TESIS – KI142502

ANALISIS KETERKAITAN ANTARA VOLATILITAS KEBUTUHAN PERANGKAT LUNAK DENGAN STABILITAS RANCANGAN PERANGKAT LUNAK BERBASIS OBJEK

Felix Handani
5114201041

DOSEN PEMBIMBING
Dr. Ir. Siti Rochimah, MT.
NIP. 196810021994032001

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

THESIS – KI142502

ANALYSIS OF RELATIONSHIP BETWEEN REQUIREMENTS VOLATILITY AND SOFTWARE COMPONENT DESIGN STABILITY IN OBJECT-ORIENTED SOFTWARE

FELIX HANDANI
5114201041

SUPERVISOR
Dr. Ir. Siti Rochimah, MT.
NIP. 196810021994032001

MASTER PROGRAM
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:

FELIX HANDANI
Nrp. 5114201041

Dengan judul :
ANALISIS KETERKAITAN ANTARA VOLATILITAS KEBUTUHAN PERANGKAT LUNAK DENGAN STABILITAS RANCANGAN PERANGKAT LUNAK BERBASIS OBJEK

Tanggal Ujian : 9-1-2017
Periode Wisuda : 2016 Gasal

Disetujui oleh:

Dr. Ir. Siti Rochimah, M.T (Pembimbing 1)
NIP. 196810021994032001

Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng. (Penguji 1)
NIP. 197411232006041001

Sarwosri, S.Kom, M.T (Penguji 2)
NIP. 197608092001122001

Rizky Januar Akbar, S.Kom, M.Eng (Penguji 3)
NIP. 198701032014041001

Direktur Program Pasca Sarjana,

an. Direktur Program Pascasarjana
Asisten Direktur

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 19611021 198603 1 001

Prof. Ir. Djauhar Manfaat, M.Sc, Ph.D
NIP. 196012021987011001

[Halaman ini sengaja dikosongkan]

ANALISIS KETERKAITAN ANTARA VOLATILITAS KEBUTUHAN PERANGKAT LUNAK DENGAN STABILITAS RANCANGAN PERANGKAT LUNAK BERBASIS

Nama Mahasiswa : Felix Handani
NRP : 5114 201 041
Pembimbing : Dr. Ir. Siti Rochimah, MT.

ABSTRAK

Arsitektur perangkat lunak adalah struktur inti dari sebuah perangkat lunak. Arsitektur perangkat lunak mencerminkan fungsi dan tujuan latar belakang dari perangkat lunak yang dibangun. Pada faktanya perangkat lunak selalu mengalami evolusi dari waktu ke waktu. Evolusi ini disebabkan berbagai hal seperti perkembangan teknologi dan perubahan kebutuhan penggunanya. Perubahan kebutuhan pengguna merupakan faktor utama terjadinya evolusi.

Dalam menyikapi evolusi, beberapa peneliti mengungkapkan rancangan perangkat lunak yang baik memiliki kecenderungan stabil terhadap perubahan minor agar menghindari terjadinya erosi perangkat lunak. Erosi menyebabkan munculnya kerusakan komponen akibat salah perubahan kode sumber yang seharusnya tidak diubah, dan pelanggaran keputusan rancangan dari komponen yang telah direncanakan pada awalnya. Hal ini menyebabkan kecacatan yang akan muncul pada kemudian hari.

Penelitian ini menjabarkan analisis mendalam tentang hubungan stabilitas arsitektur perangkat lunak dengan volatilitas kebutuhan perangkat lunak. Stabilitas rancangan perangkat lunak diukur berdasarkan tingkat penggunaan kembali komponen seperti Kelas Diagram, Paket Diagram dan Relasi antar kelas. Pengukuran dilakukan dengan memperhatikan perubahan jumlah kelas dalam satu paket dan jumlah relasi antar atau dalam paket. Pengukuran volatilitas kebutuhan dilakukan dengan memperhatikan perubahan fitur pada setiap versi. Dari dua pengukuran tersebut, peneliti mengamati perkembangan fenomena yang terjadi di dalamnya.

Terdapat tiga kondisi yang membuat ketidakstabilan rancangan perangkat lunak. Pertama, penambahan fitur perangkat lunak yang tidak terdefinisi pada awal, sehingga pada versi sebelumnya komponen perangkat lunak tidak dapat dirancangan terlebih dahulu. Hal ini dapat melebarkan lingkup perangkat lunak itu sendiri dan memungkinkan adanya degradasi pada komponen tertentu. Hal ini terlihat pada kasus AB_{2-3}, AB_{3-4}, $AB_{6-7,}$ AB_{7-8}, AB_{9-10}, NS_{1-2}, dan NS_{2-3}. Namun apabila rancangan sudah dibentuk pola perancangan yang lebih general, maka nilai stabilitas rancangan akan lebih tinggi dibanding dengan rancangan yang tidak memiliki pola perancangan. Faktor kedua adalah pengubahan fitur yang berdampak pada pelebaran lingkup perangkat lunak. Hal ini tercermin pada kasus AB_{2-3}, dan $NS_{3-4.}$ Faktor terakhir adalah penghapusan fitur inti yang dimiliki oleh lebih dari 50% fitur dari seluruh versi yang ada. Hal ini tercermin pada kasus $AB_{3-4,}$ dan $NS_{2-3.}$

Kata Kunci: Analisis mendalam, Metrik, Rancangan perangkat lunak, Teknik pengujian statistika, Volatilitas fitur perangkat lunak.

[Halaman ini sengaja dikosongkan]

ANALYSIS OF RELATIONSHIP BETWEEN REQUIREMENTS VOLATILITY AND SOFTWARE COMPONENT DESIGN STABILITY IN OBJECT-ORIENTED SOFTWARE

Student Name : Felix Handani
NRP : 5114 201 041
Supervisor : Dr. Ir. Siti Rochimah, MT.

ABSTRACT

Software architecture is the core structure of a software. Software architecture reflects functional background and purpose of the software is built. In fact the software is always evolving from time to time. This evolution is due to various things such as technological development and the needs for minor changes. The user need afford a change is a major factor in the evolution.

In addressing the evolution, some researchers revealed that architectural and component design good software has a stable trend towards minor changes in order to avoid erosion of the software. Erosion caused the emergence of elemental damage as a result of changes to the source code that should not be changed, and violations of the architectural design decisions that had been initially planned. It causes disability that will appear at a later date.

In this study, I conducted a thorough analysis on relationship between the software component stability and volatility of the software requirements. The stability of the software component is measured according to a study presented by Aversano and Constantinou. Component measurement done by considering the number of classes in one package and the number of relationships among or within the package. Volatility measurements conducted with respect to the needs of the changing needs of each version. From the two measures, the researcher observes the development of phenomena that occur in it.

There are three conditions that create instability software design. First, the addition of software features that are not defined at the beginning, so that the previous version of the software component can't be designed first. It can widen the scope of the software itself and allows the degradation of certain components. This is seen in the case of AB_{2-3}, AB_{3-4}, AB_{6-7}, AB_{7-8}, AB_{9-10}, NS_{1-2}, and NS_{2-3}. However, if the software component is already established design patterns that are more general, then the value of the stability of the design will be higher than with a design that does not have a pattern design. The second factor is the conversion feature that impact on widening the scope of the software. This is reflected in the case AB_{2-3}, and NS_{3-4}. The last factor is the elimination of the core features that are owned by more than 50% of the features of the entire existing version. This is reflected in the case AB, and NS_{2-3}.

Keywords: Analysis, Measurement of the stability of the software component, Reuseability, Statistical analysis, Volatility software requirement.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Tuhan Yesus dan Bunda Maria, yang telah melimpahkan rahmat, kasih dan karunia-Nya sehingga penulis bisa menyelesaikan Tesis yang berjudul "Analisis Kuantitatif Keterkaitan Tingkat Perubahan Kebutuhan Dengan Tingkat Stabilitas Rancangan Arsitektur Perangkat Lunak Berbasis Objek" sesuai dengan hasil yang diharapkan.

Pengerjaan Tesis ini merupakan suatu kesempatan yang sangat berharga bagi penulis untuk belajar memperdalam ilmu pengetahuan. Dalam proses penelitian Tesis ini, tidak terlepas dari bantuan dan dukungan semua pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yesus dan Bunda Maria atas rahmat, kasih, karunia dan roh Kudus sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Bapak Jacobus Budidarma, Ibu Lani Rahardjo dan saudari Yohana, selaku keluarga penulis yang selalu mendukung dan mendoakan agar selalu diberikan kelancaran dan kemudahan dalam menyelesaikan Tesis ini dengan baik.
3. Ibu Dr. Ir. Siti Rochimah, M.T. selaku dosen wali dan pembimbing yang telah memberikan kepercayaan, motivasi, bimbingan, nasehat, perhatian serta semua bantuan yang telah diberikan kepada penulis dalam menyelesaikan Tesis ini.
4. Bapak Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng, Bapak Rizky Januar Akbar, S.Kom, M.Eng, Ibu Sarwosri, S.Kom, M.T dan Ibu Ratih Nur Esti Anggraini, S.Kom, M.Sc selaku dosen penguji yang telah memberikan bimbingan, saran, arahan, dan koreksi dalam pengerjaan Tesis ini.
5. Bapak Waskitho Wibisono, S.Kom., M.Eng., PhD selaku ketua program pascasarjana Teknik Informatika ITS, Ibu Dr. Chastine Fatichah, M.Kom, selaku dosen wali penulis dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Ibu Lina, Bapak Kunto dan segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.
7. Emilia Tungary yang selalu memotivasi, membantu dan mendoakan penulis agar tetap fokus menjalan penelitian tesis ini.
8. Surveyor dan Validator baik dari akademisi, Mahasiswa atau Praktisi dari pra-penelitian untuk dukungan menghimpun dataset pada penelitian ini.

9. Rekan penulis Moh. Farid Naufal, Siska Arifiani, dan Devi Yolanda yang selalu menjadi teman diskusi yang baik.
10. Rekan-rekan angkatan 2014 Pasca Sarjana Teknik Informatika ITS yang telah menemani dan memberikan bantuan serta motivasi untuk segera menyelesaikan Tesis ini.
11. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu disini yang telah membantu terselesaikannya Tesis ini.

Penulis menyadari bahwa Tesis ini masih jauh dari kesempurnaan dan memiliki banyak kekurangan, sehingga dengan segala kerendahan hati, penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Desember 2016

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

Pada bab ini dijelaskan mengenai beberapa hal dasar dalam pembuatan proposal penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah

1.1. Latar Belakang

Perubahan perangkat lunak sering terjadi pada setiap fase pengembangannya. Bila satu komponen mengalami perubahan, maka komponen yang lain perlu penyesuaian kembali. Chen bersaudara (Chen & Chen, 2009) menganalisis dampak dari perubahan dan memberikan solusi dari sisi manajemen atau teknis. Penelitian yang dilakukan Ferreira (Ferreira, 2009) juga memberikan pendalaman konsep tentang efek dari kebutuhan yang berubah-ubah dengan permodelan analitis dan simulasi "Software Process". Penelitian yang dilakukan Williams (Williams & Carver, 2010) melakukan literatur review secara sistematik terhadap karakteristik perubahan dari arsitektur perangkat lunak. Dari literatur di atas, faktor dominan yang menyebabkan perubahan perangkat lunak adalah faktor keinginan manusia yang terus berkembang, faktor perkembangan teknologi, faktor budaya dan peraturan tempat sistem digunakan. Faktor-faktor di atas disusun dalam sebuah dokumen kebutuhan perangkat lunak. Dalam dokumen tersebut tersusun atas kebutuhan dari sistem beserta fitur-fitur yang akan diimplementasi.

Permasalahan yang sering terjadi bila suatu perangkat lunak berubah tanpa adanya kontrol adalah munculnya erosi. Pengertian erosi pada perangkat lunak adalah terjadinya penyimpanan kebutusan pada perangkat lunak. Kebusukan yang terjadi disebabkan oleh kelalaian pengembang perangkat lunak untuk menghapus atau memodifikasi kode tertentu dan kurangnya transfer ilmu antara pengembang antar versi sehingga kaidah rancangan perangkat lunak atau batasan-batasan yang ditentukan pada versi awal tidak sepenuhnya dipatuhi pada pengembangan versi berikutnya. Kebusukan tersebut dapat menurunkan performa sistem atau menurunkan respon system sehingga berakibat sistem perangkat lunak mengalami

kegagalan eksekusi perintah, tidak layak guna dan butuh untuk perbaikan yang memakan biaya dan waktu lebih tinggi. (Williams & Carver, 2010).

Penelitian tentang menjaga stabilitas sebuah perangkat lunak menjadi topik penelitian tersendiri dari proses pengembangan perangkat lunak. Topik ini cenderung bersifat preventif terhadap perubahan-perubahan yang terjadi. Karena perubahan tidak dapat terelakan, untuk menjaga stabilitas perangkat lunak perlu ada komponen dari perangkat lunak yang dikontrol agar tidak mengalami perubahan signifikan. Komponen yang dimaksud adalah rancangan arsitektur. Breivold *et all* (Breivold, Crnkovic, & Larsson, 2012) mendefinisikan arsitektur perangkat lunak sebagai sebuah fondasi dasar dari perangkat lunak. Secara definitif, arsitektur perangkat lunak disimbolkan sebagai tingkat abstraksi yang paling cocok pada saat awal pembuatan proyek perangkat lunak (Jamshid, Ghafari, Ahmad, & Pahl, 2013). Dengan penggunaan model arsitektural diharapkan memiliki manfaat seperti mengetahui mana saja kebutuhan yang telah diakomodasi, secara sistematik membantu komunikasi dengan stakeholder dan mampu membuat dasar dari penilaian kualitas, mendukung rancangan kristalisasi dan keputusan untuk melakukan evolusi, pengontrol evolusi, dan memperpanjang umur hidup perangkat lunak. Bila ada perubahan pada rancangan arsitektur, tentu berdampak pada artefak lain dan menyebabkan biaya perawatan menjadi membengkak dan waktu pengerjaan terlambat.

Beberapa peneliti juga telah meneliti hubungan keterkaitan antara kebutuhan dengan arsitektur. Huang, *et all.* (Cleland-Huang, Hanmer, Supakkul, & Mirakhorli, 2013) menginisiasi seminar internasional bertajuk “Twin Peak of Requirements and Architecture”. Seminar tersebut berfokus pada hubungan erat antara kebutuhan dan tahap perancangan arsitektur agar efisiensi sumber daya manusia dan waktu pekerjaan dapat terkontrol dengan baik dan mengurangi pembengkakakan biaya akibat kebutuhan yang tidak terakomodasi pada sistem. Selain Huang, de Boer (de Boer & van Vliet, 2009) mengenalkan konsep berpikir tentang kesamaan antara kebutuhan dari perangkat lunak dengan arsitektur pada implementasinya. Oleh karena itu, penelitian ini berfokus pada analisis mendalam mengenai hubungan volatilitas dengan rancangan perangkat lunak pada tingkat komponen pada setiap evolusi perangkat lunak.

1.2. Perumusan Masalah

Secara teoritis, suatu perangkat lunak yang tangguh mampu mudah beradaptasi dengan mudah pada perubahan lingkungannya. Penambahan, pengubahan dan pengurangan kebutuhan dari sebuah sistem tidak berdampak signifikan pada rancangan arsitektur apabila arsitek memperhatikan kaidah-kaidah rancangan. Namun pada prakteknya, konsep tersebut sulit dicapai. Sebuah rancangan arsitektur perangkat lunak dipengaruhi oleh banyak faktor seperti keahlian arsitek, pengetahuan arsitek, kebutuhan yang akan diimplementasi, kebergantungan pada teknologi tertentu, batasan-batasan lingkungan sistem dan faktor nonteknis, seperti proses bisnis, kebijakan dan faktor manajerial sebuah perusahaan. Dengan faktor-faktor yang kompleks tersebut, seorang arsitek sedikit banyak melanggar kaidah-kaidah dalam merancang perangkat lunak.

Peneliti berhipostesis bahwa besarnya perubahan kebutuhan pada perangkat lunak berpengaruh pada perubahan rancangan perangkat lunak dan sebaliknya pada kondisi tertentu. Dengan kata lain, tingkat volatilitas kebutuhan dengan tingkat stabilitas rancangan perangkat lunak saling mempengaruhi pada kondisi tertentu.

Definisi kebutuhan perangkat lunak beragam. Kebutuhan perangkat lunak menurut Leffingwell (Leffingwell & Widrig, 2000) mengungkapkan fitur-fitur yang lebih rinci, di mana setiap kebutuhan harus dipenuhi saat fase implementasi sehingga menjadi fitur yang didapat disampaikan kepada pengguna. Fitur disampaikan dalam bentuk abstraksi level tinggi untuk mempermudah menggambarkan lingkup sistem secara menyeluruh, tetapi sulit dipakai untuk fase pengembangan karena tidak spesifik. Sebaliknya kebutuhan perangkat lunak adalah spesifik sehingga dari data kebutuhan menjadi bahan materi untuk implementasi dan validasi saat waktu pengujian perangkat lunak. Pada artikel Leffingwell (Leffingwell D. , 2001) lainnya, Kebutuhan (*requirements*) perangkat lunak terdiri atas dua yaitu: permasalahan yang akan dipecahkan (*needs*) dan solusi-solusi dalam bentuk fitur untuk memecahkan permasalahan (*features*). Peneliti menyimpulkan bahwa kebutuhan perangkat lunak adalah kemampuan perangkat lunak yang dibutuhkan oleh pengguna untuk memecahkan masalah untuk mencapai tujuan tertentu. Definisi lain yang diungkapkan adalah

kemampuan perangkat lunak yang harus dipenuhi atau dimiliki oleh sistem atau komponen sistem untuk memenuhi kontrak, standar, spesifikasi atau dokumentasi resmi lainnya. Semua kebutuhan tersebut terangkum dalam dokumen spesifikasi kebutuhan perangkat lunak. Dalam garis besar dokumen kebutuhan, terdapat subbab mengenai fitur sistem. Subbab itu digunakan sebagai penjelasan singkat dari fitur dan menunjukkan faktor kepentingan di dalamnya seperti resiko, keuntungan, biaya dan pinalti. Dalam penelitian ini, pengukuran perubahan kebutuhan dilakukan pada tingkat fitur dari sistem, karena tingkat fitur mewakili sebuah kebutuhan tertentu dan fitur berada pada tingkat solusi sehingga dekat dengan implementasi sistem.

Dari penjabaran naratif di atas, maka permasalahan yang akan diselesaikan dalam penelitian ini terangkum dalam 3 poin sebagai berikut.

1. *Bagaimana membentuk data uji untuk pengukuran rancangan perangkat lunak dan volatilitas kebutuhan?*
2. *Bagaimana melakukan perbandingan stabilitas rancangan dan volatilitas kebutuhan perangkat lunak?*
3. *Bagaimana perbandingan hasil analisis keterkaitan volatiltias kebutuhan dan stabilitas rancangan perangkat lunak?*

1.3. Batasan Masalah

Arsitektur memiliki berbagai tingkat yaitu tingkat abstraksi/ konsep seperti pada subbab 2.1 dan tingkat implementasi dengan memperhatikan diagram komponen seperti diagram paket, diagram kelas dan isi dari masing-masing kelas. Masing-masing tingkatan memiliki berbagai faktor untuk evaluasi kualitas. Salah satu evaluasi adalah tingkat stabilitas dari rancangan perangkat lunak.

Menurut penelitian Vishnyakov, perancangan perangkat lunak adalah kunci utama dari salah satu proses pengembangan perangkat lunak (Vishnyakov & Orlov, 2015). Perancangan perangkat lunak sendiri terdiri atas 2 bagian yaitu: rancangan arsitektural dan rancangan rincian dari arsitektur. Prinsip pada perancangan perangkat lunak menurut buku SWEBOK v3.0 (Bourque & Fairley) adalah (1) abstraksi; (2) *coupling* dan *cohesion*; (3) dekomposisi dan modularisasi; (4) enkapsulasi; (5) pembagian antara rancangan antarmuka dengan

implementasi; (6) kelengkapan dan ketersediaan data yang terwakili dalam rancangan; (7) pembagian berdasarkan level konsentrasi dengan penggunanya. Untuk perancangan arsitektural, terdapat beberapa poin penting lainnya seperti ciri arsitektural (biasa disebut: *Architectural Styles*), perancangan pola (biasa disebut: D*esign Pattern*) dan keputusan rancangan arsitektur (biasa disebut: *Architecture Design Decision*). Keputusan rancangan arsitektur ini adalah sebuah proses kreatif seorang perancangan atau analis sistem dalam membangun perangkat lunak, sehingga kurang lebihnya ada faktor subyektif dari masing-masing individu perancang. Penelitian ini mengeluarkan faktor keilmuan dan kreativitas seorang arsitek. Penelitian ini hanya terbatas dengan melihat kualitas produk arsitektur dengan mengukur tingkat koupling, kohesi dan besarnya kelas yang diimplementasi di dalam paket. Beberapa poin yang menjadi batasan penelitian adalah sebagai berikut.

a. Produk yang akan menjadi pengujian adalah produk berbasis paradigma obyek.
b. Peneliti mengesampingkan faktor-faktor subyektif setiap arsitek dalam merancang perangkat lunak.
c. Data kebutuhan yang diukur adalah fitur dari kebutuhan fungsional yang ditangkap dari data pengamatan.
d. Rancangan perangkat lunak diukur dari tingkat abstraksi yang rendah yaitu komponen seperti diagram paket, diagram kelas dan isi dari kelas.
e. Stabilitas perangkat lunak diukur pada tingkat penggunaan kembali komponen dari versi satu dengan versi yang lain.

