BAB II
LANDASAN TEORI

2.1 Perbandingan Dengan Penelitian Terdahulu

Penelitian terdahulu digunakan sebagai bahan referensi dalam melakukan penelitian Pengembangan Sistem NKMD Pada Modul Kuesioner Dan *Setting* Dengan *Software Re-engineering.* Penelitian terdahulu yang dijadikan refrensi memiliki topik yang sama yaitu, merekayasa ulang atau *reengineering* sistem lama menjadi sistem yang baru. Adapun penelitian tersebut adalah milik Trianto Satria dengan judul Renovasi Konsul Sistem Otomasi Bangunan (BAS) Sub-Unit Sistem Pemanas HVAC (2016). Penelitian tersebut menjelaskan tentang penggunaan *software re-engineering* dan tahapan-tahapan yang dilakukan dalam melakukan proses penelitian.

Dalam penelitian Trianto Satria (2016), menerangkan tentang pengembangan atau pembaharuan sistem HVAC yang meliputi dari segi perangkat keras dan perangkat lunak agar dapat menyesuaikan dengan sistem HVAC sub-unit pemanas yang baru. Namun, sistem HVAC yang digunakan disini masih berupa simulator berbasis PLC Modicon TM221CE40R yang telah dirancang sedemikian rupa sehingga dapat menggambarkan proses kerja dari HVAC itu sendiri. Selain itu, dijelaskan juga tahapan-tahapan yang dilakukan dalam penelitiannya yaitu, *reverse engineering* dan *forward engineering* yang merupakan tahapan-tahapan pada *software re-engineering.* Rincian tahapan *reverse engineering* yang dilakukan adalah sebagai berikut.

a. Mengamati konstruksi dan cara kerja sistem yang terdahulu.
b. Cek dokumen rancangan sistem yang terdahulu.
c. Cek kesesuaian spesifikasi komponen yang terpasang terhadap dokumen rancangan.
d. Mengidentifikasi beberapa kelemahan sistem yang terdahulu.

Rincian tahapan *forward engineering* yang dilakukan adalah sebagai berikut.

a. Menentukan spesifikasi sistem yang baru.
b. Merancang ulang sistem (pembuatan gambar dan *bill of quantity* yang baru).
c. Pembuatan *hardware*.
d. Pembuatan *software.*
e. Pengujian

Hasil dari penelitian ini adalah sistem yang dibangun telah berhasil dan berjalan sesuai dengan rancangan sistem dari penelitian tersebut.

Kesimpulan dari hasil penelitian terdahulu yaitu terdapat beberapa kesamaan dengan penelitian ini dan dalam membangun ulang suatu sistem, *software re-engineering* bisa digunakan sebagai metode dalam melakukan aktivitas rekaysa ulang atau *reengineering.* Hal ini dikarenakan dalam tahapannya, dilakukan analisis dan identifikasi pada sistem yang lama (*reverse engineering)* dan kemudian perancangan serta implementasi terhadap sistem yang baru (*forward engineering)*. Perbedaan penelitian terdahulu dengan penelitian sekarang yaitu tahapan yang dilakukan pada penelitian terdahulu berupa *reverse engineering* dan *forward engineering* dan mencakup dari segi *software* dan *hardware*. Sedangkan penelitian sekarang menggunakan seluruh tahapan yang ada pada *software re-engineering* dan mencakup dari segi *software* saja. Meskipun objek penelitiannya berbeda, namun tujuan penelitian ini sama yaitu untuk merekayasa ulang suatu sistem untuk mendapatkan pembaharuan dan penambahan fitur. Adapun tabel yang menguraikan perbandingan penelitian dapat dilihat pada Tabel 2.1.

