PERBANDINGAN PROSES PENGEMBANGAN PERANGKAT LUNAK MODEL SPIRAL DAN CLEANROOM

I Made Sunia Raharja [1)], **Azhari SN** [2)]
[1,2)] Program Studi Monodisiplin S2/S3 Ilmu Komputer UGM
Gedung SIC Lt.3 FMIPA UGM Sekip Utara Bulaksumur Yogyakarta 55281 Telp/Fax:(0274)555133
e-mail : i.made.s.r@mail.ugm.ac.id[1)], arisn@ugm.ac.id[2)]

Abstrak
Perangkat lunak saat ini sangat berperan dalam kehidupan manusia diberbagai bidang baik secara individu maupun sosial. Perangkat lunak bahkan memiliki peran yang sangat kritis dalam bidang tertentu dimana kesalahan atau error menjadi hal yang sangat serius. Kebutuhan manusia yang berubah-ubah juga berpengaruh terhadap perkembangan perangkat lunak yang mau tidak mau harus dapat ikut berubah. Model proses pengembangan perangkat lunak evolusioner memungkinkan adanya perubahan terhadap perangkat lunak. Model spiral dan Cleanroom adalah model pengembangan evolusioner dan memberikan perhatian yang lebih terhadap kesalahan perangkat lunak. Dua model ini sudah sangat diperhitungkan dan banyak digunakan pada proyek skala besar dan kritis. Paper ini akan membandingkan Model proses pengembangan spiral dan cleanroom.

***Kata Kunci** : Perangkat Lunak, Model Proses Perangkat Lunak, Spiral, Cleanroom*

1. PENDAHULUAN

Perangkat lunak sudah lama menjadi bagian dari masyarakat modern, perangkat lunak menjadi alat yang mengendalikan pengambilan keputusan di dalam dunia bisnis, berfungsi sebagai dasar dari semua bentuk pelayanan atau penelitian keilmuan modern dan sudah banyak dilekatkan dalam segala bentuk sistem, misalnya sistem transportasi, medis, telekomunikasi, militer, proses industri, hiburan(Pressman, 1997). Saat ini suatu organisasi sangat bergantung dari perangkat lunak sebagai akibat dari berkembanganya teknologi komputer. Perangkat lunak sangat membantu dalam melakukan suatu proses bisnis yang kompleks menjadi lebih mudah, cepat dan efisien. Selain untuk proses bisnis perangkat lunak sangat populer digunakan untuk permainan dan media sosial.

Perangkat lunak dikembangkan menggunakan teknologi rekayasa perangkat lunak. Pengembangan perangkat lunak meliputi teknologi yang mengelompokkan proses, metode teknis, serta alat-alat otomatisasi, rangkaian proses merupakan fondasi untuk rekayasa perangkat lunak, proses-proses tersebut menentukan kerangka kerja untuk serangkaian area proses yang harus dibangun sehingga akan mengefektifkan penggunaan teknologi pengembangan perangkat lunak(Paulk, 1993).

Proses perangkat lunak merupakan suatu metodologi yang digunakan dalam proses pembuatan perangkat lunak. Metodologi ini membentuk model perencanaan dan pengendalian dalam proses pembuatan perangkat lunak. Sejak abad ke 20 hingga abad 21 sekarang ini sudah banyak proses perancangan perangkat lunak dikembangkan dengan berbagai model dan tipe, walaupun memiliki model dan tipe yang bervariasi, tujuan dari rekayasa perangkat lunak adalah untuk menciptakan suatu kerangka kerja yang sesuai untuk membangun perangkat lunak yang berkualitas(Mohammed, 2010).

Perangkat lunak seperti semua sistem kompleks yang lain, dapat berkembang dari satu periode waktu ke periode waktu lainya(Gilb,1988). Kebutuhan dan proses bisnis dapat berubah seiring dengan laju perkembangan penggunaanya. Untuk mengatasi situasi ini perancang perangkat lunak membutuhkan model proses yang dapat mengakomodasi perkembangan produk sepanjang waktu(evolusioner). Beberapa model proses perangkat lunak evolusioner telah dikembangkan salah satunya adalah pengembangan perangkat lunak model *spiral*. Pengembangan evolutioner adalah pengembangan dengan iterasi proses yang memungkinkan perekayasa perangkat lunak mengembangkan perangkat lunak menjadi lebih baik. Proses perangkat lunak yang menggunakan pendekatan iteratif lainya adalah pengembangan perangkat lunak *cleanroom.* Proses *cleanroom* menjamin perangkat lunak yang bebas cacat/kesalahan dengan menggunakan metode formal.

