Pemanfaatan Graph Data Flow Diagram pada Perancangan Rekayasa Perangkat Lunak

Fiqie Ulya Sidiastahta
Program Sarjana Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
fiqieulya@students.itb.ac.id

***Abstrak*—Seorang Software Engineer dituntut untuk dapat menciptakan sebuah sistem. Pada kenyataanya Software Enggineer juga dituntut untuk dapat melayani keinginan customer. Bukan hanya sebatas sebuah sistem dapat dipakai. Namun Software Engineer juga dituntut untuk dapat menciptakan sebuah sistem yang efisien dari segi waktu, modal, dan fungsi yang ada dalam sistem. Skill saja tidak cukup, perlu adanya management kerja yang baik. Salah satunya dengan memahami konsep rekayasa perangkat lunak. Salah satu elemen penting dari Rekayasa perangkat lunak yakni tahap analisis model. Tahap analisis menuntut Software engineer untuk berfikir sistematis. Salah satunya tahap analisis fungsional menggunakan DFD. Pada pemanfaatannya ternyata teori grap digunakan dalam Data Flow Diagram. Pemanfaatan Graf ini ternyata sangat membantu seorang software engineer dalam menyusun proses aliran data dari suatu sistem. Graf ini yang nantinya akan memodelkan permasalahan yang akan dihadapi dalam sebuah sistem, khususnya mengenai pengaliran data. Hal ini yang nantinya memudahkan sofware engineer untuk membentuk fungsi yang efisien.**

***Keywords*—matematika diskrit, graph, Data flow diagram, software engineer, context diagram.**

I. PENDAHULUAN

Pada dasarnya dalam suatu perancangan suatu sistem, seorang software engineer membutuhkan proses analisis. Proses analisis ini membantu seorang software engineer untuk memodelkan suatu masalah agar lebih mudah dipahami dan dicarikan solusi. Analisis ini mencakup permasalahan untuk siapa sistem ini digunakan, apa yang sistem lakukan, kapan dan dimana sistem ini digunakan.

Pada tahap analisisi, seorang *software engineer* akan memodelkan permasalahan dalam 3 model, yakni permodelan data (Entity Relationship Diagram), Permodelan Behaviour (State Transition Diagram), dan permodelan fungsional (Data Flow Diagram). Salah satu model yang akan dibahas yakni permodelan fungsional yang akan di representasikan dengan model Data Flow Diagram (DFD).

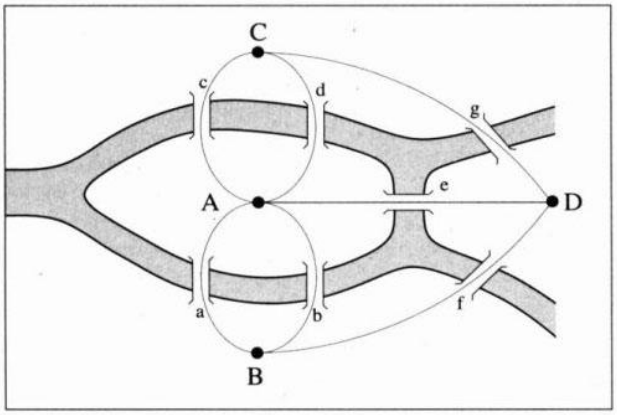
permodelan fungsional diharapkan dapat memudahkan, seorang software engineer dalam memahami hubungan antara entitas eksternal, proses transformasi data, dan penyimpanan data yang dibutuhkan dalam sistem. Dibutuhkan sebuah representasi sederhana yang dapat mencakup seluruh komponen fungsional yang dibutuhkan, oleh karena itu permodelan fungsional ini di representasikan dengn sebuah graf berarah. Graf berarah dipilih karena dapat mengambarkan kepada pembacanya, alur yang sistematis serta dalam permodelannya lebih sederhana.

II. DASAR TEORI

1. GRAF

Graf merupakan struktur diskrit yang terdiri dari simpul, dan busur yang menghubungkan sebuh simpul terhadap simpul lainnya. Graf secara matematis di representasikan dengan G = (V,E), denga V himpunan tidak kosong dari vertices (simpul) dan E himpunan busur yang menghubungkan 2 simpul.

Graf pada awalnya diperkenalkan oleh seorang matematikawan asala swiss bernama Leonhard Euler pada tahun 1736, sebagai solusi dari jembatan Koningsberg.



