**BAB II**

**TINJAUAN PUSTAKA**

1. **Penelitian Terkait**

Wang et al. (2007) melakukan penelitian *text stream mining* dalam menentukan pola topik pada berita berbahasa Inggris dan bahasa China dengan menggunakan gabungan model dari pola teks dan aliran teks walau berbeda bahasa. Tahapan yang dilakukan yaitu (1) menemukan topik di setiap *text stream*; (2) mengalokasikan aliran teks dan topik pada waktu yang sama; (3) mengatur pencarian topik pada *stream* agar lebih efisien pada topik yang berhubungan. Dan hasil penelitiannya adalah menghasilkan cerminan *event* yang sebenarnya dan ringkasan yang baik dari aliran teks tersebut.

Eric, Rajput, dan Veda (2015) melakukan penelitian mengenai *text stream mining* menggunakan *Suffix Tree*. Penelitian ini menggunakan setiap frekuensi *n-grams* untuk mendeteksi *event* pada satu *text stream*. Dengan memperhitungkan nilai p di setiap *n-grams* yang memungkinkan *n-grams* memiliki tingkatan frekuensi secara statistik. Hasilnya adalah mampu mendeteksi *event* dengan cepat.

Satu tahun berikutnya, penelitian Jejurkar dan Kshirsagar (2016) mengenai pengelompokan topik yang akan digunakan sebagai mesin pencari. Penelitian ini menggunakan metode *Suffix Tree Clustering*. Pemrosesan awal dilakukan dengan pembersihan dokumen dan pengembalian kata ke bentuk kata dasar. Tahap selanjutnya yaitu pembentukkan *base cluster* dengan cara menemukan *share phrase* antar dokumen. Untuk menemukan *share phrase* digunakan struktur data *suffix tree*. Dengan menggunakan struktur data ini, maka setiap dokumen akan direpresentasikan menjadi suatu kalimat. Tahapan terakhir yaitu melakukan kombinasi *base cluster* dengan menghitung *similarity* antar *base cluster* terlebih dahulu dan kemudian menemukan keterhubungan antar *base cluster*.

# **2.2 Landasan Teori**

* + 1. ***Trending Topic***

*Trending topic* telah berhasil menarik minat tidak hanya pengguna tetapi jurnalis, pengembang aplikasi *real-time*, dan peneliti sosial. Dengan mengetahui topik yang sedang dibahas pada waktu tertentu dapat mengetahui secara *update* mengenai masalah terkini, dan menemukan masalah utama dalam masyarakat (Zubiaga et al,. 2013). *Trending topic* merupakan urutan kejadian terbanyak yang dibahas dan umumnya terdiri dari frasa pendek atau *keyword* dan tidak memuat banyak penjelasan topik tersebut (Han, Chung, & Kang, 2014).

* + 1. ***Suffix Tree Clustering***

Inti dari suatu hasil pencarian yang menerapkan *clustering* adalah penggunaan algoritma *clustering. Suffix Tree Clustering* (STC) adalah algoritma pertama yang menggunakan frasa (*multi-word terms*) sehingga prosesnya lebih sederhana dibandingkan dengan algoritma yang lain. *Suffix Tree Clustering* juga merupakan algoritma *incremental*, kompleksitas waktu perhitungannya *linear* O(n) dan memenuhi kriteria *document clustering*. Algoritma *Suffix Tree Clustering* (STC) memiliki dua kunci utama, yaitu :

1. Menggunakan *phrase* sebagai dasar pembentukan *cluster*nya*.*

2. Menggunakan suatu definisi *cluster* sederhana.

*Suffix Tree Clustering* memiliki dua langkah utama. Dalam langkah pertama, pencarian *shared phrase* untuk semua dokumen berita yang dikoleksi. Kita menyebut *shared phrase* sebagai *phrase cluster* atau *base cluster,* yang ditemukan dengan menggunakan suatu struktur data yang dinamakan *suffix tree.* Dalam langkah kedua, kita mengkombinasikan *base cluster-base cluster* ke dalam suatu *cluster.* Penggabungan antar dua *base cluster* didasarkan pada jumlah dokumen yang melakukan *overlap* di antara kedua *base cluster* tersebut (Zamir and Etzioni, 1998). Suatu *phrase* yang dimaksud dalam konteks algoritma ini adalah urutan satu atau lebih kata-kata. STCmemiliki tiga langkah utama, yaitu :