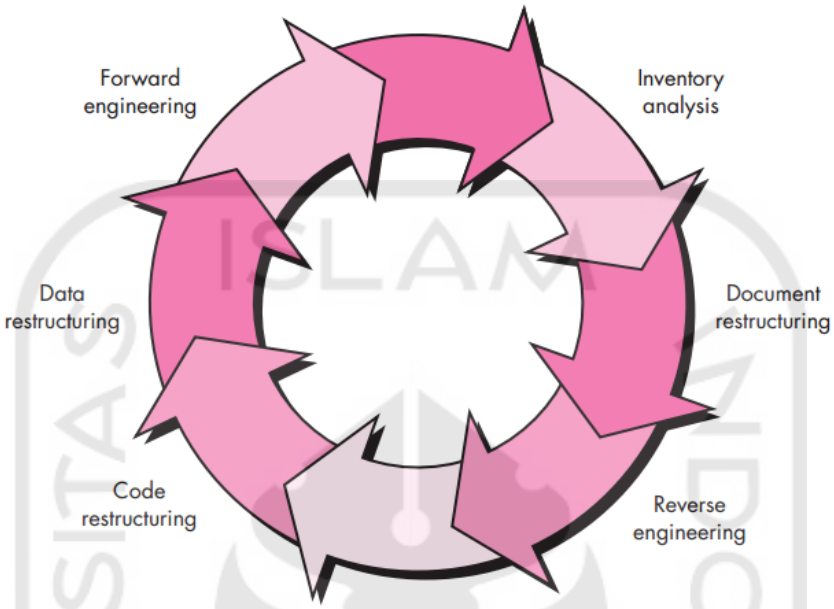
Tabel 2.1 Perbandingan penelitian

Pengarang	**Tahun**	**Bahasan**	**Objek**	**Tahapan Software Re-engineering**
Trianto Satria	2016	Renovasi Konsul Sistem Otomasi Bangunan (BAS) Sub-Unit Sistem Pemanas HVAC	Pengembangan sistem HVAC yang bisa melakukan *monitoring* dan *controlling*	*Reverse Engineering* dan *Forward Engineering*
Lalu Kismara Hadi	2019	Pengembangan Sistem NKMD Pada Modul Kuesioner Dan *Setting* Dengan Metodologi *Software Re-Engineering*	Pengembangan Sistem NKMD berbasis *web* di Fakultas Teknologi Industri	*Inventory Analysis, Reverse Engineering, Document Restructuring, Code Restructuring, Data Restructuring*, dan *Forward Engineering*

2.2 *Software Re-engineering*

Reengineering adalah aktivitas yang dilakukan untuk membangun ulang atau merekayasa ulang(Pressman, 2010). Sedangkan *software re-engineering* adalah rekyasa ulang sebuah sistem yang diubah menjadi sistem dalam bentuk baru. *Software re-engineering* merupakan gabungan dari *reverse engineering* dan *forward engineering*(Satria, 2016). Dalam

melakukan aktivitas *software re-engineering*, terdapat 6 model proses atau tahapan yang dapat dilakukan. Adapun ke-enam model proses tersebut dapat dilihat pada Gambar 2.1.



Gambar 2.1 Model proses *software re-engineering*

Sumber: (Pressman, 2010)

Pada Gambar 2.1 digambarkan bahwa model proses tersebut dilakukan secara berurutan namun, ada kalanya *reverse engineering* dilakukan terlebih dahulu sebelum melakukan restrukturisasi(Pressman, 2010). Pada beberapa literatur dijelaskan bahwa tahapan utama dari *software re-engineering* adalah *reverse engineering* dan *forward engineering*(Zamzami & Budiardjo, 2011).

2.2.1 *Inventory Analysis*

Inventory anlysis adalah tahapan dalam menganalisis atau menggali informasi terkait dengan rincian deskripsi sistem yang akan dikembangkan seperti ukuran, usia, nama sistem, dan lain-lain sehingga, hal-hal yang akan direkayasa ulang akan muncul(Pressman, 2010).

