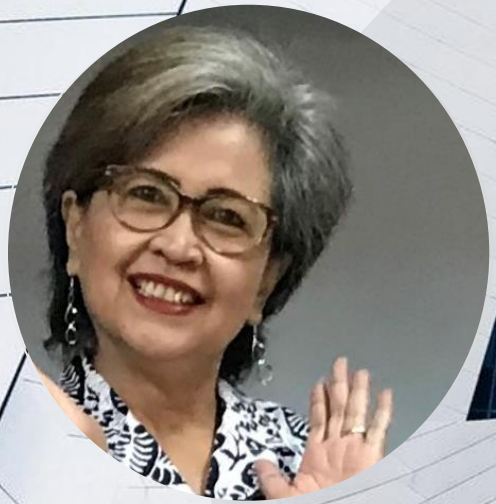
Pertemuan 3:

Agility & Process

Meuthia Rachmaniah
Departemen Ilmu Komputer, FMIPA IPB

Software Engineering: A Practitioner's Approach, Ninth Ed.
Roger S. Pressman dan Bruce R. Maxim

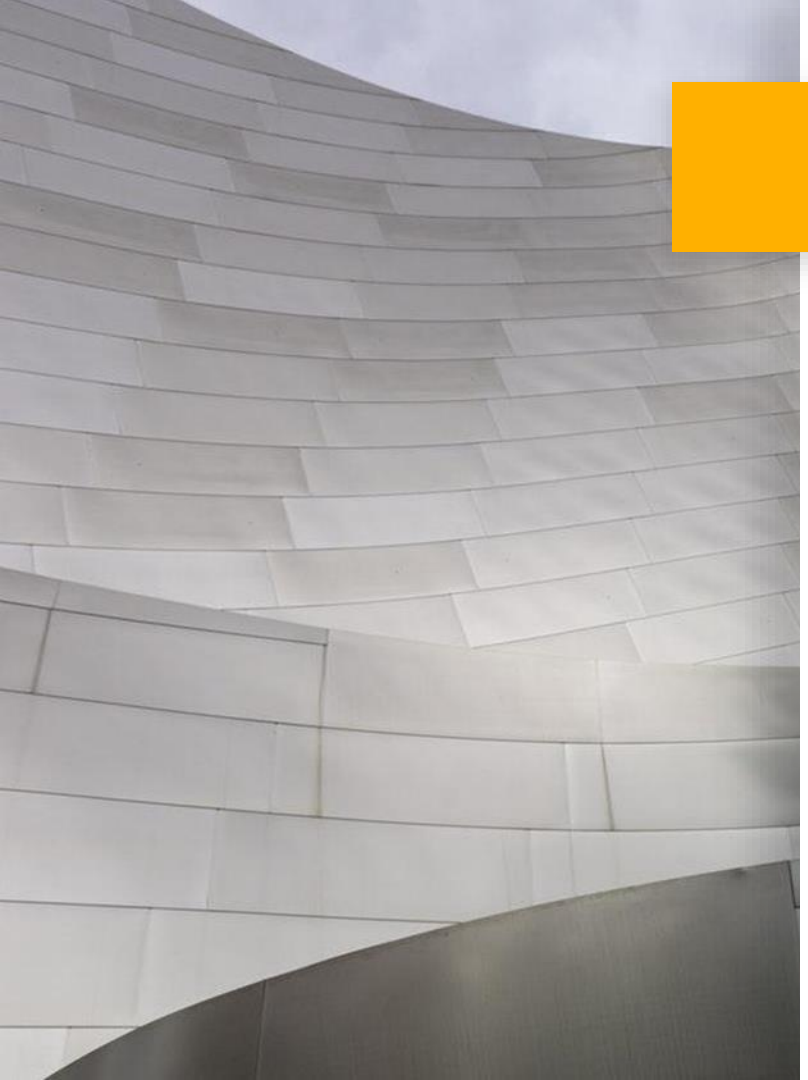


Hello!

Ir. Meuthia Rachmaniah, M.Sc.
Associate Professor
NIP 195907111984032006

Computer Science Department
Software Engineering & Information Sciences
meuthiara@apps.ipb.ac.id

Satyalancana Karya Satya 20 Tahun
Satyalancana Karya Satya 30 Tahun



Cakupan Materi

Quick Look

What Is It?

