumber: [7]

Metode Scrum merupakan metode yang termasuk dalam *agile software development*. *Agile software development* dikembangkan berdasarkan pola pengembangan *iterative and*

incremental untuk dapat beradaptasi pada perubahan secara cepat. Dengan menerapkan metode Agile, tim dapat menghasilkan nilai lebih untuk bisnis melalui penggunaan sumber daya yang efisien dan proses pengembangan yang cepat dan produktif.

Scrum adalah metode Agile yang memungkinkan tim untuk fokus menghasilkan nilai bisnis maksimal dalam waktu yang minimal. Metode Scrum pertama kali digunakan dalam pengembangan perangkat lunak oleh Easel Corporation pada tahun 1993. Pada metode Scrum, proses pengembangan perangkat lunak dibagi menjadi tahapan-tahapan yang disebut dengan *sprint.* Pada setiap *sprint* yang umumnya berdurasi 2-4 minggu, perangkat lunak didesain, dibuat, dan dites sepanjang *sprint* berlangsung. Peran-peran yang berkontribusi dalam metode Scrum di antaranya:

1. *Product owner*

   Product owner merupakan orang yang bertanggung jawab untuk menentukan fitur-fitur produk dan menentukan prioritas dari fitur-fitur tersebut sesuai nilai bisnis. *Product owner* juga bertugas menentukan tanggal rilis produk dan menerima/menolak hasil pengembangan produk.

2. *Scrum master*

   Scrum master merupakan representasi manajemen sebuah proyek yang bertanggung jawab memastikan keberlangsungan nilai dan praktik *scrum.*

3. *Scrum team*

   Scrum team umumnya terdiri atas 5-10 orang yang terdiri atas orang-orang dari latar belakang yang bersilangan, seperti *quality assurance,* programmer, UI *designer,* dan sebagainya.

Untuk dapat menerapkan metode Scrum, orang-orang yang terlibat dalam proyek berkomitmen untuk melakukan aktivitas-aktivitas yang disebut sebagai *scrum ceremonies* di antaranya:

1. *Sprint planning meeting*

   Pada aktivitas ini, *sprint backlog* (akan dijelaskan kemudian) dibuat dan tujuan dari *sprint* akan ditentukan. *Scrum team* juga akan menentukan poin dari setiap fitur pada *product backlog* berdasrkan metrik tertentu untuk kemudian ditentukan prioritas pengembangan fitur-fitur tersebut.

2. *Sprint*

   Pada setiap *sprint,* produk dikembangkan sesuai fitur yang tertera di *sprint backlog*. Setiap *sprint* harian dimulai dengan *daily scrum meeting* untuk melaporkan kemajuan dan kendala dalam pengembangan.

3. *Sprint review meeting*

   Setelah suatu *sprint* selesai, tim akan mempresentasikan/mendemokan fitur yang dibuat ke pelanggan pada *sprint review meeting.*

Dalam Scrum, terdapat beberapa alat yang digunakan untuk merepresentasikan pekerjaan yang disebut sebagai artefak, di antaranya:

1. *Product backlog*

   Product backlog berisi daftar pekerjaan/fitur yang harus diselesaikan dalam keseluruhan proyek pengembangan perangkat lunak. *Product backlog* dikelola oleh *product owner* dan bersifat terus berubah seiring perubahan pasar. Daftar pekerjaan/fitur dalam *product backlog* terurut sesuai prioritas pengembangan.

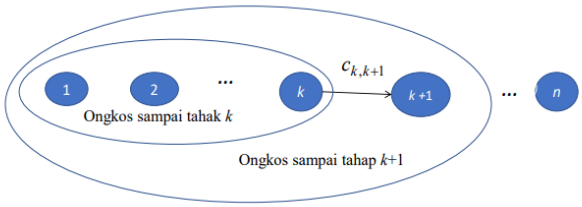
2. *Sprint backlog*

   Sprint backlog berisi daftar pekerjaan/fitur untuk diimplementasi pada suatu siklus *sprint.*

B. *Program Dinamis*

Program dinamis (*dynamic programming)* adalah metode penyelesaian masalah dengan cara menguraikan solusi menjadi tahapan-tahapan sedemikian rupa sehingga solusi persoalan menjadi serangkaian keputusan yang saling berkaitan. Algoritma ini dirancang untuk mengoptimasi msalah yang dapat dimodelkan sebagai persoalan matematis.