Tujuan dari makalah ini adalah untuk mendeskripsikan proses pengembangan perangkat lunak, tahap-tahap pengembangan, persamaan, perbedaan, keunggulan dan kelemahan dari proses pengembangan model *spiral* dan *cleanroom*.

2. TINJAUAN PUSTAKA

2.1 PROSES PERANGKAT LUNAK

Proses perangkat lunak adalah sekumpulan aktifitas yang bertujuan untuk menghasilkan suatu produk perangkat lunak. Proses perangkat lunak sangat kompleks dan sangat tergantung dari penilaian dan pengambilan

keputusan yang kreatif dari pengembang. Aktifitas dasar yang umum dari proses perangkat lunak adalah(Sommerville, 2004) :

a. Spesifikasi Perangkat Lunak:
   Mendefinisikan fungsionalitas dan operasi-operasi perangkat lunak.
b. Desain dan Implementasi Perangkat Lunak: Menghasilkan produk perangkat lunak sesuai dengan permintaan pengguna.
c. Validasi Perangkat Lunak:
   Perangkat lunak harus divalidasi apakah sudah sesuai dengan yang diinginkan pengguna.
d. Evolusi Perangkat Lunak:
   Perangkat lunak harus berkembang untuk memenuhi perubahan kebutuhan pengguna.

2.2 MODEL PROSES PERANGKAT LUNAK

Model proses perangkat lunak adalah representasi abstrak dari proses perangkat lunak. Setiap model proses merepresentasikan sebuah proses pengembangan perangkat lunak dari sudut pandang tertentu(Sommerville, 2004). Model proses pengembangan perangkat lunak berbeda dengan metodologi pengembangan perangkat lunak. Suatu metodologi melakukan pengendalian melalui suatu tahapan(spesifikasi data, pengalokasian kebutuhan, pengendalian,dll), sedangkan model mempunyai kepentingan dalam memberikan pedoman dalam urutan tertentu(beberapa tahapan) dalam melaksanakan tugas-tugas yang perlu dilakukan untuk menyelesaikan suatu proyek perangkat lunak(Boehm, 1988). Model proses pengembangan perangkat lunak yang paling pertama dipublikasikan adalah model Waterfall, model ini merupakan turunan dari proses rekayasa sistem secara umum(Royce, 1997). proses pengembangan perangkat lunak model ini melalui beberapa tahapan secara sekuensial linier. Pada model ini proses dalam suatu tahapan harus benar-benar selesai sebelum memulai proses pada tahap selanjutnya.

Kebutuhan proses bisnis dan kebutuhan produk perangkat lunak dapat berubah-ubah selama proses pengembangan, hal ini dapat dianalogikan bahwa pengembangan perangkat lunak membidik sasaran yang bergerak(berubah-ubah)(Davis, 1988), sehingga membuat suatu produk sebagai hasil akhir dari proses pengembangan menjadi tidak realistis, selain itu batas waktu permintaan pasar yang ketat terhadap penyelesaian suatu produk perangkat lunak yang komprehensif menjadi suatu hal yang tidak mungkin. Namun untuk menanggulangi tekanan bisnis dan pasar yang kompetitif diperlukan suatu ukuran produk sebagai hasil akhir proses. Pengembang perangkat lunak memerlukan model proses yang dapat mengakomodasi kebutuhan yang berevolusi terus-menerus. Proses pengembangan evolusioner memungkinkan perekayasa perangkat lunak mengembangkan suatu produk perangkat lunak menjadi versi yang lebih lengkap secara bertahap(Pressman, 1997).

Model proses perangkat lunak terus-menerus dikembangkan untuk menghadapi permasalahan dan tantangan dalam pembuatan sistem perangkat lunak. Salah satu tantangan dalam pengembangan sistem perangkat lunak adalah penanganan kesalahan(*error*). Penanganan kesalahan menjadi penting dalam pengembangan sistem yang kompleks dan kritis. Salah satu variasi model proses perangkat lunak yang penting adalah pengembangan sistem model formal. Pengembangan sistem model formal ini melibatkan penggunaan metode formal dalam pembuatan perangkat lunak(Sommerville, 2004). Pengembangan model formal meliputi sekumpulan aktivitas yang bertujuan untuk spesifikasi matematis dari sistem perangkat lunak. Sebagai contoh adalah penggunaan notasi matematika dalam proses pembangunan dan verifikasi sistem. Metode formal memungkinkan perekayasa perangkat lunak untuk membangun produk sistem perangkat lunak yang bebas dari kesalahan.