1.4. Tujuan

Tujuan dari tesis ini adalah memberikan pembuktian dan penjelasan secara analitis mengenai keterkaitan antara volatilitas kebutuhan pada tingkat fitur dengan stabilitas rancangan perangkat lunak. Diharapkan proses analisis dan evaluasi memberikan referensi pada setiap perancangan perangkat lunak mengenai poin-poin yang perlu diperhatikan pada merancang perangkat lunak.

1.5. Manfaat Penelitian

Dari penelitian ini yang diharapkan menjadi penelitian awal dari topik kebutuhan dengan arsitektur perangkat lunak. Hasil berupa seberapa besar keterkaitan antara volatilitas kebutuhan dengan stabilitas rancangan dapat menjadi nilai pertimbangan untuk pengembangan penelitian berikutnya.

1.6. Kontribusi Penelitian

Kontribusi dalam penelitian ini adalah:

(1) Data Set Program berupa metadata volatilitas kebutuhan dan Stabilitas Kebutuhan;
(2) Kerangka penghitungan kuantitatif tentang volatilitas kebutuhan dengan stabilitas rancangan perangkat lunak; dan
(3) Analisis mendalam yang menghubungkan volatilitas kebutuhan dengan stabilitas rancangan.

BAB 2
DASAR TEORI DAN KAJIAN PUSTAKA

Pada Bab 2 ini dijelaskan dasar teori dan kajian pustaka yang digunakan sebagai landasan ilmiah penelitian. Kajian pustaka yang dilakukan mencakup arsitektur perangkat lunak dan perancangan, pengamatan literatur terkait, metrik stabilitas pada arsitektur perangkat lunak, pengukuran tingkat perubahan kebutuhan, dan analisis data pengamatan.

2.1. Arsitektur Perangkat Lunak dan Perancangan Perangkat Lunak

Arsitektur Perangkat Lunak adalah sebuah struktur paling tinggi dari sebuah sistem perangkat lunak yang mengatur aturan-aturan rancangan dari komponen sistem (Clements, 2010). Pada penerapan pada industri, Arsitektur Perangkat Lunak dapat dianalogikan sebagai Arsitektur pada pembangunan sipil yang mengatur struktur bangunan secara menyeluruh sehingga dari rancangan arsitektur dapat dilihat purwa-rupa dari perangkat lunak yang akan dibuat (Perry & Wolf, 1992). Pengetahuan terhadap gaya dari arsitektural dari perangkat lunak wajib dimiliki oleh seorang arsitek perangkat lunak. Menurut SWEBOK v3.0 (Bourque & Fairley) beberapa penggolongan gaya dalam arsitektural sebagai berikut.

1. Struktur umum, seperti blackboard, layers, dan pipes.
2. Struktur terdistribusi, seperti client-server, three-tiers, dan broker.
3. Struktur interaktif, seperti Model-View-Controller, Presentation-Abstraction-Control).
4. Struktur adaptasi, seperti reflection, dan microkernel.
5. Struktur lainnya, seperti intepreter, batch, process-control, rule-based.

Gaya arsitektural ini mengakomodasi level tertinggi dari organisasi sebuah perangkat lunak. Selain gaya dari arsitektural, arsitektur perangkat lunak memiliki pola-pola beragam. Pola-pola ini mengisi gaya dari arsitektural yang dipakai sebagai level tertinggi. Pada SWEBOK v3.0 (Bourque & Fairley), terdapat 3 golongan pola arsitektur yaitu: (1) pola pembuatan, seperti builder, factory method, prototype, singleton; (2) pola struktur, seperti adapter, bridge,

composite, decorator, facade, fly-weight, proxy; dan (3) pola behavioral, seperti intepreter, iterator, mediator, memento, observer, state, strategy, tempate, visitor. Pembagian tersebut berdasarkan kegunaan dan maksud pola dibentuk. Setiap arsitektur perangkat lunak bebas memilih gaya arsitektural dan pola arsitektural untuk produk perangkat lunak yang akan dikembangkan. Proses merancang arsitektural merupakan proses kreativitas dari masing-masing individu pembuat. Faktor-faktor yang memperngaruhi hasil dari produk rancangan yang dikemukakan oleh Breivold (Breivold, Crnkovic, & Larsson, A systematic review of software architecture evolution research, 2012) adalah: faktor individu, seperti kedalaman ilmu pengetahuan, pengalaman bekerja dan kreativitas; faktor manajerial seperti kepuasan pengguna, tujuan/ motiviasi bisnis; faktor teknis seperti keterbatasan pada tingkat implementasi dan kualitas perangkat lunak, di mana faktor stabilitas masuk di dalamnya.

2.2. Pengamatan Literatur Terkait Stabilitas Rancangan Perangkat Lunak

Penelitian mengenai kualitas dan evaluasi kualitas sebuah produk perangkat lunak telah berkembang luas. Beberapa penelitian memiliki tujuan untuk membahas secara kuantitatif dengan mengumpulkan beberapa metrik seperti kompleksitas, waktu estimasi, tingkat keeratan rancangan dengan modul-modul di dalamnya dan beberapa metrik lain yang berkaitan. Penelitian tersebut berfokus pada aspek internal dari kualitas rancangan perangkat lunak. Di sisi lain, penelitian yang berfokus pada aspek arsitektural juga diusung oleh beberapa penelitian. Biasanya penelitian ini ditujukan untuk kepentingan penggunaan kembali komponen atau elemen-elemen dari perangkat lunak, sehingga semakin banyak elemen yang terpakai kembali, dapat disimpulkan stabilitas perangkat lunak tinggi (Aversano, Molfetta, & Tortorella, 2013).

Menurut ISO 9126 yang berfokus pada fase perawatan perangkat lunak, terdapat 6 komponen besar yang dapat diukur. Stabilitas adalah salah satu komponen dari fase perawatan dan pemeliharaan dari perangkat lunak tersebut. Metrik Stabilitas mengindikasi sebuah atribut untuk memprediksi bagimana kestabilan dari sebuah produk perangkat lunak setelah adanya modifikasi. Terdapat 2 metrik yang ada pada ISO 9126 terkait pengukuran stabilitas. Pertama,

mengukur frekuensi dampak buruk setelah modifikasi. Pengukuran pada persamaan (1) dilakukan dengan menghitung munculnya dampak buruk dalam bentuk angka numerik dan dibandingkan dengan jumlah modifikasi yang dilakukan persatuan kelas/paket/persentase atau pengukuran lain. Kedua, pengukuran seberapa besar dampak dari modifikasi pada produk perangkat lunak. Pengukuran persamaan (2) yaitu menghitung jumlah variable yang berubah dengan jumlah variable keseluruhan dari produk.

$$X = 1 - \frac{A}{B} \quad 1)$$

A = jumlah dampak buruk yang muncul setelah modifikasi

B = jumlah perubahan yang dilakukan

Rentang hasil adalah $0 \leq X \leq 1$, di mana mendekati angka 1 dikatakan lebih stabil

$$X = \frac{A}{B} \quad 2)$$

A = jumlah yang terimbas perubahan

B = jumlah seluruh variable pada produk

Rentang hasil adalah $0 \leq X \leq 1$, di mana mendekati angka 0 dikatakan lebih sedikit terkena imbas

Penelitian yang menitikberatkan pada aspek rancangan arsitektur pada tahun 2000 dimulai dari Alshayeb (Alshayeb & Li, 2005). Penelitian ini berfokus pada pembelajaran dalam mengenai penggunaan metrik stabilitas desain sistem, yang selanjutnya disebut *System Design Instability* (SDI), untuk kepentingan evolusi berikutnya dengan metode "eXtreme Programming". Dari hasil perumusan tersebut, peneliti kemudian mengkomparasi tingkat hubungan antara aktifitas pada XP dengan hasil pengukuran dari metrik SDI.

Penelitian yang dilakukan Aversano *et all* (Aversano, Molfetta, & Tortorella, 2013) menjelaskan sistematika evaluasi sebuah arsitektur perangkat lunak pada tingkat stabilitasnya. Pertama, penulis membagi versi dari project menjadi 3 tahap yaitu initial point, middle point dan final point. Kedua, penulis menganalisis jumlah paket dan relasi antar paket dalam sistem. Dalam perhitungan stabilitasnya, penulis menggunakan konsep *Design Instability*. Pada konsep tersebut, terdapat 2 metrik yaitu: *Core Design Instability* (CDI) dan *Core Calls Instability* (CCI). CDI mengevaluasi perubahan pada paket di sistem.

Parameter yang diperlukan adalah jumlah paket pada rilis tertentu, jumlah paket baru yang muncul dan jumlah paket yang hilang. Rumus CDI dilihat pada persamaan (3). CCI (Aversano, Molfetta, & Tortorella, 2013) mengevaluasi relasi dan interaksi antar paket. Parameter yang diperlukan jumlah pemanggilan antar kelas pada rilis tertentu, jumlah relasi pemanggilan baru dan jumlah relasi yang dihapus. Rumus CCI (Aversano, Molfetta, & Tortorella, 2013) dilihat pada Persamaan (4).

$$CDI = \frac{b + c}{m} \quad 3)$$

m = jumlah paket pada rilis ke-N

b = jumlah paket baru yang ditambahkan pada rilis ke-N+1

c = jumlah paket yang ada pada rilis ke-N tetapi hilang pada rilis ke N+1

$$CCI = \frac{x + y}{z} \quad 4)$$

z = jumlah relasi antar paket pada rilis ke N

x = jumlah relasi baru antar paket yang muncul pada rilis ke-N+1

y = jumlah relasi yang dihapus atau hilang dari rilis ke-N+1

Penelitian Aversano dikembangkan kembali oleh Constantinou *et all* (Constantinou & Stamelos, 2015). Penelitian ini mengusung sebuah metrik stabilitas baru yang memprioritaskan pada evaluasi *architectural core* berdasarkan nilai *fan-in*. Tujuan dari pengembangan Contantinou adalah mengukur seberapa banyak paket, kelas dan relasi antara elemen yang dapat dipakai kembali pada versi berikutnya. Metrik stabilitas yang dikemukakan memberikan pengukuran kedua hal yaitu: “External Stability” (ES) dan “Internal Stability” (IS).

Metrik External Stability menampilkan nilai dari arsitektural yang stabil pada pengukuran versi tertentu dengan versi setelahnya di satu sistem yang sama. Perumusan “External Stability” (Constantinou & Stamelos, 2015) terlihat pada persamaan (5), (6) dan (7).

$$ES(i, i+1) = ES_R(i, i+1) * ES_c(i, i+1) \quad 5)$$

$$ES_R(i,i+1) = 1 - \frac{\sum_{P_j \in PS_R(i,i+1)} |P_j|}{|S_i|} \quad 6)$$

$$ES_C(i,i+1) = 1 - \frac{\sum_{P_j \in PS_B(i,i+1)} |P_j(i)| - |P_j(i+1)|}{|S_i|}, \quad with |P_j(i)| > |P_j(i+1)| \quad 7)$$

$PS_R(i,i+1)$ adalah satu set paket yang dihapus pada versi ke-i dan tidak ada di versi ke-i+1

$PS_B(i,i+1)$ adalah satu set paket yang tetap ada baik di versi ke-i atau versi ke-i+1

$|P_j(i)|$ adalah jumlah dari kelas pada satu paket pada versi ke-i

$|S(i)|$ adalah jumlah seluruh kelas pada sistem pada versi ke-i

Metrik "Internal Stability" menampilkan nilai interaksi antara arsitektural yang terbentuk antar versi saat ini dengan versi setelahnya. Perumusan Internal Stability (Constantinou & Stamelos, 2015) terlihat pada persamaan (8), (9) dan (10).

$$IS(i,i+1) = \frac{\sum_{REL(P_j,P_k) \neq \emptyset} \frac{PS_A(i,i+1) + PS_R(i,i+1)}{2}}{|REL_i(P_j,P_k), REL_{i+1}(P_j,P_k) \neq \emptyset|} \quad 8)$$

$$PS_A(i,i+1) = 1 - \frac{|REL_{i+1}(P_j,P_k) \notin REL_i(P_j,P_k)|}{|REL_{i+1}(P_j,P_k)|} \quad 9)$$

$$PS_R(i,i+1) = 1 - \frac{|REL_i(P_j,P_k) \notin REL_{i+1}(P_j,P_k)|}{|REL_i(P_j,P_k)|} \quad 10)$$

$PS_R(i,i+1)$ adalah satu set paket yang dihapus pada versi ke-i dan tidak ada di versi ke-i+1

$PS_A(i,i+1)$ adalah satu set paket yang ditambahkan pada versi ke-i+1dan tidak ada versi ke-i

$REL_i(P_j,P_k)$ adalah satu paket dari hubungan antara elemen paket ke-j dan paket ke-k

Pengukuran stabilitas (Constantinou & Stamelos, 2015) dari persamaan (5) dan (8) digabungkan dalam persamaan (11) berikut ini.

$$Stability(i,i+1) = \frac{ES(i,i+1) + IS(i,i+1)}{2} \quad 11)$$

Tabel 2.1 menunjukkan analisis kritis terkait literatur-literatur untuk mengukur stabilitas arsitektur perangkat lunak.

Tabel 2. 1. Analisis Kritis Literatur terkait Metrik Stabilitas Rancangan Perangkat Lunak

Peneliti	Fokus	Metrik
Alshayeb	Pengukuran stabilitas dengan melihat keterkaitan dengan proses perancangan evolusi dengan metodologi XP	SDI
Aversano	Teknik pengukuran stabilitas dengan melihat komponen utama dengan tujuan penggunaan kembali	CDI dan CCI
Constantinou	Pengukuran stabilitas dengan melengkapi limitasi dari penelitian Aversano pada level *architectural core* dengan melihat parameter fan-in	ES dan IS

2.3. Pengukuran Volatilitas Kebutuhan

Pengukuran perubahan kebutuhan perangkat lunak dalam ISO 9126 digunakan untuk mengukur seberapa stabil suatu spesifikasi kebutuhan fungsional selama hidup perangkat lunak tersebut. Metrik yang digunakan berupa perhitungan jumlah fungsional yang berubah akibat penambahan, perubahan atribut atau penghapusan terhadap keseluruhan jumlah fungsional yang ada. Selain dari ISO 9126, Monperrus *et all.* (Monperrus, Baudry, Champeau, Hoeltzener, & Jezequel, 2013) memformulasikan pengukuran model dari kebutuhan secara otomatis. Mannaert *et all* (Mannaert, Verelst, & Ven, 2011) memperkenalkan bentuk permodelan *software primitives* yang berasal dari hasil transformasi kebutuhan untuk mendukung stabilitas dari evolusi perangkat tersebut.

Pada ISO 9126, terdapat pengukuran untuk menghitung tingkat volatility dari sebuah perubahan elemen perangkat lunak, baik kebutuhan, elemen implementasi ataupun kode. Nama Metrik pada ISO 9126 adalah “Functional Spesification Stability”. Metrik tersebut ditujukan untuk mengukur seberapa stabil kebutuhan fungsional selama proses daur hidup perangkat lunak. Persamaan dapat ditinjau pada persamaan (12). Namun untuk kepentingan pengukuran volatilitas perangkat lunak, maka rumus persamaan (12) dinegasikan dengan nilai 1 atau 100%.

$$X = 1 - \frac{A}{B} \qquad 12)$$

A= jumlah fungsi/fitur yang berubah (baik ditambahkan, dihapus atau diubah)

B= jumlah fungsi/fitur yang ada pada dokumen kebutuhan

2.4.Analisis Data Pengamatan

Penelitian adalah sebuah kegiatan untuk mengumpulkan informasi, memaparkan fakta, mendokumentasikan penemuan lapangan dan menganalisis informasi. Tujuan dari penelitian adalah untuk mengamati fenomena-fenomena yang terjadi sesuai dengan objek / lingkungan yang dimiliki sehinga dapat diambil kesimpulan baik dari sisi tren, kebiasaan fenomena dan berbagai hal lain. Penelitian memerlukan data fakta yang diolah sedemikan rupa sesuai dengan metode penelitian yang dipakai peneliti. Penelitian memiliki berbagai metode pendekatan data. Terdapat 3 pendekatan penelitian umum yang dikemukakan oleh Carrie Williams (Williams C. , 2007). Pendekatan itu yaitu pendekatan kualitatif, kuantitatif dan pendekatan campuran dengan kerangka berpikir masing-masing bidang ilmu.

Penelitian kuantitif telah diperkenalkan sejak abad 12. Dengan data kuantitatif ini mengajak para peneliti dan para investigasi untuk menemukan bidang keilmuan baru, sehingga mendapatkan rumus-rumus baku seperti rumus phitagoras dan rumus lainnya. Data penelitian kuantitatif berasal dari survei dan eksperimen sesuai dasar teori keilmuan masing-masing. Data yang tersaji dalam bentuk numerik atau statistik. Dengan penelitian dengan pendekatan hasil yang diharapkan untuk menilai penelitian dari sisi objektifitas produk yang diteliti.

Penelitian kualitatif berfokus pada pendekatan holistik dari obyek penelitian. Salah satu contohnya adalah meneliti fenomena sosial media diinvestigasi dari kebergunaan pengguna dalam menyampaikan pendapat. Penelitian tahap kualitatif ini merupakan lanjutan dari tahap penelitian kuantitatif dengan menitikberatkan pada deskripsi pada level yang lebih tinggi, sehingga penelitian kualitatif dapat membangun sebuah konsep teori baru.

2.5. Analisis Statistik

Analisis data statistik memiliki beberapa teknik. Data yang diperoleh dengan mengukur lebih dari satu variabel kriteria pada setiap anggota sampel disebut data multivariat. Teknik nalisis statistik yang memperlakukan sekelompok variabel kriteria yang saling berkorelasi sebagai suatu sistem dengan memperhitungkan korelasi antar variabel disebut Analisis Multivariat (Johnson &

Wichern, 2007). Tujuan dari analisis data multivariat adalah (1) menemukan dan menafsirkan struktur atau ciri yang mendasari data agar struktur atau ciri yang ditemukan bermakna; dan (2) mencari variabel baru yang jumlahnya lebih kecil tetapi mampu menghasilkan penjelasan tentang variasi dari variabel semua yang jumlahnya lebih banyak. Dalam analisis data multivariat terdapat 2 kelompok metode yaitu Kelompok "Metode Ketergantungan" dan "Metode Salingketergantungan". Kelompok "Metode Ketergantungan" memiliki sifat persoalan tentang hubungan antara 2 kelompok variabel yang satu sebagai variabel bebas dan yang lainnya sebagai variabel terikat. Sebaliknya kelompok "Metode Salingketergantungan" adalah kelompok metode yang variabel datanya tidak dibedakan antara variabel bebas dan terikatnya. Kelompok "Metode Ketergantungan" memiliki 5 teknik analisis sebagai berikut.

a. Analisis logit apabila data bertipe nominal atau ordinal.
b. Analisis varians apabila data bertipe nominal atau ordinal.
c. Analisis diskriminan apabila data bertipe interval atau rasio/ordinal.
d. Analisis regresi apabila data bertipe interval atau rasio.
e. Analisis korelasi apabila data bertipe interval atau rasio.

Sedangkan untuk kelompok "Metode Salingketergantungan" memiliki 4 teknik analisis sebagai berikut.

a. Analisis loglinear apabila data bertipe nominal atau ordinal
b. Analisis kelompok apabila data bertipe nominal atau ordinal/rasio
c. Analisis komponen Utama apabila data bertipe interval atau rasio
d. Analisis faktor apabila data bertipe interval atau rasio

Dari penjabaran di atas, penelitian ini menggunakan 2 analisis yaitu analisis korelasi dan regresi. Kedua analisis ini dibutuhkan untuk melihat tren pada stabilitas rancangan pada persentase volatilitas fitur dari sebuah perangkat lunak setiap versinya.