Prosedur yang digunakan dalam pemrograman dinamis adalah prosedur rekursif, yakni setiap kali keputusan diambil harus memperhatikan keadaan yang dihasilkan dari keputusan sebelumnya. Artinya, keputusan optimal sebelumnya merupakan landasan bagi keputusan optimal berikutnya. Rangkaian keputusan yang optimal tersebut merupakan penerapan prinsip optimalitas yang menyebut "jika solusi optimal, maka bagian solusi sampai ke tahap-k juga optimal". Dengan berprinsip optimalitas, jika kita mencari solusi dari tahap k ke tahap k+1, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke awal. Dengan begitu, ongkos pada tahap k+1 merupakan penjumlahan antara ongkos yang dihasilkan pada tahap-k dan ongkos dari tahap k ke tahap k+1.



Gambar 2. Gambaran perhitungan pada program dinamis
Sumber: [1]

Pada setiap tahapan program dinamis, terdapat sejumlah status (macam kemungkinan masukan pada suatu tahap) yang berhubungan dengan tahap tersebut. Hasil keputusan pada seluruh tahap kemudian ditransformasikan untuk digunakan sebagai status pada tahap berikutnya. Sebagai konsekuensi dari transformasi tersebut, ongkos total akan meningkat secara teratur seiring dengan bertambahnya jumlah tahapan. Prosedur penyelesaian masalah dengan program dinamis dapat didekati dengan 2 macam pendekatan:

1. Program dinamis maju (*forward* atau *up-down)*

   Pada program dinamis maju, perhitungan dimulai dari tahap awal ke tahap akhir, yakni dimulai dengan mencari nilai keuntungan di tahap 1, kemudian dilanjutkan di tahap 2, dan seterusnya sampai tahap n.

2. Program dinamis mundur (*backward* atau *bottom-up)*

   Pada program dinamis mundur, perhitungan dimulai dari tahap akhir ke tahap awal, yakni dimulai dengan mencari nilai keuntungan di tahap n, kemudian dilanjutkan di tahap n-1, dan seterusnya sampai tahap 1.

Untuk menyelesaikan persoalan dengan pendekatan program dinamis, digunakan langkah-langkah sebagai berikut:

1. Karakteristikkan struktur solusi optimal. Kararkter tersebut dapat terdiri atas tahap, variabel keputusan, status, dll.
2. Definisikan hubungan nilai optimal antartahap secara rekursif
3. Hitung nilai solusi optimal secara maju atau mundur menggunakan tabel
4. Rekonstruksi solusi optimal (opsional)

C. Persoalan 0/1 Knapsack

Knapsack merupakan kantung yang memiliki kapasitas tertentu dan digunakan sebagai tempat menyimpan objek-objek berbobot. Pada persoalan knapsack, kita diharuskan memilih beberapa objek dengan ketentuan total bobot kurang dari kapasitas kantung yang menghasilkan keuntungan terbesar. Terdapat persoalan knapsack yang memungkinkan pemasukan hanya sebagian objek yang dikenal sebagai *fractional knapsack problem*. Sementara, persoalan 0/1 *knapsack* mengharuskan pemasukan objek secara keseluruhan. Dengan kata lain, pilihan yang diberikan hanya dua, yaitu memasukkan atau tidak memasukkan objek ke kantung. Persoalan 0/1 *knapsack* dapat dimodelkan sebagai berikut

Fungsi maksimum:

$$Z = \sum_{i=1}^{n} p_i x_i$$

Fungsi kendala:

$$\sum_{i=1}^{n} w_i x_i \leq M$$

Keterangan:

Z = keuntungan total

n = banyak jenis objek

p = keuntungan tiap objek

w = berat objek

M = kapasitas maksimal objek

x = 0, jika objek-i tidak dimasukkan dan 1 jika objek-i dimasukkan

Persoalan knapsack dapat didekati oleh beberapa algoritma. Salah satu pendekatan yang popular adalah pendekatan greedy yang memungkinkan kita untuk mengambil objek secara greedy berdasarkan bobot, keuntungan, ataupun densitas objek. Namun, pendekatan secara greedy hanya optimal untuk persoalan *fractional knapsack* dan tidak dapat memberikan hasil optimal untuk persoalan 0/1 knapsack.

