MODEL *REQUIREMENT TRACEABILITY* UNTUK METODE PENGEMBANGAN PERANGKAT LUNAK *FEATURE DRIVEN DEVELOPMENT* (FDD)

Fildzah Shabrina [(1)*], Widodo [(2)], Bambang Prasetya Adhi [(3)]
[1] Dosen Prodi Sistem Komputer, Fakultas Ilmu Komputer, Universitas Borobudur Jakarta
[2,3] Dosen Prodi Pendidikan Teknik Informatika dan Komputer, Teknik Elektro, FT – UNJ
email : [1] fildzah@borobudur.ac.id, [2] widodo@unj.ac.id, [3] bambangpadhi@unj.ac.id

Abstract

A recent study conducted by the Standish Group in 2012 shows that in the preparation of requirements becomes one of the critical success factors of a software. Requirements traceability is the method used to explore inter-linkages or inter-relations requirement, so that when an error occurs in one or more functions on a software, or changes in the requirements, then it can easily be identified requirements which are problematic. Feature Driven Development (FDD) is one of the Agile method that does not have a fixed rule in the formation of inter-search requirement models and inter-feature. By using grounded theory in research, such as open coding, axial coding, and selective coding, and sorting. The research began with the determination of the review and identification of the problem, collect relevant data, perform data analysis, and the final document the results. Based on the test results, the modeling traceability requirements in the FDD method can assist in tracing linkages, inter-relations requirements as well as inter-feature during software development. And can be used as documentation of a software development work.

Keywords : ***requirement, traceability, relation, Feature Driven Development, and grounded theory*.**

Penelitian terbaru yang dilakukan oleh Standish Group pada tahun 2012 menunjukkan bahwa dalam penyusunan requirements menjadi salah faktor penentu keberhasilan sebuah perangkat lunak. Requirement traceability merupakan metode yang digunakan untuk menelusuri keterkaitan atau interkoneksi antar-requirement, sehingga ketika terjadi kesalahan dalam satu atau lebih fungsi pada sebuah perangkat lunak, atau terjadi perubahan requirement, maka dengan mudah dapat diidentifikasi requirement mana yang bermasalah. Metodologi pengembangan perangkat lunak Feature Driven Development (FDD) merupakan salah satu bagian dari metode Agile dimana pendekatan FDD dilakukan secara adaptif dalam pengembangan sistem. FDD merupakan Agile Method yang belum mempunyai aturan baku dalam pembuatan model penelusuran antar-requirement maupun antar-feature. Requirement pada FDD tidak secara eksplisit dapat digali, dicari, dan diatur. Namun dokumentasi requirement dibutuhkan pada proses pembentukan feature list. Oleh sebab itu dibutuhkan suatu dokumentasi model penelusuran antar requirements. Pembuatan model penelusuran requirement dengan menggunakan metode grounded theory yaitu diantaranya adalah open coding, axial coding, selective coding, dan sorting. Penelitian bermula dengan penentuan bidang kaji dan identifikasi masalah, mengumpulkan data-data terkait, melakukan analisis data, dan terakhir mendokumentasikan hasil. Berdasarkan hasil uji, pembuatan model penelusuran requirement pada metode FDD dapat memudahkan penelusuran jika terjadi kesalahan dalam testing hasil akhir program dan dapat dijadikan sebagai acuan pembuatan fungsi dalam pengembangan proyek selanjutnya.

Kata kunci : ***requirement*, penelusuran, keterkaitan, *Feature Driven Development*, dan *grounded theory*.**

1. PENDAHULUAN

Perkembangan *software* (perangkat lunak) saat ini cukup pesat. Para pelaku bisnis berusaha memberikan fitur-fitur menarik dan canggih pada *software* yang mereka kembangkan. Berbagai cara digunakan untuk mengembangkan bisnis perangkat lunak mereka, salah satunya dengan menerapkan metodologi pengembangan perangkat lunak yang sesuai dengan masalah yang

dihadapi. Metodologi pengembangan perangkat lunak digunakan untuk mengetahui proses, analisis masalah, penelitian pasar (*market research*), pengumpulan *requirement*, implementasi, ujicoba, dokumentasi, *deployment*, serta pemeliharaan perangkat lunak.