2.5.1. Analisis Korelasi

Pada statistik, metode pengukuran kebergantungan antar 2 variabel atau lebih yang dimodelkan dalam bentuk rasio adalah *Pearson Product-Moment* (PPM). PPM (Sarwono, 2009) menghasilkan rentang nilai $-1 \leq r \leq 1$, di mana

r adalah koefisien korelasi. Koefisien korelasi positif terbesar = 1 dan koefisien korelasi negatif terbesar adalah -1, sedangkan yang terkecil adalah 0. Bila besarnya antara dua variabel atau lebih itu mempunyai koefisien korelasi = 1 atau -1, maka hubungan tersebut sempurna. Rumus PPM yang digunakan terlihat pada persamaan (13) .

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum {x^2}_i - (\sum x_i)^2}\sqrt{n\sum {y^2}_i - (\sum y_i)^2}} \quad 13)$$

n = jumlah data

x_i= data set pertama

y_i= data set kedua

Menurut Sarwono, interval kekuatan hubungan korelasi terbagi atas 6 bagian seperti yang dijelaskan pada Tabel 2.2. Untuk korelasi bertanda negatif (-), interval kekuatan korelasi juga sama.

Tabel 2. 2. Interval Kekuatan Korelasi

Koefisien Korelasi	Kategorisasi
0	Tidak ada korelasi
$\mathbf{0} < r \leq 0{,}25$	Korelasi sangat lemah
$\mathbf{0{,}25} < r \leq 0{,}50$	Korelasi cukup
$\mathbf{0{,}50} < r \leq 0{,}75$	Korelasi kuat
$\mathbf{0{,}75} < r < 1$	Korelasi cukup kuat
1	Korelasi sempurna

Untuk melakukan analisis korelasi (Setiawan A. , 2010), sampel data berpasangan (x,y) berasal dari sampel data acak dan merupakan data kuantitatif. Lalu pasangan data tersebut harus merupakan distribusi normal, sehingga sebelum melakukan analisis koefisien korelasi perlu dilakukan uji normal kepada kedua data tersebut. Kedua syarat di atas dapat dicek secara visual dengan menggunakan diagram scatterplots. Bila tidak memenuhi uji normal, maka analisis yang digunakan adalah korelasi Spearman

Spearman-rank adalah metode untuk menentukkan besarnya koefisien korelasi jika data yang digunakan berskala ordinal. Mula-mula data yang diurutkan berdasarkan pesebaran data yang ada.

2.5.2. Analisis Regresi

Analisis Regresi mengukur hubungan satu variabel yang disebut sebagai variabel yang diterangkan dengan satu atau dua variabel yang menerangkan. Analisis regresi adalah salah satu analisis yang paling populer dan luas pemakaiannya. Analisis regresi dipakai secara luas untuk melakukan prediksi, tren, dan ramalan dengan penggunaan yang saling melengkapi. Analisis ini juga digunakan untuk memahami variabel bebas mana saja yang berhubungan dengan variabel terikat, dan untuk mengetahui bentuk-bentuk hubungan tersebut. Pada penelitian ini, analisis regresi dilakukan untuk melihat model perkembangan tingkat stabilitas dari rasio perubahan kebutuhan dalam suatu bentuk model matematika.

2.5.3. Uji Validitas Data

Dalam teori peluang dan statistik, validitas data perlu diperhitungan secara kuantitatif. Pada saat pengambilan data secara praktek lapangan tentu bisa ditemukan berbagai macam hal yang membuat data tersebut tidak valid. Pada bidang teknologi informasi, faktor kesalahan manusia mendominasi seperti kelalaian pengisi data, kesalahan perhitungan dan beberapa kesalahan teknis lain. Oleh karena itu, dari pengambilan data secara berulang tersebut perlu dijustifikasi apakah nilai yang terambil sudah valid. Koefisin variasi (Everitt, 1998) adalah perbandingan antara standar deviasi dengan nilai rata-rata data. Besar koefisien variasi berpengaruh pada kualitas sebaran data. Bila koefisien variansi kecil berarti data semakin homogen dan jika koefisien variansi besar berarti data semakin heteorgen. Persamaan KV dapat dilihat pada persamaan (14).

$$KV = \frac{S}{\bar{X}} \times 100\% \qquad 14)$$

KV = Koefisien varias

S= Simpangan Baku

$\bar{X}$= Rata-rata

2.5.4. Penentuan Jumlah Pengulangan untuk Penelitian

Menurut Gasperz (Gasperz, 1991) dan Sudjana (Sudjana, 1985), terdapat 2 jenis rancangan penelitian yaitu rancangan acak lengkap (RAL) dan rancangan

acak berkelompok (RAK). Perbedaan penggunaan RAL atau RAK adalah pada objek yang diteliti apakah pada setiap percobaan/ pengukuran mengalami perubahan. Bila meneliti suatu bahan pengan yang cepat busuk, penelitian cenderung menggunakan RAK, namun apabila mengukur suatu feasibilitas program (benda mati) maka bisa dilakukan RAL.

Penentuan rancangan penelitian ini berpengaruh pada jumlah pengulangan sebagai uji validasi data. Jumlah pengulangan dilakukan dengan 3 alasan yaitu: menyediakan penaksiran nilai toleransi kesalahan, mengurangi kesalahan, dan menyediakan taksiran lebih teliti. Secara sederhana penentuan jumlah pengulangan didasarkan pada derajat bebas dari masing-masing rancangan. Derajat ini merupakan konstanta baku yang dipergunakan peneliti pada dunia teknik pada umumnya. Untuk RAL memiliki derajat bebas sebesar 20 sedangkan RAK memiliki derajat bebas 15. Nilai pengulangan RAL dan RAK berdasarkan rumus persamaan (15) dan (16).

$$db_{RAL} = t\,(r-1) \tag{15}$$

$$db_{RAK} = (r-1)(t-1) \tag{16}$$

Keterangan:
db adalah derajat bebas dari masing-masing RAL atau RAK
t adalah jumlah perlakuan yang ada pada penelitian pada satu produk
r adalah jumlah repetisi atau pengulangan minimal yang dilakukan

[Halaman ini sengaja dikosongkan]

BAB 3
METODOLOGI PENELITIAN

3.1. Tahapan Penelitian

Pada bab ini dijelaskan Metodologi Penelitian. Langkah-langkah yang akan dilakukan pada penelitian ini adalah sebagai berikut: (1) studi literatur; (2) perancangan penelitian; (3) implementasi penelitian; (4) dan analisis; dan (5) dokumentasi penelitian seperti tertera pada Gambar 3.1

Gambar 3. 1. Tahap Metodologi Penelitian.

3.2. Studi Literatur

Dalam tahap studi literatur, dikaji berbagai referensi yang berkaitan dengan pengukuran stabilitas rancangan dan perkembangannya, teknik pengukuran dari perubahan kebutuhan pada setiap rilis dari produk dan hubungan analisis keterkaitan antara kedua elemen. Dari tahap ini, diharapkan dapat memberikan gambaran lengkap dan memberikan dasar kontribusi tentang hubungan relasi antara kebutuhan dan rancangan. Pada tahap ini studi literatur dilakukan secara sistematik (*Systematic Literature Review*). Literatur diambil dari jurnal dan konferensi internasional yang dipublikasi melalui portal *sciencedirect*, IEEE atau *springer*. Jurnal yang diambil dengan menggambil kosa-kata terkait pengukuran arsitektural perangkat lunak dan stabilitas arsitektur perangkat lunak. Studi literatur yang dilakukan yaitu mencari dan mempelajari beberapa referensi sebagai berikut.

1. Penelitian terdahulu yang terkait dengan evaluasi stabilitas perangkat lunak
2. Landasan teori dari aspek dari perancangan perangkat lunak, perancangan arsitektur dan metrik-metrik perancangan secara umum.
3. Metrik yang menganalisis pengukuran stabilitas perangkat lunak yang terkait dengan evolusi.

4. Metrik yang menganalisis pengukuran volatilitas kebutuhan perangkat lunak antar evolusi.
5. Landasan teori terkait pengujian hipotesis untuk analisis korelasi antara dua hingga lebih faktor.

3.3. Perancangan dan Implementasi

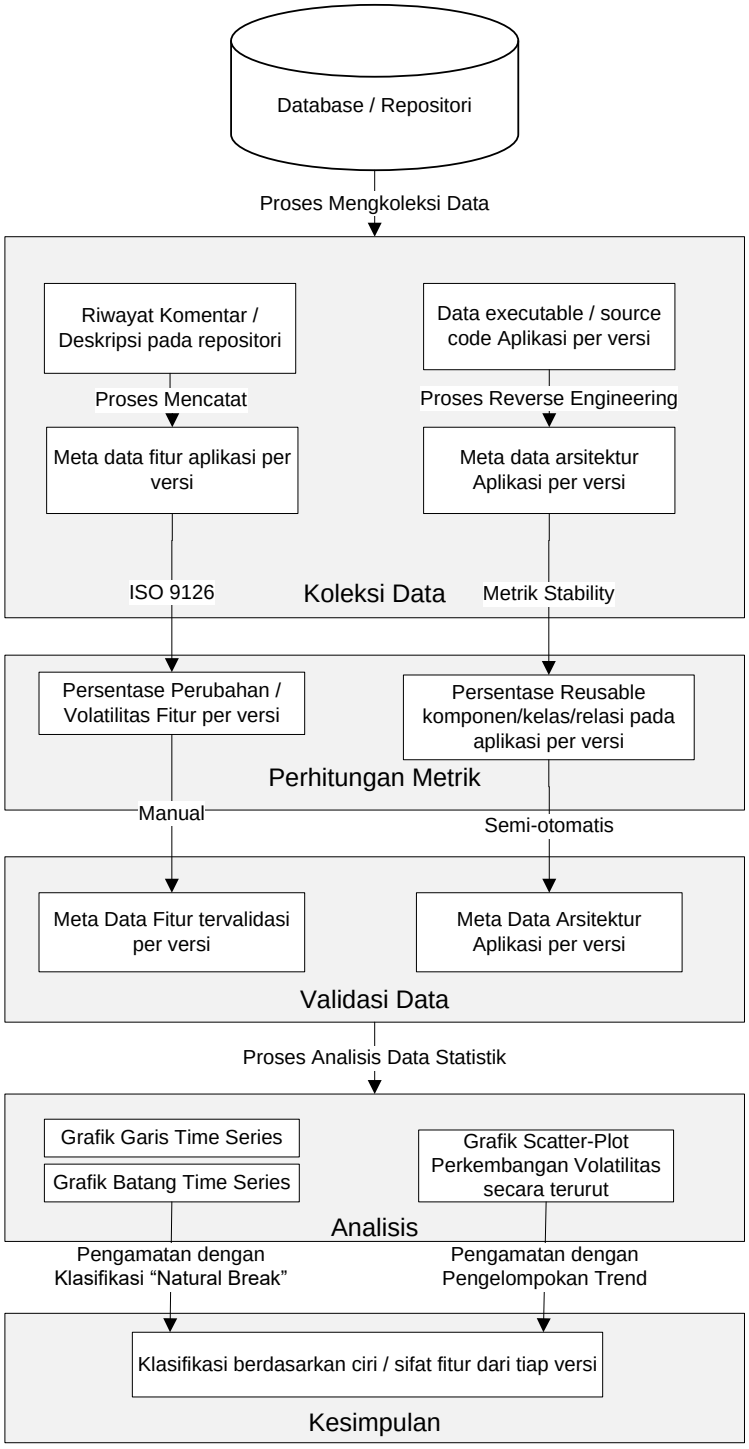
Rancangan penelitian ini dibuat dengan mengambil nilai volatilitas kebutuhan dan stabilitas rancangan pada perangkat lunak. Data awal penelitian adalah berupa kode sumber dari sebuah repositori yang diekstrak kelas, paket dan kebutuhan fitur di dalamnya. Secara keseluruhan, perancangan dan implementasi terbagi atas tiga bagian yaitu: (1) Tahap Koleksi Data; (2) Validasi Data; dan (3) Tahap Perhitungan Metrik. Alur penelitian dapat dilihat pada Gambar 3.2.

3.3.1. Koleksi Data

Data set yang digunakan dalam penelitian ini adalah dua buah program yang memiliki masing-masing memiliki versi lebih dari 5. Data set diperlukan cukup banyak versinya agar pergerakan data volatilitas dan stabilitas dapat terlihat. Data set merupakan program berbentuk aplikasi yang pernah dibuat oleh penelitian bersama tim terdahulu yang tersimpan dalam publik repositori. Data set tersebut dapat diakses pada Anaboost Official Repository[1] dan NastySteroid Official Repository[2].

Anaboost (AB) adalah proyek pertama yang digunakan pada penelitian ini. AB dibuat dalam kerangka kerja .NET 4.0 yang dibuat pada tahun 2012 hingga 2013. AB adalah kakas bantu untuk analisis kebutuhan hingga mampu membentuk diagram UML sebagai data pada tahap analisis. AB memiliki 10 versi. NastySteroid (NS) adalah proyek kedua yang digunakan pada penelitian ini. NS dibuat dalam kerangka kerja .NET 3.5 yang dibuat pada tahun 2011 hingga 2013. NS adalah kakas bantu untuk analisis model basis data hingga dapat dibangkitkan dalam bentuk berkas berekstensi (*.sql). NS memiliki 7 versi.

[1] https://sourceforge.net/projects/anaboost/
[2] https://sourceforge.net/projects/nastysteroid/



Gambar 3. 2. Tahap Rancangan dan Implementasi Penelitian.

Pembagian versi dari setiap proyek merupakan subyektifitas dari pengembang. Hal ini berarti pembagian versi tidak berdasarkan jumlah *commit*

selama proses pengembangan, namun rilis versi yang sudah ditetapkan oleh pengembang pada fase pengembangan terdahulu.

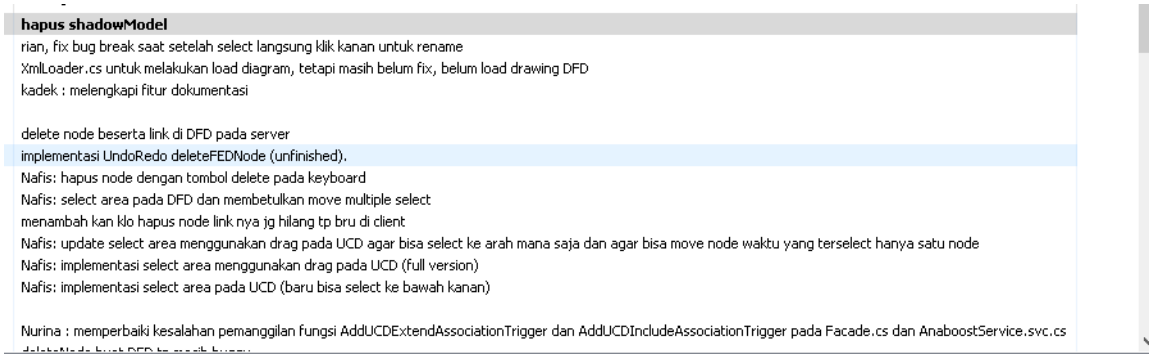
Data fitur diambil dari *browser Commit* (Gambar 3.3) dan *feature description* (Gambar 3.4) yang didapat dari dokumen fitur pengembang atau tertera pada laman resmi dari proyek. Data fitur pada masing-masing data set diambil setiap rilis dari versi. Pada proses pencatatan data fitur, peneliti memasukkan data ke dalam form yang memiliki 2 kolom yaitu: nama fitur dan status. Nama Fitur adalah fitur yang telah terimplementasi pada versi program tersebut. Status terbagi atas 3 bagian yaitu: ADD (A), MODIFY (M), REMOVE (R). Status ADD adalah status fitur bertambah dimana fitur pada versi n+1 tidak dimiliki pada versi n. Status MODIFY (M) adalah status fitur yang mengalami improvisasi, di mana fitur pada versi n+1 dimiliki pada versi n, tetapi ada perubahan tingkah laku atau mekanisme. Status REMOVE (R) adalah status fitur yang mengalami reduksi/penghapusan, di mana fitur pada versi n tidak ditemukan pada versi n+1. Tujuan adanya tiga identifikasi golongan ini adalah mempertajam analisis pengamatan pada penelitian ini.

Data arsitektur perangkat lunak diambil dari hasil rekayasa ulang/ *reverse engineering* dari produk proyek berupa kode sumber atau hasil eksekusi program. Dalam merekayasa ulang dilakukan secara semi-otomatis. Semi-otomatis adalah proses dilakukan dengan bantuan kakas bantu namun tidak menghasilkan meta data yang dapat dipakai pada proses penelitian. Peneliti memerlukan proses manual untuk pencatatan dari kakas bantu tersebut ke sebuah berkas. Kakas bantu yang digunakan peneliti adalah ILSpy[3].

ILSpy adalah kakas bantu yang digunakan untuk merekayasa ulang/ *decompiler* program yang dibuat dalam kerangka kerja .NET (Gambar 3.5). ILSpy memerlukan file masukan berupa file eksekusi atau file berekstensi *.dll. Dari file tersebut, ILSpy menampilkan dalam dua bentuk. Bentuk pertama adalah berbentuk pohon (disebut: *treeview*) yang berada di sebelah kiri. Informasi yang dapat diambil adalah referensi paket, paket, kelas, tipe kelas, nama fungsi, visibilitas fungsi, atribut dan visibilitas atribut. Bentuk kedua adalah berbentuk

[3] http://ilspy.net

editor tekstual yang berisi isi lebih rinci dari sebuah kelas, paket, relasi dan lain sebagainya. Bentuk kedua terletak di sebelah kanan dari bentuk pertama.



Gambar 3. 3. Ilustrasi Data Kebutuhan dari *Browser Commit* pada Repositori

Version	Release Date	Musician	Change Log	Announcement	DB Version
0.70	May 27, 2003		Changelog	Blog	
0.71	June 9, 2003		Changelog	Blog	
0.711	June 25, 2003		Changelog	Blog	
0.72	October 11, 2003		Changelog	Blog	
1.0	January 3, 2004	Miles Davis	Changelog	Blog	
1.0.1	January 25, 2004		Changelog	Blog	
1.0.2	March 11, 2004	Art Blakey	Changelog	Blog	
1.2	May 22, 2004	Charles Mingus	Changelog	Blog	
1.2.1	October 6, 2004		Changelog	Blog	
1.2.2	December 15, 2004		Changelog	Blog	2540
1.5	February 17, 2005	Billy Strayhorn	Changelog	Blog	2541
1.5.1	May 9, 2005		Changelog	Blog	2541

Gambar 3. 4. Ilustrasi Data Kebutuhan dari Laman Resmi Proyek

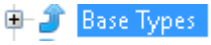
Untuk membentuk data set yang sesuai pada implementasi penelitian, peneliti melakukan pencatatan beberapa hal terkait hasil dari kakas bantu ILSpy. Pengamatan data tersebut dilakukan pada satu versi setiap proyek. Data yang diperlukan adalah sebagai berikut:

a. Nama Paket. Pada ILSpy terlihat ada simbol {}. Simbol itu merupakan paket dari proyek. Dalam bahasa teknis, simbol tersebut adalah *namespace*.
b. Nama Kelas yang ada di setiap paket. Pada ILSpy terdapat simbol . Simbol tersebut menunjukkan bahwa komponen yang dimaksud

adalah kelas dari sebuah paket. Abstrak kelas dan statik kelas juga dikategorikan ke dalam kelas.