Kita dapat selalu mendapatkan hasil optimal menggunakan pendekatan *brute force*. Namun, algoritma *brute force* sangat tidak efektif karena untuk n objek, kita perlu membangkinkan 2^n solusi. Untuk mengoptimasi brute force, pendekatan *backtracking* dapat digunakan karena kita dapat mengabaikan solusi yang tidak layak. Pendekatan lain adalah dengan algoritma *branch and bound*. Algoritma branch and bound merupakan algoritma berbasis *backtracking* yang bekerja lebih baik dibandingkan brute force dengan mengabaikan solusi yang tidak layak. Algoritma *branch and bound* sangat baik digunakan, tetapi untuk kasus terburuk kita perlu membangkitkan seluruh ruang solusi.

Selain itu, program dinamis juga dapat digunakan untuk menyelesaikan persoalan knapsack. Pada pendekatan program dinamis, proses pengambilan keputusan dilakukan secara bertahap dan perhitungan dilakukan di tahap yang berbeda melalui rekursif.

D. Pendekatan Program Dinamis untuk Persoalan 0/1 Knapsack

Pada persoalan 0/1 *knapsack* menggunakan pendekatan program dinamis, tahap k dimodelkan sebagai proses memasukkan objek ke dalam kantung *knapsack*. Sementara, status (y) merupakan kapasitas maksimal *knapsack* yang tersisa setelah proses pemasukan objek k-1 pada tahap sebelumnya. Langkah-langkah pemecahan masalah yang digunakan mengikuti mekanisme berikut:

1. Prosedur pemecahan masalah dimulai dari tahap ke-1, yaitu ketika kita memasukkan objek pertama ke dalam *knapsack* hingga batas maksimal *knapsack*
2. Pada setiap tahap k, kapasitas muat *knapsack* saat ini adalah y-w_k
3. Untuk mengisi kapasitas sisa *knapsack*¸ prinsip optimalitas digunakan dengan mengacu pada solusi optimum dari tahap sebelumnya yaitu $f_{k-1} = (y - w_k)$
4. Pada setiap tahap, kita bandingkan nilai keuntungan pada tahap k (p_k) ditambah dengan nilai keuntungan pada tahap sebelumnya $(f_{k-1}(y - w_k))$ dengan keuntungan pengisian bila k tidak dimasukkan $(f_{k-1}(y))$
5. Dari perhitungan sebelumnya, jika $p_k + f_{k-1}(y - w_k)$ bernilai lebih kecil daripada $f_{k-1}(y)$, maka objek ke-k tidak dimasukkan ke dalam *knapsack*. Sementara bila sebaliknya, objek ke-k dimasukkan

Relasi rekurens yang berlaku untuk persoalan ini adalah:

$f_0(y) = 0, \quad y = 0,1, 2, \dots, M \ (basis)$

$f_k(y) = -\infty, \ y < 0 \ (basis)$

$f_k(y) = \max\{f_{k-1}(y), p_k + f_{k-1}(y - w_k)\}, k = 1,2, \dots, n \ (rekurens)$

yang dalam hal ini,

$f_k(y)$ merupakan nilai optimum berupa keuntungan pada tahap k

$f_0(y) = 0$ adalah nilai persoalan *knapsack* kosong

$f_k(y) = -\infty$ adalah nilai persoalan *knapsack* dengan kapasitas negatif.