2.1.1 PENGEMBANGAN EVOLUSIONER

Ide dari proses pengembangan evolusioner adalah membangun suatu implementasi awal, memperlihatkan hasil implementasi ke pengguna dan mempertimbangkan ide-ide dan masukan yang diberikan. Perbaikan dan penambahan dilakukan melalui pembuatan beberapa versi yang berbeda sampai sistem yang memenuhi keinginan pengguna sudah tercapai.

Dua tipe dasar pengembangan evolusioner adalah(Sommerville, 2004) :

a. *Exploratory development*:
   Sasaran tipe ini adalah bekerjasama dengan pelanggan. Pembangunan sistem dimulai dari bagian sistem yang sudah dipahami terlebih dahulu. Sistem kemudian berkembang dengan penambahan-penambahan yang diinginkan oleh pelanggan.
b. *Throwaway prototyping*:
   Sasarannya adalah memahami kebutuhan yang diinginkan pelanggan dan kemudian membuat definisi yang lebih baik untuk pembuatan sistem. Pembuatan *prototype* adalah fokus pada eksperimen terhadap kebutuhan pelanggan yang sulit dipahami.

2.3 ITERASI PROSES

Dalam proyek perangkat lunak berskala sedang dan besar kadang perubahan dari perangkat lunak tidak dapat dihindari, kebutuhan perangkat lunak berubah seiring dengan penyesuaian bisnis proses agar dapat memenuhi tuntutan eksternal. Perubahan dapat terjadi diberbagai sisi, prioritas manajemen bisa berubah, desain dan implementasi dapat berubah sesuai dengan perkembangan teknologi. Aktivitas proses terjadi berulang-ulang secara berkala agar dapat menangani perangkat lunak yang merespon perubahan yang terjadi.

Dua jenis proses yang sudah secara eksplisit didesain untuk mendukung iterasi proses adalah(Sommerville, 2004):

a. *Incremental delivery* : diamana proses pesifikasi perangkat lunak, desain dan implementasi, dipecah-pecah menjadi rangkaian-rangkaian tahapan yang akan dikerjakan sesuai dengan urutan.
b. *Spiral Development* : proses pengembangan sistem yang berbentuk spiral bergerak secara spiral mulai dari pemikiran-pemikiran awal yang ada pada bagian dalam bergerak keluar menuju ke bagian paling luar yang merupakan proses penyelesaian pengembangan sistem.

2.4 MODEL FORMAL

Peran konkurensi proses dalam suatu sistem yang kompleks dapat menjadi sangat krusial. Pengembangan perangkat lunak yang kompleks dengan menggunakan metode konvensional, kesalahan tidak dapat dihindari. Cara yang digunakan untuk mencegah adanya kesalahan adalah dengan menuntut praktisi pengembang agar perangkat lunak yang dibangun dapat diselesaikan dan dieksekusi secepatnya sehingga bila ada kesalahan dapat ditanggulangi(Groote, 2011). Walaupun sudah ada penanggulangan kesalahan untuk kasus tertentu, kesalahan tetap akan muncul pada kasus-kasus yang belum diujikan.

Model Formal adalah model pengembangan perangkat lunak yang menerapkan metode formal. Metode formal yang digunakan dalam pengembangan perangkat lunak adalah tehnik untuk mendeskripsikan atribut-atribut sistem dengan berlandaskan teori matematika. Metode formal memberikan suatu kerangka kerja sehingga pengembang dapat menspesifikasikan, mengembangkan dan memverifikasi sistem secara sistematis. Metode formal dapat mendefinisikan pengertian dari konsistensi, ketuntasan, spesifikasi, implementasi dan kebenaran yang lebih relevan(Marciniak, 1994).

Metode formal memungkinkan suatu pengembangan sistem yang kompleks dengan pondasi matematis yang kokoh sehingga dapat menghasilkan produk perangakat lunak yang berkualitas lebih baik dibandingkan dengan perangkat lunak yang dibangun dengan metode konvensional. Sebagai contoh teknik sudah banyak diaplikasikan sebagai verifikasi terhadap perilaku diskrit dari bermacam-macam sistem industri kritis. Telah diketahui bahwa beberapa kesalahan yang fatal tidak akan ditemukan melalui pengujian secara tradisional, dan dalam situasi tertentu kesalahan yang tidak ditemukan akan memberikan kerusakan serius(Boehm, 1988).