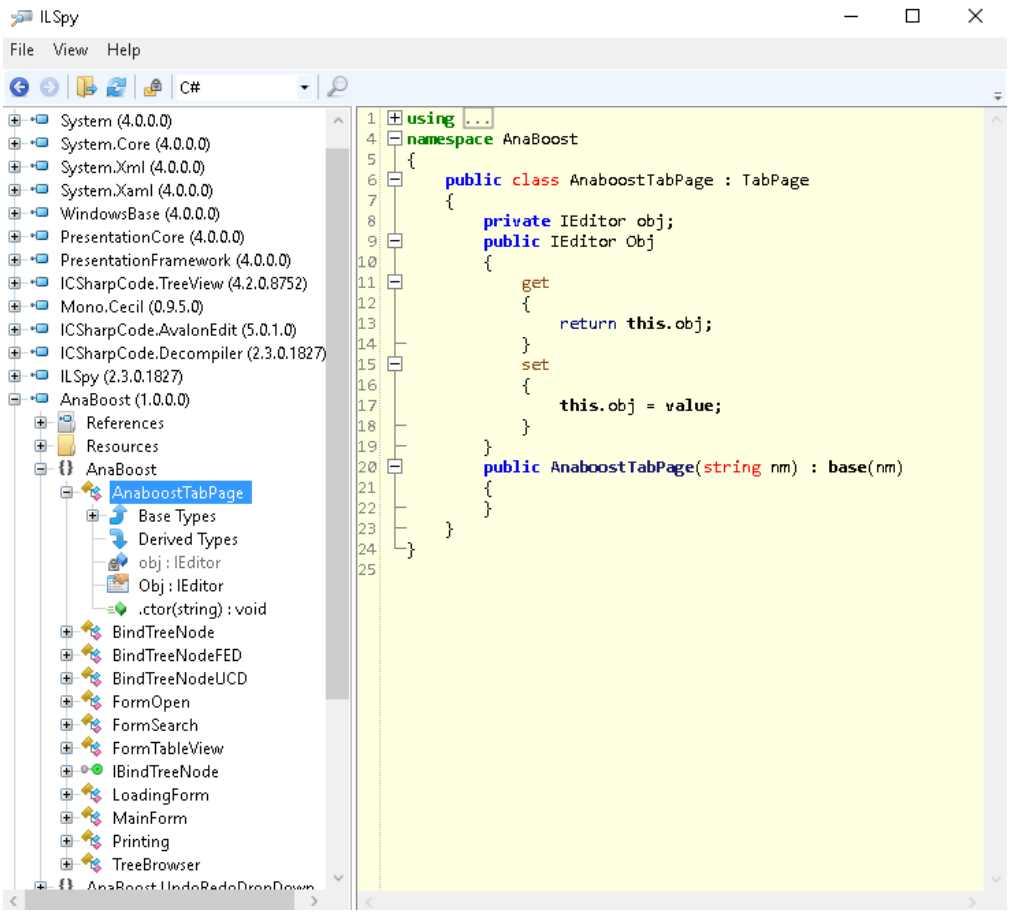
c. Nama Interface yang ada di setiap Paket. Interface diumpamakan sebagai kelas namun "interface" tidak memiliki atribut dan *body* dari fungsi. Simbol terlihat dengan gambar  .

d. Nama Atribut yang ada di setiap kelas. Atribut disimbolkan dalam bentuk  . Atribut yang memiliki visibilitas private, publik atau protected semua dicatat. Simbol gembok pada gambar tersebut memiliki arti bahwa visibilitasnya "private". Bila "public" maka simbol gembok akan hilang.

e. Nama base Kelas dari setiap kelas. Dalam ILSpy simbol seperti . Simbol 

pada aplikasi ILSpy menunjukkan kelas / interface apa saja yang diekstens oleh kelas tersebut.

f. Relasi antar kelas baik dalam paket atau antar paket. Relasi kelas dilihat dari atribut yang terdapat pada kelas tersebut. Relasi tersebut peneliti bagi menjadi 3 bagian yaitu: Relasi "Extend", Relasi "Composition", Relasi "Association". Relasi "Extend" adalah relasi saling mewariskan atribut antar kelas. Simbol sama seperti pada poin e. Relasi "Aggregation" dan Relasi "Composition" diperoleh dari pengamatan atribut pada setiap kelas. Perbedaan antara Relasi "Aggregation" dan Relasi "Composition" adalah waktu instansiasi awal. Instansiasi awal ini adalah proses pembuatan awal sebuah objek yang sudah dideklarasi sebelumnya. Dalam bahasa teknis digunakan sintak *new();*. Relasi dikatakan "Composition" apabila pada saat kelas tersebut diinstansiasi, atribut yang berelasi tersebut juga diinstansiasi awal. Dalam bahasa teknis, pada konstraktor atribut tersebut terdapat sintaks *new()*; Relasi dikatakan "Aggregation" apabila kelas tersebut memiliki atribut yang tidak diinstansiasi awal pada konstruktor dari kelas.



Gambar 3. 5. Gambar Kakas Bantu ILSpy

Form pencatatan data arsitektur terbagi atas dua yaitu formulir “Internal Stability” dan formulir “Eksternal Stability”. Form “Eksternal Stability” digunakan untuk menghitung tingkat penggunaan kembali (*reusability*) komponen dalam bentuk kelas dan paket. Form Interal Stability digunakan untuk menghitung tingkat penggunaan kembali relasi antar kelas. Form Eksternal Stability terbagi atas tiga kolom yaitu “Kode Paket”, “Nama Kelas” dan “Status Kelas”. “Kode Paket” adalah nama paket yang ada pada versi aplikasi. “Nama Kelas” adalah nama-nama kelas yang ada di dalam paket pada setiap versi aplikasi. “Status kelas” terbagi atas dua golongan yaitu: ADD (A) dan REMOVE (R). Status ADD adalah kelas yang ditemukan pada versi n+1 dan tidak ditemukan pada versi n. Status REMOVE adalah kelas yang ditemui pada versi n dan tidak ditemukan pada versi n+1.

Versi - i		Versi - J		
Kode Paket	Nama Class	Kode Paket	Nama Class	Status Class
k1	classA	k1	classA	A/R (ADD/REMOVE)
k2	classB	k2	classB	A/R (ADD/REMOVE)

(a)

Versi - i							Versi J					
P1		P2		REL	Sifat	Status	P1		P2		REL	Status
An	AnaboostTabPage	Me	IEditor	obj	Aggregation	A/R	An	Anaboost	Me	IEditor	obj	Aggregation
An	BindTreeNode	M	Process	node	Aggregation	A/R	An	BindTreeN	M	Process	node	Aggregation
						A/R	An	BindTreeN	MP	Process	node	Aggregation

(b)

Gambar 3. 6. Form Metadata (a) Fitur dan (b) Komponen Perangkat Lunak

3.3.2. Proses Hitung Metrik Volatilitas dan Metrik Stabilitas

Data fitur dan Data komponen yang sudah tervalidasi kemudian dilakukan perhitungan volatilitas dan perhitungan stabilitas. Perhitungan volatilitas dilakukan menggunakan metrik stabilitas fitur perangkat lunak.

Perhitungan stabilitas fitur kebutuhan sistem memberikan hasil antar 0% hingga 100%. Rumus perhitungan dapat dilihat pada bagian "Pengukuran Volatilitas Kebutuhan" pada subbab 2.3. Namun karena penelitian ini memberikan topik volatilitas maka hasil dari metrik stabilitas dinegasikan dengan nilai 100%, sehingga nilai 0% pada metrik volatilitas berarti tidak ada perubahan pada sisi fitur. Sebaliknya nilai 100% artinya bahwa sebagian besar atau bahkan semua fitur mengalami perubahan. Perubahan fitur ini berkaitan dengan adanya penambahan, pengurangan atau pengubahan isi dalam fitur.

Perhitungan stabilitas rancangan perangkat lunak dapat dilihat pula pada bagian "Pengamatan Literatur Terkait" pada subbab 2.2. Dari hasil perhitungan tersebut menghasilkan nilai 0% hingga 100%. Nilai 0% berarti hanya 0% elemen dari arsitektur perangkat lunak yang dapat digunakan kembali (*reusable*). Komponen perangkat lunak yang diukur pada penelitian ini adalah paket, kelas dan relasi antara kelas. Nilai 0% dapat diartikan pula kestabilan arsitektur perangkat lunak mengalami ketidakstabilan yang signifikan. Sebaliknya nilai 100% dapat diartikan bahwa 100% elemen dari arsitektur perangkat lunak yang

dapat digunakan kembali (*reusable*). Hal itu juga dapat diartikan bahwa tingkat stabilitas rancangan perangkat lunak tinggi.

3.3.3. Validasi Data

Validasi data adalah proses pemeriksaan data yang digunakan pada penelitian sehingga data yang digunakan pada penelitian ini memiliki keabsahan yang dapat dipertanggungjawabkan. Data validasi diambil dari jumlah pengulangan *reverse engineering* yang dilakukan peneliti bersama tim. Validasi ini dapat dilakukan oleh mahasiswa yang mengetahui ilmu komputer dan pemrograman berbasis objek atau sarjana teknik dengan ketentuan sudah dilakukan pelatihan atau pengantar ilmu kmputer. Validasi dilakukan oleh 3 orang dan dilakukan selama tiga bulan. Penentuan jumlah orang didasarkan pada penentuan rancangan penelitian yaitu RAL. Peneliti menggunakan RAL karena produk yang diteliti adalah homogen yaitu berupa berkas ekseuksi program dengan kakas bantu bernama ILSpy sebagai pendukung. Pada data AB dibutuhkan minimal tiga data pengulangan, sedangkan NS dibutuhkan minimal 4 data pengulangan sehingga peneliti memutuskan untuk semua data diulang sebanyak 4 kali. Validator Objek yang divalidasi terdapat 2 buah yaitu: data fitur dan data komponen perangkat lunak.

Setiap validator dilakukan pelatihan terlebih dahulu untuk menyelaraskan pola pikir dan hasil yang akan didapat. Validator terbagi atas 2 bagian yaitu: validasi data fitur dan validasi data arsitektur. Kelompok validasi data fitur akan diberikan hasil program yang dapat digunakan dan dokumen fitur. Kelompok validasi data komponen akan diberikan hasil program, kakas bantu ILSpy dan dokumen komponen.

Proses validasi dilakukan terkontrol oleh peneliti. Kontrol validasi dilakukan dengan bertatap muka langsung pada saat validasi berlangsung. Validasi dapat dilakukan masing-masing individu saat validator sudah memahami rinci mengenai validasinya.

Untuk menilai keabsahan kita menghitung persebaran data dari data-data yang diperoleh dari tiga kegiatan proses rekayasa ulang oleh tiga ahli tersebut. Perhitungan persebaran pada penelitian ini menggunakan perhitungan Koefisien Varian. Besarnya koefisien varian akan berpengaruh terhadap kualitas sebaran

data. Jika koefisien varian semakin kecil maka data semakin homogen, artinya nilai data hasil pengamatan memang telah valid. Batas yang peneliti pakai untuk dikatakan data dari hasil rekayasa ulang valid bila nilai koefisien varian di bawah 10%.

3.4. Analisis Pengamatan

Tujuan dari penelitian ini adalah mengamati data volatilitas fitur dengan tingkat stabilitas rancangan perangkat lunak. Perangkat lunak terdiri atas beberapa versi yang saling berkaitan antara satu versi dengan versi lain. Untuk mengamati fenomena volatilitas fitur dan stabilitas arsitektur, peneliti menyajikan dalam bentuk dua buah grafik yaitu grafik garis dan grafik plot. Grafik garis terjadi dengan dua buah garis yang saling terhubung. Garis itu menunjukkan pergerakan persentase volatilitas dan persentase stabilitas setiap versi pada sebuah perangkat lunak. Grafik plot menyajikan tren data perubahan volatilitas, dari volatilitas rendah hingga volatilitas tinggi, terhadap stabilitas rancangan pada setiap versi perangkat lunak.

BAB 4
HASIL dan EVALUASI

Pada bab ini dibagi menjadi tiga bagian yaitu implementasi penelitian, pemaparan hasil, dan analisis. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang digunakan untuk implementasi.

4.1. Implementasi Penelitian

Penelitian ini dibangun dalam lingkungan pengembangan sebagai berikut:

Sistem operasi : Windows 8 32 bit

RAM : 2 GB

Processor : Intel Core i7

Tools : Microsoft Excel 2013

4.2. Hasil Penelitian dan Pengujian

Pada subbab ini menjelaskan tentang hasil rekayasa ulang dan hitung parameter terkait pada setiap pengulangan, dan uji validasi dataset sebagai sumber data penelitian yang sah.

4.2.1. Hasil Rekayasa Ulang dan Hasil Pengukuran

Dataset yang digunakan adalah repositori hasil penelitian yang dilakukan oleh peneliti sebelumnya. Sumber repositori telah dijelaskan pada subbab yang terdahulu. Penelitian ini menggunakan 2 buah proyek yang memiliki versi lebih dari 4. Pada Tabel 4.1 menunjukkan deskripsi singkat proyek yang menjadi dat aset penelitian ini.

Tabel 4. 1. Rangkuman Data Set Program

Program	Jumlah versi	Dibuat pada
Anaboost	10	September 2012 – Januari 2013
NastySteroid	7	Oktober 2011 – Juli 2013

Proses pengambilan data komponen dan data fitur dilakukan 4 kali pengulangan. Pengulangan dilakukan pihak peneliti dan dibantu oleh rekan validator yang telah dilatih terlebih dahulu. Proses pengambilan masing-masing

data sesuai dengan deskripsi pada subbab perancangan penelitian. Hasil meta data fitur dan data arsitektur dari setiap pengulangan dapat dilihat pada lampiran 1.

Dari masing-masing metadata tersebut, peneliti menghitung tingkat volatilitas fitur dan tingkat stabilitas arsitektur dengan metrik ISO 9126 dan penelitian Constantinu. Tabel 4.2, Tabel 4.3, Tabel 4.4 dan Tabel 4.5 adalah hasil perhitungan volatilitas data Fitur dan stabilitas rancangan dari data komponen AB pada pengulangan pertama, kedua, ketiga dan keempat. Tabel 4.6, Tabel 4.7, Tabel 4.8 dan Tabel 4.9 adalah hasil perhitungan data Fitur dan rancangan komponen NS pada pengulangan pertama, kedua, ketiga dan keempat.

Tabel 4. 2. Volatilitas Fitur dan Stabilitas Rancangan AB pada pengulangan 1

Versi AB	Pengulangan 1							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	4	2	0	17	0.3529412	1	0.861111	0.930556
Versi 2-3	8	1	0	21	0.4285714	0.869565	0.22963	0.549597
Versi 3-4	3	3	4	29	0.3448276	1	0.358333	0.679167
Versi 4-5	10	2	3	28	0.5357143	0.888889	0.873214	0.881052
Versi 5-6	19	0	0	35	0.5428571	0.95082	0.529167	0.739993
Versi 6-7	9	0	0	55	0.1636364	1	0.589671	0.794835
Versi 7-8	3	0	0	61	0.0491803	1	0.485993	0.742996
Versi 8-9	10	0	0	65	0.1538462	1	0.832162	0.916081
Versi 9-10	15	4	1	75	0.2666667	1	0.230949	0.615475

Tabel 4. 3. Volatilitas Fitur dan Stabilitas Rancangan AB pada pengulangan 2

Versi AB	Pengulangan 2							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	4	2	0	17	0. 3529412	1	0.94444444	0.97222222
Versi 2-3	8	1	0	21	0.380952381	0.869565217	0.25396825	0.56176673
Versi 3-4	3	3	4	29	0.344827586	1	0.356349206	0.67817460
Versi 4-5	10	2	3	28	0.535714286	0.888888889	0.87321428	0.88105158
Versi 5-6	19	0	0	35	0.542857143	0.950819672	0.52916666	0.73999317
Versi 6-7	9	0	0	55	0.163636364	1	0.59325998	0.79662999
Versi 7-8	3	0	0	61	0.049180328	1	0.49400518	0.74700259
Versi 8-9	10	0	0	65	0.153846154	1	0.78817340	0.8940867
Versi 9-10	15	4	1	75	0.266666667	1	0.28823405	0.64411702

Tabel 4. 4. Volatilitas Fitur dan Stabilitas Rancangan AB pada pengulangan 3

Versi AB	Pengulangan 3							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	4	2	0	17	0. 3529412	1	0.944444	0.972222
Versi 2-3	8	1	0	21	0.380952381	0.869565217	0.285714286	0.577639752
Versi 3-4	3	3	4	29	0.344827586	1	0.362698413	0.681349207
Versi 4-5	10	2	3	28	0.535714286	0.888888889	0.681179138	0.785034014
Versi 5-6	19	0	0	35	0.542857143	0.950819672	0.537654321	0.744236997
Versi 6-7	9	0	0	55	0.163636364	1	0.579623617	0.789811809
Versi 7-8	3	0	0	61	0.049180328	1	0.442861387	0.721430694
Versi 8-9	10	0	0	65	0.153846154	1	0.770183983	0.885091992
Versi 9-10	15	4	1	75	0.266666667	1	0.271139601	0.635569801

Tabel 4. 5. Hasil Volatilitas Fitur dan Stabilitas Rancangan AB pada pengulangan 4

Versi AB	Pengulangan 4							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	4	2	0	17	0. 3529412	1	0.944444	0.972222
Versi 2-3	8	1	0	21	0.380952381	0.869565217	0.285714286	0.577639752
Versi 3-4	3	3	4	29	0.344827586	1	0.362698413	0.681349207
Versi 4-5	10	2	3	28	0.535714286	0.888888889	0.681179138	0.785034014
Versi 5-6	19	0	0	35	0.542857143	0.950819672	0.537654321	0.744236997
Versi 6-7	9	0	0	55	0.163636364	1	0.579623617	0.789811809
Versi 7-8	3	0	0	61	0.049180328	1	0.442861387	0.721430694
Versi 8-9	10	0	0	65	0.153846154	1	0.770183983	0.885091992
Versi 9-10	15	4	1	75	0.266666667	1	0.271139601	0.635569801

Tabel 4. 6. Hasil Volatilitas Fitur dan Stabilitas Rancangan NS pada pengulangan 1

Versi NS	Pengulangan 1							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	11	6	0	57	0.29824561	0.98709677	0.77479111	0.880943942
Versi 2-3	5	2	2	66	0.13636364	1	0.68301235	0.841506175
Versi 3-4	0	4	0	68	0.05882353	0.9955157	0.78947368	0.89249469
Versi 4-5	6	3	0	73	0.12328767	0.996	0.97312528	0.984562638
Versi 5-6	1	3	0	78	0.05128205	1	0.97312528	0.986562638
Versi 6-7	2	1	0	79	0.03797468	1	0.98600668	0.993003342

Tabel 4. 7. Hasil Volatilitas Fitur dan Stabilitas Rancangan NS pada pengulangan 2

Versi NS	Pengulangan 2							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	11	6	0	57	0.298245614	0.987096774	0.77479111	0.880943942
Versi 2-3	5	2	2	66	0.136363636	1	0.683012349	0.841506175
Versi 3-4	0	4	0	68	0.058823529	0.995475113	0.789473684	0.892474399
Versi 4-5	6	3	0	73	0.123287671	0.996	0.973125275	0.984562638
Versi 5-6	1	3	0	78	0.051282051	1	0.973125275	0.986562638
Versi 6-7	2	1	0	79	0.037974684	1	0.986006683	0.993003342

Tabel 4. 8. Hasil Volatilitas Fitur dan Stabilitas Rancangan NS pada pengulangan 3

Versi NS	Pengulangan 3							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	11	6	0	57	0.298245614	0.987096774	0.77479111	0.880943942
Versi 2-3	5	2	2	66	0.136363636	1	0.683012349	0.841506175
Versi 3-4	0	4	0	68	0.058823529	0.995515695	0.789473684	0.89249469
Versi 4-5	6	3	0	73	0.123287671	0.996	0.973125275	0.984562638
Versi 5-6	1	3	0	78	0.051282051	1	0.973125275	0.986562638
Versi 6-7	2	1	0	79	0.037974684	1	0.986006683	0.993003342

Tabel 4. 9. Hasil Volatilitas Fitur dan Stabilitas Rancangan NS pada pengulangan 4

Versi NS	Pengulangan 4							
	Volatilitas					Stabilitas		
	add	mod	remove	totalF	Volatilitas	ES	IS	Stabilitas
Versi 1-2	11	6	0	57	0.298245614	0.987096774	0.77479111	0.880943942
Versi 2-3	5	2	2	66	0.136363636	1	0.683012349	0.841506175
Versi 3-4	0	4	0	68	0.058823529	0.995515695	0.789473684	0.89249469
Versi 4-5	6	3	0	73	0.123287671	0.996	0.973125275	0.984562638
Versi 5-6	1	3	0	78	0.051282051	1	0.973125275	0.986562638
Versi 6-7	2	1	0	79	0.037974684	1	0.986006683	0.993003342

4.2.2. Uji Validitas DataSet

Untuk pengujian Validitas Data set, peneliti mengukur koefisien variasi. Hasil dari koefisien variasi data AB dan NS seperti terlihat pada Tabel 4.10 dan Tabel 4.11. Dari hasil uji validitas dengan koefisien variasi terlihat bahwa tingkat

toleransi kesalahan (*threshold*) pada penelitian ini di bawah 10%. Hasil tersebut menunjukkan pada penelitian ini data set yang digunakan penelitian ini sudah valid dan dapat dianalisis mendalam pada tahap berikutnya.

Pada tahap analisis, peneliti menggunakan data set yang memiliki nilai kesalahan mendekati atau di bawah dari nilai koefisien. Pada data AB dan NS peneliti menggunakan pada pengulangan ke-3 dan 4. Hal ini dapat dilihat dari Tabel 4.12 dan Tabel 4.13. Terdapat status TRUE dan FALSE. TRUE di sini adalah nilai volatilitas fitur dan nilai stabilitas rancangan berada di antara interval batas atas dan batas bawah dari persebaran data.