1. *Cleaning* Dokumen*.*

2. Identifikasi *Base cluster* menggunakan *Suffix tree.*

3. Mengkombinasikan *Base cluster* ke dalam suatu *cluster*.

Beberapa karakteristik yang membuat S*uffix Tree Clustering* cocok digunakan untuk pengelompokan dokumen*.* Pertama adalah men*generate* *cluster-cluster* untuk pengelompokan dokumen berdasarkan *phrase. Phrase* juga bermanfaat untuk membangun uraian dan keakuratan deskripsi dari *cluster*. Kedua, tidak tergantung pada model data. Hal itu mengasumsikan hanya dokumen dengan topik yang sama yang akan memiliki *shared phrase.* Ketiga, STC memperbolehkan adanya *overlaping* *cluster*. Hal itu sangat penting untuk menghindari pembatasan bahwa setiap dokumen hanya memiliki satu *cluster* saja, karena sering kita jumpai satu dokumen mempunyai lebih dari satu topik, dengan demikian terdapat kemiripan yang lebih dari satu kelompok dokumen. Keempat, STC menggunakan definisi *cluster* yang sederhana. Semua dokumen yang berisi salah satu *phrase cluster* akan menjadi anggota dari *cluster* tersebut.

STC menggunakan *phrase* untuk mendeteksi kemiripan antar dokumen. STC menggunakan *suffix tree* untuk mengidentifikasi *phrase.* Fitur yang membuat suksesnya STC sebagai algoritma *clustering* adalah adanya *overlaping cluster*. Kualitas *cluster* yang terbentuk dari algoritma STC ini akan menurun jika tanpa menggunakan *multiword* *phrase* dan tidak memperbolehkan adanya *overlaping cluster*. Metode ini terdiri dari 3 proses yaitu *stemming,* identifikasi *base cluster,* dankombinasi *base cluster*.

* + - 1. ***Document Cleaning***

*Document Cleaning* adalah tahap awal dalam algoritma *Suffix Tree Clustering*. Pada tahap ini, dokumen yang telah didapat dari proses download akan dibersihkan dan dipersiapkan untuk tahap selanjutnya. Tahapan yang dilakukan terdiri dari *preprocessing*  dan  *stemming* bahasa Indonesia.

* + - * 1. ***Preprocessing***

*Preprocessing* adalah proses awal pengolahan data sebelum proses utama dilakukan. Tahap-tahap yang dilakukan pada *preprocessing* adalah *casefolding*, *tokenizing* dan *stopword removal.*

***Casefolding***

*Casefolding* adalah tahap penyamaan pada dokumen, berupa perubahan huruf besar menjadi huruf kecil dan penghapusan karakter selain angka dan huruf. Hanya huruf ‘a’ hingga ‘z’ yang diterima. Karakter selain angka dan huruf akan dianggap sebagai *delimiter*. Contoh: “Tarif Ruas Tol Ini Bakal Naik pada 2017” menjadi “tarif ruas tol ini bakal naik pada 2017”

***Tokenizing***

*Tokenizing* adalah proses pemisahan kalimat menjadi sekumpulan kata. Contoh: Kalimat “tarif ruas tol ini bakal naik pada 2017” menjadi sekumpulan kata berikut “tarif”, “ruas”, “tol”, “ini”, “bakal”, “naik”, “pada”, dan “2007”.

***Stopword Removal***

*Stopword**Removal* adalah proses menyeleksi kata-kata yang tidak memiliki makna dan tanda baca pada teks atau kalimat yang akan diganti atau dibuang karena tidak diperlukan untuk proses selanjutnya. Contoh “tarif ruas tol ini bakal naik pada 2017” menjadi “tarif ruas tol bakal naik 2017”

* + - * 1. ***Stemming* Bahasa Indonesia**

Tahap *stemming* pada teks berbahasa Indonesia akan menggunakan Algoritma Nazief dan Andriani. Algoritma ini mengacu pada aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan, yaitu imbuhan yang diperbolehkan atau imbuhan yang tidak diperbolehkan. Pengelompokan ini termasuk imbuhan kata awalan (prefiks), imbuhan kata akhiran (sufiks), imbuhan kata sisipan (infiks) dan kombinasi imbuhan pada awal dan akhir kata (konfiks) (Asian, Williams, & Tahaghoghi, 2005). Algoritma ini menggunakan kamus kata keterangan yang digunakan untuk mengetahui bahwa proses *stemming* telah mendapatkan kata dasar. Setiap kata dalam bahasa Indonesia umumnya memiliki kombinasi 2 awalan (prefiks) dan 3 akhiran (sufiks).

