Proses perangkat lunak mendefinisikan pendekatan yang harus diambil saat perangkat lunak di rekayasa. Proses perangkat lunak juga mencakup teknologi-teknologi yang dibutuhkan oleh proses perangkat lunak. pada proses perangkat lunak terdapat beberapa pendekatan yang terangkum dalam aktivitas kerangka kerja perangkat lunak. Pada bab 2 ini akan dibahas model-model pendekatan yang dapat digunakan pada proses perangkat lunak.

Secara garis besar model proses yang telah dibahas pada bab 1 adalah model proses generik yang memiliki 5 aktivitas kerangka kerja dan ditunjang dengan sejumlah aktivitas – akivitas pendukung. Pada dasarnya proses dari rekayasa perangkat lunak dapat menjadi sebuah aliran proses yang masing – masing memiliki urutan akivitas kerja yang berbeda-beda disesuaikan berdasarkan lingkup proyeknya, Pressman menggambarkan pada model proses generik terdapat empat aliran proses.

Pada proses perangkat lunak maka ada beberapa hal yang perlu diperhatikan agar proses perangkat lunak dapat berjalan dengan baik (Roger S. Pressman, 2012):

1) Mendefinisikan aktivitas kerangkat kerja, pendefinisian ini dilakukan untuk dapat menentukan tindakan-tindakan yang sesuai dengan aktivitas kerangka kerja yang sesuai dengan permasalahan yang akan diselesaikan dan menentukan karakteristik orang-orang yang akan mengerjalan serta para stakeholder yang akan membiayai proyek.

2) Mengidentifikasi himpunan pekerjaan, mendefiniskan pekerjaan – pekerjaan yang harus di selesaikan untuk memenuhi sasaran tertentu dari suatu aksi rekayasa perangkat lunak.

3) Pendiskripsian pola – pola proses, mendiskripsikan permasalahan-permasalahan yang terkait dengan proses, yang dijumpai selama pekerjaan rekayasa perangkat lunak berlangsung. Pola-pola proses ini menyediakan template untuk menyelesaikan suatu masalah sehingga tim perangkat lunak dapat menyelesaikan permasalahan – permasalahan dan membentuk proses yang paling sesuai dengan kebutuhan dari suatu proyek yang sedang dikerjakan.

4) Penilaian dan perbaikan proses, proses perangkat lunak dapat dinilai untuk memastikan bahwa perangkat lunak tersebut telah memenuhi sejumlah kriteria proses

dasar yang esensial. Ada beberapa pendekatan penilaian proses perangkat lunak diantarnya:

SCAMPI (Standard CMMI Assesment Method for Process Improvement) menyediakan model penilaian dengan lima tahapan yaitu melakukan pemberian nilai awal (initiating), melakukan diagnosa (diagnosing), penetapan (establising), bertindak (acting), dan belajar (learning).

CBA IPI (CMM-Based Appraisal for Internal Process Improvement), Pendekatan yang menggunakan teknik diagnosa untuk melakukan penilaian kematangan perangkat lunak.

ISO 9001:2000 for software, Merupakan standar generik penilaian untuk melakukan perbaikan kualitas produk perangkat lunak.

Model Proses Perspektif Disebut model proses perspektif karena pendekatan ini memberikan sejumlah elemen proses yaitu berupa aktivitas-aktivitas kerangka kerja, tindakan-tindakan rekayasa perangkat lunak, produk-produk kerja, penilaian jaminan kualitas dan mekanisme kendali perubahan untuk setiap proyek perangkat lunak. berikut adalah beberapa pendekatan dari model proses perspektif:

Model Air Terjun (waterfall) atau siklus hidup klasik (classic life cycle), Menurut Rosa model air terjun ini menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung. Sedangkan Pressman mengemukakan bahwa model air terjun ini menyiratkan pendekatan yang sistematis dan berurutan pada perangkat lunak yang dimulai dari perencanaan (planning), pemodelan (modeling), konstruksi (construction), serta penyerahan perangkat lunak kepada pelanggan (deployment), yang diakhiri dengan dukungan berkelanjutan pada perangkat lunak lengkap yang dihasilkan. Model air terjun ini dapat digambarkan sebagai berikut (Roger S. Pressman, 2012).

Pada perkembangannya Bucanac mengemukakan model air terjun ini di variasi menjadi model-V, yang pada dasarnya model-V ini menyediakan secara visual bagaimana tindakan – tindakan verifikasi dan validasi yang seharusanya diterapkan pada bagian – bagian rekayasa perangkat lunak yang lebih awal.

Pada implementasinya pendekatan menggunakan model air terjun ini banyak sekali ditemukan kekurangan. Hanna mengemukakan beberapa kekurangan dari model air terjun ini antara lain pada kenyataannya proyek perangkat lunak jarang mengikuti aliran sekuensial seperti yang diusulkan model air terjun, sering kali pelanggan sulit menetapkan semua spesifikasi kebutuhan secara eksplisit sedangkan model air terjun sulit untuk mengakomodasi ketidakpastian yang selalu harus pada tahapan proyek, dan kebanyakan pelanggan tidak memiliki kesabaran untuk melihat program sampai proyek berakhir (Roger S. Pressman, 2012).

Bradac juga mengemukakan kekurangan dari model air terjun ini, menurutnya tahapan pada model air terjun cenderung terkunci sehingga banyak anggota tim proyek perangkat lunak harus menghabiskan waktu untuk menunggu anggota tim lainnya bekerja hingga dapat menyelesaikan pekerjaannya sendiri (Roger S. Pressman, 2012).

Keuntungan dari Model Waterfall:

Model Waterfall sangat sederhana dan mudah dimengerti dan menggunakan metode yang sangat bermanfaat untuk pemula atau pengembang pemula;

Sangat mudah untuk dikelola, karena kekakuan model. Selain itu, setiap fase memiliki kiriman spesifik dan proses peninjauan individu;

Dalam fase model ini diproses dan diselesaikan sekaligus dalam satu waktu sehingga menghemat banyak waktu;

Jenis model pengembangan ini bekerja lebih efektif dalam proyek-proyek yang lebih kecil di mana persyaratan sangat dipahami dengan baik;

Pengujian lebih mudah karena dapat dilakukan dengan mengacu pada skenario yang didefinisikan dalam spesifikasi fungsional sebelumnya;

Kekurangan Model Air Terjun:

Model ini hanya dapat digunakan ketika persyaratan depan yang sangat tepat tersedia;

Model ini tidak berlaku untuk jenis proyek pemeliharaan;

Kelemahan utama dari metode ini adalah bahwa sekali aplikasi berada dalam tahap pengujian, tidak mungkin untuk kembali dan mengedit sesuatu;

Tidak ada kemungkinan untuk menghasilkan perangkat lunak yang bekerja sampai mencapai tahap terakhir dari siklus;

Dalam model ini, tidak ada pilihan untuk mengetahui hasil akhir dari keseluruhan proyek;

Model ini bagus untuk proyek kecil tetapi tidak cocok untuk proyek yang panjang dan berkelanjutan;

Tidak ideal untuk proyek yang persyaratannya sangat moderat, dan ada ruang lingkup yang bagus untuk modifikasi.