3. METODE PENELITIAN

3.1 JENIS PENELITIAN

Penelitian ini adalah studi literatur untuk membandingkan dua buah model pengembangan sistem perangkat lunak yaitu model spiral dan cleanroom. Apakah kedua model tersebut memiliki suatu persamaan atau perbedaan sebagai suatu model proses pengembangan perangkat lunak. Penelitian yang dilakukan adalah mengidentifikasi persamaan-persamaan, perbedaan-perbedaan, keunggulan, kelemahan dan penerapan model pengembangan sistem perangkat lunak *Spiral* dan *Cleanroom* yang sudah pernah dilakukan.

3.2 HIPOTESA

Hipotesa awal dalam penelitian ini adalah model proses *Spiral* dan *Cleanroom* suatu persamaan tertentu karena kedua model tersebut adalah model proses pengembangan perangkat lunak evolusioner, selain itu kedua model proses itu memiliki elemen-elemen yang berkaitan dengan kualitas perangkat lunak yang dihasilkan.

3.3 DATA

Jenis data yang digunakan dalam penelitian ini adalah buku dan makalah-makalah yang berkaitan dengan rekayasa perangkat lunak dengan model Spiral dan Cleanroom.

Pada penelitian ini data model proses pengembangan perangkat lunak ditinjau aktifitas yang dilakukan pada setiap tahapnya sebagai suatu proses perangkat lunak. Kemudian melakukan pengumpulan informasi mengenai penerapan kedua model proses pada proyek yang sudah pernah dilakukan.

4. HASIL DAN PEMBAHASAN

4.1 PROSES PENGEMBANGAN MODEL SPIRAL

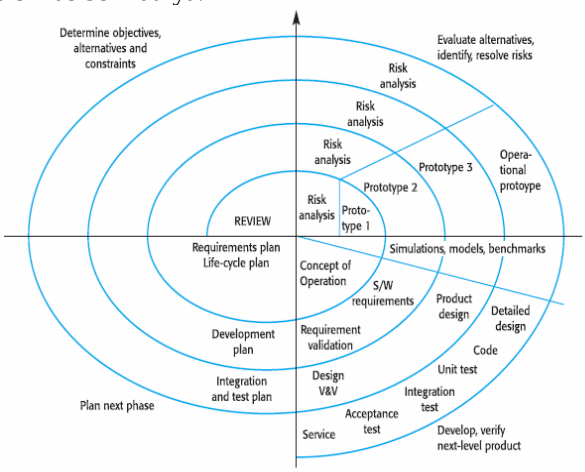
Model spiral ditemukan oleh Barry Boehm, berdasarkan pengalaman dari perbaikan-perbaikan model Waterfall yang diaplikasikan ke proyek pengembangan sistem perangkat lunak berskala besar(Awad, 2005). Pengembangan model spiral merupakan salah satu model pengembangan perangkat lunak evolusioner(Boehm, 1988) dan pembuatan *prototype* dengan dikombinasikan dengan iterasi proses. *Prototype* dibuat dengan kontrol

dan aspek sistematis dari model sekuensial linier(Pressman, 1997). Fokus tujuan dari model spiral adalah mengunakan kemudahan dan menghindarkan hambatan yang ada pada model lainya dengan mengarahkan manajemen ke penekanan terhadap evaluasi resiko dan resolusi(Oriogun, 2000).

Sebagai salah satu proses pengembangan perangkat lunak, proses model spiral melalui proses spesifikasi perangkat lunak, proses spesifikasi perangkat lunak dalam model spiral terjadi dalam tahap awal pengembangan perangkat lunak, proses spesifikasi ini ditujukan untuk bentuk dasar atau *prototype* perangkat lunak. Tahap awal pengembangan dapat dikatakan sebagai proses identifikasi diantaranya adalah menentukan sasaran ukuran produk yang akan dikembangkan(performa, fungsionalitas, kemampuan menakomodasi perubahan, dll). Menentukan alternatif-alternatif dalam implementasi misalnya(penggunaan komponen yang sudah ada atau membuat yang baru, alternatif-alternatif desain dll). Menentukan batasan-batasan yang ada dalam menggunakan alternatif yang sudah didapatkan sebelumnya(biaya, jadwal, dll). Setelah tahap awal kemudian dilanjutkan dengan tahap evaluasi terhadap alternatif yang sudah digunakan apakah sudah sesuai dengan sasaran dan tidak melewati batasan-batasan yang sudah ditentukan, pada umumnya proses ini rentan ditemukannya ketidakpastian dan resiko-resiko pengembangan perangkat lunak, karena itu diperlukan analisis resiko dan tindakan penanggulangan resiko, diantaranya adalah pembuatan *prototype*, simulasi, perbandingan, melakukan kuisioner ke pengguna dll. Dalam model spiral ini desain, implementasi dan evaluasi sudah dilakukan pada tahap ini produk perangkat lunak sudah dibuat namun masih dalam bentuk *prototype*, *prototype* ini dikomunikasikan dengan pelanggan atau pengguna sistem.

