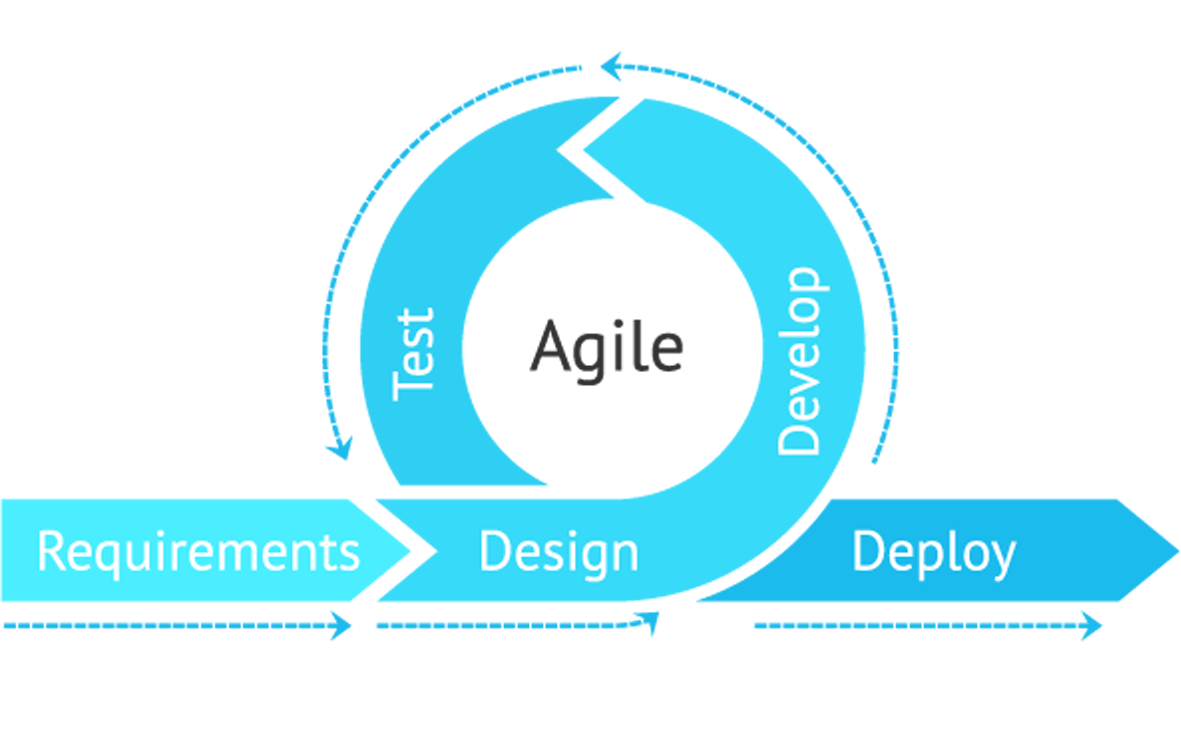
Ridwan Rizky Ramadhan  
A11.2019.12075  
A11.4606

**TUGAS PROYEK PERANGKAT LUNAK**

1. **Metode Agile**



Agile adalah metode pengembangan perangkat lunak yang dilakukan secara bertahap dan berulang (iterasi). Agile Development sering disebut sebagai kerangka kerja karena di dalamnya memang terdapat berbagai metode yang bisa digunakan sesuai kebutuhan. Kami akan membahas lebih lanjut di bawah. Setiap iterasi pada Agile juga berbeda-beda durasinya, tergantung proyek yang Anda lakukan dan metode yang digunakan. Namun, umumnya iterasi berjalan antara satu sampai empat minggu. Iterasi pada Agile fokus pada upaya pengembangan perangkat lunak yang cepat sesuai dengan perubahan kebutuhan konsumen dengan melibatkan semua tim. Contohnya, Anda sedang mengembangkan aplikasi peta. Lalu, tiba-tiba terjadi pandemi yang memunculkan kebutuhan baru di masyarakat tentang update informasi sebaran virus. Agile merupakan alternatif dari metode Waterfall yang linear dan tak bisa diubah di tengah proses pengembangan. Masalahnya, kebutuhan pasar mulai berubah dengan cepat. Efeknya, banyak software dengan metode waterfall yang tak lagi dibutuhkan di pasar saat dirilis.