Model inkremental atau disebut dengan model penambahan sedikit demi sedikit ini menggabungkan elemen – elemen aliran proses dari model generik yaitu aliran proses linier dan aliran proses paralel (Roger S. Pressman, 2012). Masing – masing urutan linier menghasilkan bagian penambahan dari perangkat lunak dengan cara serupa dengan penambahan sedikit demi sedikit yang dihasilkan oleh suatu aliran proses yang bersifat evolusioner. Contohnya adalah perangkat lunak pengolah kata yang dikembangkan dengan cara penambahan sedikit demi sedikit dengan menambahkan fungsi-fungsi pengelolaan berkas, penyuntingan dan fungsi-fungsi dokumen yang dibuat pada tahap pertama, kemudian fungsi penyuntingan dan fungsi produksi yang lebih canggih pada tahap kedua. Pada setiap aliran proses setiap penambahan dapat digabungkan dengan pembentukan prototipe. Produk yang dihasilkan pada tahap pertama merupakan produk inti, produk yang dihasilkan pada tahap kedua dan selanjutnya merupakan fitur-fitur tambahan yang merupakan pengembangan dari produk tersebut.

Pada pengembangan perangkat lunak dengan model proses inkremental ini sangat bermanfaat pada strategi pemanfaatan sumber daya, dan juga dapat direncanakan dengan mempertimbangkan risiko-risiko teknik tertentu.

Model proses evolusioner adalah model proses perangkat lunak yang dirancang untuk mengakomodasi suatu produk perangkat lunak yang akan berubah secara perlahan sepanjang waktu. Model proses evolusioner ini digunakan untuk menjawab tantangan perubahan kebutuhan perangkat lunak yang semakin kompleks. Model proses

evolusioner ini bersifat iteratif yang dicirikan dalam bentuk yang memungkinkan pengembang perangkat lunak untuk mengembangkan produknya menjadi semakin kompleks pada versi – versi berikutnya.

Model proses evolusioner ini terbagi menjadi beberapa paradigma pendekatan yaitu: Pembuatan prototipe (prototyping).

Pengguna perangkat lunak kadang mendefinisikan kebutuhannya secara umum, dan tidak dapat mendefinisikan kebutuhan secara rinci terkait fitur – fitur atau fungsi – fungsi yang nantinya akan dikembangkan. Dengan melihat kasus ini maka pengembang tidak memiliki kepastian terkait efisiensi algoritma yang digunakan untuk mengembangkan perangkat lunak yang dibutuhkan oleh pengguna. Pada kasus tersebut, pendekatan yang paling baik adalah menggunakan paradigma pembuatan prototipe. Dalam hal ini paradigma pembuatan prototipe seringkali membantu tim pengembang perangkat lunak dan para stakeholder untuk memahami kebutuhan perangkat lunak yang akan dikembangkan.

Pembuatan protipe dimulai dengan dilakukannya komunikasi antara tim pengembang perangkat lunak dengan pelanggan, kemudian menetapkan sasaran pengembangan secara keseluruhan dan mengidentifikasi spesifikasi kebutuhan. Pembuatan prototipe direncanakan dan dirancang dengan cepat.

Rancangan kemudian di kontruksi untuk membuat prototipe. Prototipe kemudian diserahkan kepada stakeholder dan mereka melakukan evaluasi terhadap prototipe tersebut, yang pada akhirnya memberikan umpan balik yang akan digunakan untuk memperbaiki atau mengembangkan spesifikasi kebutuhan perangkat lunak. Pengulangan yang terjadi pada saat prototipe diperbaiki secara tidak langsung memenuhi kebutuhan stakeholder, dan pada saat yang sama memungkinkan pengembang memahami lebih dalam kebutuhan dari perangkat lunak yang dikerjakan. Semakin banyak pengembangan maka perangkat lunak berevolusi memenuhi kebutuhan pengguna.

Keuntungan Model Prototype:

Ketika prototipe ditunjukkan kepada klien, mereka mendapatkan pemahaman yang jelas dan menyelesaikan 'rasa' fungsionalitas perangkat lunak;

Metode ini secara signifikan mengurangi risiko kegagalan, karena potensi risiko dapat diidentifikasi pada tahap awal dan langkah-langkah moderasi dapat diambil dengan cepat;

Komunikasi antara tim pengembangan perangkat lunak dan klien membuat lingkungan yang sangat baik dan kondusif selama proyek;

Ini membantu dalam pengumpulan kebutuhan dan analisis kebutuhan ketika ada kekurangan dokumen persyaratan.

Kekurangan Model Prototype:

Pembuatan prototipe biasanya dilakukan dengan biaya pengembang, jadi harus dilakukan dengan menggunakan sumber daya minimal jika biaya pengembangan organisasi terlalu banyak;

Terlalu banyak keterlibatan klien tidak selalu disukai oleh pengembang perangkat lunak;

Terlalu banyak modifikasi mungkin tidak baik untuk proyek, karena mudah mengganggu alur kerja seluruh tim pengembangan perangkat lunak.

Model spiral merupakan suatu model proses perangkat lunak evolusioner yang menggabungkan pendekatan prototyping yang bersifat iteratif dengan aspek-aspek sistematis dan terkendali yang di jumpai pada model air terjun. Menggunakan model spiral, perangkat lunak dikembangkan mengikuti peluncuran produk yang bersifat evolusioner. Selama tahap awal, produk perangkat lunak yang di luncurkan mungkin berupa sebuah model atau suatu prototipe. Pada langkah-langkah iterasi berikutnya versi – versi perangkat lunak yang semakin lengkap akan dihasilkan (Roger S. Pressman, 2012).

Model pengembangan spiral dimulai dengan fase perencanaan, yang mengevaluasi elemen kerja utama dari proyek seperti tujuan, kondisi pasar dan banyak lagi. Fase berikutnya, analisis risiko mencoba memecahkan masalah dalam lebih dari satu cara. Fase ketiga adalah menyelesaikan pekerjaan, diikuti oleh ulasan pelanggan yang bergantung pada umpan balik dari pelanggan. Iterasi berhenti ketika penyampaian akhir dikembangkan. Dengan cara ini baik klien dan tim pengembangan berinteraksi

secara berkala, meningkatkan produk yang dapat dikirimkan satu per satu (Receptives, 2018).

Model spiral dibagi menjadi sejumlah aktivitas kerangka kerja yang didefinisikan oleh pengembang perangkat lunak. Lintasan pertama di sekitar spiral mungkin menghasilkan sesuatu yang penting untuk spesifikasi produk, lintasan-lintasan selanjutnya di sekitar spiral mungkin digunakan untuk mengembangkan suatu prototipe dan kemudian pada lintasan – lintasan berikutnya tim pengembang bisa secara progresif bergerak ke versi-versi perangkat lunak selanjutnya yang semakin canggih.

Walaupun model spiral seolah-olah dapat memberikan kemudahan dalam pengembangan perangkat lunak namun Pressman menyampaikan bahwa model spiral membutuhkan pakar-pakar yang bisa melakukan penilaian risiko dengan cara yang sempurna, dan model ini mengandalkan dirinya pada pakar – pakar terkait agar berhasil, jika risiko utama tidak tersingkap dan tidak bisa dikelola dengan baik, permasalahan-permasalahan yang buruk sangat mungkin akan terjadi (Roger S. Pressman, 2012).