Jika *prototype* belum dapat mencapai ukuran produk atau hanya dapat menangani resiko minimal maka proses dilanjutkan dengan perencanaan terhadap pengembangan *prototype* supaya dapat mencapai sasaran ukuran yang diinginkan. Kemudian pada siklus spiral berikutnya proses identifikasi akan dilewati dan proses berlanjut ke fokus terhadap analisis resiko dan pembuatan *prototype* berikutnya. Namun jika *prototype* sudah memenuhi ukuran produk maka proses dilanjutkan dengan pengembangan *prototype* sebagai evolusi dari perangkat lunak. Proses kemudian ditujukan untuk pengembangan prototype menjadi produk perangkat lunak yang utuh, analisis resiko lebih kepada resiko dengan cakupan yang lebih luas, tahap pembuatan *prototype* tidak dilakukan dan diganti dengan proses yang berbeda.

Pengaturan resiko yang terbagi-bagi pada model spiral dapat memberikan strategi penentuan tertentu melihat dari resiko program dan efektivitas dari teknik penanganan resiko yang bersifat relatif. Selain itu pertimbangan yang dilakukan dalam manajemen resiko juga dapat menentukan ukuran waktu dan usaha khusus untuk aktifitas proyek tertentu misalnya perencanaan, manajemen konfigurasi, jaminan kualitas, verifikasi formal, dan pengujian. Misalnya pada spesifikasi berorientasi resiko dapat memberikan variasi ukuran ketuntasan dan formalitas tergantung tingkat resiko yang relatif, apakah terlalu banyak atau sedikit dalam melakukan spesifikasi. Hal yang terpenting dalam model spiral adalah pembahasan hasil dari proses pengembangan sistem dalam satu siklus spiral yang melibatkan orang atau organisasi yang penting dalam pengembangan perangkat lunak. Pembahasan yang dilakukan mencakup semua hasil dari siklus proses sebelumnya termasuk perencanaan untuk semua yang diperlukan pada siklus berikutnya.



Gambar 1. Proses Model Spiral(Sommerville, 2004)

4.2 PROSES PENGEMBANGAN MODEL CLEANROOM

Pengembangan sistem model Cleanroom merupakan sebuah pendekatan untuk memenuhi kebutuhan akan software yang bebas kesalahan sejak masih dalam tahap pengembangan. Daripada menggunakan siklus klasik (analisis, desain, coding, pengetesan, dan proses debugging), pendekatan cleanroom menggunakan sudut pandang yang berbeda. Yang menjadi perhatian utama dari proses pengembangan model *cleanroom* adalah menghilangkan proses debugging yang memerlukan biaya cukup besar dengan meningkatan kebenaran penulisan kode sejak awal dan memverifikan kebenarannya sebelum pengetesan dilakukan. Dengan peningkatan kebenaran

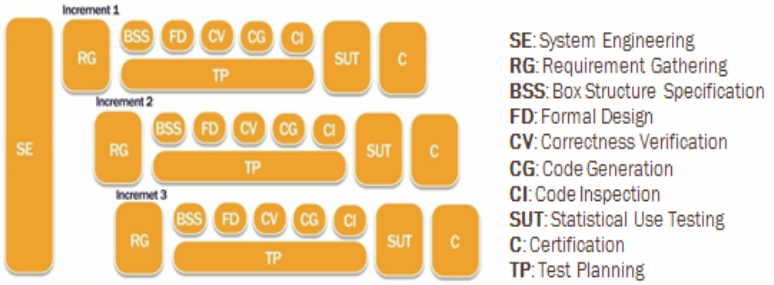
ini diharapkan nantinya akan dihasilkan produk perangkat lunak yang memiliki tingkat kehandalan yang bermutu. Proses pengembangan model *cleanroom* ditemukan oleh Harlan Mills dan Alan Hevner(Mills, 1987).