* **12 Prinsip Utama dari Agile Software Development**

**Pengembangan perangkat lunak menggunakan metode Agile memiliki 12 prinsip utama yang dikenal dalam Agile Manifesto. Berikut ini merupakan kedua belas prinsip yang dimiliki oleh Agile Software Development :**

1. Menekankan pada kepuasan dari klien dengan menggunakan sebagai prioritas utama dalam menghasilkan produk lebih awal dan berkelanjutan.

2. Menerima segala bentuk perubahan saat proses pengembangan perangkat lunak meskipun berada di tahap akhir pengembangan.

3. produk berupa perangkat lunak yang dibuat dengan jangka waktu sedikit (2 minggu – 2 bulan), dengan kualitas yang teruji.

4. Terjadinya proses kerjasama yang baik antara pihak pengembang dengan pebisnis selama proyek berlangsung.

5. Membangun suasana lingkungan yang berisi orang – orang bermotivasi tinggi. Agar dapat menyelesaikan proyek dengan efektif dan efisien.

6. Komunikasi secara langsung sangat dibutuhkan dalam proses pengembangan sebuah perangkat lunak

7. Software yang bekerja dengan baik dan sempurna merupakan sebuah ukuran darikemajuan proyek.

8. Metode Agile dapat mengembangkan perangkat lunak secara berkelanjutan dari dukungan setiap pihak seperti sponsor, pengguna dan pengembang itu sendiri.

9. Keunggulan dari segi teknis menjadi keutamaan dalam pengembangan perangkat lunak menggunakan metode Agile.

10. Kesederhanaan disini penting bagi Agile sendiri dalam memaksimalkan sumber daya (sumber daya) yang ada.

11. Segala kebutuhan dari segi arsitektur dan kebutuhan software sangat tergantung pada manajemen setiap tim pengembang.

12. Secara berkala, setiap tim pengembang melakukan evaluasi diri (refleksi) untuk bekerja lebih efektif dan mengatur pola kerja mereka.

* **Tujuan Metode Agile**

Lalu, untuk apa sih Agile diciptakan padahal sudah ada metode pengembangan software lain? Secara garis besar, tujuan metode Agile adalah:

1. High-value and working App system — Software yang diciptakan berfungsi baik, nilai jualnya tinggi, dan biaya pembuatannya rendah.
2. Iterative, incremental, evolutionary — Pengembangan software terbuka dengan perubahan di mana tim mampu bekerja dengan singkat dalam menambah fitur sesuai kebutuhan konsumen.
3. Cost control & value-driven development — Software dikembangkan dengan sesuai kebutuhan pengguna dengan waktu dan biaya pengembangan yang dikontrol.
4. High-quality production — Kualitas software bisa terjaga dengan melakukan tes menyeluruh di setiap iterasinya.
5. Flexible & risk management — Proses pengembangan software disiapkan untuk menghadapi perubahan berbekal kemampuan adaptasi yang baik. Jadi, dapat meminimalisir kegagalan.
6. Collaboration — Kolaborasi tim lebih maksimal karena rutin bertemu untuk membahas perkembangan project.
7. Self-organizing, self-managing teams — Tim bisa mengatur dirinya sendiri dengan support dari pimpinan sehingga tercipta tim yang solid.

* **Kelebihan dan Kekurangan Metode Agile**

**Kelebihan metode Agile**

Berikut adalah beberapa manfaat Agile yang membuatnya diterapkan oleh berbagai tim di seluruh dunia. Apa saja?  
  