Keuntungan dari Model Spiral:

Tingginya jumlah analisis risiko karenanya, menghindari kemungkinan risiko tentu berkurang;

Model ini bagus untuk ukuran besar dan proyek penting;

Dalam model spiral, fungsi tambahan dapat ditambahkan di kemudian hari;

Ini lebih cocok untuk proyek-proyek berisiko tinggi, di mana kebutuhan bisnis dapat berbeda dari waktu ke waktu.

Kekurangan Model Spiral:

Ini tentu model mahal untuk digunakan dalam hal pengembangan;

Keberhasilan keseluruhan proyek tergantung pada fase analisis risiko sehingga kegagalan dalam fase ini dapat merusak keseluruhan proyek;

Ini tidak sesuai untuk proyek berisiko rendah;

Risiko besar dari metodologi ini adalah bahwa ia dapat terus berlanjut tanpa batas dan tidak pernah selesai.

Model pengembangan perangkat lunak konkuren, memungkinkan tim perangkat lunak untuk menggunakan unsur-unsur yang bersifat berulang atau iteratif dan konkuren atau berjalan bersamaan dalam setiap model proses. Berikut gambaran aktivitas model konkuren (Roger S. Pressman, 2012).

Model konkuren sering lebih tepat untuk proyek rekayasa sistem di mana tim teknik yang berbeda terlibat. Model konkuren direpresentasikan dengan skema sebagai series dari kerangka aktivitas, aksi software engineering dan juga tugas dari jadwal.

Aktivitas dikerjakan secara bersamaan, setiap proses memiliki beberapa pemicu kerja dari aktivitas. Pemicu dapat berasal dari awal proses kerja maupun dari pemicu yang lain karena setiap pemicu akan saling berhubungan. Model ini digambarkan secara skematik dengan diagram modeling activity sebagai rangkaian dari teknis utama, tugas dan hubungan antar bagian.

Model konkuren sering digunakan sebagai paradigma untuk pengembangan aplikasi client/server. Model konkuren memiliki kelebihan hasil sistem yang sangat baik karena terdapat perancangan yang besar, terencana dan matang. Sedangakan kekurangan dari model konkuren memungkinkan terjadinya perubahan besar – besaran yang akan membuat biaya dan waktu yang diperlukan membengkak. Jadi model konkuren ini suatu cara atau langkah kerja untuk membuat suatu sistem yang dikerjakan secara besar – besaran namun tetap mempertahankan kualitas sesuai dengan permintaan customer, bila ada permintaan lain dari customer maka langkah-langkah kerja dihentikan sementara untuk memaksimalkan hasil akhir dari model ini (IT, 2014).

Simpulan pada proses-proses evolusioner. Perhatian model-model evolusioner adalah mengembangkan perangkat lunak berkualitas tinggi dalam arti interatif dan bersifat penambahan sedikit demi sedikit (inkremental).

Meski demikian, merupakan hal yang mungkin untuk menggunakan proses yang evolusioner yang menenkankan pada fleksibilitas, perluasan serta kecepatan pengembangan. Tantangan bagi tim perangkat lunak dan para manajernya adalah menetapkan keseimbangan antara proyek yang bersifat kritis dan parameter-parameter produk dan kepuasan pelanggan (Roger S. Pressman, 2012).

Model Proses yang Terkhususkan Model proses terkhususkan mengambil karakteristik – karakteristik dari satu atau lebih model – model yang tradisional yang telah dibahas sebelumnya.

Pengembangan Berbasis Komponen, Model ini biasa disebut CBD atau componentbased development. Model ini kebanyakan menggunakan karakteristik model spiral. Model CBD mengkonstruksi perangkat lunak menggunakan komponen-komponen yang telah ada sebelumnya dengan cara merakitnya ke perangkat lunak yang sedang di kembangkan.

Langkah-langkah pengembangan perangkat lunak dengan menggunakan pendekatan CBD adalah pertama komponen – komponen yang akan diintegrasikan diteliti dan diintegrasi kemudian isu integrasi komponen dipertimbangkan, setelah dipertimbangkan kemudian dirancang arsitektur perangkat lunak yang dapat mengakomodasi komponen, kemudian komponen tersebut diintegrasikan ke arsitektur dan dilakukan pengujian-pengujian yang bersifat komprehensif untuk melihat apakah komponen yang diintegrasikan membarikan fungsionalitas yang baik.

Model CBD ini memberikan peluang untuk penggunaan ulang komponen yang dapat menguntungkan rekayasawan perangkat lunak (Roger S. Pressman, 2012).

Proses Terpadu (Unified Proces), Ivar Jacobson, Graddy Booch dan James Rumbaugh dalam (Roger S. Pressman, 2012) membahas tentang kebutuhan untuk suatu proses perangkat lunak yang bersifat dikendalikan oleh use case, berpusat pada arsitektur, bersifat interatif menyatakan: “saat ini perangkat lunak cenderung semakin besar dan kompleks, ini di sebabkan karena perangkat keras yang semakin powerful sehingga para pengguna menjadi berharap banyak darinya. Kecendrungan ini juga dipengaruhi dengan internet yang digunakan sebagai budaya bertukar informasi. Dan para pengguna menginginkan perangkat lunak yang mampu beradaptasi lebih baik dengan kebutuhannya.”.

Proses terpadu (unified process) berusaha menggunakan karakteristik-karakteristik terbaik model proses perangkat lunak tradisional, tetapi menggabungkannya dengan prinsip – prinsip terbaik yang dimiliki pengembang perangkat lunak yang cepat. Proses terpadu mengenali pentingnya komukasi dengan para pelanggan dan menekankan deskripsi sistem dari sudut pandang pelanggan (melalui diagram – diagram usecase).

Sejarah singkat, Pressman mengemukakan pada tahun 1990-an James Rumbaugh, Graddy Booch dan Ivar Jacobson mulai mengembangkan metode terpadu yang menggabungkan fitur terbaik dari masing-masing metode analisis dan perancangan berorentasi objek dan juga mengadopsi fitur-fitur tambahan yang diusulkan oleh para pakar dalam pemodelan berorentasi objek. Hasilnya adalah diagram-diagram UML (Unified Modeling Language), yang memuat notasi yang cukup tangguh untuk digunakan sebagai sarana pemodelan dan pengembangan perangkat lunak berorentasi objek (Roger S. Pressman, 2012).

UML menyediakan teknologi yang diperlukan untuk mendukung rekayasa perangkat lunak berorentasi objek, tetapi UML tidak menyediakan kerangka kerja proses untuk memandu tim – tim proyek dalam mengaplikasikan teknologi. Saat ini proses terpadu dan UML digunakan secara meluas pada proyek – proyek berorientasi objek yang beragam. Model yang bersifat interatif dan inkremental diusulkan oleh UP dan sebaliknya diadaptasi untuk memperoleh kebutuhan-kebutuhan proyek perangkat lunak.

Tahapan dari proses terpadu adalah sebagai berikut: tahapan pengenalan, tahapan elaboration, tahapan construction, tahapan transition, tahapan production.