Pendekatan model cleanroom merupakan pengembangan dari model rekayasa sistem konvensional yang bersifat ingkremental, proses ingkremental berdasarkan pada "*pipeline of software increments*"(Linger, 1994) yang melibatkan tim rekayasa perangkat lunak untuk setiap pipeline(jalur) ingkremen. Model proses cleanroom diawali oleh tim yang menganalisa dan mengklarifikasi kebutuhan(requirements) dari pelanggan, jika terjadi keraguan maka tim membuat suatu prototipe untuk memperoleh masukkan secara iteratif(Linger, 1994). Proses dilanjutkan ke spesifikasi fungsional, setelah spesifikasi fungsional dari sistem sudah didapatkan maka pipeline(jalur) setiap inkremen dari proses cleanroom sudah dapat dikerjakan.

Proses spesifikasi fungsional dari model cleanroom sesuai dengan prinsip-prinsip analisis operasional(Pressman, 1977), dimana dalam proses ini dilakukan pemodelan terhadap data, fungsi dan prilaku dari sistem. Prinsip-prinsip analisis operasional diterapkan dengan metode *box structure specification*(Hevner, 1993), dalam hal ini sebuah *box* merupakan enkapsulasi dari sistem atau sebagian/aspek dari sistem. Sebuah *box* dapat bertipe *black box*, yaitu perilaku dari sistem atau sebagian/aspek dari sistem yang merupakan respon(output) dari rangsangan spesifik(*event*) sebagai input dengan mengunakan aturan transisi tertentu. Bertipe *state box*, yaitu keadaan dan operasi data, yang juga memiliki input dan output, *state box* juga dapat menjadi bagian dari *black box*. Bertipe *clear box*, adalah pendefinisian fungsi transisi operasi yang ada dalam *state box*, dengan kata lain *clear box* berisi desain prosedural untuk *state box*.

Sudah diketahui bahwa program dienkapsulasi dalam sebuah abstraksi(box) yang dijalankan oleh suatu subfungsi(operasi), konsep dari engkapsulasi ini kemudian digunakan sebagai desain untuk data program. Desain prosedur(subfungsi) dilakukan pada saat pembentukan suatu *clear box*. Proses perbaikan dan validasi dilakukan pada tahap desain ini. Perbaikan dan validasi dilakukan oleh tim melalui verifikasi formal dengan menerapkan *correctness conditions* pada struktur kontrol program(Linger, 1994). Setelah diverifikasi maka *box* ini kemudian diimplementasikan dengan bahasa pemrograman yang sesuai.

Pengujian pada model cleanroom ada dua yaitu *statistical use testing* dan *Certification*. *statistical use testing* dilakukan melalui penentuan distribusi probabilitas dari penggunaan sistem, dalam hal ini spesifikasi suatu sistem(*black box*) dari setiap ingkremen akan dianalisa dan ditentukan sejumlah input(rangsangan atau event) berupa skenario penggunaan dan probabilitasnya. Setiap input ini kemudian dicobakan beberapa kali dengan urutan yang acak, dan dilihat hasilnya apakah sudah menghasilkan output yang sesuai atau tidak dengan ukuran waktu tertentu pada setiap urutan. Hasil ini dapat memberikan informasi mengenai *mean-timeto failure*(MTTF). Jika MTTF rendah maka sistem memiliki tingkat kehandalan yang tinggi. *Certification* lebih kepada pengesahan terhadap hasil dari *statistical use testing* yang telah dilakukan pada setiap komponen, sehingga komponen ini dapat digunakan lagi secara sah jika diperlukan.



Gambar 2. Proses Model Cleanroom(Pressman, 1997)

Rangkaian kerja untuk setiap igkremen:

- Requierements gathering : Pengumpulan kebutuhan yang detail dari pelanggan
- Box structure spesification : Metode spesifikasi yang menggunakan struktur box untuk spesifikasi fungsional.
- Formal design : Penggunaan metode formal dalam pendekatan struktur box
- Correctness Verification : Tim pengembang melakukan verifikasi terhadap desain
- Code Generation dan Code Inspection : Dilakukan implementasi dengan bahasa pemrograman yang sesuai dan verifikasi terhadap kode
- Statistical Use Testing : Pengujian *statistical use testing* pada sistem
- Certification : Setiap komponen disertifikasi/disahkan dan siap untuk diintegrasikan
- Test Planning : Skenario dan probabilitas dari sistem dianalisa dan ditentukan, aktivitas ini berjalan paralel dengan spesifikasi, verifikasi dan implementasi.