Tabel 4. 10. Koefisien Variasi AB (a) Volatilitas Fitur (b) Stabilitas Rancangan

Versi AB	**Analisis Volatilitas Fitur**				**KV**
	Pengulangan ke-				
	1	**2**	**3**	**4**	
Versi 1-2	0.353	0. 353	0. 353	0. 353	**0.00%**
Versi 2-3	0.429	0. 429	0. 429	0. 429	**0.00%**
Versi 3-4	0.345	0.344828	0.3448276	0.344827586	**0.00%**
Versi 4-5	0.536	0.535714	0.5357143	0.535714286	**0.00%**
Versi 5-6	0.543	0.542857	0.5428571	0.542857143	**0.00%**
Versi 6-7	0.164	0.163636	0.1636364	0.163636364	**0.00%**
Versi 7-8	0.049	0.04918	0.0491803	0.049180328	**0.00%**
Versi 8-9	0.154	0.153846	0.1538462	0.153846154	**0.00%**
Versi 9-10	0.267	0.266667	0.2666667	0.266666667	**0.00%**

(a)

Versi AB	**Analisis Stabilitas Rancangan**				**KV**
	Pengulangan ke-				
	1	**2**	**3**	**4**	
Versi 1-2	0.931	0.972	0.972222	0.972	**2.17%**
Versi 2-3	0.55	0.562	0.57764	0.578	**2.40%**
Versi 3-4	0.679	0.678	0.681349	0.681	**0.24%**
Versi 4-5	0.881	0.881	0.785034	0.785	**6.65%**
Versi 5-6	0.74	0.74	0.744237	0.744	**0.33%**
Versi 6-7	0.795	0.797	0.789812	0.79	**0.44%**
Versi 7-8	0.743	0.747	0.721431	0.721	**1.87%**
Versi 8-9	0.916	0.894	0.885092	0.885	**1.63%**
Versi 9-10	0.615	0.644	0.63557	0.636	**1.92%**

(b)

Tabel 4. 11. Koefisien Variasi NS (a) Volatiltias Fitur (b) Stabilitas Rancangan

Versi NS	Analisis Volatilitas Fitur				KV
	pengulangan ke-				
	1	2	3	4	
Versi 1-2	0.298245614	0.298245614	0.298	0.298	**0.0%**
Versi 2-3	0.136363636	0.136363636	0.136	0.136	**0.00%**
Versi 3-4	0.058823529	0.058823529	0.059	0.059	**0.00%**
Versi 4-5	0.123287671	0.123287671	0.123	0.123	**0.00%**
Versi 5-6	0.051282051	0.051282051	0.051	0.051	**0.00%**
Versi 6-7	0.037974684	0.037974684	0.038	0.038	**0.00%**

(a)

Versi NS	Analisis Stabilitas Rancangan				KV
	pengulangan ke-				
	1	2	3	4	
Versi 1-2	0.880943942	0.880943942	0.880943942	0.880943942	0.00%
Versi 2-3	0.841506175	0.841506175	0.841506175	0.841506175	0.00%
Versi 3-4	0.89249469	0.892474399	0.89249469	0.89249469	0.00%
Versi 4-5	0.984562638	0.984562638	0.984562638	0.984562638	0.00%
Versi 5-6	0.986562638	0.986562638	0.986562638	0.986562638	0.00%
Versi 6-7	0.993003342	0.993003342	0.993003342	0.993003342	0.00%

(b)

Tabel 4. 12. Hasil Keputusan Pemilihan Data AB yang Dianalisis

Versi AB	Analisis Nilai				Pengulangan			
	Volatiltias Fitur		Stabiltias Rancangan					
	Batas bawah	**Batas atas**	**Batas bawah**	**Batas atas**	**1**	**2**	**3**	**4**
Versi 1-2	*0.35294*	*0. 35294*	*0.94097*	*0.982639*	FALSE	TRUE	**TRUE**	**TRUE**
Versi 2-3	*0. 428571*	*0. 428571*	*0.55304*	*0.580277*	FALSE	TRUE	**TRUE**	**TRUE**
Versi 3-4	*0.344828*	*0.344828*	*0.67841*	*0.681609*	TRUE	FALSE	**TRUE**	**TRUE**
Versi 4-5	*0.535714*	*0.535714*	*0.77761*	*0.888479*	TRUE	TRUE	**TRUE**	**TRUE**
Versi 5-6	*0.542857*	*0.542857*	*0.73966*	*0.744565*	TRUE	TRUE	**TRUE**	**TRUE**
Versi 6-7	*0.163636*	*0.163636*	*0.78928*	*0.796268*	TRUE	FALSE	**TRUE**	**TRUE**
Versi 7-8	*0.04918*	*0.04918*	*0.71951*	*0.74692*	TRUE	FALSE	**TRUE**	**TRUE**
Versi 8-9	*0.153846*	*0.153846*	*0.88046*	*0.909711*	FALSE	TRUE	**TRUE**	**TRUE**
Versi 9-10	*0.266667*	*0.266667*	*0.62052*	*0.644842*	FALSE	TRUE	**TRUE**	**TRUE**

Tabel 4. 13. Hasil Keputusan Pemilihan Data NS yang Dianalisis

Versi NS	Analisis Nilai				Pengulangan			
	Volatiltias Fitur		Stabiltias Rancangan					
	Batas bawah	Batas atas	Batas bawah	Batas atas	1	2	3	4
Versi 1-2	*0.29824561*	*0.29824561*	*0.88094394*	*0.880943942*	TRUE	TRUE	TRUE	TRUE
Versi 2-3	*0.13636364*	*0.13636364*	*0.84150617*	*0.841506175*	TRUE	TRUE	TRUE	TRUE
Versi 3-4	*0.05882353*	*0.05882353*	*0.89247947*	*0.892499762*	TRUE	FALSE	TRUE	TRUE
Versi 4-5	*0.12328767*	*0.12328767*	*0.98456264*	*0.984562638*	TRUE	TRUE	TRUE	TRUE
Versi 5-6	*0.05128205*	*0.05128205*	*0.98656264*	*0.986562638*	TRUE	TRUE	TRUE	TRUE
Versi 6-7	*0.03797468*	*0.03797468*	*0.99300334*	*0.993003342*	TRUE	TRUE	TRUE	TRUE

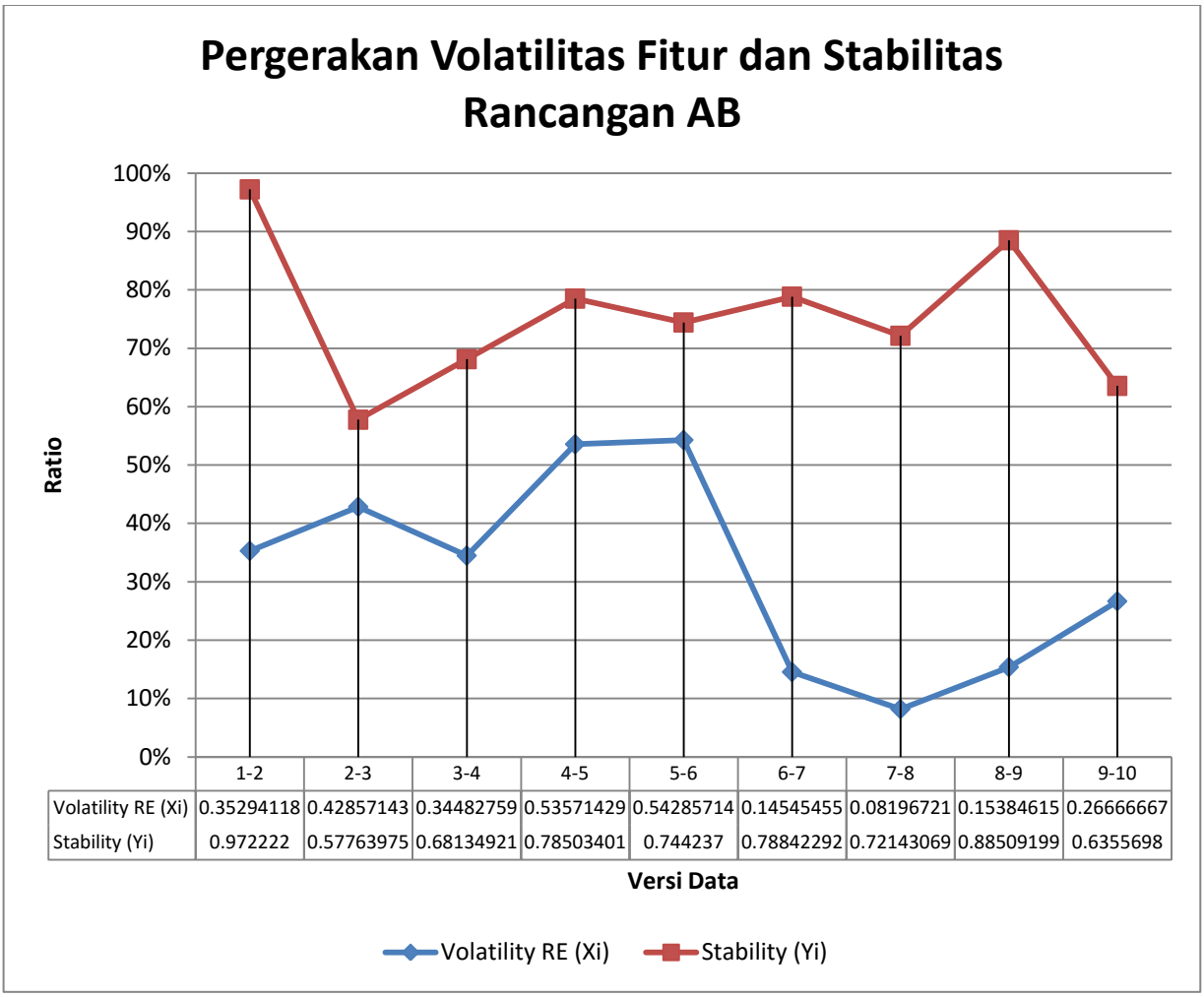
4.3. Analisis Hasil

Hasil dari penelitian ini tidak dapat disimpulkan menjadi satu nilai mutlak, apakah data volatilitas fitur mempengaruhi pada stabilitas arsitektur. Hal ini dilatarbelakangi hasil korelasi total data volatilitas dan hasil data stabilitas seperti data bila diukur secara bersamaan.

Dari Tabel 4.14 menunjukkan nilai korelasi pada data AB menujukkan angka -0.127. Nilai itu berarti korelasi data volatilitas fitur dengan data stabilitas arsitektur bila diukur secara bersamaan tidak memiliki korelasi yang kuat. Dari Gambar 4.1 juga menunjukkan tidak dapat ditarik kesimpulan langsung untuk menujukkan hasil yang mutlak.

Tabel 4. 14. Hasil Analisis Korelasi total data AB keseluruhan

Versi	Volatilitas	Stabilitas	"a"	"b"	a^2	b^2	a*b
Versi 1-2	0.352941176	0.972222	0.037581	0.217512415	0.001412335	0.047312	0.008174345
Versi 2-3	0.428571429	0.577639752	0.113211	-0.17706983	0.012816799	0.031354	-0.020046307
Versi 3-4	0.344827586	0.681349207	0.029467	-0.07336037	0.000868331	0.005382	-0.00216174
Versi 4-5	0.535714286	0.785034014	0.220354	0.030324429	0.048555956	0.00092	0.006682114
Versi 5-6	0.542857143	0.744236997	0.227497	-0.01047258	0.051754893	0.00011	-0.00238248
Versi 6-7	0.163636364	0.789811809	-0.15172	0.035102224	0.0230201	0.001232	-0.00532584
Versi 7-8	0.049180328	0.721430694	-0.26618	-0.03327889	0.070851685	0.001107	0.008858169
Versi 8-9	0.153846154	0.885091992	-0.16151	0.130382407	0.026086763	0.017	-0.02105858
Versi 9-10	0.266666667	0.635569801	-0.04869	-0.11913978	0.002371053	0.014194	0.005801328
Mean	*0.315360126*	*0.754709585*					
Total / SUM					0.237737915	0.11861	-0.1277
Koefisien Korelasi							**-0.1277**

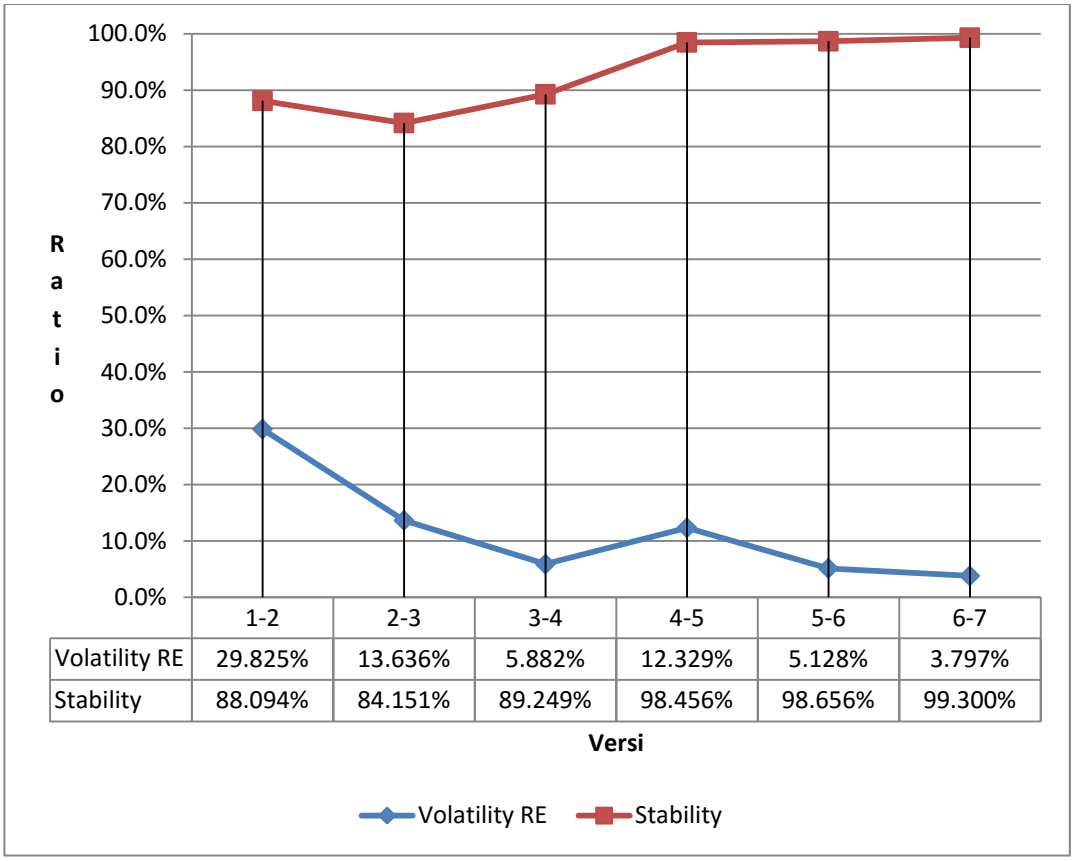


Gambar 4. 1. Grafik Pergerakan Volatilitas Fitur dan Stabilitas Rancangan AB

Dari Tabel 4.15 menunjukkan nilai korelasi pada data NS menujukkan angka -0.56371. Nilai itu memiliki arti korelasi data volatilitas fitur dengan data stabilitas rancangan bila diukur secara bersamaan memiliki korelasi yang cukup. Artinya tingkat volatilitas dengan stabilitas memiliki keterkaitan yang cukup kuat dan berkebalikan, di mana bila tingkat volatilitas fitur menurun, maka tingkat stabilitas meningkat. Dari pengamatan korelasi dan Gambar 4.2 juga menunjukkan bahwa terdapat hal-hal atau faktor-faktor yang menyebabkan adanya ketidakkorelasian antara volatilitas fitur dengan stabilitas rancangan.

Tabel 4. 15. Hasil Analisis Korelasi total data NS keseluruhan

Variansi Versi	Volatility RE (Xi)	Stabilitas (Yi)	"a"	"b"	a2	b2	a*b
1-2	31.579%	88.094%	0.185222149	-0.0489016	0.034307244	0.0023914	-0.0090577
2-3	13.636%	84.151%	0.005796312	-0.0883394	3.35972E-05	0.0078038	-0.000512
3-4	10.448%	89.249%	-0.026089713	-0.0373509	0.000680673	0.0013951	0.0009745
4-5	13.636%	98.456%	0.005796312	0.0547171	3.35972E-05	0.002994	0.0003172
5-6	5.195%	98.656%	-0.078619273	0.0567171	0.00618099	0.0032168	-0.0044591
6-7	3.846%	99.300%	-0.092105786	0.0631578	0.008483476	0.0039889	-0.0058172
Means	*0.130567325*	*0.92984557*			*0.008286596*	*0.0036317*	*-0.003092*
					Korelasi		**-0.56371**



Gambar 4. 2. Grafik Pergerakan Volatilitas Fitur dan Stabilitas Rancangan NS

Analisis data tren volatilitas terhadap stabilitas diukur dengan analisis regresi. Dalam uji regresi beberapa parameter yang perlu diperhatikan untuk menentukkan apakah model dari regresi bisa dipakai untuk melihat tren dan

peramalan. Ada 2 parameter utama yaitu uji signifikasi dengan anova dan "R-Square". Uji signifikasi digunakan untuk melihat tingkat signifikan variabel bebas dan variabel terikat. Uji signifikasi dinyatakan signifikan bila taraf lebih kecil dari 5% (<5%) dan "R-Square" yang diharapkan mendekati nilai 1 (dalam persentase 100%).

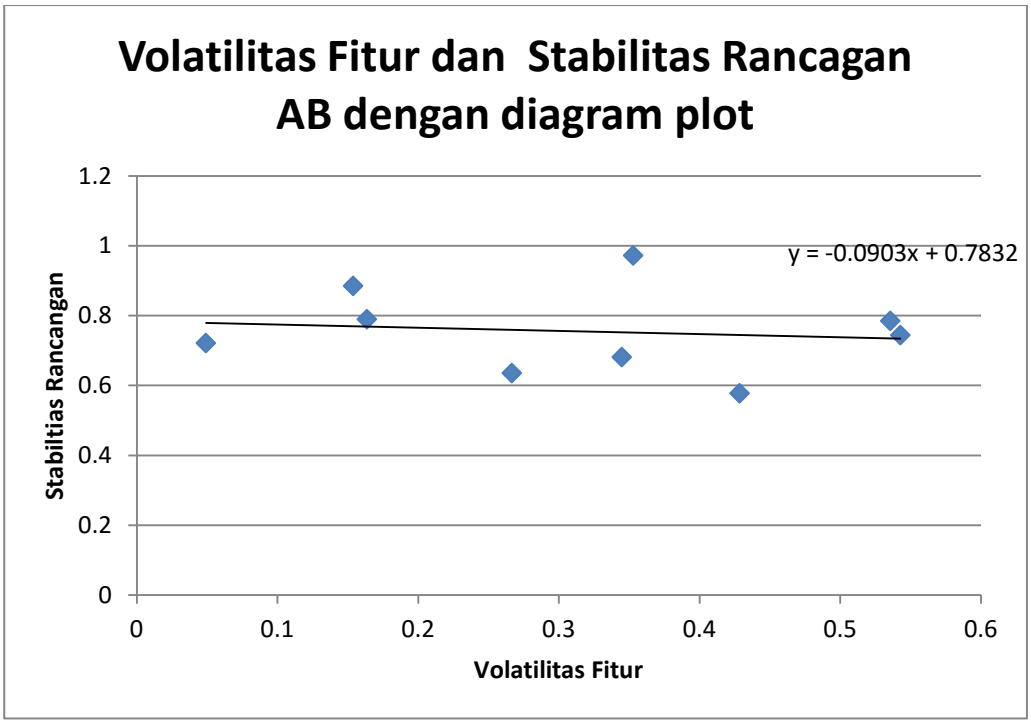
"R-Square" pada AB (Tabel 4.15) menunjukkan nilai 1,6%, berarti stabilitas rancangan tidak independen dengan variabel volatiltias fitur. "R-Square" mendekati 100% maka faktor variabel bebas bisa dipastikan independen dan memiliki korelasi sehingga fit sebagai data regresi. Uji signifikasi bernilai 74.3% melebihi 5% dari batas. Gambar 4.3 menunjukkan visualisasi dengan grafik plot untuk pergerakan volatilitas dari nilai terendah hingga tertinggi. Bagian ini tidak menunjukkan runtutan waktu, tetapi menghasilkan nilai tren dari pergerakan volatilitas fitur dengan pergerakan stabilitas rancangan pada data AB.