Tahapan pengenalan (inception) dari proses terpadu membahas tentang komunikasi dengan para pelanggan dan juga membahas aktivitas-aktivitas perencanaan. Dengan cara berkolaborasi dengan para stakeholder, spesifikasi-spesifikasi bisnis untuk perangkat lunak dapat diidentifikasi, arsitektur garis besar untuk sistem dapat diusulkan dan suatu rencana untuk tahapan-tahapan yang bersifat iteratif dan inkremental yang berkaitan dengan proyek mulai dikembangkan. Spesifikasi-spesifikasi bisnis yang mendasar dijelaskan melalui sejumlah use case awal yang deskripsikan fitur-fitur serta fungsi – fungsi yang mana untuk masing-masing kelas utama yang diinginkan oleh para pengguna. Arsitektur pada titik ini tidak lebih dari gambaran garis besar tentative tentang subsitem-subsitem utama dan fungsi-fungsi serta fitur-fitur yang diperlukan untuk membentuk arsitektur sistem. Perencanaan sumber-sumber daya, melakukan penilaian terhadap risiko-risiko utama, mendefinisikan jadwal, serta menetapkan suatu dasar bagi tahapan – tahapan yang akan diaplikasikan saat pengembangan sedikit demi sedikit perangkat lunak (inkremental) dikembangkan.

Tahapan elaboration menggunakan aktivitas-aktivitas komunikasi dan pemodelan milik model proses generik. Tahap elaboration digunakan untuk menghaluskan dan mengembangkan use case awal yang dikembangkan pada tahap pengenalan dan mengembangkan representasi arsitektural dengan melibatkan 5 sudut pandang yang berbeda dari suatu perancangan perangkat lunak yaitu model use case, model spesifikasi kebutuhan, model perancangan, model imlementasi dan model penyebaran komponen (deployment model).

Tahapan construction pada metode proses terpadu identik dengan aktivitas yang sama yang didefinisikan untuk proses perangkat lunak generik. Pada tahapan konstruksi ini, tim pengembang perangkat lunak akan mengembangkan komponen-komponen perangkat lunak yang akan membuat masing-masing use case bersifat operasional untuk masing-masing pengguna akhir. Untuk dapat melakukan dengan baik, model-model spesifikasi kebutuhan dan perancangan yang diperoleh pada tahap elaboration dilengkapi untuk mencerminkan versi terakhir dari perangkat lunak yang pada dasarnya dikembangkan secara inkremental. Semua fitur-fitur dan fungsi-fungsi yang penting dan diperlukan untuk produk diimplementasi dalam bentuk kode-kode dalam bahasa pemograman berorientasi objek tertentu yang dipilih. Setelah komponen-komponen diimplementasikan, unit-unit pengujian dirancang dan dieksekusi.

Juga aktivitas-aktivitas yang berkaitan dengan pengintegrasian sistem dilaksanakan. Use case - use case akan digunakan untuk membentuk sejumlah paket pengujian yang akan dieksekusi sebelum semua masuk pada tahapan proses terpadu yang berikutnya.

Tahapan transition pada proses terpadu adalah penyerahan komponen dan umpan balik. Perangkat lunak diserahkan kepada pengguna akhir untuk pengujian beta dan untuk mendapatkan umpan balik dari pengguna tentang hal – hal yang berkaitan dengan cacat – cacat program dan perubahan – perubahan yang diperlukan.

Tahapan production pada tahapan ini perangkat lunak dipantau, dukungan untuk lengkungan operasional atau infrastruktur yang disediakan, dan laporan tentang cacat program dan permintaan untuk perubahan – perubahan dikirimkan dan dievaluasi.

Perlu dicatat bahwa tidak semua pekerjaan yang diidentifikasi untuk suatu aliran kerja proses terpadu harus dikerjakan pada setiap proyek perangkat lunak. Tim perangkat

lunak akan mengadaptasi proses – proses sesuai kebutuhan proyek – proyek perangkat lunak secara spesifik.

Rekayasa perangkat lunak cepat menggabungkan suatu falsafah dan serangkaian tuntunan pengembangan. Falsafah tersebut mengupayakan kepuasan pelanggan dan penghantaran perangkat lunak yang meningkat, tim – tim proyek yang kecil dan termotivasi, metode – metode informal, produk kerja rekayasa perangkat lunak yang minimal, dan keseluruhan kesederhanaan pengembangan. Panduan pengembangan menekankan penghantaran daripada analisa dan perancangan, komunikasi yang berkelanjutan dan aktif antara pengembang dan pelanggan. Pengembangan cepat dapat memberikan manfaat, namun tidak selalu cocok untuk semua situasi.

Konteks kecepatan dalam rekayasa perangkat lunak, suatu tim cepat adalah suatu tim yang cekatan yang mampu merespon perubahan – perubahan dengan cepat dan tepat. Perubahan berarti perubahan dalam perangkat lunak yang dibuat, perubahan pada anggota tim, perubahan karena teknologi baru, perubahan semua hal yang mungkin berdampak pada produk perangkat lunak yang di kembangkan. Tim cepat tahu bahwa perangkat lunak yang dikembangkan oleh individu – individu yang bekerja dalam tim, keahlian, kolaborasi merupakan inti menuju keberhasilan proyek perangkat lunak. Kecepatan mendorong terbentuknya struktur tim dan mendorong tercapainya sikap yang menjadikan komunikasi menjadi lebih mudah dan kondusif. Kecepatan menekankan penghantaran yang cepat dari perangkat lunak, mengadopsi pelanggan sebagai salah satu tim pengembang. Proses cepat yang mencakup penghantaran perangkat lunak ke pengguna yang meningkat, serta dipasangkan dengan pengujian unit yang berkelajutan akan dapat menekan biaya pengembangan perangkat lunak. Proses cepat merupakan proses yang harus dapat beadaptasi, namun adaptasi berkelanjutan tanpa adanya kemajuan tidak dapat diharapakan. Untuk itu proses perangkat lunak yang cepat harus beradaptasi secara meningkat. Untuk dapat beradaptasi secara meningkat, suatu tim cepat sangat membutuhkan umpan balik yang datang dari pelanggan.

Ada 12 prinsip kecepatan dalam pengembangan perangkat lunak:

1) Prioritas tertinggi adalah kepuasan pelanggan melalui penghantaran perangkat lunak dalam waktu yang lebih awal dan berkesinambungan;

2) Menyambut kebutuhan-kebutuhan yang berubah-ubah. Proses cepat memanfaatkan perubahan-perubahan untuk mencapai keunggulan kompetitif;

3) Sering mengahantarkan perangkat lunak;

4) Pelaku bisnis dan pengembang perangkat lunak harus bekerja sama setiap hari selama proyek perangkat lunak dilaksanakan;

5) Bangun proyek perangkat lunak di lingkungan individu yang termotivasi;

6) Metode yang paling efisien dan efektif untuk menyampaikan informasi ke dan dalam tim pengembangan adalah percakapan tatap muka;

7) Perangkat lunak yang berjalan dengan baik adalah tolok ukur utama dari kemajuan suatu proyek perangkat lunak;

8) Proses pengembangan perangkat lunak cepat mendukung pengembangan berkelanjutan;

9) Peningkatan terus menerus mengenai keunggulan teknis dan perencanaan yang baik dapat meningkatkan kecepatan pengembangan perangkat lunak;

10) Kesederhanaan;

11) Arsitektur, persyaratan dan rancangan perangkat lunak terbaik muncul dari tim perangkat lunak yang mengatur dirinya secara mandiri;

12) Pada interval waktu yang teratur, tim perangkat lunak mencerminkan keadaan bagaimana caranya mereka menjadi lebih efektif, lalu menyesuaikan perilakunya.