4.3 MODEL SPIRAL DAN CLEANROOM

Sebagai sebuah proses pengembangan perangkat lunak, pada model proses spiral dan *cleanroom* melalui beberapa aktifitas umum dalam proses perangkat lunak :

Tabel 1 : Aktifitas Model proses

Aktifitas proses perangkat lunak	Model Spiral	Model Cleanroom
Spesifikasi	• Proses identifikasi sistem, dalam proses ini dilakukan penentuan sasaran ukuran produk, menentukan alternatif untuk proses implementasi dan Menentukan batasan-batasan yang ada dalam menggunakan alternatif yang sudah didefinisikan.	• Proses spesifikasi pada model ini dilakukan pemodelan terhadap data, fungsi dan prilaku dari sistem. Prinsip-prinsip analisis operasional diterapkan dengan metode *box structure specification*.
Desain dan implementasi	• Pada Model Spiral proses desain dan implementasi ditujukan untuk pembuatan *prototipe*. Prototipe ini diperlukan untuk analisis resiko dan penanggulangan terhadap resiko.	• Proses desain ditujukan untuk mendesain proses dan data, proses dan data didesain menggunakan model formal berdasarkan *box structure specification*. • Implementasi dilakukan setelah proses desain proses dan data diverivikasi, desain dalam bentuk struktur *box* tersebut kemudian diimplementasikan menggunakan bahasa pemrograman yang sesuai.
Validasi	• Validasi perangkat lunak dilakukan sebagai bagian dari analisis dan manajemen resiko. Dalam hal ini tim pengembang akan berinteraksi dengan pengguna sehingga mendapatkan masukan untuk pengembangan prototipe	• Dilakukan proses verifikasi terhadap desain, tahap ini dilakukan sebelum proses implementasi dan menggunakan metode verifikasi formal.
Evolusi	• Evolusi perangkat lunak difasilitasi dengan adanya konsep *prototype*, dalam hal ini prototype akan dikonsultasikan ke pengguna untuk mendapatkan masukkan dalam hal pengembangan sistem.	• Pada model ini juga dapat dilakukan pembuatan suatu *prototype*, penerimaan masukkan dari anggota tim dan pengguna dalam hal pengembangan sistem lebih jauh dilakukan setelah proses *Certification*.

Model spiral telah digunakan pada pendefinisian dan pengembangan TRW Software Productivity System(TRW-SPS). Software Productivity System yang dibangun menggunakan model spiral tebebas dari resiko yang sudah diprediksi dan sudah dapat mencapai semua sasaran yang diinginkan. SPS sudah berkembang dengan tambahan 300 fitur dengan jumlah instruksi mencapai 1.300.000 instruksi, 93 persen instruksi berasal dari proyek sebelumnya yang digunakan kembali, lebih dari 25 proyek sudah menggunakan sistem ini secara penuh dan berhasil meningkatkan produktivitas sebanyak 50 persen(Boehm, 1988).

Hal ini dapat tercapai dikarenakan beberapa keunggulan dari model spiral diantaranya adalah : Prioritas yang tinggi terhadap analisis resiko, sangat cocok digunakan untuk proyek skala besar dan kritis dan mekanisme yang memungkinkan perangkat lunak dibuat dibagian awal siklus pembangunan sistem(Mohammed, 2010).

Walaupun begitu model spiral juga memiliki kekurangan diantaranya adalah merupakan model yang membutuhkan biaya yang besar, analisis resiko membutuhkan pakar khusus, kesuksesan proyek tergantung dari tahap analisis resiko dan tidak terlalu bagus digunakan pada proyek dengan skala kecil(Mohammed, 2010).

Sedangakan model pengembangan *cleanroom* lebih banyak digunakan pada proyek-proyek yang dibuat oleh IBM dan NASA diantaranya adalah The Corse/Fine Attitude Determination System(CFADS) merupakan proyek pertama dari NASA goddard Space Flight Centre yang menggunakan model *cleanroom*, diketahui telah berhasil meningkatkan produktivitas sebesar 80 persen dari sebelumnya dan 60 persen program di-*compile* dengan benar pada percobaan pertama, Proyek IBM AOEXPERT/MVS merupakan fasilitas pendukung keputusan yang menggunakan *Artificial Intelligence* untuk memprediksi dan menanggulangi permasalahan pada *MVS*

environment, proyek diketahui telah berhasil meningkatkan produktivitas sebesar 486 LOC perorang-perbulan dan tidak ada kesalahan operasional dilaporkan dari *beta test* hingga masa awal penggunaan(Linger, 1994).

Keunggulan yang paling penting dari model *cleanroom* adalah pengurangan/pencegahan kesalahan atau *error* yang ditemukan selama tahap pengujian, hal ini akan mengurangi proses pengerjaan ulang selama proses pembangunan sistem, selain itu tingkat spesifikasi dan pembuatan model yang sangat detail dan formal pada produk perangkat lunak membuat perangkat lunak itu sendiri dapat digunakan dalam jangka waktu yang lama.