- RPL Agile mengkombinasikan filosofi dan set pedoman pengembangan
- Filosofi RPL Agile mengutamakan customer satisfaction dan pengiriman awal dari increment PL; tim proyek yang kecil, sangat bermotivasi; metode informal; work products RPL minimal; dan kesederhanaan seluruh pengembangan
- Pedoman pengembangan menekankan penyampaian pada analisis dan desain (walaupun kegiatan ini tidak dianjurkan)

Who Does It

- Software Engineer dan stakeholder proyek lainnya (manajer, pelanggan, pengguna akhir) bekerja sama dalam tim agile—tim yang mengatur dan mengendalikan diri dan nasibnya sendiri
- Tim agile memupuk komunikasi dan kolaborasi di antara semua yang terlibat di dalamnya

What Are The Steps

- Framework Aktivitas—komunikasi, perencanaan, pemodelan, konstruksi, dan deployment ada. Tetapi mereka berubah menjadi kumpulan tugas minimal yang mendorong tim proyek menuju konstruksi dan pengiriman

Why Is It Important?

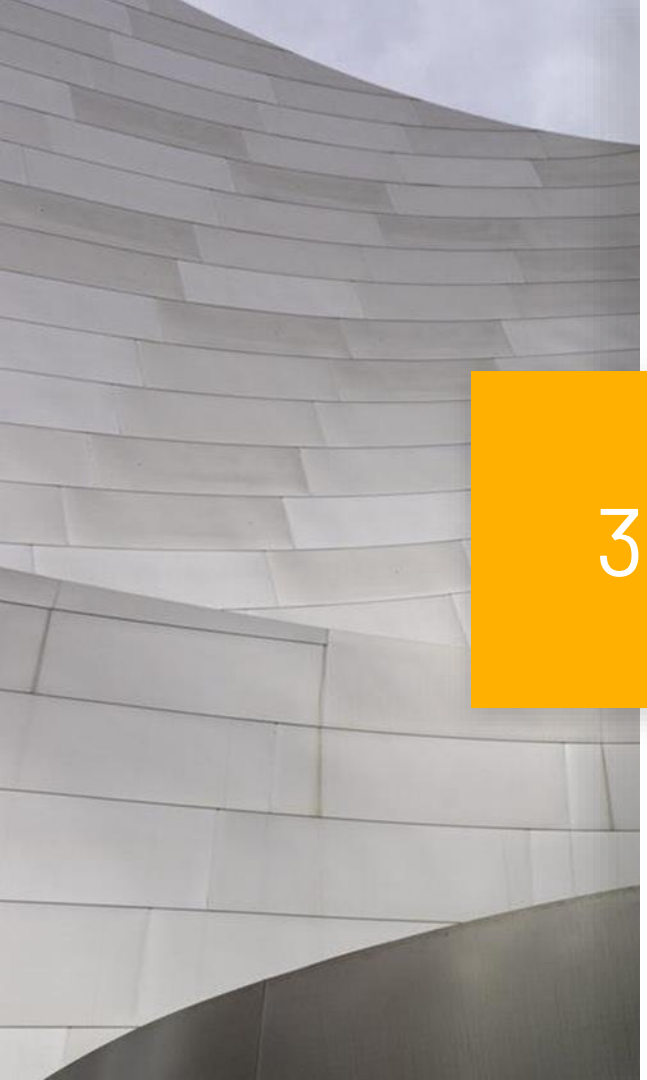
- Lingkungan bisnis modern menelurkan sistem berbasis komputer dan produk PL yang serba cepat dan selalu berubah
- RPL agile merupakan alternatif yang masuk akal untuk RPL konvensional. Telah ditunjukkan untuk memberikan sistem yang sukses dengan cepat

What Is The Work Product

- Work product paling penting ialah ‘software increment” yang operasional yang diserahkan ke konsumen tepat waktu
- Dokumen paling penting ialah user story dan kasus uji terkait

How Do I Ensure That I’ve Done It Right

- Jika tim agile memenuhi bahwa prosesnya berhasil, dan tim menghasilkan peningkatan PL yang dapat dikirimkan yang memuaskan pelanggan, maka Anda telah melakukannya dengan benar