Gambar 1 Koningsberg Bridges
(http://physics.weber.edu/carroll/honors/konigsberg.htm)

Graf memiliki berbagai macam jenis, graf pada dasarnya dibedakan berdasarkan ada tidaknya busur ganda (adanya lebih dari 1 busur yang menghubungkan 2 simpul) dan ada tidaknya busur gelang (sebuah sisi yang terhubung pada 1 simpul) jenis ini dibagi menjadi 2 graf

sederhana dan graf tidak sederhana.

1.1. Graf Sederhana
Graf yang tidak memiliki busur ganda maupun gelang. Hanya terdiri dari simpul dan sebuah busur untuk setiap pasangan simpul.

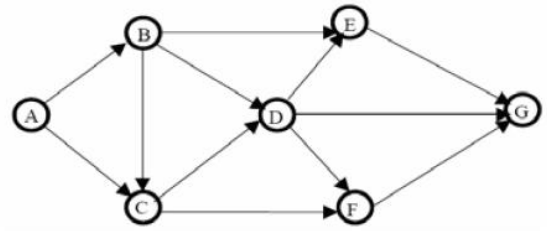
1.2. Graf Tak Sederhana
Graf yang bisa memiliki busur ganda maupun memiliki gelang.

Graf juga dibedakan berdasarkan orientasi grafnya, menjadi graf tidak berarah, dan graf berarah.

1.3. Graf Tidak Berarah
Graf yang tidak memiliki orientasi arah pada setiap busur.

1.4. Graf Berarah
Graf yang memiliki orientasi arah pada setiap busurnya.

Pada pembahasan kali ini, akan digunakan jenis representasi graf berarah. Penggunaan graf berarah ini didasarkan dengan permodelan DFD yang menganalisa sebuah permodelan fungsional secara sistematis, dan memperhatikan urutan suatu proses.



Gambar 2 Graf Sederhana Berarah (http://elsimelyna.blog.upi.edu/)

Beberapa terminologi tentang graf:

1) Ketetanggaan (Adjacent)
Dua buah simpul dikatakan bertetangga bila dihubungkan dengan sebuah busur. Pada Gambar 2 A bertetangga dengan B dan C.
2) Bersisian (Incidency)
Untuk sembarang busur e = (Va,Vb), e bersisian dengan simpul Va, e bersisian dengan Vb.
3) Derajat (Degree)
Jumlah busur yang bersisian dengan suatu simpul. Notasi d(A), pada Gambar 2 d(A) = 2, karena A bersisian dengan 2 buah busur.
4) Lintasan(Path)
Panjang dari suatu simpul awal ke simpul tujuan.
5) Siklus (Cycle)
Lintasan yang berawal dan berakhir pada 1 simpul yang sama.
6) Terhubung (Connected)
2 simpul dikatakan terhubung bila dihubungkan dengan sebuah busur. Sedangkan Graf dikatakan terhubung bila simpul-simpul pada suatu graf terhubung dengan 1 atau lebih simpul. Bila ada simpul yang tidak terhubung dengan simpul lain, maka graf tersebut bukan graf terhubung.
7) Graf Berbobot (wieghted graph)
Graf yang sisinya memiliki nilai, atau panjang sisi.

2. DATA FLOW DIAGRAM (DFD)

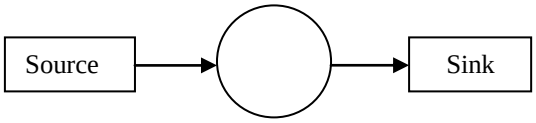
Diagram aliran data (DFD) adalah representasi grafis yang menggambarkan aliran informasi yang bergerak dari suatu proses input ke proses output. DFD juga dikenal sebagai data flow graph atau grafik gelembung, karena proses dalam DFD digambarkan seperti sebuah gelembung.

DFD ini juga dapat mewakili setiap tingkat abstraksi dari suatu sistem perangkat lunak, bahkan DFD juga dapat dipartisi lagi sehingga dapat merepresentasikan ke tingkat yang lebih detail secara fungsional, dan memenuhi prinsip analisis operasional (yaitu , menciptakan model fungsional) yang telah dibahas sebelumnya. Pada kasus ini DFD tidak dapat menampilkan detail prosedural seperti kondisi dan loop.