Dapat ditulis menjadi :

**P1 + P2 + KD + S1 + S2 + S3**

(II.1)

dimana pada persamaan (II.1) :

P1, P2 = Awalan (prefiks) kata pertama dan kedua.

KD = Kata Dasar.

S1, S2, S3 = Akhiran (suffiks) kata pertama, kedua, dan ketiga.

Algoritma Nazief & Adriani yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut:

1. Pertama cari kata yang akan distem dalam kamus kata dasar. Jika ditemukan maka diasumsikan kata adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   1. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   2. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   1. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   2. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan Recoding.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya. Contohnya pada aturan awalan “te” pada table II-2. Misalkan kata “terlambat” distem. Setelah menghapus awalan “te” akan menjadi “rlmabat”. Awalan pertama diambil berdasarkan aturan Set 1. Dalam hal ini, setelah awalan “te” adalah huruf “r”, dan sesuai dengan lima aturan yang sama pada Set 1. Setelah huruf “r” diikuti huruf “l” (Set 2), dan sesuai dengan tiga aturan sama pada Set 2 yaitu bukan vokal atau “-r-” . Setelah huruf “l” adalah kata “ambat”, sehingga pada Set 3 menjelaskan bahwa tidak ada huruf “-er-“ yang mengikuti dan bisa ditentukan bahwa awalan pada kata “terlambat” adalah “ter”.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah bukan “none” maka awalan dapat dilihat pada Tabel 2. Hapus awalan jika ditemukan.

Tabel II-1. Kombinasi Awalan Akhiran Yang Tidak Diijinkan

|  |  |
| --- | --- |
| Awalan | Akhiran yang tidak diizinkan |
| be- | -i |
| di- | -an |
| ke- | -i, -kan |
| me- | -an |
| se- | -i, -kan |

Tabel II-2. Cara Menentukan Tipe Awalan Untuk awalan “te-”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aturan Huruf | | | | Tipe Awalan |
| Set 1 | Set 2 | Set 3 | Set 4 |
| “-r-“ | “-r-“ | – | – | none |
| “-r-“ | vokal | – | – | ter-luluh |
| “-r-“ | bukan (vokal atau “-r-”) | “-er-“ | vokal | ter |
| “-r-“ | bukan (vokal atau “-r-”) | “-er-“ | bukan vokal | ter- |
| “-r-“ | bukan (vokal atau “-r-”) | bukan “-er-“ | – | ter |
| bukan (vokal atau “-r-”) | “-er-“ | vokal | – | none |
| bukan (vokal atau “-r-”) | “-er-“ | bukan vokal | – | te |

Tabel II-3. Jenis Awalan Berdasarkan Tipe Awalannya

|  |  |
| --- | --- |
| Tipe Awalan | Awalan yang harus dihapus |
| di- | di- |
| ke- | ke- |
| se- | se- |
| te- | te- |
| ter- | ter- |
| ter-luluh | ter |

Untuk mengatasi keterbatasan pada algoritma di atas, maka ditambahkan aturan-aturan dibawah ini:

1. Aturan untuk reduplikasi.

* Jika kedua kata yang dihubungkan oleh kata penghubung adalah kata yang sama maka *root word* adalah bentuk tunggalnya, contoh : “buku-buku” *root word*-nya adalah “buku”.
* Kata lain, misalnya “bolak-balik”, “berbalas-balasan, dan ”seolah-olah”. Untuk mendapatkan *root word*-nya, kedua kata diartikan secara terpisah. Jika keduanya memiliki *root word* yang sama maka diubah menjadi bentuk tunggal, contoh: kata “berbalas-balasan”, “berbalas” dan “balasan” memiliki *root word* yang sama yaitu “balas”, maka *root word* “berbalas-balasan” adalah “balas”. Sebaliknya, pada kata “bolak-balik”, “bolak” dan “balik” memiliki *root word* yang berbeda, maka *root word*-nya adalah “bolak-balik”.