Ada perdebatan besar tentang manfaat dan kecocokan pengambangan perangkat lunak cepat yang berlawanan dengan proses rekayasa perangkat lunak yang lebih konvensional.

Ketika kecepatan dianggap penting bagaimana caranya membangun perangkat lunak yang memenuhi kebutuhan pelanggan hari ini dan menunjukkan karakteristik kualitas yang memungkinkan perangkat lunak untuk diperluas dan diukur untuk memenuhi kebutuhan pelanggan dalam jangka panjang? tentunya tidak ada jawaban yang mutlak dari pertanyaan tersebut, namun banyak konsep cepat sebenarnya hanya diadaptasi dari konsep – konsep rekayasa perangkat lunak yang baik. Pada dasarnya

ada banyak hal yang dapat diperoleh dengan mempertimbangkan hal – hal yang terbaik.

Pengembangan perangkat lunak secara cepat berfokus pada kemampuan individu, membentuk proses bagi tim. Untuk itu sejumlah ciri – ciri kunci harus ada diantara tim cepat yaitu (Roger S. Pressman, 2012):

1) Kompetensi, konteks kompetensi mencakup bakat, ketrampilan khusus yang berhubungan dengan perangkat lunak dan mencakup seluruh pengetahuan proses yang dipilih tim perangkat lunak untuk diterapkan.

2) Fokus bersama, meskipun anggota tim memiliki kecerdasan yang dapat melakukan tugas yang berbeda-beda, semuanya pada dasarnya harus dipusatkan pada satu tujuan yaitu untuk meningkatkan kualitas perangkat lunak yang akan dihantarkan kepada pelanggan dalam jangka waktu yang dijanjikan. Untuk mencapai tujuan ini, tim cepat akan berfokus pada adaptasi – adaptasi yang berkelanjutan yang akan menyesuaikan proses perangkat lunak dengan kebutuhan tim perangkat lunak.

3) Kolaborasi, rekayasa perangkat lunak adalah tentang penilaian, analisis, dan penggunaan informasi yang dikomunikasikan kepada tim dan dia antara anggota-anggota tim perangkat lunak, tentang pembuatan informasi yang membantu semua stakeholder untuk memahami kerja tim, dan tentang bagaimana cara membangun atau mengembangkan informasi – informasi yang memberikan nilai bisnis tertentu bagi para pelanggan. Untuk mencapai tugas – tugas ini, anggota tim cepat harus saling bekerjasama dengan stakeholder lainnya.

4) Kemampuan pengambilan keputusan. Setiap tim perangkat lunak yang baik harus memungkinkan memiliki kebebasan mengendalikan nasibnya sendiri. Hal ini berarti bahwa tim perangkat lunak harus diberikan otonomi secukupnya maksudnya otoritas pengambilan keputusan untuk isu – isu proyek maupun isu – isu yang terkait dengan masalah teknis.

5) Kemampuan pemecahan masalah yang kabur. Manajer perangkat lunak menyadari bahwa tim cepat akan tetap harus berhadapan dengan ambiguitas dan akan diterpa perubahan – perubahan yang harus dihadapinya, namun demikian, pelajaran yang di petik dari aktivitas pemecahan masalah mungkin bermanfaat bagi tim perangkat lunak saat nantinya mereka melakukan tugas – tugas penting dalam proyek.

6) Saling percaya dan menghormati. Tim cepat harus menjadi tim kental yaitu menunjukkan kepercayaan dan penghargaan yang diperlukan untuk menjadikan mereka begitu kuat dan rapat sehingga keseluruhan merupakan bagian yang lebih besar daripada kumpulan bagian-bagian kecil.

7) Pengorganisasian diri. Menyiratkan tiga hal yaitu tim cepat mampu mengorganisaksn dirinya sendiri untuk pekerjaan yang akan diselesaikan, tim cepat mampu mengatur proses perangkat lunak untuk dapat mengakomodasikan dengan sangat baik lingkungan lokalnya, tim cepat mampu mengatur jadwal kerja untuk mencapai penghantaran perangkat lunak ke para pelanggan dengan cara yang sangat baik.

Pemograman Extrem (Extreme Programming) yaitu suatu pendekatan yang paling banyak digunakan untuk pengembangan perangkat lunak cepat. Nilai – nilai yang membentuk dasar dari XP adalah komunikasi, kesederhanaan, umpan balik, keberanian dan rasa hormat.

Untuk mencapai komuniasi yang efektif yang seharusnya terjadi antara pengembang dan para stakeholder, XP menekankan kolaborasi informal antara pelanggan dan pengembang perangkat lunak, menekankan pentingnya pembentukan metafora-metafora yang efektif untuk mengkomunikasikan konsep – konsep yang penting, menekankan pentingnya adaptasi terhadap umpan balik yang berkesinambungan, dan menekankan pentingnya dokumentasi yang produktif sebagai seuatu media komunikasi. Untuk mencapai kesederhanaan, XP membatasi pengambang perangkat lunak melakukan perancangan hanya untuk kebutuhan – kebutuhan yang sifatmya mendesak, tujuannya adalah untuk menciptakan rancangan yang sederhana yang dapat dengan mudah diimplementasikan dalam bentuk kode – kode program yang sangat cepat, jika rancangan tersebut harus ditingkatkan, maka rancangan tersebut dapat di refactorisasi di waktu lain.

Umpan balik pada umumnya berasal dari tiga sumber yaitu dari perangkat lunak yang diimplementasikan sendiri, dari pelanggan dan dari anggota tim perangkat lunak.

Kepatuhan ketat dengan praktik – praktik XP tertentu keberanian. Sebuah tim XP yang cepat harus memiliki disiplin untuk melakukan perancangan hanya untuk saat ini dan mnegakui kebutuhan di masa depan berubah secara drastis, sehingga menuntut pengerjaan ulang substansial atas rancangan – rancangan yang telah dibuat serta

menuntut pengerjaan ulang atas kode – kode program komputer yang telah diimplementasikan.

Dengan mengikuti masing – masing nilai penting tersebut, tim cepat menanamkan rasa hormat diantara para anggota tim lainnya dan anggota tim perangkat lunak secara tidak langsung menanamkan rasa hormat untuk perangkat lunak itu sendiri.

XP menggunakan suatu pendekatan berorentasi objek sebagai paradigma pengembangan yang diinginkan dan mencakup di dalamnya seperangkat aturan praktik – praktik yang terjadi didalam konteks empat kerangka kerja yaitu: Perencanaan, Perancangan, Pengkodean, Pengujian.