Tabel 4. 16. Hasil Analisis Korelasi Total Data AB keseluruhan

Regression Statistics	
Multiple R	0.127790864
R Square	**0.016330505**
Adjusted R Square	-0.124193709
Standard Error	0.129102904
Observations	6

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.001937	0.001937	0.116211	**0.743183419**
Residual	7	0.116673	0.016668		
Total	8	0.11861			



Gambar 4. 3. Grafik Pergerakan Volatilitas Fitur dan Stabilitas Arsitektur AB

R-Square pada NS (Tabel 4.16) menunjukkan nilai 1%, berarti stabilitas rancangan belum tentu independen dengan variabel volatiltias fitur. R-Square mendekati 100% maka faktor variabel bebas bisa dipastikan independen dan memiliki korelasi sehingga fit sebagai data regresi. Gambar 4.4 menunjukkan visualisasi dengan grafik plot untuk pergerakan volatilitas dari nilai terendah hingga tertinggi. Bagian ini tidak menunjukkan runtutan waktu, tetapi menghasilkan nilai tren dari pergerakan volatilitas fitur dengan pergerakan stabilitas arsitektur pada data NS.

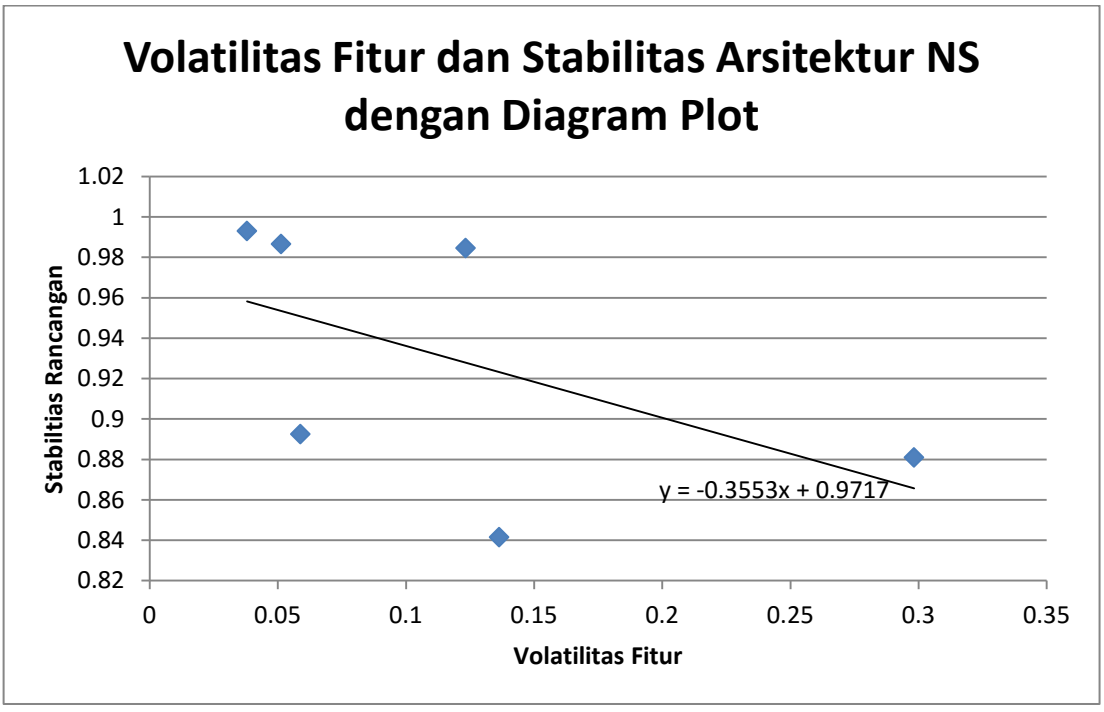
Dari analisis tren data volatilitas tingkat rendah ke tingkat lebih tinggi, hasil stabilitas arsitektur pada masing-masing data tidak menunjukkan regresi yang baik karena memiliki taraf significance F yang melebihi dari taraf toleransi (lebih dari 5%) dan R-square terlampau kecil. Idealnya R-square mendekati nilai 1. Oleh karena itu, untuk analisis data peneliti melakukan observasi lebih dalam pada faktor fitur yang berpengaruh. Peneliti mengamati satu persatu versi data dan mengamati sifat perubahan kebutuhan di dalamnya.

Tabel 4. 17. Hasil Analisis Korelasi Total Data NS keseluruhan

Regression Statistics	
Multiple R	0.570858
R Square	**0.325878**
Adjusted R Square	0.157348
Standard Error	0.062514
Observations	6

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.007557	0.007557	1.933648	**0.236728624**
Residual	4	0.015632	0.003908		
Total	5	0.023189			



Gambar 4. 4. Grafik Pergerakan Volatilitas Fitur dan Stabilitas Rancangan NS

.

4.3.1. Pengamatan Mendalam AB_{1-2}

Pada pengamatan AB_{1-2} terdapat volatilitas fitur 35,29% dengan rincian adanya 4 penambahan, 2 pengubahan dan tidak ada penghapusan fitur dari 17 fitur awal. Stabilitas rancangan cenderung stabil pada angka 97,5% dengan

rincian tingkat penggunaan kembali pada komponen sebesar 100% dan tingkat penggunaan kembali pada relasi antar komponen sebesar 94%. Dari pengamatan tersebut dapat disimpulkan bahwa dengan volatilitas sebesar 35,3% rancangan pada komponen perangkat lunak masih cenderung stabil. Perubahan pada rancangan terjadi pada perubahan relasi antar paket atau interpaketnya.

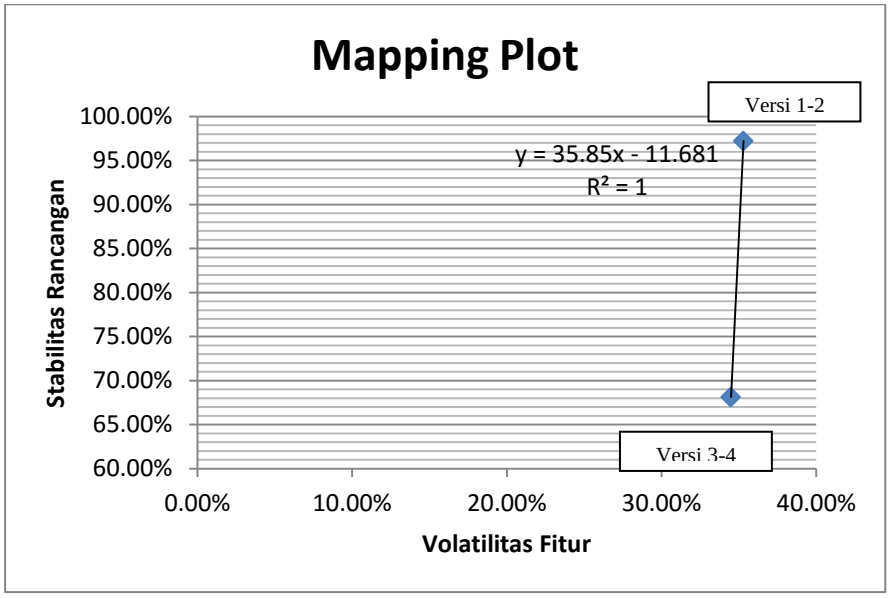
4.3.2. Pengamatan Mendalam AB_{2-3}

Pada pengamatan AB_{2-3} terdapat volatilitas 42,86% dengan rincian adanya 8 penambahan, 1 pengubahan dan tidak ada penghapusan fitur dari 21 fitur awal. Volatilitas pada versi ini mengalami penurunan sejumlah 21,43% daripada versi $AB_{1-2.}$ Stabilitas arsitektur pada AB_{2-3} mengalami ketidakstabilan pada angka -40,59% menjadi 57,76% dengan rincian penggunaan kembali pada elemen menurun menjadi 86,95% dan tingkat penggunaan kembali menurun menjadi 28,57%. Data pengamatan tersebut memberikan kesimpulan bahwa bila volatilitas fitur mengalami kenaikan, maka tingkat kestabilan rancangan cenderung mengalami penurunan. Namun pada fase ini mengalami keunikan karena stabilitas rancangan mengalami penurunan dratis dari volatilitas perangkat lunak. Dari pengamatan mendalam, peneliti menemukan beberapa faktor yaitu: (1) adanya penambahan modul sistem, pada kasus fase ini penambahan Diagram FishEye; dan (2) adanya pengikisan/reduksi kelas dalam paket "ProcessView" sebagai akibat adanya penambahan modul sistem. Reduksi ini terjadi karena pada versi yang lalu hanya diagram yang ada dalam sistem adalah "Diagram DFD", namun pada versi ke-3 diagram dikembangkan menjadi DFD dan FishEye. Oleh karena itu, kelas bernama "TreeBrowser" dan kelas bernama "IProcessViewMediator" mengalami perubahan posisi dari paket "ProcessView" menjadi paket "Anaboost" yang lebih global.

4.3.3. Pengamatan Mendalam AB_{3-4}

Pada pengamatan AB_{3-4} terdapat volatilitas 34,5% dengan rincian adanya 3 penambahan, 3 pengubahan dan 4 penghapusan fitur dari 29 fitur awal. Volatilitas fitur pada versi ini mengalami penurunan daripada versi AB_{2-3} sebanyak -19,54%. Stabilitas rancangan pada AB_{3-4} mengalami peningkatan +17,95% menjadi 68,13% dengan rincian 100% tingkat penggunaan kembali pada elemen dan 36,26% tingkat penggunaan kembali pada relasi. Namun peningkatan

ini tidak mengalami peningkatan signifikan hingga dua kali lipat. Persentase stabilitas rancangan dibanding dengan rancangan yang memiliki volatilitas hampir sama yaitu AB_{1-2} (selisih volatilitas fitur AB_{1-2} dan AB_{3-4} adalah 2,5%) juga cukup jauh yaitu -29,92% seperti terlihat pada Gambar 4.5. Penurunan berkaitan dengan adanya 4 penghapusan fitur yaitu "Editor moves Process", "Editor moves DataStore", "Editor moves External Entity" dan "System disables pallete if Use Case Diagram is active". Fitur pertama yang disebutkan adalah fitur yang telah diimplementasi pada versi-2 dan versi-3, sedangkan tiga fitur yang lain diimplementasi pada versi-3. Selain alasan tersebut, terdapat alasan lainnya yaitu 3 fitur yang disebut di awal yaitu "Editor moves" diimplementasi lagi pada versi ke-5, sehingga jumlah fitur yang terimplementasi dari total versi ke-1 hingga 10 adalah 8 buah versi. Karena terdapat fitur yang dihapus dari versi yang lebih terdahulu (hingga 2 versi sebelumnya) dan fitur tersebut memiliki bobot besar pada 10 versi sistem, maka dapat disimpulkan bahwa kestabilan dari rancangan perangkat lunak mengalami penurunan.

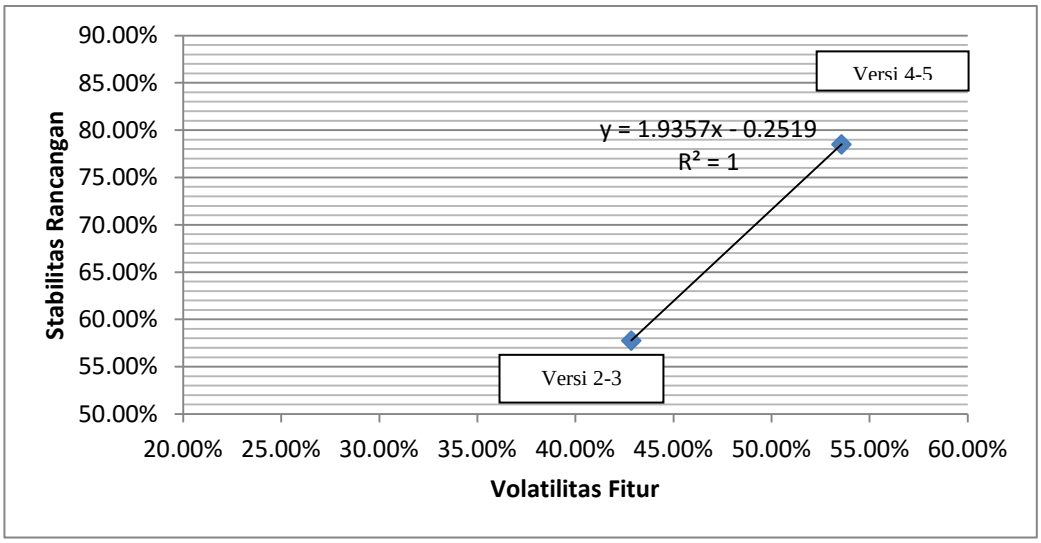


Gambar 4. 5. Grafik Perbandingan Versi (1-2) dengan Versi (3-4)

4.3.4. Pengamatan Mendalam AB_{4-5}

Pada pengamatan AB_{4-5} terdapat volatilitas 53,6% dengan rincian adanya 10 penambahan, 2 pengubahan dan 3 penghapusan fitur dari 28 fitur awal. Volatilitas AB_{4-5} mengalami kenaikan sebesar +55,36% dari versi AB_{3-4}. Bila

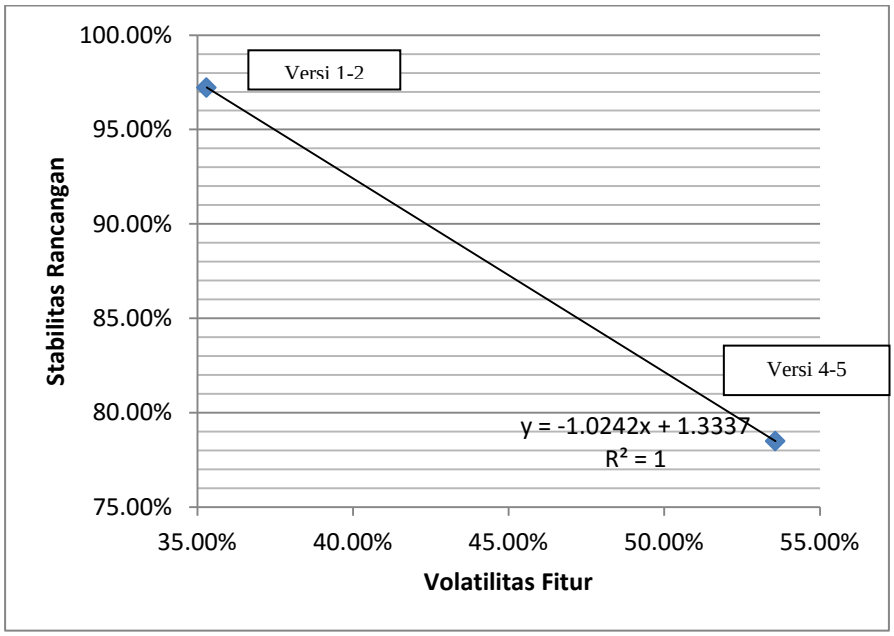
selisih stabilitas diukur dari volatilitas yang paling dekat, AB_{2-3} yang memiliki selisih dengan AB_{4-5} 25.00%, adalah +35,9%. Fenomena ini (Gambar 4.6) adalah anomali karena volatilitas fitur bertambah besar sedangkan stabilitas rancangan semakin meningkat. Hal ini disebabkan karena versi AB_{2-3} merupakan fenomena yang dikecualikan karena pada versi tersebut terdapat perubahan fitur yang besar yaitu penambahan lingkup perangkat lunak yang menyebabkan pengikisan dan pengkisian pada elemen paket "ProcessView". Peneliti menyimpulkan perbandingan anomali perbandingan AB_{2-3} dan AB_{4-5} tidak dapat dijadikan acuan perbandingan yang valid.



Gambar 4. 6. Grafik Perbandingan Versi (2-3) dengan Versi (4-5) yang merupakan anomali

Peneliti kemudian membandingkan AB_{4-5} dengan AB_{1-2} (Gambar 4.7) di mana selisih volatilias adalah terpaut jauh +51.79% dan stabilitas arsitektur adalah -19.25%. Dari perbandingan dua unit uji ini menunjukkan bahwa semakin banyak perubahan berpengaruh pada penurunan stabilitas arsitektur. Penurunan besar ini juga diakibatkan adanya tiga penghapusan fitur yaitu "System generates name of Process automatically", "System generates name of Data Store automatically, dan "System generates name of External Entity automatically". Data ini tidak diimplementasi pada pasca versi 5 dan hanya memiliki jumlah implementasi 4 versi dari 10 versi yang diteliti. Peneliti berargumen penghapusan 4 fitur ini memiliki sifat minor sehingga penghapusan pada versi 5 ini tidak

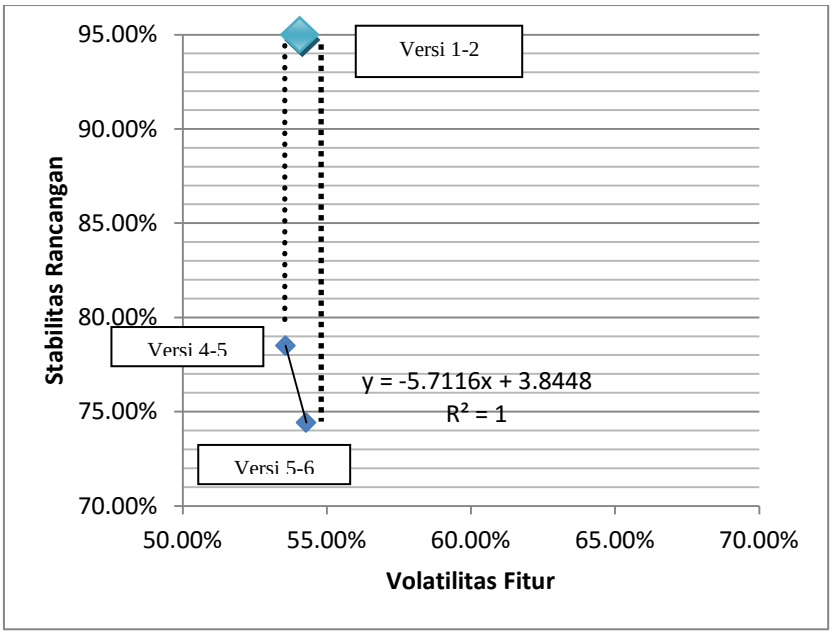
seberapa besar dengan penghapusan versi 4 sehingga penurunan stabilitas rancangan tidak sebanyak AB_{1-2} dan AB_{3-4}.



Gambar 4. 7. Grafik Perbandingan Versi (1-2) dengan Versi (4-5)

4.3.5. Pengamatan Mendalam AB_{5-6}

Pada pengamatan AB_{5-6} terdapat volatilitas 54,3% dengan rincian adanya 19 penambahan, tidak ada pengubahan dan tidak ada penghapusan fitur dari 35 fitur awal. Kita bandingan volatilitas yang mendekati AB_{5-6} yaitu pada AB_{4-5}. Perubahan volatilitas AB_{4-5} dengan AB_{5-6} adalah +1.33% sedangkan perbedaan stabilitas adalah -5.20% yang bernilai stabilitas 74.4%. Namun ada yang menarik dari perbandingan kedua unit data ini. Berdasarkan sifatnya AB_{4-5} dan AB_{5-6} memiliki faktor volatilitas fitur yang berbeda. Volatilitas AB_{4-5} terjadi akibat adanya penghapusan fitur yang bersifat minor dan tidak ada penambahan lingkup kebutuhan, sedangkan pada AB_{5-6} perubahan fitur terjadi akibat adanya penambahan lingkup yang cukup besar yaitu penambahan diagram "Use Case". Dari pernyataan tersebut, asumsi peneliti awal hingga pengamatan AB_{5-6} adalah penambahan lingkup yang besar memberikan dampak lebih besar terhadap ketidakstablian rancangan daripada penghapusan minor, seperti terlihat pada Gambar 4.8. Namun peneliti perlu pembuktian lebih dalam dengan membandingkan faktor volatilitas yang sejenis seperti pada AB_{2-3}.