1. Kualitas Software Lebih Baik

Dengan Agile Methodology, Anda bisa dengan cepat menerapkan setiap feedback dari konsumen di iterasi selanjutnya. Baik itu tentang penambahan fitur atau memperbaiki bug. Alhasil, kualitas software Anda akan membaik karena lebih sesuai dengan keinginan konsumen.

2. Konsumen Puas dan Lebih Dihargai

Software dengan kualitas yang baik akan disukai konsumen. Apalagi kalau fitur software di dalamnya sesuai dengan keinginan konsumen berkat feedback yang diberikan. Dengan begitu, mereka akan merasa lebih dihargai sebagai konsumen.

3. Fleksibilitas Tinggi

Metode Agile memungkinkan Anda melakukan perubahan rutin sesuai feedback konsumen pada software yang dikerjakan. Bila ada banyak feedback, Anda bisa memilih akan melakukan perubahan apa di setiap iterasinya. Jadi, Anda tidak harus saklek mengikuti sebuah rencana dari awal sampai akhir.

4. Software Cepat Selesai

Metode Agile berfokus mengembangkan software yang benar-benar dibutuhkan konsumen. Jadi, software akan cepat selesai karena Anda tak menghabiskan waktu menambahkan banyak fitur yang belum tentu dibutuhkan konsumen.

5. Pengembangan yang Lebih Terprediksi

Setiap Agile methodology mempunyai iterasi rutin yang fokus pada pengembangan bertahap. Jadi, pengembangan software Anda akan lebih terprediksi, sehingga Anda bisa tahu berbagai pengeluaran dengan baik. Efeknya, risiko bisnis akan berkurang.

**Kekurangan metode Agile**

Tentu saja agile juga mempunyai kekurangan yang perlu Anda ketahui, sebagai berikut:

1. Produk Akhir yang Kurang Jelas

Agile tidak membutuhkan banyak perencanaan, sehingga tim Anda akan kesulitan menentukan bentuk akhir software-nya. Apalagi, fitur baru juga selalu ditambahkan dan membuat software Anda semakin kompleks. Efeknya, selama kebutuhan konsumen berubah-ubah, Anda juga akan semakin sulit menentukan bentuk akhirnya.

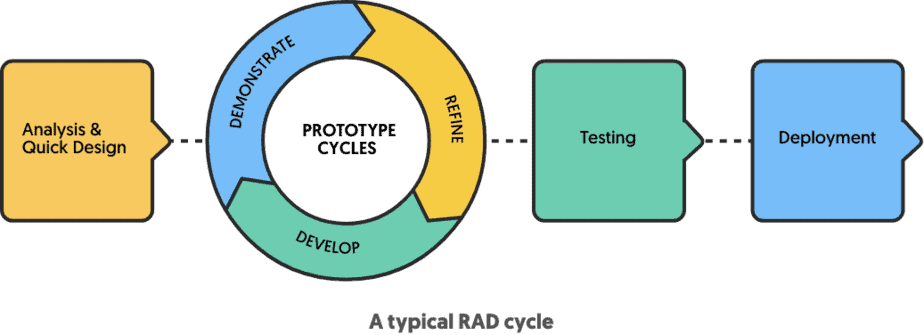
2. Bergantung Pada Komitmen Tinggi Tim

Agile mengharuskan semua pihak untuk selalu berinteraksi secara rutin agar software sesuai kebutuhan konsumen. Mulai dari memberikan feedback, menerapkan perubahan, melakukan test, dan sebagainya. Ini tentu saja membutuhkan komitmen tinggi dari setiap individu, sehingga akan menguras banyak energi dan waktu Anda.

3. Dokumentasi yang Kurang Lengkap

Dalam agile, dokumentasi dibuat dalam waktu yang singkat sehingga berujung pada hasil yang kurang lengkap. Artinya, saat ada anggota baru di tim, ia akan kesulitan memahami berbagai hal tentang project yang berjalan. Efeknya, akan timbul miskomunikasi yang bisa memperlambat project Anda.