Keberhasilan dari pembuatan sebuah perangkat lunak apabila dapat berjalan sesuai kebutuhan pengguna (*user*), rilis tepat waktu, biaya yang digunakan sesuai perencanaan, dan implementasi yang memuaskan (*Standish Group International*: 2014). Tentu *requirement* menjadi bagian penting dalam pembuatan perangkat lunak, apabila dalam penyusunannya mengacu pada pembuatan *Requirement Engineering* (RE) yang baik. RE menjadi salah satu fase krusial dalam mendesain perangkat lunak dan salah satu tahap dalam pengumpulan data-data sistem yang diinginkan oleh *customer*.

Hasil penelitian Standish Group pada tahun 1994 dan 1995 mencatat bahwa terdapat beberapa aspek penyebab kegagalan proyek pembuatan perangkat lunak, yangmana sebagian besar disebabkan oleh *requirements* dan spesifikasinya. Kegagalan terjadi akibat ketidakkonsistenan (*inconsistent*), ketidaklengkapan (*incomplete*), maupun ketidakbenaran (*incorrect*) dalam penyususunan *software requirement*.

Definisi RE menurut Zave (1997) adalah cabang dari *software engineering* yang mengurusi masalah yang berhubungan dengan: tujuan (dunia nyata), fungsi, dan batasan-batasan pada sistem *software*. Tujuan dari RE adalah untuk menemukan, mengembangkan, menelusuri, menganalisis, hingga mengolah dokumen *software requirements*. RE mengacu pada seluruh kegiatan hidup *requirement*.

Salah satu hal yang dilakukan oleh RE ialah melakukan manajemen *requirement* (*Requirement Management*). *Requirement Management* (RM) berkaitan dengan semua proses yang terlibat dalam perubahan sistem *requirement* (Sommerville, 1997). Perubahan *requirement* pada saat proses pengembangan perangkat lunak menjadi salah satu masalah yang sering terjadi. Tugas dari RM ialah mendokumentasi, menganalisis, mengaitkan, memprioritaskan, dan memahami segala perubahan *requirement* serta mengontrol perubahan *requirement* tersebut relevan atau tidak.

Requirement Traceability (RT) ialah salah satu bagian dari tahap requirement management. Ramesh dan Jarke menjelaskan bahwa *requirement traceability* adalah "Karakteristik dari sebuah sistem dimana *requirements* yang jelas dikaitkan dengan sumber-sumber dan artefak yang dibuat selama siklus hidup (*life cycle*) pengembangan sistem berdasarkan *requirements* yang ada". RT mengacu pada kemampuan untuk mendefinisikan, menangkap, dan menelusuri *requirements* pada unsur-unsur lain dari sekitar pengembangan *software* serta menelusuri elemen-elemen pada *requirements* (Pinheiro, 1996).

Metodologi pengembangan perangkat lunak *Feature Driven Development* (FDD) merupakan salah satu bagian dari metode *Agile*. FDD menerapkan pendekatan secara adaptif dalam mengembangkan sistem. FDD tidak menyangkup keseluruhan proses pengembangan, melainkan berfokus pada desain dan *building phase* (Palmer dan Felsing, 2002). Menurut Palmer, FDD adalah proses yang didisain dan dilaksanakan untuk menyajikan (*delivery*) hasil kerja secara berulang-ulang dalam waktu tertentu dan dapat diukur. Tidak seperti beberapa metodologi *Agile* lain, FDD diklaim dapat diterapkan pada pengembangan sistem kritikal (Palmer dan Felsing, 2002). FDD merupakan sebuah model pengembangan perangkat lunak yang dibuat berdasarkan fitur yang akan dibuat.

FDD merupakan bagian dari *Agile Method* yang mana memiliki pendokumentasian yang sedikit (*minimal documentation)*. Bentuk pendokumentasi FDD hanya berupa sebuah diagram UML (*Unified Modeling Language*) dan daftar fitur (*feature list*). Diagram UML menjelaskan kejadian atau aktivitas pada sistem perangkat lunak. UML dibuat sebagai pemula atau *intial modeling* dari suatu proses. UML menggambarkan komponen yang menggambarkan *Feature Set Progress* (Coad, 1999). Sedangkan daftar fitur menjadi dasar untuk penjadwalan proyek (*project timelines*) dan juga untuk menelusuri perkembangan dari proyek tersebut.