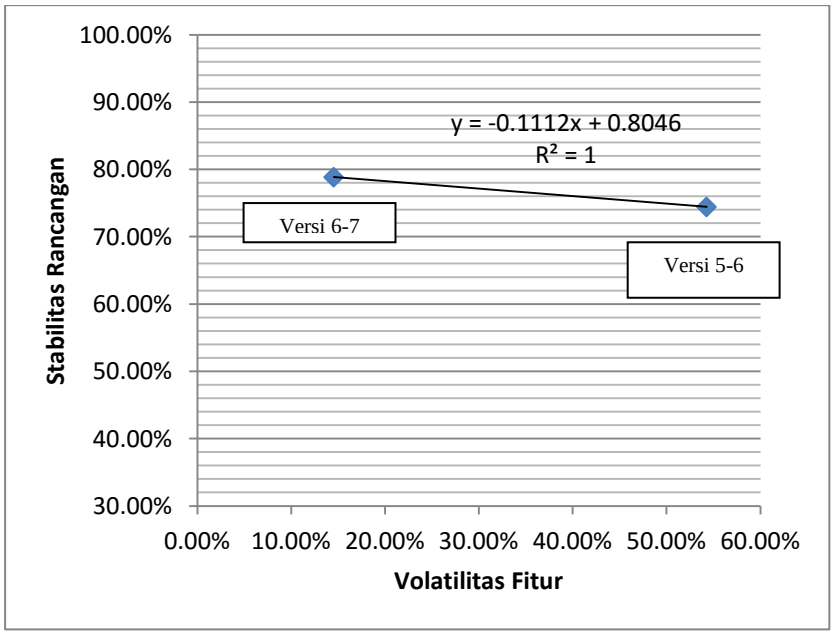
Gambar 4. 8. Grafik Perbandingan Versi (4-5) dengan Versi (5-6) terhadap versi (1-2)

Peneliti kemudian membandingkan AB_{5-6} dengan AB_{2-3} yang memiliki faktor volatilitas sama yaitu adanya penambahan skope komponen diagram dan tidak ada penghapusan . Volatilitas penambahan pada AB_{2-3} sebesar 8 fitur dari 21 total fitur (sekitar 38%) dengan sebagian besar penambahan untuk diagram "FishEye". Volatilitas penambahan AB_{5-6} sebesar 19 fitur dari 35 total fitur (sekitar 54%) dengan sebagian besar penambahan untuk diagram "UseCase". AB_{5-6} memiliki perubahan fitur penambahan lebih besar daripada AB_{2-3}. Namun dari sisi stabilitas arsitektur AB_{2-3} lebih rendah dari AB_{5-6} dengan nilai perbandingan 57,76% : 74,42%. Hal ini dikarenakan penambahan skope bergantung pula pada keputusan rancangan arsitektur. Pada AB_{5-6} tidak mengalami pengikisan elemen atau relasi terlampau besar karena sebagian besar kelas telah digeneralisasi ke lingkup paket yang lebih global. Tidak ada kasus seperti AB_{2-3} yang mengalami relokasi kelas pada paket "ProcessView" yang berpindah ke paket yang lebih global yaitu "Anaboost". Keputusan rancangan yang dimaksud pada pernyataan ini adalah keputusan rancangan kelas yang diletakkan pada paket tertentu yang sesuai.

4.3.6. Pengamatan Mendalam AB_{6-7}

Pada pengamatan AB_{6-7} terdapat volatilitas 14,5% dengan rincian adanya 8 penambahan, tidak ada pengubahan dan tidak ada penghapusan fitur dari 55 fitur awal. Volatilitas fitur pada versi ini hanya penambahan fitur untuk memenuhi beberapa kebutuhan yaitu: "DFD–DragDrop pada TreeBrowser", "FED-Balanced Node Algorithm" dan "Undo Redo". Penambahan fitur ini berkisar pada lingkup DFD dan FED namun ada penambahan kebutuhan yang perlu diimplementasi. Stabilitas yang diperoleh dari AB_{6-7} adalah 78,84%.

Bila dibanding dengan faktor volatilitas yang sejenis seperti pada AB_{5-6}, yang memiliki 54,28%, AB_{6-7} memiliki nilai volatilitas jauh lebih rendah dari AB_{5-6}, namun nilai tingkat penggunaan kembali tidak memiliki selisih jauh (seperti pada Gambar 4.9). Stabilitas AB_{6-7} dan AB_{5-6} memiliki perbandingan 78.88% : 74,42%. Dari pengamatan tersebut, penambahan satu butir fitur yang mewakili satu butir kebutuhan memiliki dampak cukup signifikan untuk membuat rancangan tidak stabil. Hal ini dapat dibuktikan dengan adanya penambahan paket pada AB_{6-7} untuk mengimplementasikan fungsi "Undo Redo" yaitu paket "ProcessView.Command", penambahan banyak kelas pada paket "UseCaseView" dan penambahan banyak kelas pada paket "Mechanism". Faktor jumlah paket dan jumlah kelas dalam paket yang sudah lebih besar menjadi bahan pertimbangan.



Gambar 4. 9. Grafik Perbandingan Versi (6-7) dengan Versi (5-6)

4.3.7. Pengamatan Mendalam AB_{7-8}

Pada pengamatan AB_{7-8} terdapat volatilitas 8,2% dengan rincian adanya 5 penambahan, tidak ada pengubahan dan tidak ada penghapusan fitur dari 61 fitur awal. Data AB_{7-8} ini memiliki ciri yang mirip dengan $AB_{6-7.}$ Kesamaan terjadi karena volatilitas fitur memiliki nilai yang kecil namun nilai stabilitas arsitektur berada pada nilai 70-80%. Stabilitas tidak dapat menjangkau nilai stabilitas di atas 80%. Stabilitas pada AB_{7-8} berada pada nilai 72,124% sedangkan AB_{6-7} berada pada 78,84%. Hasil pada perbandingan AB_{6-7} dan AB_{7-8} adalah semakin bertambahnya nilai volatilitas semakin stabil. Hal ini merupakan kasus pengecualian karena 5 fitur yang ditambahkan merepresentasikan 4 fungsional besar dan 1 fungsional pelengkap dari versi AB_{6-7} yaitu: "Penambahan Warning", "Fitur Search pada semua diagram", "Undo Redo pada UseCase", "Ekspor diagram" dan "Panning/Zooming pada masing-masing diagram". Dari 5 fungsional tersebut berdampak pada penambahan, dan pengubahan paket, kelas dan relasi di dalamnya sebagai contoh ada penambahan paket "ProcessViewWarning", "FishEyeWarning" dan "UseCaseWarning".

4.3.8. Pengamatan mendalam AB_{8-9}

Pada pengamatan AB_{8-9} terdapat volatilitas 15,8% dengan rincian adanya 10 penambahan, tidak ada pengubahan dan tidak ada penghapusan fitur dari 65 fitur awal. Pada versi AB_{8-9} memiliki kesamaan sifat seperti pada AB_{6-7} dan AB_{7-8} yaitu hanya memiliki fitur penambahan saja. Perbedaan yang mencolok adalah fitur penambahan ini merepresentasikan lingkup fungsional yang sudah ada sebelumnya. Artinya penambahan ini tidak menambah lingkup fungsional baru sehingga walau nilai volatilitas pada AB_{8-9} merupakan nilai volatilitias fitur paling tinggi dibanding dengan AB_{6-7} dan AB_{7-8}, stabilitas AB_{8-9} adalah yang paling tinggi yaitu 88,5%.

4.3.9. Pengamatan mendalam AB_{9-10}

Pada pengamatan AB_{9-10} terdapat volatilitas 26,7% dengan rincian adanya 15 penambahan, 4 pengubahan dan 1 penghapusan fitur dari 75 fitur awal. Volatilitas terbagi atas 2 bagian yaitu: fitur yang merepresentasikan kebutuhan baru dan fitur yang menjadi suplemen atau tambahan dari kebutuhan yang telah terimplementasi sebelumnya. Pada AB_{9-10} terdapat beberapa skope baru yang

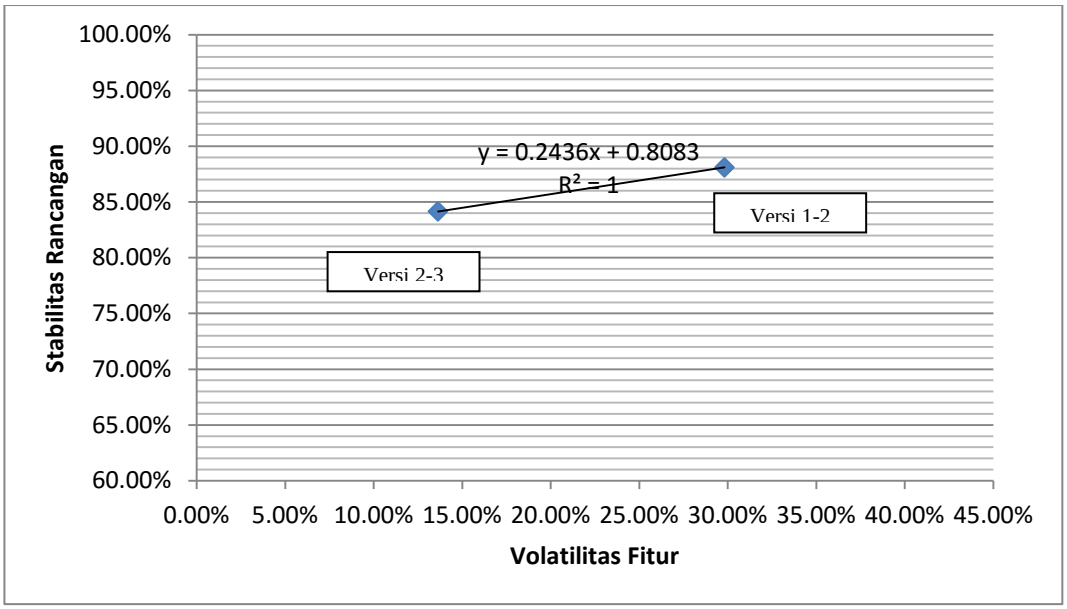
diimplementasi. Lingkup kebutuhan yang mengalami penambahan meliputi "Collaboration Editing". Dan ada pula 1 penghapusan fitur yang telah diimplementasi dari versi 2 hingga versi 9. Tentu akibat adanya 2 hal ini, stabilitas mengalami penurunan hingga menjadi 63,5%.

4.3.10. Pengamatan mendalam NS_{1-2}

Pada pengamatan NS_{1-2} terdapat volatilitas 29,8% dengan rincian adanya 11 penambahan, 6 pengubahan dan adanya tidak ada penghapusan fitur dari 57 fitur awal pada versi ke-1. Pengamatan stabilitas rancangan NS_{1-2} terdapat nilai 88,094% dengan adanya degradasi tingkat penggunaan kembali pada paket tertentu sebesar 98,7% dan degradasi tingkat penggunaan kembali pada relasi antar paket sebesar 77,5%. Penambahan terjadi pada skope PDM untuk menambah kebutuhan akan pendektisan warning pada rancangan CDM dan PDM. Dari hasil tersebut, rancangan cenderung stabil dari perubahan versi ke-1 dan versi ke-2 dengan tingkat penggunaan mencapai 88,09% walaupaun ada sekitar 31% perubahan yang terjadi.

4.3.11. Pengamatan Mendalam NS_{2-3}

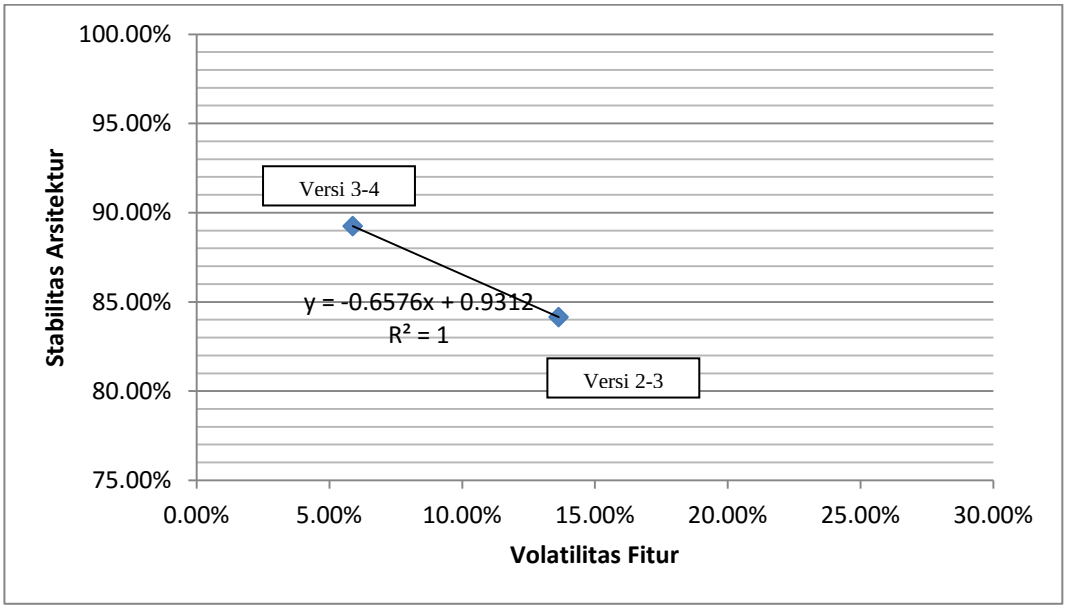
Pada pengamatan NS_{2-3} terdapat volatilitas 13,6% dengan rincian adanya 5 penambahan, 2 pengubahan dan 2 ada penghapusan fitur dari 66 fitur awal pada versi ke-2. Sedangkan untuk stabilitas rancangan malah mengalami penurunan menjadi 84,15%. Turunnya stabilitas ini menjadi anomali dari hipotesis awal di mana volatilitas mengalami penurunan, maka stabilitas semakin meningkat. Sama seperti kasus pada AB, anomali terjadi karena (1) ada beberapa fitur yang dihapuskan dan (2) ada penambahan lingkup. Pada NS_{2-3} ini terjadi penghapusan pada skope PDM yaitu fitur penggunakan VIEW dan fitur "export to XML" sebagai fungsi simpan proyek. Untuk penambahan terjadi yaitu penambahan lingkup pada SYSTEM yaitu kebutuhan UNDO REDO, "Zoom Canvas" dan "Generate Documentation to document file". Walaupun secara nominal perubahan fitur sedikit, namun dampak ke sistem cukup besar karena mencakup kebutuhan, tidak menyangkut ranah fitur/solusi. Hal ini terlihat pada Gambar 4.10.



Gambar 4. 10. Grafik Perbandingan Versi (1-2) dengan Versi (2-3)

4.3.12. Pengamatan Mendalam $NS_{3\text{-}4}$

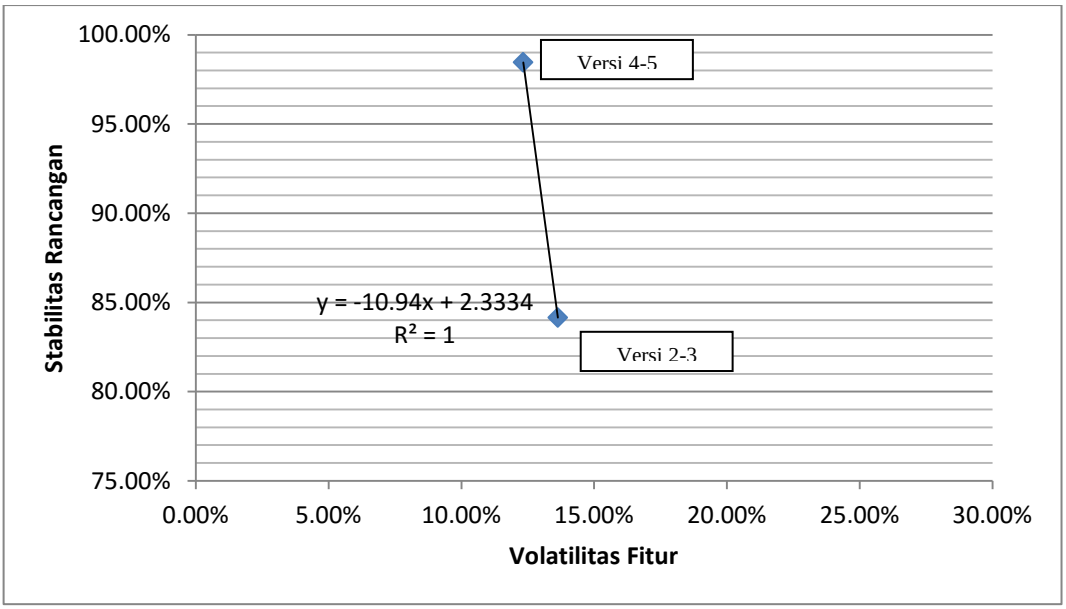
Pada pengamatan $NS_{3\text{-}4}$ terdapat volatilitas 5,8% dengan rincian adanya tidak ada penambahan, 4 pengubahan dan tidak ada penghapusan fitur dari 68 fitur awal. Pada pengamatan $NS_{3\text{-}4}$ terdapat stabilitas rancangan sebesar 89,3%. Fenomena ini sesuai dengan hipotesis peneliti dan tidak terjadi anomali karena semua data mengalami perubahan / improvisasi sehingga pada implementasi tidak ada penambahan paket / kelas yang terlampau banyak.



Gambar 4. 11. Grafik Perbandingan Versi (2-3) dengan Versi (3-4)

4.3.13. Pengamatan Mendalam NS_{4-5}

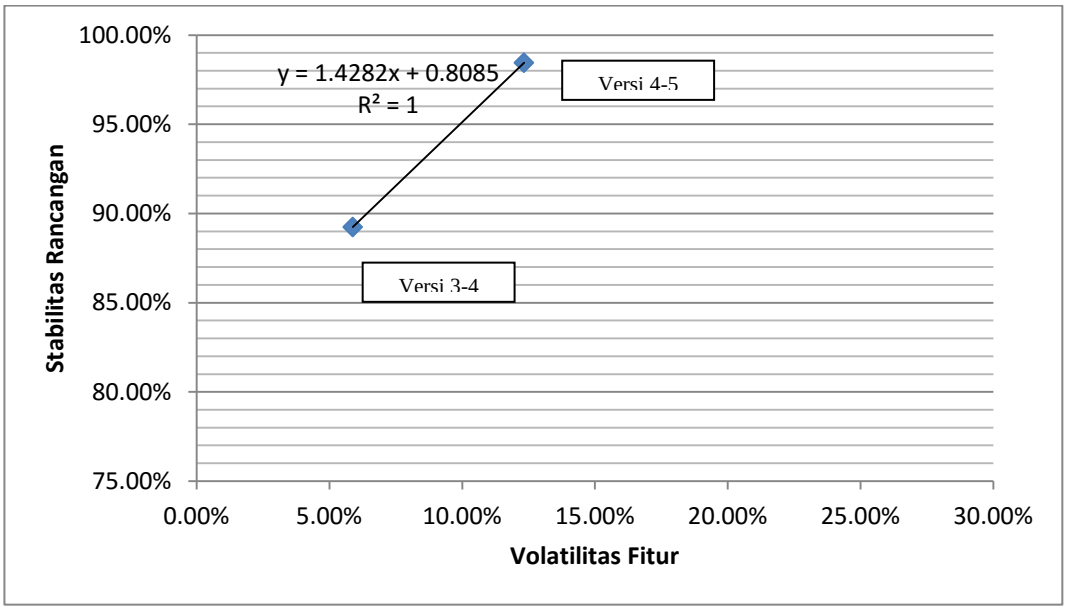
Pada pengamatan NS_{4-5} terdapat volatilitas 12,3% dengan rincian 6 fitur bertambah, 3 mengalami perubahan dan tidak adanya penghapusan data. Namun tingkat stabilitas meningkat tajam menjadi 98.46%. Fitur yang bertambah adalah penambahan elemen pada PDM seperti PROCEDURE, TRIGGER dan INDEX. Penambahan elemen PDM tersebut secara rancangan arsitektur memiliki kemiripan implementasi rancangan dengan komponen PDM yang telah ada, sehingga tingkat penggunaan kembali pada versi ke-4 dan ke-5 tinggi. Penggunaan kembali elemen LoadXML yang belum terimplementasi sempurna juga menambah nilai stabilitas rancangan. Peneliti tidak bisa dengan NS_{2-3} yang memiliki nilai volatilitas fitur mendekati nilai 12,3% yaitu 13,64% karena pada NS_{2-3} sifat/faktor perubahan volatilitas berbeda karena adanya perubahan dan penghapusan fitur yang membuat stabilitas menurun hingga 84,15% (Gambar 4.12).



Gambar 4. 12. Grafik Perbandingan Versi (2-3) dengan Versi (4-5)

Bila peneliti membandingkan NS_{4-5} dengan NS_{3-4} munculnya anomali positif karena pada NS_{3-4} pengubahan fitur mengakibatkan persentase tingkat penggunaan kembali rancangan tidak meningkat signifikan. Anomali ini disebabkan rancangan arsitektur dan kelas pada NS_3 tidak dibuat global apabila ada perubahan di dalamnya. Contoh kasusnya adalah fitur UNDO REDO dan fitur

"Generate Dokumentasi". Pada NS_3 fitur "UNDO REDO" terjadi hanya pada masing-masing diagram tanpa ada interaksi antar diagram. Pada NS_4 pengguna meminta adanya dinteraksi antar diagram, jadi bisa saling menotifikasi antar diagram apabila ada perubahan di salah satu diagram. Fitur Generate Dokumentasi juga mengalami perubahan antar muka dan mekanisme "Generate ke dokumen". Positif ini dalam arti perubahan yang lebih tinggi tidak mengganggu stabilitas rancangan (Gambar 4.13).