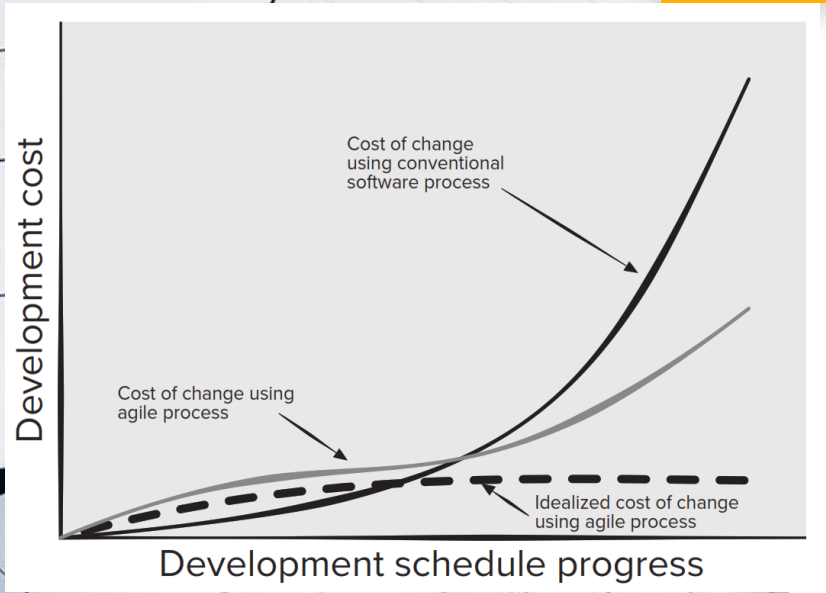
3.1 What Is Agility

3.1 What Is Agility (*Kelincahan*)

- Respons yang efektif terhadap perubahan
- Mendorong struktur dan sikap tim yang membuat komunikasi (antara anggota tim, antara teknolog dan pebisnis, dan antara perekayasa PL dan manajer) lebih mudah
- Menekankan pengiriman cepat PL yang beroperasi dan menekankan pentingnya work products menengah (tidak selalu merupakan hal yang baik)
- Mengadopsi pelanggan sebagai bagian dari tim pengembangan dan bekerja untuk menghilangkan sikap "kita dan mereka" yang terus meliputi banyak proyek perangkat lunak
- Mengakui bahwa perencanaan di dunia dan bahwa rencana proyek tidak pasti memiliki batasnya harus fleksibel
- Agility dapat diterapkan pada proses PL apa pun. Namun penting agar:
  - proses dirancang sedemikian rupa sehingga memungkinkan tim proyek untuk menyesuaikan tugas dan merampingkannya,
  - melakukan perencanaan dengan cara yang memahami fluiditas pendekatan pengembangan agile,
  - menghilangkan semua kecuali work products yang paling yang penting dan menjaganya tetap ramping,
  - menekankan strategi pengiriman increment yang memberikan PL yang berfungsi kepada pelanggan secepat mungkin untuk jenis produk dan lingkungan operasional

3.2 Agility and The Cost of Change

Perubahan biaya sebagai fungsi waktu dalam pengembangan



- Proses PL konvensional adalah bahwa biaya perubahan meningkat secara nonlinier seiring berjalannya proyek.
  - Tetapi jika di tengah pengujian validasi, stakeholder meminta perubahan fungsional utama yang memerlukan modifikasi desain arsitektur, konstruksi komponen baru, perubahan komponen lain yang sudah ada, pengujian baru dan sebagainya. Maka biaya meningkat dengan cepat
- Proses agile yang dirancang dengan baik "meratakan" kurva biaya perubahan, memungkinkan tim PL untuk mengakomodasi perubahan di akhir proyek PL tanpa dampak biaya dan waktu yang dramatis

3.3 What Is An Agile Process

Karakteristik Agile Process

1. Kesulitan Memprediksi Kebutuhan
   - Sulit untuk memprediksi sebelumnya kebutuhan PL mana yang akan bertahan dan mana yang akan berubah.
   - Sama sulitnya untuk memprediksi bagaimana prioritas pelanggan akan berubah seiring berjalannya proyek