2.1. Komponen DFD

DFD memiliki 4 komponen utama pada notasinya, yakni:

1. Data Flow, menggambarkan data apa yang sedang mengalir.
2. Proses, menangani data yang mengalir, dapat berupa sebuah aktifitas, tugas atau fungsi – fungsi.
3. Data Storage, menggambarkan tempat penyimpanan data.
4. External/Outside entities/Terminator, sumber faktor dari luar yang ikut mempengaruhi suatu sistem. Terminator nantinya dibedakan menjadi 2, yaitu Source (sumber data) yang di letakan pada posisi kiri, dan Sink(penerima data) diletakan pada posisi kanan



Gambar 3 Representasi Jenis Terminator (penulis)

2.2. Context Level DFD

Context Level DFD atau context diagram merupakan bagian teratas dari DFD, sekaligus gambaran terabstrak dari sistem (gambaran umum).

Ciri – ciri context diagram :

1. Terminator hanya akan muncul pada level ini.
2. Proses (bubble) hanya terdapat 1 buah
3. Data stores tidak pernah di tampilkan pada level ini.
4. Biasa disebut sebagai level -0

2.3. Level

Beda dengan Level-0 sebelumnya, pada level selanjutnya kita tidak menjumpai Terminator lagi, dan jumlah proses(bubble) bisa lebih dari 1 untuk masing- masing level. Jumlah proses(bubble) dalam suatu level dipisahkan berdasarkan input utama yang masuk ke dalam proses, waktu, *equipment*, dan fungsi utama.

Jika dalam sebuah sistem jumlah proses(*bubble*) terlalu besar untuk di tampilkan dalam graf tunggal atau dalam 1 level, maka pecahlah menjadi sub-sistem dan sub-sub-sistem lanjutan(bila diperlukan) atau disebut *top-down partitioning*. Proses inilah yang disebut *Leveling*. Setiap partisi yang lebih bawah (*child*) harus lebih detail dibanding proses (*parent*) pada level sebelumnya. Sehingga memungkinkan sebuah proses(*parent*) dapat memiliki child dalam bentuk graf. Proses Leveling ini akan berakhir saat suatu proses (bubble) sudah tidak dapat di partisi lagi, atau sudah dapat di bentuk algoritmanya.

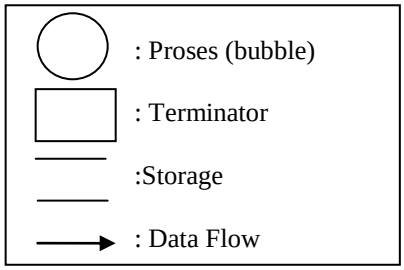
2.4. Balancing

Proses terakhir dari pembentukan DFD yakni Balancing. DFD dikatakan balance jika jumlah inputan dan output pada child sama seperti jumlah input dan output pada parent yg dipartisi. Pada proses ini seorang software enggineer akan memperhitungkan inputan dan output untuk setiap proses, hal ini untuk mengurangi data ganda dan data yang tidak terpakai.

III. IMPLEMENTASI GRAF PADA DFD

A. Notasi

Pada DFD, proses, terminator , dan store di representasikan dengan simpul pada representasi graf. Proses digambarkan dengan lingkaran(bubble), terminator digambarkan dengan bentuk kotak , dan storage dalam bentuk 2 garis. Sedangkan data flow direpresentasikan dengan busur berarah pada sebuah graf.

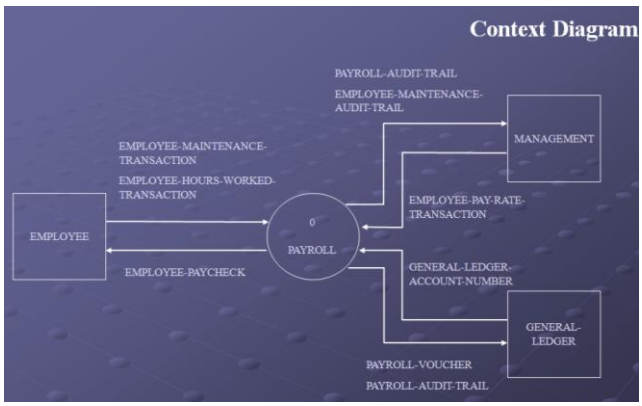


Gambar 4 Representasi Komponen DFD (penulis)