2.2.2 *Document Restructuring*

Dokumentasi yang lemah merupakan kekurang pada sistem yang diwariskan. Pada pengembangan sistem yang baru, menstruktur ulang dokumentasi dari sistem terdahulu bisa saja dilakukan tergantung dari kasus yang dihadapi. Sehingga pada tahapan ini, dokumentasi

ulang dapat dilakukan dengan mencakup keseluruhan sistem atau sebagian saja yang diperlukan. Dokumenstasi bisa saja tidak diperlukan apabila terlalu banyak yang didokumentasikan dan memakan waktu yang cukup lama(Pressman, 2010).

2.2.3 *Reverse Engineering* (Rekayasa Terbalik)

Reverse engineering merupakan tahapan pertama kali yang perlu dilakukan sebelum melakukan *forward engineering* dalam aktivitas *software reengineering*. Tahapan ini adalah analisis yang dilakukan untuk mengidentifikasi sistem terdahulu, sehingga dapat dipahami cara kerja sistem dan agar dapat dilakukan pengembangan(Satria, 2016). Terdapat beberapa faktor yang menjadi penyebab dilakukannya *reverse engineering,* yaitu(Zamzami & Budiardjo, 2011):

a. Kelengkapan desain/spesifikasi sistem kurang lengkap atau hilang.
b. Dokumentasi sistem hilang atau tidak sesuai.
c. Kompleksitas program meningkat.
d. *Source code* tidak terstruktur dengan baik.
e. Ketika program perlu diterjemahkan ke dalam bahasa pemrograman yang berbeda.

Dalam melakukan tahapan ini, terdapat beberapa rangkaian proses yang perlu dilakukan sebelum melanjutkan ke tahapan *forward engineering*. Adapun proses-proses tersebut diantaranya(Satria, 2016):

a. Menganalisis konstruksi dan cara kerja sistem terdahulu.
b. Mengecek dokumen rancangan sistem terdahulu.
c. Pengecekan terhadap kesesuaian komponen spesifikasi sistem dengan dokumen rancangan.
d. Identifikasi kelemahan pada sistem terdahulu.

2.2.4 *Code Restructuring*

Code Restructuring merupakan tahapan dalam menganalisis kode sumber sistem terdahulu untuk memahami fungsi-fungsi apa saja yang terdapat di dalam kode sumber. Setelah itu, kode sumber dapat distruktur ulang dengan bahasa pemrograman yang lebih modern jika diperlukan(Pressman, 2010).

2.2.5 *Data Restructuring*

Data Restructuring merupakan tahapan dalam menganalisis kebutuhan data sistem. Pada beberapa kasus, *data restructuring* diawali dengan *reverse engineering*. Objek data yang terdapat pada sistem dan atributnya diidentifikasi dan sturuktur data yang ada diulas untuk tujuan kualitas sistem nantinya(Pressman, 2010).

2.2.6 *Forward Engineering* (Rekayasa Maju)

Forward Engineering merupakan perancangan dan implementasi hasil dari *reverse engineering* untuk menghasilkan sistem yang baru(Satria, 2016). Tahapan ini akan meningkatkan kualitas sistem secara keseluruhan. Dalam beberapa kasus, perangkat lunak yang direkayasa ulang mengimplementasikan ulang fungsi yang telah ada dan menambahkan fungsi baru serta peningkatan pada sisi performa sistem(Pressman, 2010). Selain *reverse engineering, forward reengineering* juga memiliki rangkaian proses yang perlu dilakukan. Adapun rangkaian proses tersebut diantaranya(Satria, 2016):

a. Penentuan spesifikasi dari sistem yang baru.
b. Merancang sistem yang baru.
c. Pembuatan sistem yang baru.
d. Pengujian.