Requirement pada FDD tidak secara ekspilsit digali, dicari, dan diatur. Namun dokumentasi *requirement* dibutuhkan pada proses pembentukan *feature list*. Oleh sebab itu dibutuhkan suatu dokumentasi model penelusuran antar *requirement* pada pengembangan FDD. Model penelusuran untuk FDD nantinya dapat digunakan untuk melihat keterkaitan antara *requirement* dengan *requirement*, *feature* dengan *feature*, ataupun *requirement* dengan *feature*.

1.1. Requirement

Menurut IEEE (1998), *requirement* adalah sebuah pernyataan yang mengidentifikasi sebuah produk atau operasional proses, fungsi, karakteristik desain atau batasan, yang mana tidak ambigu, mudah dites atau diukur, serta dibutuhkan untuk produk atau dalam proses penerimaan oleh konsumen.

1.1.1. *Requirement Engineering*

Pada dasarnya, aktivitas RE hanya berhubungan pada analisis spesifikasi *requirements* dimana hal tersebut adalah tahap utama pada siklus kehidupan pengembangan perangkat lunak. Pada tahun 1990, RE diterima sebagai kunci utama pada siklus kehidupan *software* dan lingkup RE menjadi lebih luas dari konsep awal di dalam sistem analis.

Banyak peneliti menekankan bahwa proses RE adalah esensi utama dari kualitas sebuah produk *software*, ini didasarkan pada investigasi empiris dan pengalaman industri (Hoare, 1981; Brooks, 1987; Emam, 2000, diacu dalam Li Jiang, 2005: 28).

1.1.2. RE pada *Agile*

Requirement Engineering (RE) berperan dalam keberlangsungan dan keberhasilan proyek *Agile*. Berbeda dengan pelaksanaan RE pada metode tradisional yang meliputi . Berbeda dengan metodologi tradisional yang mana metode tradisional (Seperti *Waterfall*) pelanggan terlibat selama fase awal proyek, sementara pada metode *Agile* pelanggan dilibatkan dalam keseluruhan proses pengembangan.

1.2. *Requirement Traceability*

Penelusuran *requirements* (*requirements traceability*) adalah “kemampuan menelusuri secara deskriptif dengan mengikuti siklus kehidupan *requirements*, menelusuri siklus sebelumnya maupun sesudahnya, menelusuri *requirements* yang masih original maupun yang sudah dikembangkan atau disepesifikasikan, untuk dapat disesuaikan penggunaannya sesuai perubahan-perubahan yang terjadi pada siklus kehidupan suatu *requirements*. Penelusurannya dilakukan secara sekuensial (berurutan) maupun secara iteratif (pengulangan)”. (Got94, diacu dalam Mirka Palo, 2003: 1)

Selama proses perancangan sistem aplikasi, tak jarang pengembang mengalami proses-proses modifikasi. Selama masa itu, penelusuruan *requirements* dapat membantu menanggulangi jika terdapat perubahan disain sistem yang saling keterkaitan. Penelusuran *requirements* juga dapat memeberikan informasi tentang dasar kebenaran, keputusan, dan asumsi yang terlibat dibalik *requirements*.

1.2.1. Model Penelusuran *Requirement*

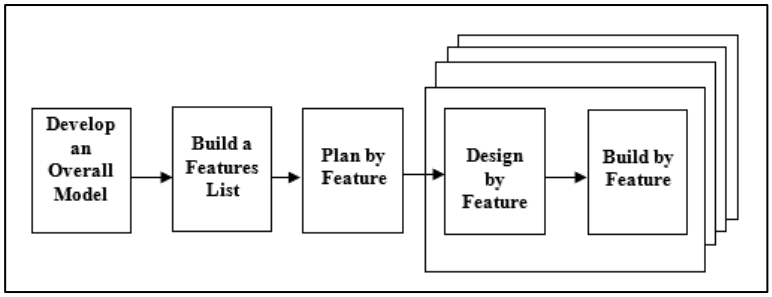
Terdapat beberapa cara untuk melakukan penelusuran *requirements*, 3 (tiga) di antaranya adalah:

1) Berdasarkan arah penelusurannya, penelusuran dilakukan dengan metode *backward and forward traceability.*
2) Berdasarkan perkembangan *requirements*, penelusuran dilakukan dengan menelusuri aspek-aspek yang terlibat, baik sebelum maupun sesudah terjadinya perubahan (*pre and post-requirements spesification traceability*), yang tercantum dalam spesifikasi *requirements.*
3) Berdasarkan tipe objek yang terlibat, penelusuran dilakukan dengan melibatkan aspek internal maupun eksternal dari penelusuruan *requirements*.