Perencanaan, kegiatan perencanaan dimulai dengan mendengarkan yaitu suatu kegiatan yang bertujuan untuk mengumpulkan kebutuhan-kebutuhan yang memungkinkan anggota tim teknis XP memahami konteks bisnis untuk perangkat lunak yang akan dikembangkan sehingga mendapatkan inspirasi terkait output, fitur dan fungsionalitas dari perangkat yang akan dikembangkan. Pelanggan memberikan suatu nilai yaitu seluruh nilai bisnis dari fitur atau fungsi. Anggota tim menilai dari apa yang telah di uraikan pelanggan, dari uraian tersebut anggota tim menetapkan biaya yang diukur dari kesulitan serta lamanya proyek pengembangan perangkat lunak. Ketika terbentuk komitmen dasar antara pelanggan dengan pengembang perangkat lunak, maka anggota tim pengembang perangkat lunak akan membuat jadwal pengembangan perangkat lunak, sehingga pelanggan dapat mengetahui perkiraan perangkat lunak tersebut siap untuk dipakai.

Perancangan, perancangan XP mengikuti prinsip “tetap sederhana”. Pada pengembangan menggunakan pendekatan XP, digunakan kartu CRC (Class-responsibility-collaborator), kartu ini digunakan untuk mengidentifikasi dan mengatur kelas – kelas dalam konteks pemograman berorientasi objek. Jika pada identifikasi kebutuhan, kesulitan dalam menentukan masalah perancangan, maka disarankan untuk membuat prototipe operasional dari perancangan tersebut. Prototipe ini selanjutnya diimplementasikan dan dievaluasi oleh pelanggan sehingga pelanggan dapat menvalidasi dari fungsionalitas serta fitur-fitur perangkat lunak yang di kembangkan.

Metode XP ini mendorong refaktorisasi yaitu perancangan perangkat lunak yang terjadi terus menerus ketika sistem di konstruksi, kegiatan ini memberikan tim pedoman bagaimana cara untuk meningkatkan kualitas rancangan.

Pengkodean, setelah cerita pelanggan di kembangkan menjadi sebuah perancangan, langkah selanjutnya adalah mengembangkan rangkaian pengujian , kemudian beralih ke kode–kode program. Kode – kode program tersebut selanjutnnya diimplementasikan. Dari implementasi tersebut akan memberikan umpan balik kepada pengembang. Pada pendekatan ini biasanya diterapkan konsep pemogram berpasangan yaitu satu komputer workstation digunakan untuk menuliskan kode - kode oleh dua orang, sehingga mereka dapat bekerjasama dalam menyelesaikan suatu masalah.

Pengujian, uji kelayakan XP sering disebut uji pelanggan, dirinci oleh pelanggan dan pada dasarnya berfokus pada fitur-fitur dan fungsionalitas-fungsionalitas sistem perangkat lunak secara keseluruhan yang dapat terlihat dan ditinjau kembali oleh pelanggan.

Keuntungan Metodologi Pemrograman Ekstrim:

Metodologi pemrograman ekstrem menekankan pada keterlibatan pelanggan.

Model ini membantu untuk menetapkan rencana dan jadwal yang rasional dan untuk membuat para pengembang secara pribadi berkomitmen untuk jadwal mereka yang tentunya merupakan keuntungan besar dalam model XP.

Model ini konsisten dengan kebanyakan metode pengembangan modern sehingga, pengembang dapat menghasilkan perangkat lunak berkualitas

Kekurangan Metodologi Pemrograman Ekstrim:

Model pengembangan perangkat lunak semacam ini membutuhkan pertemuan pada interval yang sering dengan biaya besar kepada pelanggan.

Ini membutuhkan terlalu banyak perubahan pengembangan yang sangat sulit untuk diadopsi setiap waktu bagi pengembang perangkat lunak.

Dalam metodologi ini, cenderung mustahil untuk diketahui perkiraan pasti dari upaya kerja yang diperlukan untuk memberikan penawaran, karena pada awal proyek tidak ada yang tahu tentang seluruh ruang lingkup dan persyaratan proyek.

IXP merupakan evoluasi organik dari XP. IXP diilhami dengan semangat minimaslis, berpusat pada pelanggan dan yang diarahkan oleh pengujian. IXP didalamnya mencakup enam praktik baru yang dirancang untuk memastikan proyek XP berkerja dengan sukses untuk proyek – proyek perangkat lunak yang signifikan yang ada dalam organisasi – organisasi yang besar. Berikut enam praktik dalam IXP (Roger S. Pressman, 2012):

1) Penilaian kesiapan, sebelum memulai proyek IXP, organisasi pada awalnya melakukan penilaian kesiapan yang bertujuan untuk memastikan lingkungan pengembangan perangkat lunak tepat sudah mendukung IXP, tim perangkat lunak diisi dengan stakeholder yang tepat, organisasi perangkat lunak memiliki program yang berkualitas dan mendukung peningkatannya, budaya organisasi perangkat lunak akan mendukung nilai-nilai baru yang cepat, proyek masyarakat yang lebih luas akan diisi dengan tepat;

2) Komunitas proyek, ketika XP harus diterapkan konsep tim menjadi sebuah komunitas, dalam IXP anggota komunitas perangkat lunak akan ditetapkan perannya dan mekanisme untuk komunikasi dan koordinasi antar anggota komunitas perangkat lunak yang akan dibuat;

3) Pelabelan proyek, pelabelan menguji konteks proyek untuk menentukan bagaimana konteks tersebut melengkapi, memperluas atau menggantikan sistem atau proses yang telah ada sebelumnya.

4) Pengaturan yang diarahkan oleh pengujian, pengaturan yang diarahkan melalui mekanisme pengujian akan menetapkan serangkaian tujuan yang terukur dan kemudian mendefinisikan mekanisme untuk menentukan apakah tujuan-tujuan tersebut telah tercapai atau belum.

5) Retrospektif, yaitu proses penelaahan teknis khusus setelah perangkat lunak dihantarkan ke pelanggan. Penelaahan ini merupakan proses peninjauan isu – isu dari sebuah peningkatan perangkat lunak yang bertujuan untuk meningkatkan proses IXP.

6) Pembelajaran yang berkelanjutan, anggota tim IXP di dorong untuk mempelajari metode-metode dan teknik-teknik yang baru yang dapat menghasilkan produk perangkat lunak yang berkualitas tinggi.

Selain enam praktik diatas, IXP juga melakukan modifikasi beberapa praktik yaitu:

1) Pengembangan yang didorong cerita (story-drivent development, SDD);

2) Perancangan yang didorong oleh ranah (domain–drivent development, DDD);

3) Pemasangan (pairing), Meskipun XP merupakan sebuah pendekatan yang menghasilkan suatu produk yang cepat dihantarkan ke pelanggan, namun beberapa ahli memperdebatkan beberapa isu yaitu diantaranya.

Volatilitas kebutuhan, dalam XP pelanggan masuk ke dalam tim sehingga, perubahan kebutuhan diminta secara informal.

Kebutuhan pelanggan yang bertentangan, dalam proyek perangkat lunak sering ditemukan banyak pelanggan yang memiliki berbeda-beda kebutuhan.

Kebutuhan dinyatakan secara informal.

Kurangnya perancangan formal.