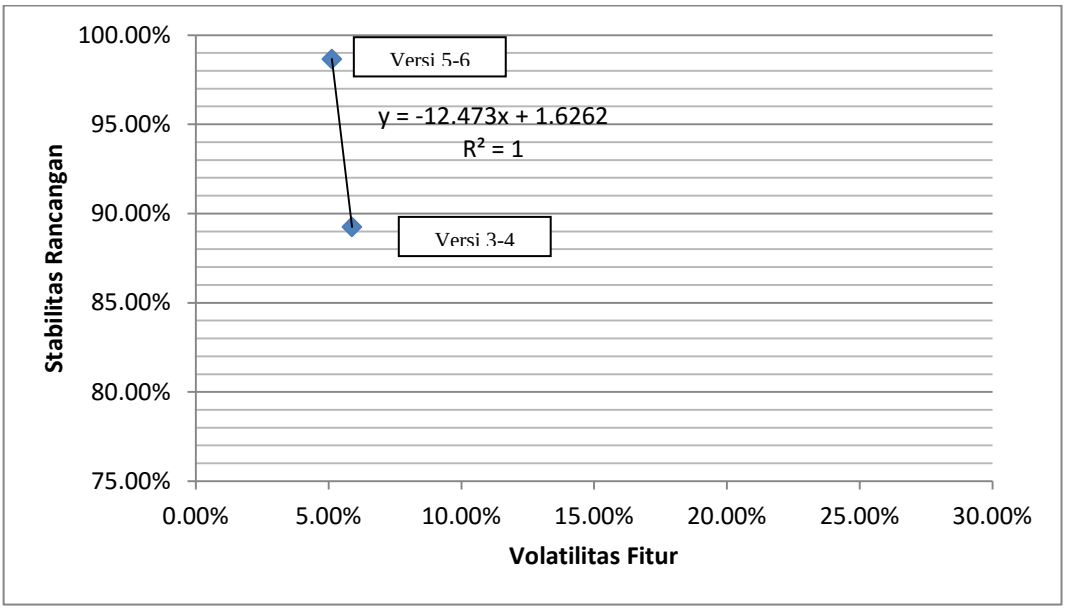
Gambar 4. 13. Grafik Perbandingan Versi (3-4) dengan Versi (4-5)

4.3.14. Pengamatan Mendalam NS_{5-6}

Pada pengamatan NS_{5-6} terdapat volatilitas 5,13% dengan rincian adanya 1 penambahan, 3 pengubahan dan tidak ada penghapusan fitur dari 78 fitur awal. Pada bagian ini stabilitas arsitektur menunjukkan peningkatan menjadi 98,66%. Hampir seluruh elemen kelas/paket dan relasi dipergunakan ulang pada versi ke-5 dan versi ke-6.

Bila dibandingkan dengan NS_{3-4} yang memiliki volatilitas 5,58%, perbedaan tingkat stabilitas cukup mencolok (Gambar 4.14). Hal ini disebabkan karena fitur yang mengalami penambahan yaitu: Change Paper Orientation and Size menggunakan komponen implementasi yang sama dengan fitur "Editor can

make new CDM Diagram, with PageSize and orientation", hanya penempatan pemanggilan saja yang mengalami perbedaan.



Gambar 4. 14. Grafik Perbandingan Versi (3-4) dengan Versi (5-6)

4.3.15. Pengamatan Mendalam NS_{6-7}

Pada pengamatan NS_{6-7} terdapat volatilitas 3,8% dengan rincian adanya 2 penambahan, 1 pengubahan dan tidak ada penghapusan fitur dari 79 fitur yang ada pada versi ke-6. Stabilitas arsitetektur meningkat sedikit menjadi 99,3% dapat dikatakan hampir seluruh elemen dipergunakan kembali. Hal ini dilatar belakangi karena perpindahan versi 6 ke versi 7 ada penambahan fitur yang sebagai pelengkap dari fitur yang telah ada. Contohnya adalah fitur "Save in Database". Konsep "penyimpanan Diagram to Database" itu hampir sama seperti penyimpanan pada fitur "Save in XML" namun lingkungan keluarannya saja yang mengalami perbedaan.

BAB 5
PENUTUP

Pada bab ini dijelaskan mengenai kesimpulan akhir yang didapat setelah melakukan serangkaian uji coba pada bab sebelumnya.

5.1. Kesimpulan

Kesimpulan yang dapat diambil dalam penelitian ini antara lain adalah sebagai berikut.

a. Pembentukan data uji untuk pengukuran dan evaluasi rancangan perangkat lunak dan volatilitas dibentuk dari data volatilitas fitur dengan data komponen diagram yang dipasangkan bersamaan sesuai dengan versinya program. Volatilitas fitur diukur dan didekomposisi ke beberapa faktor yaitu jumlah fitur yang bertambah, fitur yang berubah dan fitur yang dihapus. Hal ini mengacu pada pengertian dari subbab 2.4 sehingga mendukung pembahasan mendalam pada subbab 4.3. Data komponen diagram disajikan dalam bentuk nama paket, jumlah paket, nama kelas dalam paket tertentu dan relasi antar kelas baik di dalam atau antar paket. Hal ini berdasarkan pada literatur yang tersaji subbab 2.2 terkait pengukuran stabilitas rancangan perangkat lunak.
b. Perbandingan stabilitas rancangan dan volatilitas kebutuhan ditingkat fitur diukur secara bersamaan setiap versinya. Nilai stabilitas rancangan berdasarkan pada rumus “Stability” yang dijelaskan pada subbab 2.2 sedangkan nilai volatilitas kebutuhan perangkat lunak berdasarkan pada perhitungan ISO 9126 yang dijelaskan pada subbab 2.3. Untuk mengukur keterkaitan pad variabel stabilitas dan variabel volatilitas peneliti menggunakan uji korelasi statistik dengan Pearson-Product Moment yang landasan teorinya terletak pada subbab 2.5.1 dan hasil pengujiannya terpapar pada subbab 4.3.
c. Hasil perbandingan nilai pengukuran pengukuran volatilitas kebutuhan tidak dapat langsung dibandingkan dengan stabilitas rancangan, karena nilai korelasi volatilitas terhadap stabilitas rancangan bernilai di bawah 50% sesuai pembahasan pada subbab 4.3. Hal ini berarti pada dua kasus data yang diteliti,

perangkat lunak tidak memiliki korelasi yang kuat dan memiliki faktor-faktor lain yang menyebabkan hasil perhitungan menjadi tidak seragam. Oleh karena itu perlu didekomposisi faktor-faktor yang membangun nilai volatilitas fitur tersebut. Hal ini mengacu pada pembahasan pada subbab 4.2.1.

d. Terdapat tiga faktor yang membuat suatu ketidakstabilan rancangan perangkat lunak.
    i. Penambahan Fitur/kebutuhan sistem yang mana pada versi sebelumnya tidak ditangkap / diketahui sehingga keputusan rancangan tidak dapat dibuat generalisasi atau pola perancangan tertentu hingga dapat melebarkan skope sistem pada tingkat implementasi. Hal ini tercermin pada kasus AB_{2-3}, AB_{3-4}, $AB_{6-7,}$ AB_{7-8}, AB_{9-10}, NS_{1-2}, dan NS_{2-3}. Apabila memiliki penambahan atau perubahan kebutuhan/fitur yang tidak tertangkap namun pada implementasinya memiliki pola perancangan yang mirip dengan entitas kelas yang lain, maka tingkat stabilitas masih lebih tinggi dibanding dengan implementasi yang tidak disiapkan pola perancangan arsitektur. Kasus seperti ini dijumpai pada AB_{5-6}.
    ii. Pengubahan struktur Sistem akibat perkembangan skope kebutuhan yang semula skope kecil menjadi skope besar. Pada 2 kasus tersebut terjadi kesalahlokasi kelas yang seharusnya ditaruh pada paket yang lebih general namun sudah terlanjut pada paket lebih spesifik. Hal ini tercermin pada kasus AB_{2-3}, dan NS_{3-4}.
    iii. Penghapusan fitur/kebutuhan yang inti. Definisi Inti adalah fitur tersebut terdapat > 50% dari seluruh versi produk penelitian. Hal ini tercermin pada kasus $AB_{3-4,}$ dan $NS_{2-3.}$

5.2. Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut penelitian ini adalah pembentukan model data uji bisa dilakukan dalam bentuk TREE yang dinamis. TREE dinamis ini dapat mengetahui besar kebergantungan dengan node lain dan hubungan antar NODE sehingga dapat merepresentasikan NODE mana yang memiliki bobot terbesar.

DAFTAR PUSTAKA

Alshayeb, M., & Li, W. (2005). An empirical study of system design instability metric and design evolution in an agile software process. *The Journal of Systems and Software* , 269-274.

Aversano, L., Molfetta, M., & Tortorella, M. (2013). Evaluating Architecture Stability of Software Projects. Koblenz: IEEE.

Bourque, P., & Fairley, R. E. Software Design Principles. In IEEE, *SWEBOK v3.0* (p. 52). IEEE.

Breivold, H. P., Crnkovic, I., & Larsson, M. (2012). A systematic review of software architecture evolution research. *Information and Software Technology , 54* (Software Architecture Evolution Research), 16-40.

Chen, C.-Y., & Chen, P.-C. (2009). A holistic approach to managing software change impact. *The Journal of Systems and Software* , 2051–2067.

Cleland-Huang, J., Hanmer, R. S., Supakkul, S., & Mirakhorli, M. (2013). *The Twin Peaks of Requirements and Architecture.* IEEE.

Clements, P. (2010). *Documenting Software Architectures: Views and Beyond, Second Edition.* Boston: Addison-Wesley.

Constantinou, E., & Stamelos, I. (2015). Architectural Stability and Evolution Measurement for Software Reuse. *SAC.* Spain: ACM.

de Boer, R. C., & van Vliet, H. (2009). On the similarity between requirements and architecture. *The Journal of Systems and Software , 82* (544-550).

Everitt, B. (1998). *The Cambridge Dictionary of Statistics.* Cambridge, UK New York: Cambridge University Press.

Ferreira, S. (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *The Journal of Systems and Software* , 1568-1577.

Gasperz. (1991). *Metodologi Perancangan Percobaan.*

Jamshid, P., Ghafari, M., Ahmad, A., & Pahl, C. (2013). A Framework for Classifying and Comparing Architecture-Centric Software. *17th European Conference on Software Maintenance and Reengineering.* Ireland.

Johnson, R. A., & Wichern, D. W. (2007). *Applied Multivariate Statistical Analysis (6th).* Prentice Hall.

Leffingwell, & Widrig. (2000). *Managing software requirements: a unified approach.*

Leffingwell, D. (2001). *Features, Use Cases, Requirements, Oh My!* Rational Software.

Mannaert, H., Verelst, J., & Ven, K. (2011). The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability. *Science of Computer Programming* , 1210-1222.

Molesini, A., Garcia, A., Chavez, C. v., & Batista, T. V. (2010). Stability assessment of aspect-oriented software architectures: A quantitative study. *The Journal of Systems and Software* , 711-722.

Monperrus, M., Baudry, B., Champeau, J., Hoeltzener, B., & Jezequel, J. M. (2013). Automated measurement of models of requirements. *21*.

Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes* , 40.

Sarwono, J. (2009). *Statistik Itu Mudah: Panduan Lengkap untuk Belajar Komputasi Statistik Menggunakan SPSS 16.* Yogyakarta: Universitas Atma Jaya Yogyakarta.

Setiawan, A. (2010, 11 21). *Korelasi Pearson*. Retrieved 10 06, 2015, from Smart Statistik: https://smartstat.wordpress.com/2010/11/21/korelasi-pearson/

Sudjana. (1985). *Disain dan Analisis Eksperimen.*

Vishnyakov, A., & Orlov, S. (2015). Software Architecture and Detailed Design Evaluation. *Procedia Computer Science* (pp. 41-52). Latvia: ELSEVIER.

Williams, B. J., & Carver, J. C. (2010). Characterizing Software Architecture Changes: A Systematic Review. *Information and Software Technology , 52* (1).

Williams, C. (2007). Research Methods. *Journal of Business & Economic Research* , 65-72.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

LAMPIRAN 1 Meta data Kebutuhan yang digunakan

Formulir Data Fitur AB versi 1

Code	Status	Features Description	Scope
FRD001		Editor inserts Process	DFD
FRD002		Editor inserts Data Flow	DFD
FRD003		Editor inserts Control Flow	DFD
FRD004		Editor inserts Data Store	DFD
FRD005		Editor inserts External Entity	DFD
FRD006		System generates name of Process automatically	DFD
FRD007		Editor inserts name of Data Flow	DFD
FRD008		Editor inserts name of Control Flow	DFD
FRD009		System generates name of Data Store automatically	DFD
FRD010		System generates name of External Entity automatically	DFD
FRD011		Editor makes new DFD Diagram	DFD
FRD012		Editor picks DFD's elements from DFD's pallette	DFD
FRS001		Editor closes CASE Tools	System
FRS002		Editor minimizes CASE Tools window	System
FRS003		Editor maximizes CASE Tools window	System
FRS004		Editor restores down CASE Tools window	System
FRS005		System generates automatically items in TreeBrowser depends on diagrams	System

Karena meta data yang terlampau banyak, untuk selengkapnya mengenai AB dan NS versi berikutnya bisa diakses pada http://tesis.felix-handani.com/ atau menghubungi penulis.

Formulir Data Komponen bagian Metrik ES pada versi AB_{1-2}

V1				*V2*		
Kode Paket	**Nama Class**	**Status Class**		**Kode Paket**	**Nama Class**	**Status Class**
An	Form1	A		An	Form1	
M	ControlFlow	A		M	ControlFlow	
M	DataFlow	A		M	DataFlow	
M	DataStore	A		M	DataStore	
M	ExternalEntity	A		M	ExternalEntity	
				M	FishEyeLink	A
				M	FishEyeNode	A
M	Flow	A		M	Flow	
M	Iobserver	A		M	Iobserver	
M	ModelObject	A		M	ModelObject	
M	Node	A		M	Node	
M	Process	A		M	Process	
M	Subject	A		M	Subject	
P	BindTreeNode	A		P	BindTreeNode	
P	ControlFlowDraw	A		P	ControlFlowDraw	
				P	DataFlowDraw	A
P	DataStoreDraw	A		P	DataStoreDraw	
P	DFD	A		P	DFD	
P	ExternalEntityDraw	A		P	ExternalEntityDraw	
				P	IProcessViewMediator	A
P	ModelObjectUI	A		P	ModelObjectUI	
P	ProcessDraw	A		P	ProcessDraw	
P	TreeBrowser	A		P	TreeBrowser	

Karena meta data yang terlampau banyak, untuk selengkapnya mengenai AB dan NS versi berikutnya bisa diakses pada http://tesis.felix-handani.com/ atau menghubungi penulis.

Formulir Data Komponen bagian Metrik IS pada versi AB_{1-2}

V1								*V2*						
P1		**P2**		**REL**	**Sifat**	**Status**	**Status**	**P1**		**P2**		**REL**	**Status**
An	Form1	M	Process	root	Composition			An	Form1	M	Process	root	Composition
An	Form1	P	DFD	dfd1	Aggregation		R	An	Form1	P	DFD	dfd1	Aggregation
An	Form1	P	DFD	List<> dfds	Aggregation			An	Form1	P	DFD	List<> dfds	Aggregation
							A	An	Form1	P	IProcessViewMediator		Extends
An	Form1	P	TreeBrowser	treebrowser1	Composition			An	Form1	P	TreeBrowser	Treebrowser1	Composition
P	BindTreeNode	M	Process	node	Aggregation			P	BindTreeNode	M	Process	node	Aggregation
P	BindTreeNode	M	IObserver		Extends			P	BindTreeNode	M	IObserver		Extends
P	ControlFlowDraw	P	ModelObjectUI		Extends			P	ControlFlowDraw	P	ModelObjectUI		Extends
							A	P	DataFlowDraw	P	ModelObjectUI		Extends
P	DataStoreDraw	P	ModelObjectUI		Extends			P	DataStoreDraw	P	ModelObjectUI		Extends
P	DFD	M	Node	dest	Aggregation			P	DFD	M	Node	dest	Aggregation
P	DFD	M	Process	node	Aggregation			P	DFD	M	Process	node	Aggregation
P	DFD	M	Node	org	Aggregation			P	DFD	M	Node	org	Aggregation
P	ExternalEntityDraw	P	ModelObjectUI		Extends			P	ExternalEntityDraw	P	ModelObjectUI		Extends
P	ModelObjectUI	M	IObserver		Extends			P	ModelObjectUI	M	IObserver		Extends
P	ModelObjectUI	M	ModelObject	mObject	Aggregation			P	ModelObjectUI	M	ModelObject	mObject	Aggregation
P	ProcessDraw	P	ModelObjectUI		Extends			P	ProcessDraw	P	ModelObjectUI		Extends
P	TreeBrowser	M	IObserver		Extends			P	TreeBrowser	M	IObserver		Extends
P	TreeBrowser	M	Process	model	Aggregation			P	TreeBrowser	M	Process	model	Aggregation
							A	P	TreeBrowser	P	IProcessViewMediator	mediator	Aggregation
M	ControlFlow	M	Process	nodeDest	Aggregation			M	ControlFlow	M	Process	nodeDest	Aggregation
M	ControlFlow	M	Process	nodeOrg	Aggregation			M	ControlFlow	M	Process	nodeOrg	Aggregation

M	ControlFlow	M	Flow		Extends				M	ControlFlow	M	Flow		Extends
M	DataFlow	M	Node	nodeDest	Aggregation				M	DataFlow	M	Node	nodeDest	Aggregation
M	DataFlow	M	Node	nodeOrg	Aggregation				M	DataFlow	M	Node	nodeOrg	Aggregation
M	DataFlow	M	Flow		Extends				M	DataFlow	M	Flow		Extends
M	DataStore	M	Node		Extends				M	DataStore	M	Node		Extends
M	DataStore	M	Process	parent	Aggregation				M	DataStore	M	Process	parent	Aggregation
M	ExternalEntity	M	Node		Extends				M	ExternalEntity	M	Node		Extends
								A	M	FishEyeNode	M	Node		Extends
M	Flow	M	ModelObject		Extends				M	Flow	M	ModelObject		Extends
M	ModelObject	M	Subject		Extends				M	ModelObject	M	Subject		Extends
M	Node	M	ModelObject		Extends				M	Node	M	ModelObject		Extends
M	Process	M	Node		Extends				M	Process	M	Node		Extends
M	Process	M	Flow	List<> flows	Aggregation				M	Process	M	Flow	List<> flows	Aggregation
M	Process	M	Node	List<> objs	Aggregation				M	Process	M	Node	List<> objs	Aggregation
								A	M	Process	M	Process	parent	Aggregation
								A	M	Subject	M	IObserver	List<> Observers	Aggregation

Karena meta data yang terlampau banyak, untuk selengkapnya mengenai AB dan NS versi berikutnya bisa diakses pada http://tesis.felix-handani.com/ atau menghubungi penulis.

BIODATA PENULIS

Peneliti, Felix Handani, lahir di kota Surabaya pada tanggal 23 Agustus 1991. Penulis adalah anak pertama dari dua bersaudara dan dibesarkan di kota Surabaya, Jawa Timur.

Peneliti menempuh pendidikan formal di SD Katolik Yohannes Gabriel Surabaya (1997-2003), SMP Katolik St. Agnes Surabaya (2003-2006), dan SMA Katolik St. Agnes Surabaya (2006-2009). Pada tahun 2009-2013, peneliti melanjutkan pendidikan S1 di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Pada tahun 2014-2016, peneliti melanjutkan pendidikan Magister S2 di jurusan yang sama, yaitu Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di Jurusan Teknik Informatika, peneliti mengambil bidang minat Rekayasa Perangkat Lunak. Pada pendidikan S1 dan S2 peneliti aktif dalam organisasi kemahasiswaan dan kegiatan formal akademik seperti: Himpunan Mahasiswa Teknik Computer (HMTC), Ketua National Programming Contest 2012, Administrator Laboratorium Pemograman 1 (2011-2013), Asisten Dosen pada Mata Kuliah Sistem Digital dan Basis Data. Peneliti aktif dalam kegiatan penelitian seperti Program Karya Mahasiswa (PKM-GT) pada tahun 2010, Program PKM-Wirausaha pada tahun 2012-2013 yang didanai dan Program Mahasiswa Wirausaha pada tahun 2013 dengan membentuk sebuah startup digital.

Pada lingkup professional peneliti telah mengajar pada lembaga pendidikan di Surabaya dan pernah ikut serta sebagai tim teknis dalam berbagai proyek Sistem Informasi melalui badan swasta. Peneliti dapat dihubungi melalui alamat email felix.handani@gmail.com