1) *Proses (bubble)*
   Setiap bubble harus memiliki no unik berdasarkan level dan sub-level(bila bubble tersebut merupakan sub-sistem). Bubble juga harus memiliki nama proses dengan ketentuan merupakan kata kerja aktif diikuti objek *clause*-nya dan tidak boleh memiliki makna ganda. Contoh : *edit-customer-payment*.
2) *Data Storage*
   Ukuran garis berkisar 1- ½ inci. Untuk data flow yang menuju data storage, artinya data di simpan/ditulis, sedangkan untuk data flow yang keluar dari data storage berarti data dibaca atau diambil dari penyimpanan.
3) *Data Flow*
   Penghubung antar bubble, terminator, & data storage. Pada akhirnya data flow sama fungsinya dengan busur pada graf, untuk menghubungkan 2 simpul. Aturan penamaan haruslah unik dan deskriptif, data apa yang mengalir.

B. Context Level DFD

Pada context Level 0 ini hanya terdiri dari 1 bubble dengan no bubble 0. Terminator hanya dituliskan pada level ini pula. Terminator yang di tuliskan hanya terminator yang mempengaruhi sistem, misal dalam sebuah restoran dibutuhkan pemasok daging, tapi pada kenyataannya hanya 1 distributor pemasok daging. Pada kasus ini Pemasok daging bukan bagian dari terminator yang mempengaruhi sistem. Beda halnya bila sebuah restoran memiliki banyak pemasok daging, maka pemasok daging menjadi terminator yang berpengaruh, karena sistem akan mencatat pendistribusian daging, dimungkinkan antar 1 distributor dengan distributor lain memiliki perbedaan kontrak dengan restoran. Pada penamaan bubble level 0 ini haruslah nama proses yang bisa mendiskripsikan sistem yang dibuat secara keseluruhan.



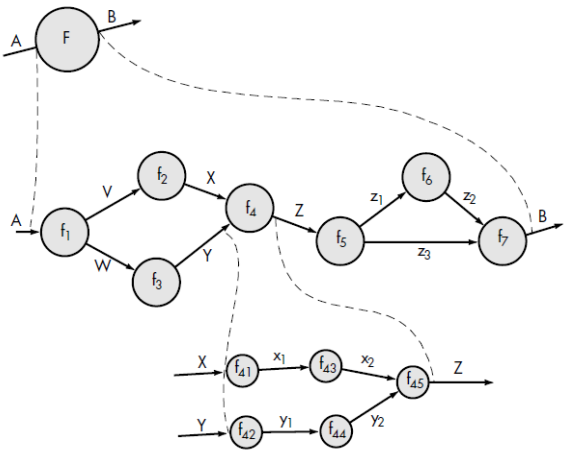
***Gambar 5** Contoh Context Diagram*
(IF ITB DS/Revisi :2008)

Pada DFD diatas terlihat bentuk implementasi dari graf berarah. Bisa kita lihat contoh sederhana simpul (*Employee*) sebagai terminator memberikan input berupa data "*EMPLOYEE-HOUR-WORKED-TRANSACTION*" yang nantinya akan di proses pada bubble "*PAYROLL*", dan sebaliknya "*PAYROLL*" memberikan output berupa "*EMPLOYE-PAYCHEK*"

terhadap simpul(*Employee*). Hal yang sama berlaku untuk simpul Terminator (*MANAGEMENT & GENERAL LEDGER*). Dengan graf berarah ini memudahkan seorang Software Engineer untuk memodelkan sebuah permasalahn

C. Leveling

Pada Level sebelumnya kita melihat simpul berupa terminator dan proses yang memanfaatkan graf berarah ganda. Pada level berikutnya kita akan memanfaatan graf berarah sederhana, untuk tiap simpul (bubble) hanya memiliki 1 busur terhadap simpul (bubble) lainnya. Hal ini dimungkinkan agar tidak terjadi looping pada DFD, mengingat DFD menggambarkan permodelan fungsional.