Pengembangan perangkat lunak adaptif (Adaptive Software Development – ASD), Fokus dari ASD adalah kolaborasi manusia. Siklus hidup ASD menggabungkan tiga fase yaitu spekulasi, kolaborasi dan pembelajaran. Spekulasi proyek dimulai dan perencanaan siklus adaptif dilakukan siklus perencanaan adaptif menggunakan informasi inisiasi proyek yaitu pernyataan misi pelanggan, kendala proyek dan kebutuhan dasar yang akan dibutuhkan untuk proyek tersebut. Orang – orang yang termotivasi akan melakukan kolaborasi sehingga dapat meningkatkan kinerja. Kolaborasi ini meliputi komunikasi, kerjasama, menekan individulaisme sehingga akan menciptakan kepercayaan antar anggota tim. Penekanan dari pengembangan komponen siklus adaptif adalah pembelajaran, pembelajaran akan membantu pemahaman anggota tim.

Scrum merupakan suatu metode pengembangan perangkat lunak yang proses pengembangannya mencakup kegiatan kerangka kerja berikut: kebutuhan, analisis, perancangan, evolusi dan penghantaran. Dalam masing – masing kegiatan kerangka kerja, tugas kerja terjadi dalam suatu pola proses yang disebut sebagai sprint. Kerja yang dilakukan dalam suatu sprint disesuaikan dengan masalah yang dihadapi dan didefinisikan dan seiring dirubah secara real time oleh tim scrum.

Backlog merupakan daftar prioritas dari kebutuhan proyek atau fitur yang menyediakan nilai bisnis bagi pelanggan. Item – item dapat ditambahkan pada backlog setiap saat. Para manajer produk menilai backlog dan prioritas update sesuai kebutuhan. Sprint terdiri atas unit kerja yang diperlukan untuk mencapai kebutuhan yang didefinisikan dalam proses backlog yang harus sesuai dengan time-box yang sudah didefiniskan sebelumnya, sprint memungkinkan anggota tim untuk bekerja dalam lingkungan jangka pendek, namun stabil.

Scrum meetings biasanya pendek waktunya, ada tiga pertanyaan yang diajukan dan dijawab anggota tim dalam meeting tersebut yaitu apa yang anda lakukan sejak pertemuan terakhir tim? kendala apa saja yang anda hadapi? apa yang anda rencanakan untuk mencapai sesuatu sebelum rapat tim berikutnya. Seorang pemimpin tim disebut scrum master, akan memimpin rapat dan menilai tanggapan yang muncul. Scrum meeting membantu tim perangkat lunak untuk mengungkap potensi terjadinya masalah – masalah sedini mungkin.

Demo memberikan peningkatan perangkat lunak untuk pelanggan sehingga fungsionalitas yang telah dilaksanakan dapat didemonstrasikan dan dievaluasi oleh pelanggan.

Keuntungan Pengembangan Scrum:

Dalam metodologi ini, pengambilan keputusan sepenuhnya berada di tangan tim Metodologi ini memungkinkan proyek di mana dokumentasi persyaratan bisnis tidak dianggap sangat signifikan untuk pengembangan yang sukses.

Ini adalah metode yang dikontrol ringan yang benar-benar berempati pada pembaruan sering dari kemajuan, oleh karena itu, langkah-langkah pengembangan proyek terlihat dalam metode ini.

Pertemuan harian dengan mudah membantu pengembang untuk memungkinkan mengukur produktivitas individu. Ini mengarah pada peningkatan produktivitas masing-masing anggota tim.

Kekurangan Pengembangan Scrum:

Model pengembangan semacam ini menyulitkan jika perkiraan biaya dan waktu proyek tidak akan akurat.

Ini bagus untuk proyek kecil yang bergerak cepat tetapi tidak cocok untuk proyek ukuran besar.

Metodologi ini membutuhkan anggota tim yang berpengalaman saja. Jika tim terdiri dari orang-orang yang pemula, proyek tidak dapat diselesaikan dalam jangka waktu yang tepat.

Metode Pengembangan Sistem, Dinamik (Dynamic System Development Methode - DSDM), Merupakan sebuah pendekatan pengembangan perangkat lunak cepat yang menyediakan kerangka kerja untuk membangun dan memelihara sistem yang memenuhi batas waktu yang ketat melalui penggunaan protipe yang secara perlahan ditambahi dan diperluas dalam lingkungan proyek terkendali. Siklus hidup dari DSDM ini meliputi: Studi Kelayakan, Studi Bisnis, Iterasi Model Fungsional, Perancangan dan Membangun Iterasi, Implementasi.

Keuntungan dari Model Pengembangan Sistem Dinamis:

Pengguna sangat terlibat dalam pengembangan sistem sehingga, mereka lebih mungkin untuk mendapatkan pegangan pada proyek pengembangan perangkat lunak.

Dalam model ini, fungsi dasar disampaikan dengan cepat, dengan lebih banyak fungsi yang dikirimkan pada interval yang sering.

Metode ini memberikan akses yang mudah oleh pengembang kepada pengguna akhir.

Dalam pengembangan semacam ini, pendekatan proyek disampaikan tepat waktu dan sesuai anggaran tertentu.

Kekurangan Model Pengembangan Sistem Dinamis:

Hal pertama adalah DSDM sangat mahal untuk diterapkan, karena membutuhkan pengguna dan pengembang untuk dilatih untuk menggunakannya secara efektif.

Ini mungkin tidak cocok untuk organisasi kecil atau proyek satu kali. Ini adalah model yang relatif baru, oleh karena itu, tidak terlalu umum dan mudah dimengerti.

Metode Crystal adalah keluarga metodologi pengembangan perangkat lunak yang dikembangkan oleh Alistair Cockburn dari studinya dan wawancara tim. Metode ini

diberi kode warna untuk menandakan risiko terhadap kehidupan manusia. Misalnya, proyek yang mungkin melibatkan risiko terhadap kehidupan manusia akan menggunakan Crystal Sapphire sementara proyek yang tidak memiliki risiko seperti itu akan menggunakan Crystal Clear.

Crystal berfokus pada enam aspek utama: orang, interaksi, komunitas, komunikasi, keterampilan, dan bakat (SANTOS, 2017).

Metode Crystal menganggap orang sebagai yang paling penting, jadi proses harus dimodelkan untuk memenuhi persyaratan tim. Ini adaptif, tanpa seperangkat alat dan teknik yang ditentukan. Ini juga ringan, tanpa terlalu banyak dokumentasi, manajemen atau pelaporan. Bobot metodologi ditentukan oleh lingkungan proyek dan ukuran tim.

Pengembangan Dikendalikan Fitur (Feature Driven Development – FDD), Feature-Driven Development (FDD) diperkenalkan pada tahun 1997 oleh Jeff De Luca ketika ia bekerja dalam proyek pengembangan perangkat lunak untuk sebuah bank Singapura yang besar. FDD juga dibangun di sekitar praktik terbaik rekayasa perangkat lunak seperti pemodelan objek domain, yang dikembangkan oleh fitur dan kepemilikan kode.