Solusi optimum dari persoalan 1/0 *knapsack* adalah $f_n(M)$

III. PEMBAHASAN

A. *Gambaran Umum Program*

Secara umum, program yang akan dibuat merupakan program yang menerapkan teknik baru pengurutan prioritas *product backlog* dengan pertimbangan faktor-faktor sebagai metrik yang mempengaruhi prioritas *product backlog*. Prioritas *product backlog* yang dikembangkan berfokus pada pengurutan prioritas untuk dimasukkan ke *sprint* tertentu. Dengan begitu, keluaran sistem akan berupa *sprint backlog yang dibuat* berdasarkan urutan prioritas dari *product backlog.*

Setiap fitur memiliki nilai keuntungan yang diperoleh dari penjumlahan nilai-nilai metrik yang telah dikaji. Teknik ini mengadopsi beberapa metrik yang digunakan pada salah satu teknik penentuan prioritas *product backlog* yakni *Weighted Shortest Job First*. Fitur-fitur tersebut juga memiliki bobot berupa kompleksitas pengembangan fitur. Sementara di sisi lain, setiap *sprint* memiliki kapasitas maksimal bobot kompleksitas fitur yang akan dikembangkan. Untuk itu, tujuan dibuatnya program adalah untuk memaksimalkan keuntungan yang diperoleh dari pengembangan fitur-fitur dengan kapasitas kompleksitas tertentu pada setiap *sprint*.

Selain keluaran berupa *sprint backlog* yang terurut berdasarkan prioritas, sistem juga akan mengelompokkan fitur yang terdapat pada *sprint backlog* dengan kriteria sebagai berikut:

- 60% = *must have*
- 20% = *should have*
- 20% = *could have*

Sementara, sisa fitur atau fitur-fitur yang tidak dapat termuat pada *sprint* kali ini akan disimpan untuk diimplementasikan pada *sprint* berikutnya. Secara umum, prosedur sistem mengikuti rancangan sebagai berikut:

Gambar 3. Rancangan prosedur sistem
Sumber: dokumentasi pribadi

Metrik yang digunakan untuk penentuan keuntungan setiap fitur pada teknik ini adalah sebagai berikut:

1. *Effected number of users*
   Metrik ini mempertimbangkan jumlah pengguna yang terdampak fitur yang dikembangkan.
2. *Business value*
   Metrik ini mempertimbangkan nilai suatu perusahaan atas produk yang diluncurkan. Pertanyaan yang dapat diajukan untuk menilai metrik ini di antaranya:
   - Apa nilai relatif bagi pelanggan atau bisnis?
   - Apakah pengguna kami lebih suka ini daripada itu?
   - Apa dampak fitur ini pada pendapatan bisnis kita?
3. *Time critically*
   Metrik ini mempertimbangkan kemungkinan penurunan nilai bisnis seiring berjalannya waktu dan tenggat waktu dari pelanggan. Pertanyaan yang dapat diajukan untuk menilai metrik ini di antaranya:
   - Bagaimana nilai pengguna/bisnis menurun seiring waktu?
   - Apakah ada tenggat waktu yang tetap?
   - Apakah pelanggan akan menunggu kita atau pindah ke solusi lain?
   - Apa efek saat ini pada kepuasan pelanggan?
4. *Risk reduction & opportunity enablement relative estimate*
   Metrik ini mempertimbangkan pengurangan risiko dan pemberdayaan peluang dari sebuah fitur untuk produk yang sedang dikembangkan. Pertanyaan yang dapat diajukan untuk menilai metrik ini di antaranya:
   - Apa ada manfaat lain bagi bisnis kita?
   - Apakah fitur ini dapat mengurangi risiko ini atau pengiriman di masa depan?
   - Akankah fitur ini memungkinkan peluang bisnis baru?
5. *Customer satisfaction*
   Terdapat tiga klasifikasi yang digunakan untuk menentukan nilai dari metrik ini, yaitu fitur yang

berupa kebutuhan dasar (*basic need)*, fitur untuk meningkatkan performa (*performance area),* dan fitur yang termasuk dalam *delightful functionality.*