Gambar 6 Contoh DFD Level 0 -2
(softwareengineering Pressman- Fifth Edition)

Pada pemanfaatan diatas bisa dilihat bahwa lintasan f1 hingga f7 merupakan hasil dari partisi F(context diagram). f1 mengirim data v untuk di proses f2, kemudian f2 menghasilkan X yang nantinya akan di proses dengan Y yang merupakan hasil proses dari f3. Pada f4 ternyata proses masih panjang, sehingga kita dapat mempartisinya lagi menjadi level 2. Pada representasi ini terlihat penggunaan graf berarah sederhana tanpa gelang, dan busur ganda.

Selain itu penggunaan graf berarah membantu DFD dalam balancing. Untuk level 0 ada terminator dari luar yang memberi inputan A, kemudian memunculkan output B. Sedangkan pada level 1, f1 menerima A, dan proses level 1 yang berakhir di f7 menghasilkan B. Ini membuktikan bahwa partisi dari level 0 ke level 1 balance. Bila suatu partisi input/output level parent tidak sama dengan input output pada simpul awal dan simpul akhir dari level anaknya, maka fungsi tersebut tidak valid. Pengaplikasian Graf pada DFD juga dapat mengurangi data yang tercopy ganda, dan mengurangi resiko data yang hilang atau tidak terpakai.

IV. MEMANFAATKAN GRAF DFD UNTUK MENGANALISIS FUNGSI SISTEM PEGADAIAN

Dalam sebuah kasus, diharapkan seorang software engineer dapat menganalisis data dan fungsi apa saja yang dibutuhkan dalam membentuk sebuah sistem penggadaian barang secara konvensional. Gadai konvensional yakni kredit dengan sistem gadai yang diberikan kepada semua golongan nasabah, baik untuk kebutuhan konsumtif maupun kebutuhan produktif. Untuk mendapatkan kredit nasabah hanya perlu membawa agunan berupa perhiasan emas dan barang berharga lainnya.

Software engineer harus menangani kasus proses penggadaian barang, perpanjangan gadai, pelelangan, pembuatan laporan untuk manager, dan penebusan barang.

Prosedur penggadaian barang meliputi proses:

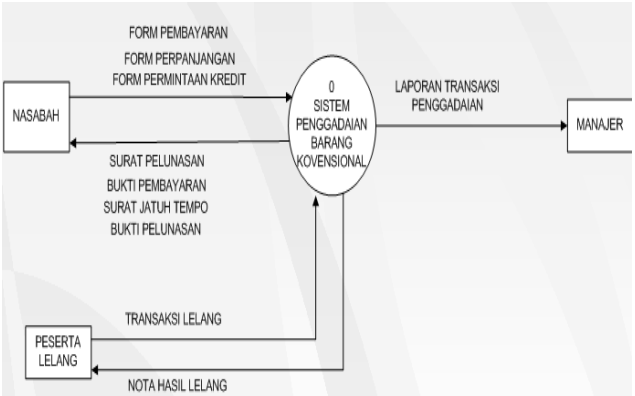
- Nasabah membawa agunan, serta kelengkapan surat-suratnya
- Barang berharga milik nasabah kemudian ditaksir oleh Penaksir Harga
- Setelah nasabah setuju, nasabah meregistrasi barang gadaiannya sesuai dengan taksiran harga barang, ditambah dengan biaya sewa berdasarkan lamanya pinjaman
- Setelah proses registrari selesai, Nasabah mendapatkan Slip transaksi Pegadaian(SBK) yang tercantum harga pinjaman serta besarnya uang tebusan barang gadai.
- Slip tersebut nantinya digunakan untuk pengambilan Uang tunai.

4.1. Menentukan Terminator

Dari kasus diatas diketahui ada 3 terminator yang berpengaruh yakni Nasabah sebagai source sekaligus sink. Manager Sebagai sink penerima laporan, dan pelelang sebagai source dan sink bila terjadi proses pelelangan.

4.2. Membentuk Context Diagram

Berdasarkan Terminator yang sudah ada maka membentuk data flow yang nantinya di butuhkan sistem. Mambentuk 1 bubble level 0 yang deskriptif, yakni "Sistem Penggadaian Barang Konvensional".



Gambar 7 Context Diagram Pegadaian (Penulis)

Bila Proses balancing Context diagram dan Level sudah seimbang, maka sofware engineer sudah dapat memperkirakan data apa saja yang dibutuhkan sistem. Data flow bukan berupa data tunggal(hanya 1 atribut), atau merupakan data yang memiliki beberapa atribut.