1. **Metode RAD**



Rapid Application Development atau RAD adalah salah satu metode pengembangan aplikasi yang kerap dipakai saat ini.

Metode ini menekankan pada proses pembuatan aplikasi berdasarkan pembuatan prototype, iterasi, dan feedback yang berulang-ulang.

Dengan begitu, aplikasi yang dibuat bisa dikembangkan dan diperbaiki dengan cepat. Sangat cocok dengan kebutuhan dan perkembangan dunia digital yang super cepat.

Berbeda sekali dengan metode pengembangan lain. Waterfall Model, misalnya, yang dianggap kurang efektif. Sebab, Waterfall model mengharuskan aplikasi dirancang dari A sampai Z. Baru kemudian bisa dieksekusi. Jadi, butuh waktu lama sekali untuk sebuah aplikasi bisa dirilis.

Ada empat tahapan RAD yang perlu dilalui developer ketika mengembangkan aplikasi. Keempat tahapan itu adalah:

* Menentukan kebutuhan proyek
* Membuat prototype
* Rapid construction dan feedback
* Implementasi dan finalisasi produk

Dari keempat tahapan tersebut memiliki penjelasan sebagai berikut :

1. **Menentukan kebutuhan proyek**

RAD dimulai dengan menentukan kebutuhan sebuah proyek (project requirements). Pada tahap ini, tim perlu menentukan kebutuhan yang ingin dipenuhi dari sebuah proyek. Kebutuhan ini tidak perlu spesifik. Tapi, sifatnya benar-benar umum dan jumlahnya bisa banyak. Baru dari situ, tim akan menentukan mana kebutuhan yang perlu diprioritaskan.

Setelah mendapatkan kebutuhan yang jelas, barulah tim menentukan hal-hal yang lebih detail. Misalkan seperti tujuan, timeline, dan budget yang diperlukan. Semua anggota tim juga perlu memikirkan apa saja masalah yang mungkin muncul dalam proses pengembangan aplikasi. Tak lupa, mereka juga memikirkan strategi yang tepat untuk menyelesaikannya. Intinya, tahap awal ini berguna untuk memberikan gambaran luas pada proyek yang ingin dikerjakan. Dalam prosesnya, tim bisa saja beralih ke hal-hal yang lebih spesifik.

Jadi, prosesnya sangat ✨ fleksibel ✨.

1. **Membuat prototype**

Hal yang selanjutnya dilakukan adalah membuat prototype. Developer secepat mungkin akan membuat prototype dari aplikasi yang diinginkan. Lengkap dengan fitur dan fungsi yang berbeda-beda. Tujuannya, sekadar untuk mengecek apakah prototype yang dibuat sudah sesuai dengan kebutuhan klien.Meski begitu, tahap ini bisa saja dilakukan berulang-ulang. Kadang juga melibatkan user untuk testing dan memberikan feedback. Proses ini memungkinkan tim mempelajari error yang mungkin muncul ke depannya. Ini berguna untuk mengurangi error dan debugging. Lewat tahapan ini, tim developer memiliki modal untuk membuat aplikasi yang mudah dipakai, stabil, tidak sering error, dan desainnya pun oke.

1. **Proses pengembangan dan pengumpulan feedback**

Setelah tahu aplikasi seperti apa yang ingin dibuat, developer mengubah prototype ke bentuk aplikasi versi beta sampai dengan final. Jadi, bisa dibilang tahap RAD inilah yang cukup intens. Developer terus-menerus melakukan coding aplikasi, melakukan testing sistem, dan integrasi dengan bagian-bagian lainnya. Karena itulah, developer menggunakan tools dan framework yang mendukung RAD agar cepat.   
Apalagi proses ini terus diulang sambil terus mempertimbangkan feedback dari klien. Baik itu soal fitur, fungsi, interface, sampai keseluruhan aspek dari produk yang dibuat. Nah, kalau prosesnya berjalan lancar, developer akan melanjutkan ke langkah berikutnya. Yaitu, finalisasi produk atau implementasi.  
Kalau pun tidak, proses ini kemungkinan akan terus diulang. Pun, kalau apes-apesnya aplikasi tidak tidak menjawab kebutuhan, developer akan kembali ke proses prototyping.