Walaupun awalnya memberikan suatu yang menjanjikan namun akhirnya model ini tidak banyak digunakan karena beberapa kekurangan yaitu : model cleanroom terlalu teoritis, terlalu matematis, dan terlalu radikal untuk digunakan di keadaan sebenarnya, tidak adanya pengujian unit perangkat lunak namun hanya verifikasi kebenaran dan kontrol kualitas secara statistik, dan model cleanroom membutuhkan penerapan yang sangat teliti sedangkan industri rekayasa perangkat lunak secara umum belum mampu menerapkan tehnik tersebut(Pressman, 1997).

5. KESIMPULAN

Setelah dilakukan perbandingan terhadap model spiral dan cleanroom telah ditemukan persamaan dan perbedaan beserta kelebihan dan kekurangannya. Persamaan dari model spiral dan *cleanroom* adalah sama-sama menggunakan prototipe untuk mengakomodasi sifat evolusi perangkat lunak, selain itu dua model proses ini juga memungkinkan untuk melibatkan pengguna dalam mendapatkan masukan mengenai perubahan atau penambahan kebutuhan perangkat lunak. Dua model pengembangan sistem ini juga memberikan penekanan terhadap penanganan kesalahan perangkat lunak, namun menggunakan cara yang berbeda, pada model spiral penanganan kesalahan dilakukan melalui analisis dan manajemen resiko sedangkan pada model *cleanroom* dilakukan pencegahan terhadap adanya kesalahan melalui penerapan metode formal dalam spesifikasi dan desain. Melihat perhatian yang signifikan terhadap penanganan kesalahan dari kedua model ini memberikan potensi keyakinan bahwa perangkat lunak yang dihasilkan dari kedua model ini memiliki kualitas yang patut diperhitungkan. Melihat dari penggunaanya, kedua model ini sudah banyak diterapkan pada pengembangan beberapa proyek khususnya yang berskala besar dan kritis, namun untuk perkembangan kedepan model proses spiral akan lebih banyak diperhitungkan dibandingkan model *cleanroom* yang kurang realistis.

DAFTAR PUSTAKA

Royce, W. W.1970, *Managing the Development of Large Software Systems: Concepts and Techniques*. Proceedings of the IEEE WESTCON (pp. 328-338). IEEE.

Sommerville, Ian.2004.*Software Engineering, Addison Wesley*, 7th edition.

Pressman, Roger S.1997, *Software Engineering : A Practitioner's Approach*, McGraw-Hill. Inc, New York, NY 10020.

Paulk, M. Et al.1993, *Capability Maturity Model for Software*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Oriogun, P. K. 2000. *A Survey of Boehm ' s Work on the Spiral Models and COCOMO II — Towards Software Development Process Quality Improvement*. Development, 53-63.

Mohammed, N., Munassar, A., & Govardhan, A.2010. *A Comparison Between Five Models Of Software Engineering*. Journal of Computer Science, *7*(5), 94-101.

Mills, H,M. Dyer and R. Linger.1987, *Cleanroom Software Engineering*. IEEE Software 4 (5): 19–25

Marciniak, J.J. , 1994, *Encyclopedia of Software Engineering*, Wiley.

Linger, R.1994, *Cleanroom Process Model*. *IEEE Software*, vol. 11, no. 2, pp. 50–58.

Hevner, A.R. and H.D. Mills. 1993 *Box Structure Methods for System Development with Objects*, IBM Systems Journal, vol. 31, no.2, pp. 232–251.

Groote, Jan Frisco. Osaiweran, Ammar. Wesselius, J H. 2011. *Analyzing The Effects of Formal Methods on the Development of Industrial Control Software*. Conference On Software Maintenance, 467-472.

Gilb, T.1988.*Principle Of Software Engineering*, Addison-Wesley.

Davis, M., & Bersoff, H. 1988. *A Strategy for Comparing Alternative Software Development Life Cycle Models. IEEE Transactions on Software Engineering*, vol. 14, no. io, october 1988

Boehm, B. 1998. *WinWin Spiral Model: A case study*, *IEEE*.

Boehm, B. 1988. *A spiral model of software development and enhancement*, Computer May, pp. 61-72.

Awad, M.A. 2005.*A Comparison between Agile and Traditional Software Development Methodologies. School of Computer Science and software Engineering*, The University of Western Australia.