Inputan yang dibutuhkan:

- Form pembayaran (Nasabah)
- Form perpanjangan (Nasabah)
- Form permintaan kredit(Nasabah)
- Transaksi Lelang (Peserta lelang)

Output yang dibutuhkan:

- Surat pelunasan (Nasabah)
- Bukti pembayaran (Nasabah)
- Surat jatuh tempo (Nasabah)
- Bukti pelunasan (Nasabah)
- Nota hasil lelang (Peserta lelang)
- Laporan transaksi penggadaian (Manager)

4.3. Leveling

Bubble level 0 nantinya akan di partisi menjadi beberapa bubble lain yang merepresentasikan proses proses yang lebih kompleks di dalam suatu sistem. Pada tingkat ini kita menggunakan representasi graf sederhana, dimana tidak ada data flow ganda maupun busur gelang. Bubble diberi label sesuai urutan berjalannya sistem, hal ini untuk memudahkan pembaca dalam menangkap representasi yang dimaksud. Untuk data flow yang menuju keluar dan datang dari luar merupakan data flow yang terbawa saat proses partisi. Berdasarkan permasalahan yang ada penulis membagi sistem menjadi 5 fungsi dasar, dan sebuah storage (Barang Gadai).

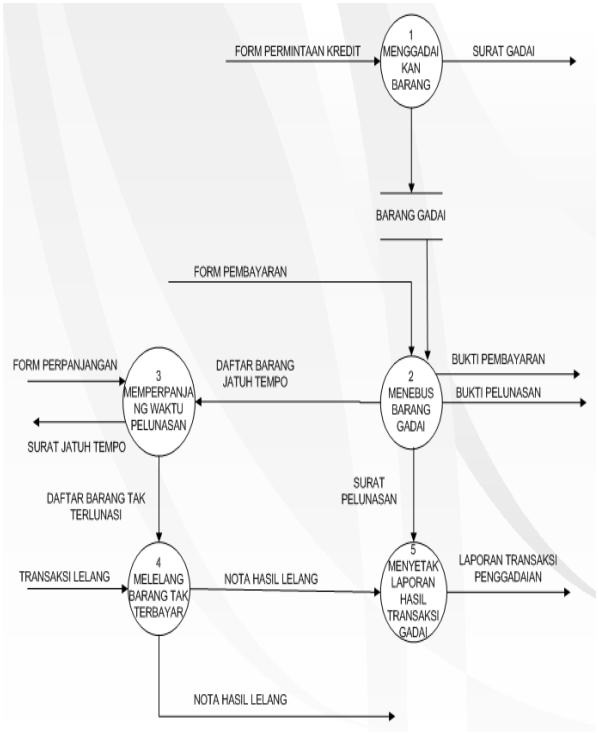
Bubble 1 menunjukkan proses penggadaian barang oleh nasabah. Data yang nantinya di proses pada bubble 1 akan menghasilkan surat gadai sebagai bukti penggadaian. Selain itu pada bubble 1, data dari nasabah akan di masukkan kedalam data base pegadaian, untuk diolah pada proses lainnya.

Bubble 2 menunjukkan proses penebusan oleh nasabah, pada proses ini selain menerima pembayaran lunas, nasabah juga dapat mengangsur. Bila pada proses ini nasabah melunasinya maka nasabah menerima bukti pembayaran.

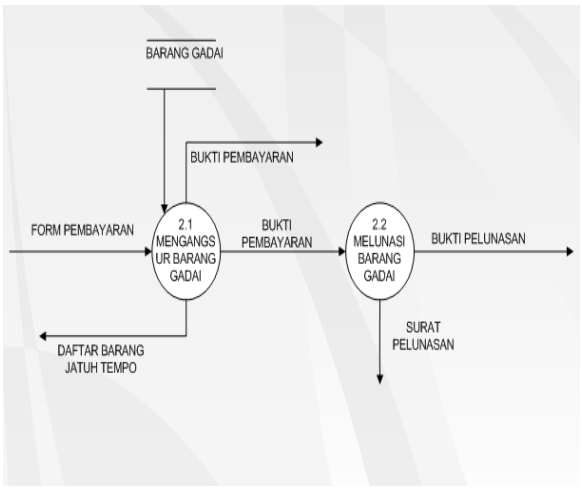
Bubble 3 menunjukkan proses perpanjangan penggadaian. Pada setiap siklusnya akan dicek data barang barang yang jatuh tempo, data ini keluar dari bubble 2 yang nantinya akan diproses dalam bubble 3.