1.3. *Feature Driven Development* (FDD)

Feature Driven Development (FDD) merupakan salah satu *Agile Method* yang berfokus pada pengembangan dan penyampaian dimana *requirement* dan fitur-fitur diekstrak / digali dalam cakupan bahasa domain agar mudah dipahami oleh klien. Jeff De Luca dan Peter Coad memperkenalkan FDD pada tahun 1997. Menurut Palmer (2001), FDD adalah proses yang didesain dan dilaksanakan untuk menyajikan (*deliver*) hasil kerja berulang-ulang dalam waktu tertentu dan dapat diukur. FDD adalah pendekatan yang mengacu pembuatan sistem menggunakan metode yang mudah dimegerti dan mudah diimplementasikan; teknik *solving*; dan pelaporan yang mudah dimengerti dan dikontrol oleh *stakeholders*.

FDD terdiri dari 5 (lima) rangkaian proses dalam mendisain serta membangun sistem. Gambar 1.1. merupakan proses FDD menurut Palmer & Festing (2002). Bagian iteratif pada prosess FDD (*Design and Buid*) mendukung pengembangan *Agile* dengan proses adaptasi cepat untuk melakukan perubahan *requirements* dan kebutuhan bisnis.



Gambar 1.1. Proses FDD (Palmer & Fesling : 2002)

1.4. *Grounded Theory*

Penelitian mengenai model *requirement* traceability untuk metodologi perangkat lunak modern *Feature Driven Development* ini, selain secara penyusunan dilandaskan pada metode kualitatif, secara penganalisisan data penelitian ini menggunakan *Grounded Theory*. *Grounded Theory* adalah sebuah penelitian kualitatif yang menghasilkan sebuah teori yang terbentuk dari data-data (Glasser dan Strauss, 1967, diacu dalam Steve Adolph, 2012). *Grounded Theory* adalah metode untuk menggeneralisasikan teori dari kumpulan data.*Grounded theory* memiliki tiga fase pengodingan yang klasik, yaitu *open coding, selective coding,* dan *theoritical coding*.

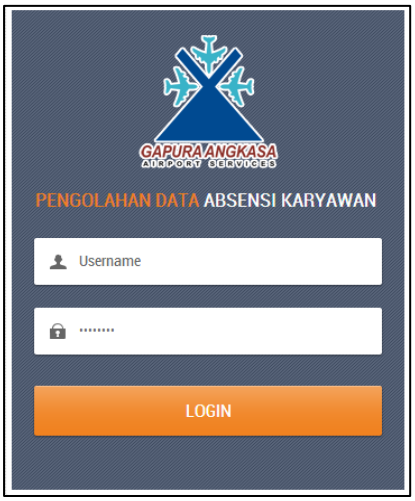
2. METODE PENELITIAN

Penelitian ini menggunakan metode kualitatif dan teori *Grounded Theory* dengan analisis *axial coding* dan *selective coding*. Berikut ini adalah prosedur yang dilakukan dalam penelitian :

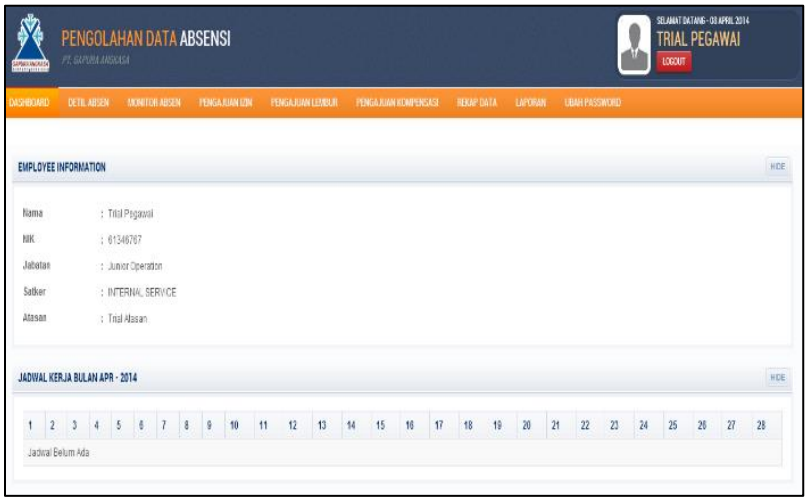
1. Mengidentifikasi Masalah
2. Menentukan Solusi
3. Mengumpulkan Data-Data Terkait