Nilai setiap metrik dapat berupa angka-angka pada deret Fibonacci yakni 1, 2, 3, 5, 8, 13, dan 21. Nilai pada deret Fibonacci merupakan nilai yang umum digunakan untuk menilai metrik dalam metode Scrum karena nilainya yang terdiri atas bilangan bulat, eksponensial, dan non-linear. Selain itu, Fibonacci mendorong *scrum team* untuk memilih "more or less" sehingga dapat digunakan sebagai pendekatan yang realistis. Sementara, khusus untuk metrik *customer satisfaction*¸ nilai hanya dapat berupa 3 (*delightful functionality)*, 8 (*performance area)*, dan 13 (*basic need).*

Misalkan sebuah *scrum team* ingin mengembangkan aplikasi *e-commerce* A. *Product owner* akan menentukan fitur-fitur yang akan dikembangkan selama proyek berlangsung dan *scrum team* beserta *product owner* menentukan nilai metrik-metrik setiap fitur. Fitur beserta nilai metrik yang dihasilkan adalah sebagai berikut:

Fitur	Metrik					*Profit*	*Weight (Complexity)*
	1	2	*3*	*4*	*5*		
Login	13	8	13	2	13	49	3
Register	8	8	13	2	13	44	3
Halaman daftar produk per toko	8	8	13	2	13	44	8
Halaman tampilan toko	5	8	5	2	13	33	3
Pembayaran	13	8	13	8	13	55	13
Mekanisme pencarian produk	13	8	13	5	13	52	13
Halaman pengaduan	2	2	1	2	3	10	2
Halaman bantuan	2	2	1	2	3	10	2
Diskon produk	13	8	5	13	3	42	5
Gratis ongkir	8	8	5	13	3	37	21
Rekomendasi produk	8	13	3	13	3	40	21
Keranjang belanja	13	8	8	2	13	44	8
Tambah dan hapus barang dari toko	8	8	5	2	13	36	5
Detail produk	13	3	3	3	13	35	5
Wishlist	3	3	2	3	3	14	8
Review pelanggan	8	13	5	5	13	44	13
Prediksi ongkos kirim	13	8	8	5	13	47	8
Halaman promo	8	13	3	13	3	40	8
Informasi produk baru	5	13	3	8	3	32	13
Fitur keamanan	20	8	20	20	13	81	8
Halaman FAQ	3	2	1	1	3	10	3
Artikel terintegrasi	3	3	1	3	3	13	5
User-friendly	20	8	13	5	5	51	8
High-resolution foto dan video	13	8	5	3	5	34	3
Find-in-store	5	3	1	2	3	14	5

Tabel 1. Data *profit* dan *weight* fitur-fitur pada pengembangan aplikasi e-commerce A
Sumber: dokumentasi pribadi

Keterangan:
1 = *Effected number of users*
2 = Business value
3 = *Time critically*
4 = *Risk reduction & opportunity enablement relative estimate*
5= *Customer satisfaction*

B. Implementasi Program

Pengurutan prioritas *product backlog* dimodelkan sebagai persoalan *knapsack* dengan dasar sebagai berikut:

- Keuntungan setiap fitur diperoleh dari penjumlahan metrik-metrik yang terdiri atas e*ffected number of users, business value, time*
- *critically, risk reduction & opportunity enablement relative estimate, dan customer satisfaction*

- Bobot setiap fitur diperoleh dari kompleksitas pengembangan fitur tersebut dari segi waktu dan biaya
- Fungsi objektifnya adalah memaksimalkan Kendala untuk maksimalisasi keuntungan adalah jumlah kompleksitas maksimal pada suatu *sprint*

Karena salah satu fokus metode Scrum adalah untuk merespon perubahan secara cepat, nilai metrik tersebut dapat diperbaharui untuk setiap iterasi. Namun, contoh yang diberikan mengasumsikan bahwa tidak ada perubahan pada nilai metrik sepanjang proyek berlangsung.