2.3 Sistem Nilai Kinerja Dosen (NKD)

Sistem NKD merupakan sistem terdahulu yang akan dikembangkan menjadi sistem yang baru yaitu sistem Nilai Kinerja Mengajar Dosen (NKMD). Sistem ini merupakan sistem yang digunakan untuk menghitung dan menilai kinerja dosen yang ada di Fakultas Teknologi Industri. Terdapat 4 faktor yang dijadikan komponen dalam menghitung nilai dari tiap-tiap dosen diantaranya, kuesioner, kehadiran mengajar, jabatan (diubah menjadi kesesuaian SAP), dan pengumpulan nilai. Ke empat komponen tersebut dikategorikan lagi berdasarkan status dosen yaitu, dosen tetap dan dosen tidak tetap(Setiaji, 2011). Sistem ini dioperasikan secara lokal yang terletak di *server* FTI UII dan telah beroperasi sejak tahun 2008. Bahasa pemrograman yang digunakan dalam pembuatannya adalah PHP versi 5.1.6 tanpa menggunakan *framework* dan basis data MYSQL.

2.4 Komponen-Komponen Nilai Kinerja Mengajar Dosen

Berdasarkan dokumen Draft Metode Pengukuran Sasaran Mutu 2019, sistem NKMD memiliki total 5 komponen yaitu tingkat kehadiran, tingkat realisasi aktivitas pembelajaran dengan rencana pembelajaran semester (RPS), rata-rata penilaian dari mahasiswa peserta kuliah (kuesioner), Kesesuaian asesmen/penliaian dengan capaian pembelajaran mata kuliah (CPMK), dan ketepatan waktu penyerahan nilai(UII, 2019). Adapun bobot dari tiap komponen dapat dilihat pada Tabel 2.2.

Tabel 2.2 Bobot komponen NKMD

Kode	Komponen	Bobot
Nd_1	Tingkat Kehadiran	10%
Nd_2	Tingkat Realisasi Aktivitas Pembelajaran Dengan Rencana Pembelajaran Semester (RPS)	30%
Nd_3	Rata-rata Penilaian Dari Mahasiswa Peserta Kuliah	20%
Nd_4	Kesesuaian Asesmen/Penilaian Dengan Capaian Pembelajaran Mata Kuliah (CPMK)	30%
Nd_5	Ketepatan Waktu Penyerahan Nilai	10%

Sumber: (UII, 2019)

Adapun perhitungan tiap komponen dijabarkan sebagai berikut(UII, 2019):

a. Komponen tingkat kehadiran dalam mengajar dosen dapat dihitung berdasarkan jumlah pertemuan yang dilakukan secara luar jaringan (luring) atau dalam jaringan (daring). Persentase kehadiran dosen dalam mengajar dihitung berdasarkan rasio realisasi kehadiran dengan jumlah pertemuan yang direncanakan. Adapun persentase penilaian komponen tingkat kehadiran (Nd_1) dapat dilihat pada Tabel 2.3.

Tabel 2.3 Bobot komponen NKMD

Persentase kehadiran : P_h	Komponen
P_h = 100%	4
85% ≤ P_h < 100%	3
75% ≤ P_h < 85%	2
P_h < 75%	0

Sumber: (UII, 2019)

b. Komponen tingkat realisasi aktivitas pembelajaran dengan rencana pembelajaran semester dapat diukur sebagai berikut:

   1. P_{RPS} dapat diartikan sebagai persentase realisasi aktivitas pembelajaran atas RPS di suatu kelas. Rumus Nd_2 ditentukan sebagai berikut.

$$Nd_2 = 4\ x\ P_{RPS} \qquad (2.1)$$

2. Jika jumlah kehadiran dosen tidak mencapai 100%, maka persentase RPS tidak boleh mencapai 100%.
3. Jika jumlah kehadiran dosen mencapai 100%, maka terdapat kemungkinan persentase RPS tidak mencapai 100%.
4. Pemeriksaan dan pengukuran capaian RPS dapat dilakukan oleh Perwakilan Dosen Kelompok Bidang Keahlian (KBK) dan/atau Sekretaris Prodi dan/atau Ketua Prodi.

c. Komponen penilaian dari mahasiswa (kuesioner) diukur dengan cara menghitung rata-rata dari formulir kuesioner yang telah diisi oleh mahasiswa. Minimal jumlah responden yang mengisi formulir kuesioner sebesar 20%.

d. Komponen kesesuaian asesmen/penilaian dengan CPMK didasarkan dari isian formulir Realisasi atas Pengukuran CPMK. Nilai kesesuaian asesmen/penilaian dengan CPMK dapat dilihat pada Tabel 2.4.