2. Pada banyak PL, Desain dan Konstruksi Disisipkan
   - Artinya, kedua kegiatan tersebut harus dilakukan secara bersamaan sehingga model desain terbukti saat dibuat.
   - Sulit untuk memprediksi berapa banyak desain yang diperlukan sebelum konstruksi digunakan untuk membuktikan desain

3. Dari sudut pandang perencanaan: analisis, desain, konstruksi, dan pengujian tidak dapat diprediksi seperti yang kita inginkan

3.3.1 Duabelas Prinsip Agility

6. Metode penyampaian informasi yang paling efisien dan efektif ke dan di dalam tim pengembangan adalah percakapan tatap muka

5. Bangun proyek di sekitar individu yang termotivasi. Beri mereka lingkungan dan dukungan yang mereka butuhkan, dan percayai mereka untuk menyelesaikan pekerjaan

4. Pebisnis dan pengembang harus bekerja sama setiap hari selama proyek berlangsung

3. Kirimkan PL yang berfungsi sesering mungkin, dari beberapa minggu hingga beberapa bulan, dengan preferensi pada skala waktu yang lebih pendek

2. Menyambut perubahan kebutuhan, walau terlambat dalam pengembangan. Proses agile memanfaatkan perubahan untuk keunggulan kompetitif pelanggan

1. Prioritas tertinggi adalah untuk memuaskan pelanggan melalui pengiriman PL yang berharga secara dini dan berkelanjutan

7. Perangkat lunak yang berfungsi adalah ukuran utama kemajuan

8, Proses agile mempromosikan pembangunan berkelanjutan. Sponsor, pengembang, dan pengguna harus dapat mempertahankan kecepatan konstan tanpa batas

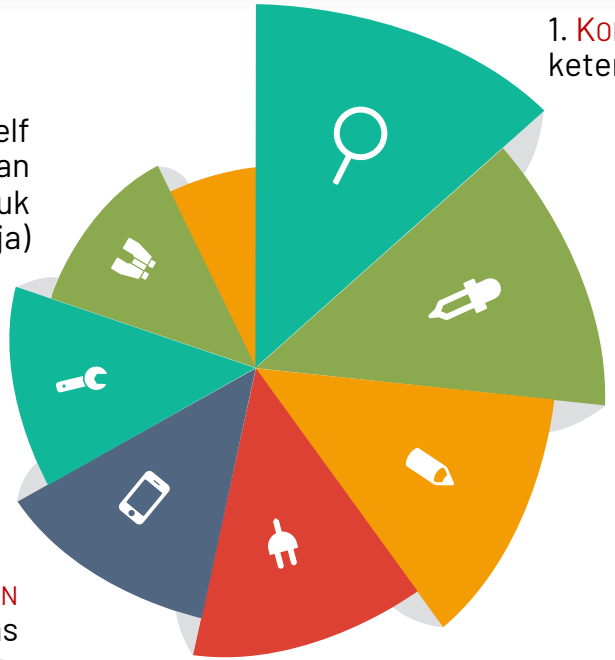
9. Perhatian terus-menerus pada keunggulan teknis dan desain yang baik meningkatkan agility

10. Kesederhanaan – seni memaksimalkan jumlah pekerjaan yang tidak dilakukan – sangat penting

11. Arsitektur, kebutuhan, dan desain terbaik muncul dari tim yang mengatur diri sendiri (self organizing)

12. Secara berkala, tim merefleksikan bagaimana menjadi lebih efektif, kemudian menyetel dan menyesuaikan perilakunya sesuai dengan itu

Human Factors



1. KOMPETENSI. (bakat, keterampilan, pengetahuan)!

2. FOKUS BERSAMA. (memberikan peningkatan PL yang berfungsi)

3. KOLABORASI. (rekan dan stakeholder)

4. KEMAMPUAN MEMBUAT KEPUTUSAN. (kebebasan untuk mengendalikan nasibnya sendiri)

5. KEMAMPUAN PEMECAHAN PROBLEM YANG FUZZY. (Ambiguitas dan perubahan konstan, problem hari ini mungkin bukan problem besok)!