1. **Implementasi dan finalisasi produk**

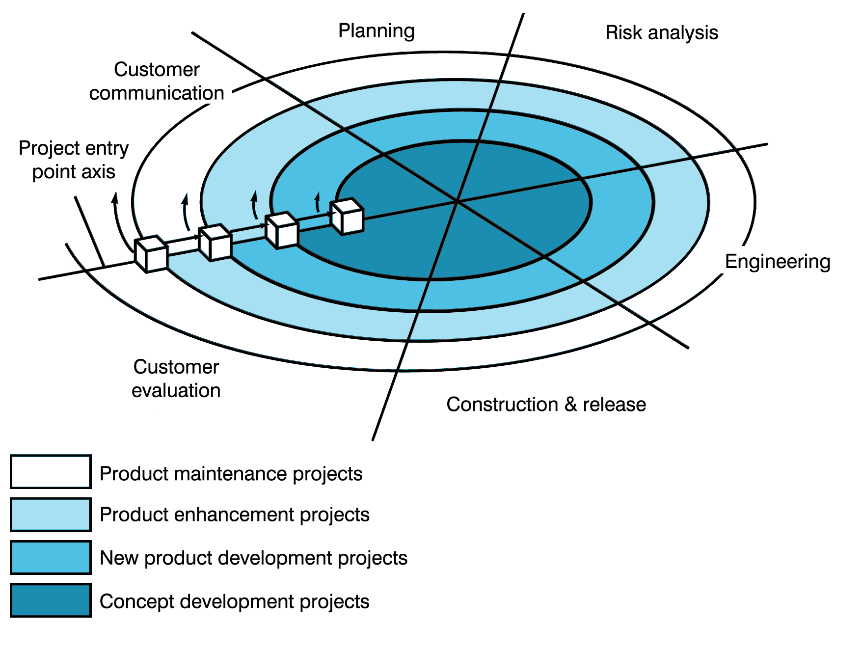
Di sini, tugas utama developer adalah menambal kekurangan yang mungkin terjadi ketika proses pengembangan aplikasi. Tugas ini termasuk melakukan optimasi untuk stabilitas aplikasinya, memperbaiki interface, hingga melakukan maintenance dan menyusun dokumentasi. Pokoknya, langkah terakhir ini dilakukan sebelum menyerahkan aplikasi ke klien.

**Kelebihan dan Kekurangan RAD**Sebagai salah satu model pengembangan aplikasi, RAD tentu saja memiliki kelebihan dan kekurangannya masing-masing. Berikut adalah penjelasan tentang kelebihan dan kekurangan RAD yang dirangkum secara singkat.

**Kelebihan Metode RAD**1. Kebutuhan aplikasi bisa berubah sewaktu-waktu.  
2. Aplikasi dikembangkan berdasarkan kebutuhan dan keinginan user.  
3. Memperkecil kemungkinan error dan hal buruk lainnya.  
4. Waktu pengembangan aplikasi bisa lebih cepat dan efektif.  
5. Mempermudah proses integrasi.