Tabel 2.4 Nilai kesesuaian asesmen/penilaian dengan CPMK

Tingkat Kesesuaian	**Nilai Nd_4**
Sangat sesuai (100% CPMK terukur dan sesuai)	4
Sesuai (75% - 99% CPMK terukur)	3
Kurang Sesuai (50% - 74% CPMK terukur)	2
Tidak Sesuai (kurang dari 50% CPMK terukur)	0

Sumber: (UII, 2019)

e. Komponen ketepatan waktu penyerahan nilai dapat diukur berdasarkan kombinasi dari jumlah lembar koreksi dan lama waktu penyerahan nilai akhir. Adapun ketentuan penilaian dari komponen ini dapat dilihat pada Tabel 2.5. Variabel "**B**" pada Tabel 2.5 diartikan sebagai jumlah lembar jawaban per kelas mata kuliah dalam bentuk angka.

Tabel 2.5 Nilai untuk komponen penyerahan nilai

Penilaian untuk ketepatan waktu penyerahan nilai akhir dalam bentuk huruf (dalam hari) dengan jumlah lembar jawaban per kelas Mata Kuliah sebanyak "B"		
Jumlah Lembar B ≤ 50	**Jumlah Lembar B > 50**	**Nilai Nd_5**
≤ 5	≤ 7	4
6 - 8	8 - 10	3
9 - 12	11 - 14	2
13 -16	15 - 17	1
> 16	> 17	0

Sumber: (UII, 2019)

2.5 *Alpha Testing*

Alpha Testing merupakan salah satu metode pengujian yang dilakukan oleh pengguna terhadap sistem yang dibangun atau dikembangkan pada situs pengembang perangkat lunak tersebut dan hasilnya dicatat dan diamati oleh pengembang dalam waktu yang sama(Sawant, Bari, & Chawan, 2012). Dari pengujian ini dapat diperoleh tanggapan dari pengguna mengenai sistem yang diuji(Ananta & Sri, 2013). Pengujian ini bertujuan untuk membuat sistem agar terhindar dari cacat penggunaan, sehingga pengguna tidak kecewa terhadap sistem yang telah dibangun setelah diluncurkan kepada pengguna umum. *Alpha Testing* merupakan pengujian terakhir sebelum perangkat lunak diluncurkan secara resmi ke pengguna(Suhartono, 2016a).

2.6 *Beta Testing*

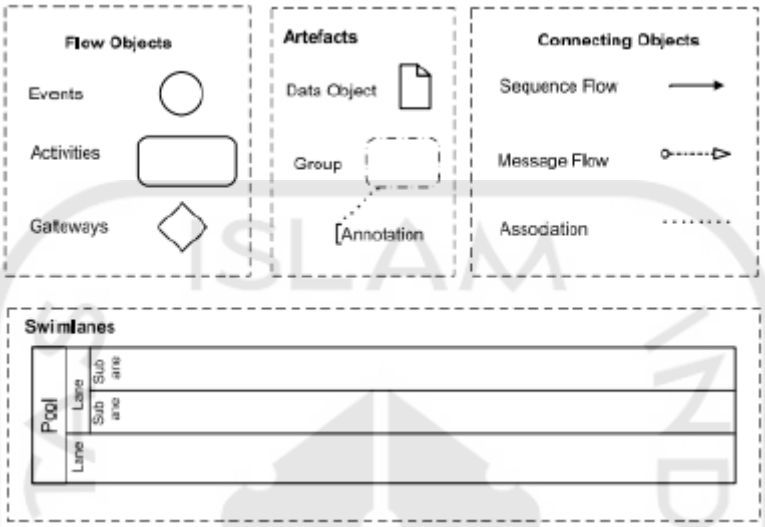
Beta Testing adalah pengujian yang dilakukan untuk memvalidasi kegunaan, fungsi, kompatibilitas, dan uji reliabilitas sistem di lokasi pengguna akhir. Pengujian *beta* dilakukan untuk memastikan dari sisi pengguna bahwa perangkat lunak tersebut bebas dari cacat dan kegagalan. Tujuan pengujian ini adalah untuk mengetahui kekurangan dan mendapatkan masukan dari sisi pengguna akhir(Suhartono, 2016b).