2.1. Sample Perangkat Lunak

Perangkat lunak yang digunakan sebagai *sample* adalah mesin **Pengolahan Data Absensi dari perusahaan PT. Gapura Angkasa.** Perangkat lunak ini merupakan alat bantu untuk memudahkan karyawan PT. Gapura Angkasa dalam melakukan kegiatan administrasi, seperti melakukan absensi, pengajuan lembur, rekapitulasi absensi, dan mengelola laporan absensi.



Gambar 1.2. Tampilan Awal Aplikasi Sistem Absensi PT. Gapura Angkasa



Gambar 1.3. Tampilan Laman Setelah Login *user*

2.2. Dokumentasi Perangkat Lunak

Bentuk dokumentasi atau kumpulan- kumpulan dokumen yang terkait dengan pengembangan sistem menjadi panduan dalam melakukan penelitian ini. Di dalamnya tersimpan proses *requirements management* seperti satu set *reqirements*, *use case, data flow diagram, database, site map, flow chart,* SOP penggunaan aplikasi, dan buku manual. Dokumentasi ini berguna untuk *user* atupun *stakeholder* yang terlibat untuk dapat mengetahui deskripsi kerja dari aplikasi.

2.2.1. Data Requirements

Requirements yang terkumpul dari 4 (empat) user yang terlibat dalam sistem sebanyak 119 *requirements*, terdiri dari :

a. 49 *requirements* untuk *user* Administrator
b. 18 *requirements* untuk *user* Atasan/ Pimpinan
c. 37 *requirements* untuk *user* Tata Usaha
d. 15 *requirements* untuk *user* Karyawan

2.2.2. Model *Requirement Traceability* pada Metodologi Konvensional

Salah satu tahapan dalam penelitian adalah membuat model *requirements traceability*.

Traceability Matrix (TM) merupakan salah satu pendekatan dalam membangun teknik penelususran diantara dua elemen yang berbeda pada dokumen/artefak *software* (Ghazarian:2008). Dokumen dan test matrix adalah contoh dari TM. TM mempermudah dalam menunjukkan proses analisis (*forward* dan *reverse*).

Requirement Traceability Matrix (RTM) berguna untuk mengetahui hubungan keterkaitan antar*requirements* dan antar*requirements* dengan kondisi lingkungan (fungsional).

Tabel 2. 1. Requirement Traceability Matrix

Req_Code	User / Subject Area	Functional Area / Test Case					Relation_Req_Code				

2.3. Melakukan Analisis Data

Poin analisis yang dilakukan adalah :

a. Analisis keterkaitan antar *requirement*
b. Analisis keterkaitan antara *requirement* dengan kondisi yang terlibat
c. Analisis penggeneralisiran *requierment*
d. Analisis keterkaitan antar *feature / feature set*
e. Analisis pemetaan *requirement traceabiity* pada *Feature Driven Development*

2.4. Menguji Model *Requirement Traceability*

Setelah model berhasil dibentuk, kemudian dilakukan pengujian terkait apakah model tersebut dapat diterapkan secara global atau tidak dengan cara membandingkan perubahan keterkaitan yang terjadi antara sebelum penggeneralisiran *requirement* dengan sesudah penggeneralisiran *requirement*.

Perbandingan ditunjukkan dengan banyaknya jumlah keterkaitan antara satu *requirement* dengan *requirement* yang lain, jika tidak terjadi perubahan jumlah, atau terjadi perubahan namun justru keterkaitannya semakin kuat (ditunjukkan dengan jumlah keterkaitanya yang lebih banyak dari sebelumnya) maka dapat dikatakan model yang dibuat tidak merubah keterkaitan antar *requirement* atau dengan kata lain model *requirements traceability* dapat diterapkan pada metodologi pengembangan perangkat lunak *Feature Driven Development*. Selain itu, secara umum, dibuatnya model *requirement traceability* tidak merubah nilai kinerja *Feature Driven Development*.