2. Tambahan bentuk awalan dan akhiran serta aturannya.

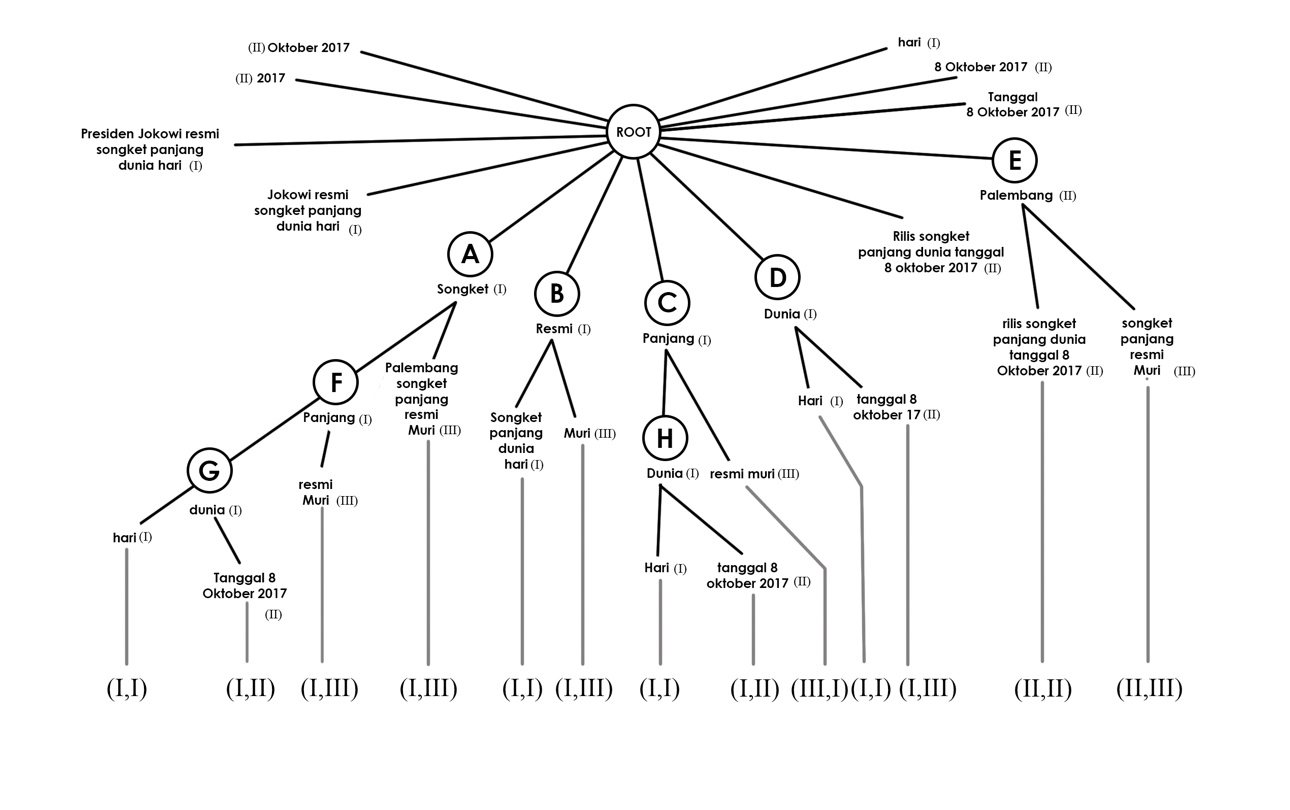
* Untuk tipe awalan “mem-“, kata yang diawali dengan awalan “memp-” memiliki tipe awalan “mem-”.
* Tipe awalan “meng-“, kata yang diawali dengan awalan “mengk-” memiliki tipe awalan “meng-”.
  + - 1. **Identifikasi *Base cluster***

Tahap identifikasi *base cluster* merupakan tahap terpenting dalam algoritma *suffix tree clustering*, karena pada tahap ini akan menghasilkan *cluster*-*cluster* dasar (Zamir & Etzioni, 1998). Pembentukkan *base cluster* dilakukan dengan cara menemukan share *phrase* antar dokumen. Untuk menemukan share *phrase* digunakan struktur data *suffix tree*. Dengan menggunakan struktur data ini, maka setiap dokumen akan direpresentasikan menjadi suatu kalimat. Untuk menemukan *base cluster* dapat dilakukan dengan cara membuat suatu *invert index* dari *phrase* untuk semua dokumen.