2.7 *Business Process Modeling Notation (*BPMN*)*

Business Process Modeling Notation merupakan bahasa standar yang digunakan dalam menggambarkan proses bisnis, khususnya pada tingkatan analisis domain dan desain sistem tingkat tinggi. BPMN memiliki kemiripan dengan *modeling standard* lainnya yaitu *Unified Modeling Language*. Pengembangan BPMN didasari dari perbaikan notasi lainnya yaitu, UML, IDEF, ebXML, RosettaNet, LOVeM, and Event-driven Process Chain. BPMN dikembangkan oleh *Business Process Management Initiative* (BPMI)(Muehlen & Recker, 2008).

BPMN mewakilkan aspek organisasi dengan *pool* dan *swimlanes*. Hirarki dari *swimlanes* terdiri dari *pool* dengan *lanes* dan *sub-lanes*. *Lanes* digunakan untuk menggambarkan entitas organisasi seperti departemen dalam organisasi. *Sub-lanes* digunakan untuk menggambarkan entitas organisasi seperti organisasi di dalam departemen. Entitas organisasi berperan untuk melakukan objek tertentu yang didefinisikan dalam bentuk grafis. Unsur-unsur notasi yang digunakan dalam BPMN dibagi menjadi empat kategori yang tiap

kategorinya terdiri dari satu set elemen(Huda, 2018). Adapun ke empat kategori tersebut dapat dilihat pada Gambar 2.2.



Gambar 2.2 Kategori elemen BPMN

Sumber: (White & Miers, 2008)

Flow Objects merupakan *building blok* proses bisnis yang di dalamnya terdapat *events, activities,* dan *gateway*. *Events* digunakan untuk mewakilkan kejadian nyata yang sesuai untuk proses bisnis atau kejadian umum yang terjadi. *Activities* adalah pekerjaan-pekerjaan yang dilakukan selaman proses bisnis. *Gateways* dapat digunakan untuk menggabung atau membagi kebiasaan diantara *events* dan *activitites.*

Artefacts adalah kategori elemen dalam BPMN yang dapat digunakan untuk mewakilkan informasi tambahan mengenai proses bisnis yang secara langsung tidak berkaitan dengan alur proses. Di dalam *artefacts* terdapat *data objects, groups,* dan *annotations. Data objects* merupakan dokumentasi data yang digunakan di dalam proses dan menggunakan nama objek sebagai penamaannya. *Annotations* adalah nama dari sebuah group dari elemen proses.

Connecting object digunakan untuk menghubungkan alur dari objek, *swimlanes,* maupun *artefacts. Connecting object* terdiri dari *sequence flow, message flow, association. Sequence flow* merupakan elemen yang digunakan untuk menunjukkan urutan alur objek. *Message flow* adalah elemen untuk mendeskripsikan alur pesan diantara *partner* bisnis yang ada pada *pools. Association* merupakan elemen dari *connecting object* yang digunakan untuk menghubungkan *artefacts* dengan elemen di diagram proses bisnis(Huda, 2018).