Bubble 4 menunjukkan proses pelelangan. Barang – barang yang tidak diperpanjang dari proses perpanjangan nantinya akan diproses pada bubble ini.bubble ini menerima dan mengeluarkan data untuk entitas peserta lelang.

Bubble 5 merupakan proses menyetak hasil laporan transaksi penggadaian yang nantinya diperuntungkan untuk manager. Terlihat pada bubble ini ada 2 data masuk dari 2 bubble lain, yakni bubble 2 dan 4, dimaksudkan untuk setiap terjadi transaksi apapun maka data tersebut harus diproses dan dicetak sebagai bukti.



Gambar 8 graf level 1 Pegadaian (Penulis)



Gambar 9 graf partisi bubble 2 (level 2)

Pada gambar 9 penulis mempartisi bubble 2 menjadi 2 proses lagi. Hal ini dilakukan karena bubble 2 belum bisa dibuat algoritmanya sehingga di perluka partisi lagi.

Dari graf DFD diatas seorang software engineer akan lebih mudah menyusun sebuah sistem yang efisien dan minimum perbaikan, hal ini disebabkan karena kemungkinan kesalahan dalam sistem hanya sebatas ide, sehingga merubah dfd sebelum mengimplementasikan dalam sebuah program mengurangi proses perbaikan bila dibandingkan dengan langsung mengimplementasi tanpa perlu perancangan.

V. KESIMPULAN

Dengan teori graf dapat memudahkan seorang software engineer dalam membuat sebuah software, salah satunya dengan pemanfaatan graf dalam Data Flow Diagram. Dengan graf DFD seorang software engineer dapat lebih mudah dalam menganalisis kebutuhan fungsional dalam membuat sebuah sistem. Graf berarah juga membantu dalam menyusun sistem fungsional secara sistematis. Selain itu dengan graf berarah yang ada pada DFD membantu software engineer dalam mengalirkan data tanpa ada data yang terbuang atau terduplikasi. Pada Kasus Software Engineering yang besar dan kompleks penggunaan graf DFD ini akan sangat berguna, karena akan mengurangi resiko perubahan masal tiap fungsi bila terjadi miss data yang sebelumnya tidak diperkirakan

VI. UCAPAN TERIMA KASIH

Pertama-tama penulis ingin bersyukur kepada Tuhan Yang Maha Esa, karena hanya oleh karena rahmat-Nya penulis dapat menyelesaikan tulisan ini. Penulis juga berterima kasih kepada orang tua dan dosen yang memberikan tugas ini Dr. Ir. Rinaldi Munir atas bimbingan dan jasa beliau yang selama ini telah mengajar dan memberikan ilmu bagi penulis, sehingga penulis mampu membuat tulisan ini. Tak lupa juga penulis berterima kasih atas rekan-rekan yang senantiasa memberikan dukungan dan kontribusi ide bagi penulis.

REFERENSI

[1] K. H. Rosen, "Discrete Mathematics and its Applications" 7th ed. New York: McGraw-Hill, 2007, pp. 641 - 644

[2] R.S. Pressman "Software Engineering A Practitioner's Approach" 5th ed. New York: McGraw-Hill,2001, pp. 299-317

[3] Munir, Rinaldi, "Matematika Diskrit", Informatika, Bandung: 2010

[4] Pegadaian. "Gadai Konvensional" from: http://www.pegadaian.co.id/pegadaian-gadai.php dibaca: 10 Des 2014

[5] B W Carroll. "Konigsberg Bridges Problem" . Departement of Physics Weber State University. from: http://physics.weber.edu/carroll/honors/konigsberg.htm dibaca 9 Des 2014

[6] Melyna, Elsi, "Pemanfaatan Metode Heuristik Pada Pencarian Jalur Terpendek dengan Algoritma Genetika" Ilmu Komputer Universitas Pendidikan Indonesia. from: http://elsimelyna.blog.upi.edu/2013/02/27/tugas-1-mata-kuliah-seminar-resume-jurnal-ke-4/ dibaca 9 Desember 2014

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, 10 Desember 2014

Fiqie Ulya Sidiastahta /13514602