**Kekurangan Metode RAD**

1. Membutuhkan tim dengan skill teknis yang mumpuni.
2. Memerlukan kolaborasi tim yang kuat.
3. Hanya cocok untuk proyek yang waktunya singkat.
4. Hanya cocok digunakan untuk mengembangkan aplikasi secara modular (fokus ke suatu fitur untuk dijadikan module terpisah).
5. Sulit diterapkan untuk mengembangkan aplikasi besar.
6. **Metode Spiral**



**Pengertian Spiral Model**

Model Spiral merupakan penggabungan dari model prototyping dan model waterfall. Model prototyping yang fokus pada penyajian atau presentasi kepada user dengan format input dan output kemudian perangkat lunak akan dievaluasi. Metode waterfall yang fokus kepada proses pengembangan perangkat lunak yang sistematis atau berurutan. Pada model spiral ini menekankan pada analisa resiko setiap tahapannya.

**Fase Spiral Model :**

1. Identification  
   Bertujuan untuk mengumpulkan kebutuhan bisnis di dasar spiral. Identifikasi persyaratan sistem, subsistem, dan unit. Fase ini juga mencakup komunikasi antar user dan development.
2. Design  
   Fase ini dimulai dengan mendesain konseptual di dasar spiral yang melibatkan desain arsitektur, desain logis dari modul, desain produk fisik, dan desain akhir.
3. Construct or Build  
   Fase ini mengarah pada produksi produk perangkat lunak yang sebenarnya pada tiap spiral.
4. Evaluation and Risk Analysis  
   Pada fase ini mengidentifikasi, memperkirakan dan memantau kelayakan teknis dan risiko manajemen, seperti jadwal dan biaya. Setelah pengujian sistem, akhir dari iterasi user akan mengevaluasi prosuk yang sudah dibangun dan akan memberikan feedback.

**Kelebihan Spiral Model**

1. Setiap tahap pengerjaan dibuat prototyping sehingga kekurangan dan apa yang diharapkan oleh user dapat diperjelas dan juga dapat menjadi acuan untuk user dalam mencari kekurangan kebutuhan.
2. Lebih cocok untuk pengembangan sistem dan perangkat lunak skala besar.
3. Dapat disesuaikan agar perangkat lunak bisa dipakai selama hidup perangkat lunak komputer.
4. Pengembang dan pemakai dapat lebih mudah memahami dan bereaksi terhadap resiko setiap tingkat evolusi karena perangkat lunak terus bekerja selama proses.
5. Menggunakan prototipe sebagai mekanisme pengurangan resiko dan pada setiap keadaan di dalam evolusi produk.
6. Tetap mengikuti langkah-langkah dalam siklus kehidupan klasik dan memasukkannya ke dalam kerangka kerja iteratif.
7. Membutuhkan pertimbangan langsung terhadp resiko teknis sehingga mengurangi resiko sebelum menjadi permaslahan yang serius.

**Kekurangan Spiral Model**

1. Banyak user yang tidak percaya bahwa pendekatan secara evolusioner dapat dikontrol oleh kedua pihak. Model spiral mempunyai resiko yang harus dipertimbangkan ulang oleh user dan developer.
2. Memerlukan tenaga ahli untuk memperkirakan resiko, dan harus mengandalkannya supaya sukses.
3. Belum terbukti apakah metode ini cukup efisien karena usianya yang relatif baru.
4. Memerlukan penaksiran resiko yang masuk akal dan akan menjadi masalah yang serius jika resiko mayor tidak ditemukan dan diatur.
5. Butuh waktu lama untuk menerapkan paradigma ini menuju kepastian yang absolute.

Kapan harus memilih metodologi spiral?

1. Dalam rekayasa perangkat lunak digunakan saat proyek besar
2. Ketika pelepasan harus sering dilakukan
3. Kapan pembuatan prototipe dapat diterapkan
4. Ketika evaluasi risiko dan biaya penting
5. Metodologi spiral berguna untuk proyek berisiko menengah hingga tinggi
6. Jika persyaratan tidak jelas dan kompleks
7. Ketika perubahan mungkin diperlukan kapan saja
8. Ketika komitmen proyek jangka panjang tidak memungkinkan karena adanya perubahan dalam prioritas ekonomi