Perpaduan dari praktik-praktik ini yang menghasilkan keseluruhan kohesif adalah karakteristik terbaik dari FDD. Ini terdiri dari lima kegiatan dasar, yaitu pengembangan model keseluruhan, pembangunan daftar fitur, perencanaan berdasarkan fitur, perancangan berdasarkan fitur, dan pembangunan berdasarkan fitur. Setiap proyek akan memiliki model uniknya sendiri, yang akan menghasilkan daftar fitur. Tiga aktivitas terakhir adalah proses berulang yang singkat, dengan fitur yang tidak membutuhkan waktu lebih lama dari dua minggu untuk dibangun. Jika akan memakan waktu lebih dari dua minggu, maka harus dipecah menjadi fitur yang lebih kecil.

Tujuan utama FDD adalah menyampaikan perangkat lunak yang nyata dan bekerja secara berulang pada waktu yang tepat. Keuntungan menggunakan FDD adalah scalable bahkan untuk tim besar karena konsep ‘just enough design first’ (JEDI). Ini adalah solusi yang bagus untuk mempertahankan kendali atas proyek yang lincah, inkremental dan inheren kompleks karena prosesnya yang berpusat pada fitur. Ini digunakan dalam proyek-proyek perusahaan serta proyek web seperti situs game online Mousebraker.com.

Keuntungan Metodologi FDD:

FDD Membantu memindahkan proyek ukuran yang lebih besar dan memperoleh keberhasilan yang berulang;

Lima proses sederhana membantu menyelesaikan pekerjaan dalam waktu singkat dan cara termudah;

Jenis model ini dibangun di atas standar yang ditetapkan untuk industry pengembangan perangkat lunak sehingga membantu pengembangan yang mudah dan praktik terbaik yang diakui industry.

Kekurangan Metodologi FDD:

Bukan metodologi yang ideal untuk proyek yang lebih kecil, jadi tidak baik untuk pengembang perorangan;

Ketergantungan yang tinggi pada pengembang utama berarti orang tersebut harus dilengkapi sepenuhnya untuk bertindak sebagai koordinator, desainer utama, dan mentor;

Tidak ada dokumentasi tertulis yang diberikan kepada klien dalam metodologi ini, sehingga mereka tidak dapat memperoleh bukti untuk perangkat lunak mereka sendiri.

Pengembangan Perangkat Lunak yang Ringkas (Lean Software Development - LSD) berfokus pada penciptaan perangkat lunak yang mudah berubah. Model Pengembangan Perangkat Lunak ini lebih strategis daripada berbagai jenis metodologi tangkas lainnya.

Tujuan dari metodologi ini adalah untuk mengembangkan perangkat lunak dalam sepertiga waktu, dengan anggaran yang sangat terbatas, dan jumlah alur kerja yang sangat sedikit.

Prinsip-prinsip ringkas yang diilhami LSD dapat diringkas menjadi menghilangkan pemborosan, membangun kualitas, menciptakan pengetahuan, menunda komitmen, cepat menghantarkan, menghargai orang dan mengoptimalkan keseluruhan.

Keuntungan Metodologi Pengembangan Lean:

Penghapusan awal efisiensi keseluruhan proses pengembangan tentu membantu mempercepat proses pengembangan perangkat lunak keseluruhan yang pasti mengurangi biaya proyek.

Menyampaikan produk lebih awal adalah keuntungan yang pasti. Ini berarti bahwa tim pengembangan dapat memberikan lebih banyak fungsi dalam waktu yang lebih singkat, sehingga memungkinkan lebih banyak proyek yang akan dikirimkan.

Pemberdayaan tim pengembangan membantu dalam mengembangkan kemampuan pengambilan keputusan dari anggota tim yang menciptakan motivasi lebih di antara anggota tim.

Kekurangan Metodologi Pengembangan Lean:

Keberhasilan dalam pengembangan perangkat lunak tergantung pada bagaimana disiplin para anggota tim dan bagaimana memajukan keterampilan teknis mereka.

Peran seorang analis bisnis sangat penting untuk memastikan dokumentasi persyaratan bisnis dipahami dengan benar. Jika ada organisasi yang tidak memiliki orang dengan analis bisnis yang tepat maka metode ini mungkin tidak berguna bagi mereka.

Dalam model pengembangan ini, fleksibilitas yang besar diberikan kepada pengembang yang pasti hebat, tetapi terlalu banyak dari itu akan dengan cepat mengarah ke tim pengembangan yang kehilangan fokus pada tujuan awalnya sehingga, hati aliran seluruh pekerjaan pengembangan proyek.

Pemodelan Cepat (Agile Methode), Scott Ambler dalam mengemukakan bahwa pemodelan cepat adalah metodologi berbasis praktik untuk pemodelan dan dokumentasi berbasis perangkat lunak yang efektif, yang di dalamnya terdapat kumpulan dari nilai – nilai, prinsip dan praktik untuk model perangkat lunak yang dapat diterapkan pada proyek pengembangan perangkat lunak secara efektif dan secara ringan.

Keuntungan Metodologi Pengembangan Agile:

Metodologi tangkas memiliki pendekatan adaptif yang mampu menanggapi perubahan persyaratan dari klien.

Komunikasi langsung dan umpan balik konstan dari perwakilan pelanggan tidak menyisakan ruang untuk dugaan apa pun dalam sistem.

Kekurangan Metodologi Pengembangan Agile:

Metodologi ini berfokus pada perangkat lunak yang bekerja daripada dokumentasi, sehingga dapat mengakibatkan kurangnya dokumentasi.

Proyek pengembangan perangkat lunak dapat keluar jalur jika pelanggan tidak begitu jelas tentang hasil akhir proyeknya.

Pemodelan Terpadu Secara Cepat (Agile Unified Process - AUP), AUP merupakan pendekatan yang menyediakan suatu susunan yang bersifat serial yaitu urutan linier dari kegiatan rekayasa perangkat lunak yang memungkinkan sebuah tim perangkat lunak memvisualkan keseluruhan aliran proses untuk sebuah model perangkat lunak, namun dalam setiap kegiatan, tim berulang untuk mencapai kecepatan dan untuk memberikan peningkatan perangkat lunak yang berarti kepada pengguna akhir secepat mungkin. Pada setiap pengulangan AUP mengacup pada kegiatan berikut: Pemodelan, representasi UML untuk bisnis dan ranah masalah diciptakan; Pelaksanaan, model tersebut diterjemahkan ke dalam kode sumber; Pengujian, seperti XP, tim perangkat lunak akan merancang dan menjalankan serangkaian pengujian untuk menemukan kesalahan-kesalahan dalam kode sumber yang telah dihasilkan melalui tahapan sebelumnya dan memastikan bahwa kode sumber memenuhi kebutuhan dan harapan para penggunanya; Pemasangan, seperti kegiatan proses generik, pemasangan dalam konteks ini berfokus pada penghantaran peningkatan perangkat lunak dan perolehan umpan balik dari pengguna akhir; Konfigurasi dan manajemen proyek, dalam konteks AUP, manajemen konfigurasi menunjukkan perubahan, manajemen risko dan pengendalian produk kerja yang terus-menerus yang dihasilkan oleh tim. Manajemen proyek menelusuri dan mengendalikan kemajuan tim dan mengoordinasikan kegiatan tim; Lingkungan manajemen, mengkoordinasikan infrastruktur proses yang meliputi standar, alat dan teknologi pendukung lainnya yang tersedia bagi tim.