3. HASIL DAN PEMBAHASAN

Berdasarkan proses penelitian yang telah dilakukan sebelumnya, berikut ini akan dibahas secara jelas mengenai pembuatan model *requirement traceability* untuk metodologi *Feature Driven Development* (FDD) :

1. Membuat daftar *requirements* (lihat Gambar 3.1) yang telah dikumpulkan sebelumnya ke dalam sebuah daftar *requirements* (*requirements list*).

Requirement_Code	Requirement_Name
FR058	Atasan dapat melakukan pengajuan kompensasi secara pribadi
FR059	Atasan dapat melihat daftar izin staff.
FR060	Atasan dapat melakukan permohonan izin staff
FR061	Atasan dapat melihat daftar kompensasi staff

Gambar 3.1. Daftar *Requirements*

2. Membuat matriks atau tabel keterkaitan antar-*requirements* berupa *Requirements Traceability Matrix*. Tabel berbentuk dokumen *checklist*, dimana disi berdasarkan *requirements* yang ada sebelumnya. Gambar 3.2. menggambarkan table RTM.

Req_Code	User/Subject Area			Functional Area/Test Case			Relation_Req_Code			
	Atasan	Pegawai	Admin	Login	Status Approval	Rekap Kehadiran	FR001	FR002	FR003	FR004
FR001										
FR002										
FR003										
FR004										

Gambar 3.2. Model *Requirement Traceability Matrix*

3. Proses pengeneralisasi pada daftar *requirements*. *Requirements* awal dikategorikan berdasarkan ciri serta dimensi yang sama. Hasil generalisasi *requirements* akan menjadikan dasar pembuatan *features set* nanti.

Req_Code	Requirements	General_Req	Gen_Code
FR023	Melihat laporan harian pegawai yang tidak masuk kerja	Menampilkan informasi tentang apakah ada pegawai yang tidak masuk	GR09
FR016	Menampilkan informasi tentang apakah ada pegawai yang tidak masuk		
FR090	Tata Usaha dapat melihat laporan harian pegawai yang tidak masuk kerja		
FR119	Pegawai dapat melihat laporan harian pegawai yang tidak masuk kerja		
FR038	Admin dapat melihat laporan pembayaran lembur tiap pegawai pada daftar tersebut terdapat golongan dan gaji pokok tiap pegawai	Admin dapat melihat laporan pembayaran lembur tiap pegawai pada daftar tersebut terdapat golongan dan gaji pokok tiap pegawai	GR21
FR039	Admin dapat melihat detail laporan pembayaran lembur per bulan		

Gambar 3.3. Generalisasi *Requirements*

4. Membuat *feature list* dari tiap *requirements* terbaru (hasil genaralisir). Sebelumnya *requirements* hasil generalisasi diubah template penulisannya menjadi bentuk kalimat fitur. Kemudian mengelompokkan fitur berdasarkan general *requirements.*

Gen_Code	General_Req	Feature_ Number	Feature_List
GR21	Admin dapat melihat laporan pembayaran lembur tiap pegawai pada daftar tersebut terdapat golongan dan gaji pokok tiap pegawai	F023	Melihat laporan pembayaran lembur tiap pegawai pada daftar tersebut terdapat golongan dan gaji pokok tiap pegawai
GR22	Admin dapat melihat daftar pembayaran detail tiap pegawai per bulan	F024	Melihat daftar pembayaran detail tiap pegawai per bulan
GR23	melihat total kehadiran pegawai dalam satu bulan, dan melihat laporan bulanan tersebut secara detail	F015	Melihat total kehadiran pegawai dalam satu bulan, dan melihat laporan bulanan tersebut secara detail

Gambar 3.4. Tabel Gabungan General *Requirements* dan *Feature List*

5. Setelah membuat daftar fitur, selanjutnya memetakan masing-masing fitur ke dalam tabel baru yang disebut tabel *feature* (*feature set*) berisi : *"Feature_number" , "subject_area",* dan *"feature_list".*

Feature Number	Subject Area				Feature List
	Admin	Atasan	Tata Usaha	Pegawai	
F001	v	v	v	v	Melihat statistik kehadiran pegawai berdasarkan periode laporan
F002	v				Melihat dan mengontrol data pegawai yang bekerja pada PT Gapura Angkasa secara detail
F003	v				Melihat data pegawai yang melakukan revisi absen, serta memonitor kevalidan dari laporan revisi absen tersebut.
F004	v				Melihat daftar pengajuan izin yang dilakukan pegawai beserta status approval