Tuwohingide dkk. (2016) menyatakan bahwa proses pembentukan *phrase* *suffix tree* terdiri dari beberapa langkah, yaitu:

* 1. Mengidentifikasi semua kalimat yang terdapat dalam dokumen.
  2. Setiap akhiran kalimat akan diidentifikasi sebagai *phrase.*
  3. Melakukan pengecekan apakah *phrase* sudah ada pada struktur suffix tree yang telah terbentuk. Apabila *phrase* sudah ada, maka dilakukan penambahan informasi *phrase* berupa nomor dokumen dan posisi *phrase* ke dalam *node* yang mewakili *phrase* tersebut. Apabila *phrase* tidak ada pada struktur *suffix tree*, maka *phrase* akan ditambahkan sebagai *node* baru disertakan dengan label *phrase* yang baru ditambahkan.

Contoh pembentukkan *suffix tree* untuk kalimat Presiden Jokowi telah meresmikan songket terpanjang di dunia hari ini pada dokumen 1, Palembang telah merilis songket terpanjang di dunia pada tanggal 8 Oktober 2017 pada dokumen 2, dan sonket Palembang telah menjadi songket terpanjang yang diresmikan MURIpada dokumen 3 ditunjukkan pada Gambar II-1. Pada Gambar II-1 menunjukkan adanya internal *node* yang terbentuk. Setiap internal *node* merepresentasikan suatu kelompok dokumen dan share *phrase* untuk kelompok tersebut. Oleh karena itu, setiap internal *node* juga merepresentasikan *base cluster* yang terbentuk.

Gambar II-1 Pembentukan *Suffix Tree*

Pada tahap pemilihan *base cluster*, setiap *phrase* yang terlabeli di internal *node* dan *leaf node* yang memiliki anak akan dianggap sebagai *base cluster*, sementara *phrase* yang terlabeli di *leaf*  *node* yang tidak memiliki anak akan diabaikan. Pada contoh Gambar II-1, *node* yang akan terpilih sebagai *base cluster* adalah *node* A, B, C, D, E. *Node* F, G, H, I tidak akan diproses karena *phrase* yang terlabeli di *node* tersebut telah di terlabeli di internal *node*. Proses ini dilakukan untuk mereduksi jumlah *phrase* yang akan menjadi *base cluster* dan menghapus *node* yang terlabeli dengan *phrase* yang sama. Semua *base cluster* yang terbentuk dapat ditunjukkan pada Tabel II-4.

Tabel II-4. *Base cluster* yang Terbentuk

|  |  |  |
| --- | --- | --- |
| ***Base cluster*** | ***Phrase*** | **Documents** |
| A | Songket | I,II,III |
| B | Resmi | I,III |
| C | Panjang | I,II,III |
| D | Dunia | I,III |
| E | Palembang | II,III |
| F | Songket Panjang | I,II,III |
| G | Songket Panjang Dunia | I,II |
| H | Panjang Dunia | I,II,III |

Setiap *base cluster* yang terbentuk memiliki suatu *score*. Penghitungan score merupakan suatu fungsi dari jumlah dokumen yang masuk anggota *base cluster* dan jumlah kata yang menyusun *phrase* dari *base cluster*. Fungsi untuk menghitung *score* *base cluster* ditunjukkan oleh Persamaan (II.2).

*s(B) = |B|.f(|P|)* (II.2)

dimana pada persamaan (II.2) :

*|B|* =jumlah dokumen di dalam *base cluster* *B* dan

*|P|* = jumlah kata yang menyusun frase *P*.

* + 1. **Metode Pengembangan Perangkat Lunak**

Metode pengembangan perangkat lunak dalam tugas akhir ini menggunakan metode Rational Unified Process (RUP). Adapun penjelasan tentang metode RUP tersebut dijelaskan sebagai berikut.

* + - 1. ***Rational Unified Process* (RUP)**

Metode RUP merupakan paradigma pengembangan perangkat lunak yang bersifat dinamis dengan pemodelan menggunakan bahasa pemodelan *Unified Modeling Language* (UML)*.* Metode RUP dalam stuktur alur kegiatannya dapat melakukan iterasi/perulangan ke fase sebelumnya tanpa harus menunggu seluruh alur kegiatan dalam proses selesai dilakukan. Dengan demikan, alur kegiatan pengembangan perangkat lunak metode RUP dinamakan juga dengan struktur dinamis *(dynamic structure)* (Kruchten, 2000).