Misalkan pada *sprint* pertama, *product owner* menentukan poin kompleksitas maksimal dari *sprint* ini adalah 45. Berikut merupakan keluaran yang dihasilkan untuk *sprint* pertama:

```
SPRINT BACKLOG 1

Maximum sprint point complexity: 45
Total features complexity: 44
Total features profit: 228

Must be included in this sprint:
    Pembayaran (profit: 55, complexity: 13)
    Keranjang belanja (profit: 44, complexity: 8)
    Halaman daftar produk per toko (profit: 44, complexity: 8)
    Halaman promo (profit: 40, complexity: 8)
    Detail produk (profit: 35, complexity: 5)

Should be included in this sprint:
    Halaman bantuan (profit: 10, complexity: 2)

Could be included in this sprint:
    None
```

Gambar 4. Keluaran program untuk *sprint* pertama dengan poin kompleksitas maksimal = 45
Sumber: dokumentasi pribadi

Program menampilkan *sprint backlog* yang berisi fitur-fitur yang diimplementasikan pada *sprint* pertama. Total kompleksitas dari fitur-fitur tersebut adalah 44, kurang dari jumlah maksimal poin kompleksitas *sprint* ini yaitu 45. Berikut merupakan hasil *screenshot* program untuk *sprint* kedua, ketiga, keempat, hingga kelima dengan poin maksimal kompleksitas setiap sprint yang bervariasi:

```
SPRINT BACKLOG 2

Maximum sprint point complexity: 48
Total features complexity: 47
Total features profit: 238

Must be included in this sprint:
    Pembayaran (profit: 55, complexity: 13)
    Keranjang belanja (profit: 44, complexity: 8)
    Halaman daftar produk per toko (profit: 44, complexity: 8)
    Halaman promo (profit: 40, complexity: 8)
    Detail produk (profit: 35, complexity: 5)

Should be included in this sprint:
    Halaman FAQ (profit: 10, complexity: 3)
    Halaman bantuan (profit: 10, complexity: 2)

Could be included in this sprint:
    None
```

Gambar 5. Keluaran program untuk *sprint* kedua dengan poin kompleksitas maksimal = 48
Sumber: dokumentasi pribadi

```
SPRINT BACKLOG 3

Maximum sprint point complexity: 43
Total features complexity: 39
Total features profit: 128

Must be included in this sprint:
    Mekanisme pencarian produk (profit: 52, complexity: 13)
    Review pelanggan (profit: 44, complexity: 13)
    Informasi produk baru (profit: 32, complexity: 13)

Should be included in this sprint:
    None

Could be included in this sprint:
    None
```

Gambar 6. Keluaran program untuk *sprint* ketiga dengan poin kompleksitas maksimal = 43
Sumber: dokumentasi pribadi

```
SPRINT BACKLOG 4

Maximum sprint point complexity: 50
Total features complexity: 50
Total features profit: 91

Must be included in this sprint:
    Rekomendasi produk (profit: 40, complexity: 21)
    Gratis ongkir (profit: 37, complexity: 21)
    Wishlist (profit: 14, complexity: 8)

Should be included in this sprint:
    None

Could be included in this sprint:
    None
```

Gambar 7. Keluaran program untuk *sprint* keempat dengan poin kompleksitas maksimal = 50
Sumber: dokumentasi pribadi

```
SPRINT BACKLOG 5

Maximum sprint point complexity: 50
Total features complexity: 10
Total features profit: 27

Must be included in this sprint:
    Find-in-store (profit: 14, complexity: 5)
    Artikel terintegrasi (profit: 13, complexity: 5)

Should be included in this sprint:
    None

Could be included in this sprint:
    None
```

Gambar 8. Keluaran program untuk *sprint* kelima dengan poin kompleksitas maksimal = 50
Sumber: dokumentasi pribadi

Sprint kelima merupakan iterasi terakhir karena fitur-fitur aplikasi di *product backlog* sudah selesai dikembangkan.

C. *Potensi Penerapan Program*

Program ini memiliki potensi untuk dapat diterapkan dan diintegrasikan dengan sistem/aplikasi *project management* sehingga dapat memberikan solusi *end-to-end* kepada tim yang akan menerapkan metode Scrum dalam mengembangkan perangkat lunak. Nantinya, aplikasi tersebut dapat digunakan sebagai alat pembantu utama yang berisi segala artefak Scrum sekaligus dapat mengelola artefak-artefak tersebut secara otomatis. Selain itu, dapat dilakukan riset lebih jauh mengenai metrik-metrik yang dapat digunakan sebagai faktor penentu nilai keuntungan setiap fitur pada *product backlog*.