Gambar 3.5. *Feature Set*

6. Analisis kekuatan keterkaitan *requirements* pada *feature set*. Analisis yang dilakukan meliputi :
   - *3.1.* Analisis keterkaitan antar- *requirements*
   - 3.2. Analisis keterkaitan antar-*requirement* dengan kondisi yang terlibat
   - *3.3.* Analisis pengeneralisiran *requirement*
   - *3.4.* Analisis keterkaitan antar-*feature*
   - *3.5.* Analisis pemetaan *requirement traceability* pada *Feature Driven Development*

7. Tahap akhir adalah membuat *Requirement Traceability Matrix* untuk *Feature Driven Development*. Isi kolom terdiri dari : *"Feature_Number","Subject_Area", "Domain_Area",* dan *"Relation_Req_Code".*

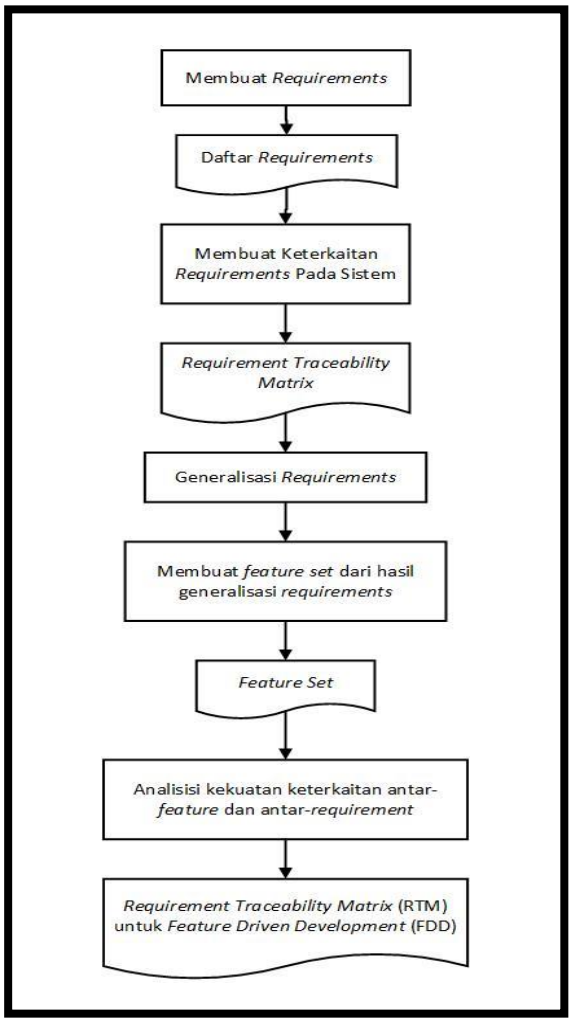
Feature Number	Subject Area				Domain Area					Relation_Req_Code			
	Admin	Atasan	Tata Usaha	Pegawai	Login	Data Kehadiran	Status Approval	Data Pembayaran	Pengajuan Lembur	FR001	FR002	FR003	FR004
F001													
F002													
F003													
F004													

Gambar 3.6. RTM untuk *Feature Driven Development*

Keterangan :

- *Feature Number* : penomoran untuk tiap fitur yang terdapat pada sistem
- *Subject Area* : *user* atau pengguna yang terlibat pada sistem
- *Domain Area* : berisi hasil pengategorian *requirements* berdasarkan kesamaan ciri dan dimensi
- *Relation Req Code* : keterkaitan antar *requirement* dan *feature*

Alur proses pembentukkan model RTM di atas dapat digambarkan melalui model berikut ini :



Gambar 3.7. Alur Pembentukan Model RTM untuk FDD

KESIMPULAN

Berdasarkan hasil pembahasan penelitian pada skripsi ini, maka dapat diambil kesimpulan bahwa pembuatan model penelusuran keterkaitan *requirement* dapat dibuat atau diterapkan pada metode pengembangan perangkat lunak *Feature Driven Development* (FDD). Dengan tujuan, untuk dapat memudahkan penelusuran jika terjadi kesalahan dalam *testing* hasil akhir sebuah program dan dapat digunakan sebagai acuan pembuatan fungsi dalam pengembangan proyek selanjutnya.