UML merupakan pemodelan aktivitas dari perangkat lunak dalam bentuk visualisasi, spesifik, dan dokumentasi dari perangkat lunak dalam bentuk grafis. UML menjelaskan kepada pengembang tentang ruang lingkup dari komponen perangkat lunak seperti: keterkaitan antara suatu kelas dengan kelas yang lain, skema basis data, dan skema bisnis proses. Dengan demikian, UML merupakan dokumentasi cetak biru *(blue print)* dari perangkat lunak.

Metode RUP memiliki tiga karakteristik, yaitu *use-case driven*, *architecture-centric*, dan *iterative and incremental*. *Use case driven* merupakan pemodelan perangkat lunak yang dikembangkan berlandaskan pada *use case*. *Use case* berfungsi untuk menjembatani komunikasi antara pengguna dengan perangkat lunak. Sehingga, *use case* mampu menyajikan apa yang seharusnya perangkat lunak lakukan dari sudut pandang pengguna.

*Architecture centric* merupakan komposisi dari struktur elemen-elemen yang memiliki hubungan keterkaitan satu dengan yang lainnya, baik antarelemen maupun elemen dengan perangkat lunak. Selain itu, arsitektur dapat juga menjelaskan funsionalitas dan unjuk kerja *(performance)* dari perangkat lunak. Selanjutnya, *iterative* merupakan perulangan yang dilakukan pada fase-fase tertentu untuk memenuhi kebutuhan dari perangkat lunak. Sedangkan, *incremental* merupakan jumlah dari perulangan yang dilakukan pada fase-fase tertentu. Penjelasan di atas dapat digambarkan pada Gambar II-3.



Gambar II-3 Arsitektur RUP

(Sumber: Kruchten, 2000)

Struktur proses dalam arsitektur RUP secara keseluruhan terdiri atas dua dimensi. Berdasarkan Gambar II-3, dimensi pertama digambarkan secara horizontal. Dimensi ini menjelaskan aspek dinamis dari proses pengembangan perangkat lunak yang meliputi: siklus, fase, iterasi, dan *milestones*. Dimensi ini terdiri dari beberapa tahapan, yaitu: insepsi *(inception)*, elaborasi *(elaboration)*, konstruksi *(construction)*, dan transisi *(transition)*. Dimensi kedua ditunjukkan melalui garis vertikal. Dimensi ini menjelaskan aspek statis dari metode RUP yang meliputi: aktivitas, alur kerja, dan pekerja yang bertanggung jawab terhadap suatu proses. Alur kerja pada dimensi kedua ini terdiri atas: *business modeling*, *requirement*, *analysis and design*, *implementation*, *test*, *deployment*, *configuration and change management*, *project management* dan *environment*.

Aktivitas penelitian secara umum dalam metode RUP pada setiap fasenya dapat dijelaskan sebagai berikut:

1. Fase insepsi *(inception)* merupakan tahapan awal dalam pengembangan perangkat lunak. Aktivitas dalam fase ini berfokus pada pemodelan bisnis. Pemodelan bisnis tersebut antara lain: mendefinisikan masalah, membatasi ruang lingkup proyek, dan membangun *business case*;
2. Fase elaborasi *(elaboration)* merupakan tahap kedua dari pengembangan perangkat lunak. Fase elaborasi menjelaskan aktivitas yang paling dominan pada analisis kebutuhan perangkat lunak, perancangan, dan mendaftar resiko-resiko yang mungkin terjadi selama masa pengembangan perangkat lunak berlangsung;
3. Fase konstruksi *(construction)* merupakan tahapan yang mengimplementasikan kebutuhan perangkat lunak ke dalam modul-modul dalam bentuk kode program. Selain itu, aktivitas pada fase ini juga berfokus terhadap pengujian perangkat lunak. Sehingga, perangkat lunak yang dihasilkan berkualitas dan sesuai dengan kebutuhan;
4. Fase transisi *(transition)* merupakan fase terakhir dalam metode RUP. Aktivitas yang paling dominan dilakukan pada fase ini adalah mempersiapkan lingkungan sistem operasi perangkat lunak, melakukan perawatan perangkat lunak, persiapan fase *closing*, dan penyelesaian dokumentasi perangkat lunak.