IV. SIMPULAN

Persoalan penentuan prioritas *product backlog* dapat dimodelkan sebagai persoalan 1/0 *knapsack* yang dapat diselesaikan dengan pendekatan program dinamis. Dengan metrik berupa *effected number of users, business value, time critically, risk reduction & opportunity enablement relative estimate, customer satisfaction* sebagai nilai *profit* fitur dan *complexity* sebagai nilai bobot fitur, teknik yang digunakan dapat menghasilkan daftar fitur prioritas dari *product backlog* yang dimuat dalam *sprint backlog*.

Ke depannya, teknik ini dapat diintegrasikan dengan sistem *project management* sehingga dapat memberikan solusi menyeluruh untuk memudahkan pengembangan perangkat lunak menggunakan metode Scrum.

VIDEO LINK AT YOUTUBE

Penjelasan makalah ini dalam bentuk video dapat diakses pada https://youtu.be/pZk5ClEAHIM

UCAPAN TERIMA KASIH

Puji dan syukur penulis ucapkan kepada Allah SWT. atas segala berkah dan rahmat-Nya yang tidak pernah berhenti mengiringi hidup penulis hingga penulis dapat menyelesaikan pembuatan makalah ini dengan baik. Ucapan terima kasih penulis sampaikan kepada orang tua dan keluarga yang telah mendukung penulis dari segi moral maupun material.

Penulis mengucapkan terima kasih sebesar-besarnya kepada seluruh pihak yang telah menyediakan sarana dan prasarana serta memberikan dukungan dalam penyelesaian makalah ini. Terima kasih kepada Bapak Rinaldi Munir selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas K03 dan kepada seluruh tim pengajar mata kuliah IF2211yang telah memberikan bantuan dan ilmunya kepada penulis dan teman-teman penulis hingga kami dapat menyelesaikan pembuatan masalah dengan baik dan tepat waktu.

Terima kasih kepada rekan-rekan penulis, terutama segenap mahasiswa program studi Teknik Informastika Institut Teknologi Bandung yang telah memberikan motivasi, dukungan, dan saran kepada penulis selama pengerjaan makalah.

REFERENSI

[1] Munir, Rinaldi. (n.d.). *Program Dinamis (Bagian 1)* . Diakses pada 20 Mei 2022 dari *https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf*

[2] Irmeliyana, Bangun, P. J., Pratamawati, D., & Septiani, W. (2017). PENERAPAN ALGORITMA DYNAMIC PROGRAMMING PADA PERMASALAHAN KNAPSACK 0-1. 2.

[3] Mega Arista, Winda. (2013). *PENERAPAN ALGORITMA GREEDY DAN DYNAMIC PROGRAMMING PADA PERMASALAHAN INTEGER KNAPSACK [Undergraduate thesis, Universitas Jember]*. Unej Repository. Diakses pada 21 Mei 2022 dari https://repository.unej.ac.id/bitstream/handle/123456789/6944/Winda%20Mega%20Arista%20-%20071810101019.pdf?sequence=1

[4] Anonymous. (n.d.). *How does weighted shortest job first (WSJF) work?* Diakses pada 21 Mei 2022 dari https://agility.im/frequent-agile-question/how-does-weighted-shortest-job-first-wsjf-work/

[5] Velasquez, Robert. (2017). *5 Reasons Using The Fibonacci Sequence Makes You Better At Agile Development.* . Diakses pada 21 Mei 2022 dari https://elearningindustry.com/using-the-fibonacci-sequence-makes-better-agile-development-5-reasons

[6] Tim Pengajar IF2250. (2020). *Agile Process dan Scrum.*

[7] Anonymous. (n.d.). *The Agile Journey: A Scrum Overview.* . Diakses pada 23 Mei 2022 dari https://www.pm-partners.com.au/the-agile-journey-a-scrum-overview/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2022

Addin Nabilal Huda
13520045