Dari hasil penelitian yang dilakukan, sekiranya dapat dikembangkan lagi. Pengembangan lain yang dapat dilakukan baiknya memperhatikan aspek-aspek berikut :

a. Model penelusuran *requirement* dapat dikembangkan untuk metodologi pengembangan perangkat lunak modern *Agile* lainnya.
b. Model penelusuran tidak hanya dapat dikembangkan dengan parameter kekuatan keterkaitan antara-*requirements* saja, namun juga dapat dikembangkan degan parameter lain seperti keserasian atau keselarasan satu *requirements* dengan *requirements* yang lain.
c. Analisis penelusuran *requirements* tidak hanya berbentuk sebuah tabel atau matriks, dapat juga diaplikasikan dalam bentuk lain, misalnya penelusuran berbentuk grafik .

DAFTAR PUSTAKA

Abrahamsson, P., Salo, O., & dkk. (2002). *Agile Software Development Methods.* Finlandia: VTT Publication.

Adolph, S., Hall, W., & Kruchten, P. (2011). Using Grounded Theory to Study The Experience of Software Development. *Empirical Software Engineering, 16*(4), 487-513.

Arimbawa, I. A. (2014). Feature Driven Development (FDD), Apakah Bisa Disebut Agile ? *STMIK Lombok*, 1-5.

Aurum, A., & Wohlin, C. (2005). Engineering and Managing Software Requirements. *Requirements Engineering for Agile Methods, 18*, 309-325.

Coad, P., & De Luca, J. (1999). *Java Modeling in Color With UML.* Upper Saddle River: NJ: Prentice Hall PTR.

Elizabeth, K. (2010). *Requirement Engineering: Third Edition.* London: Springer.

Ghazarian, A. (2008). Traceability Pattern: An Approach to Requirement-Component Traceability in Agile Software Development. *The 8th WSEAS International Conference on APPLIED COMPUTER SCIENCE* (hal. 236-241). Canada: University of Toronto.

Goyal, S. (2007/2008). *Major Seminar on Feature Driven Development: Agile Techniques for Project Management and Software Engineering.* Munich: Technical University Munich.

Hoda, R., Noble, J., & Marshall, S. (2011). Developing a Grounded Theory to Explain The Practices of Self-Organizing Agile Teams. *Empirical Software Engineering, 17*(6), 609-639.

Hull, E., Jackson, K., & Dick, J. (2011). *Requirements Engineering: Third Edition.* London: Springer.

IEEE Computer Society. (2004). *SWEBOK: Guide to the Software Engineering Body of Knowledge.* California: Angela Burgess.

IEEE STD 1220-1998. (1998). *Standard for Application and Management of the Systems Engineering Process.* New York: IEEE.

Jiang, L. (2005). *A Framework for The Requirements Engineering Process Development.* Alberta: University of Calgary.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements Engineering and Agile Software Development. *Proceedings of The 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (hal. 1-5). IEEE Computer Society.

Palmer, S. R., & Felsing, J. (2001). *A Practical Guide to Feature-Driven Development*. Dipetik November 2015, dari www.step- 10.com

Palmer, S. R., & Felsing, J. M. (2002). *A Practical Guide tp Feature-Driven Development.* Upper Saddle River, NJ: Prentice-Hall.

Palo, M. (2003). *Requirements Traceability*. Department of Computer Science University of Helsinki.

Pinheiro, F., & Goguen, J. A. (1996). *An Object- Oriented Tool for Tracing Requirements*. IEEE Software, 13(2), 52-64.

Ramesh, B., & Jarke, M. (2001). *Towards Reference Models for Requrements Traceability*. IEEE Transactions on Software Engineering, 27(1), 58-93.

Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering: A Good Practice Guide*. England: John Wiley & Sons.

Strauss, A., & Corbin, J. (2003). *Dasar-Dasar Penelitian Kualitatif*. (Terjemahan oleh M. Shodiq, & I. Muttaqien) Yogyakarta: PUSTAKA BELAJAR.

The Standish Group. (2012). *Chaos Manifesto 2012: The Year of the Executive Sponsor*. The Standish Group.

Young, R. R. (2004). *The Requirements Engineering Handbook*. London: Artech House.

Zave, P. (1997). *Classification of Research Efforts in Requirements Engineering*. ACM Computing Surveys, 29(4), 315-321.