6. SALING PERCAYA DAN MENGHORMATI

7. MENGATUR DIRI SENDIRI (self organization((diri untuk pekerjaan yang dilakukan, proses untuk lingkungan lokalnya, jadwal kerja)

3.3.2 The Politics of Agile Development

- **Pro Agility**: "Metodologi tradisional adalah sekelompok orang yang tidak tahu apa-apa yang lebih suka menghasilkan dokumentasi tanpa cacat daripada sistem kerja yang memenuhi kebutuhan bisnis"
- **Pro Konvensional**: "Ahli metodologi yang ringan, eh, 'gesit' adalah sekelompok peretas yang dimuliakan yang akan mendapat kejutan besar ketika mereka mencoba meningkatkan mainan mereka ke dalam perangkat lunak tingkat perusahaan."

Tidak ada yang menentang agility. Pertanyaan agility yang sebenarnya adalah:

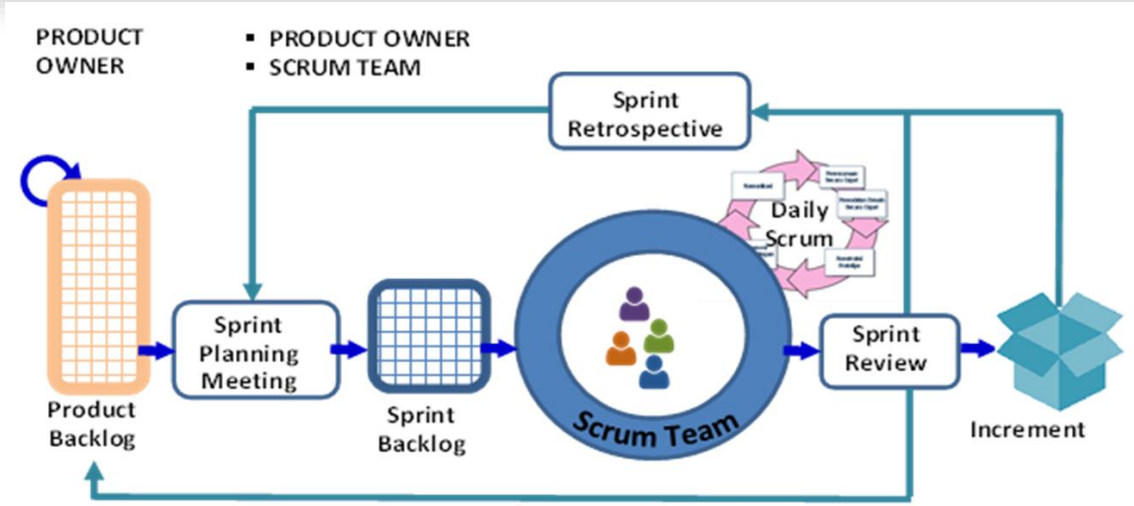
- Apa cara terbaik untuk mencapainya?
- Ingatlah bahwa PL yang berfungsi itu penting, tetapi jangan lupa bahwa itu juga harus menunjukkan berbagai atribut kualitas termasuk keandalan, kegunaan, dan kemudahan perawatan.
- Bagaimana Anda membangun PL yang memenuhi kebutuhan pelanggan saat ini dan menunjukkan karakteristik kualitas yang memungkinkannya diperluas dan ditingkatkan untuk memenuhi kebutuhan pelanggan dalam jangka panjang?

3.4 Scrum

3.4 SCRUM

- Scrum merupakan metode Agile untuk manajemen proyek yang dikembangkan oleh Ken Schwaber dengan tujuan untuk secara dramatis meningkatkan produktivitas dari tim.
- Scrum menerapkan metode empirisme ilmiah.
- Scrum menggantikan pendekatan algoritmik terprogram secara heuristik, yang memperhatikan orang dan organisasi untuk mengatasi ketidakpastian dan memecahkan problem yang kompleks.
- Gambar pada slide 15 mewakili *Scrum in Action* seperti yang dijelaskan oleh Schwaber & Sutherland (2012) dalam buku *Software in 30 Days*.
- Pada Scrum terdapat lima pertemuan untuk membahas *backlog refinement meeting*, *sprint planning meeting*, *daily scrum meeting* (15 menit *standup*), *sprint review meeting*, dan *sprint retrospective meeting*.
- Gambar pada slide 16 menjelaskan alur proses Scrum yang dielaborasi Pressman dan Maxim (2020)

Framework Scrum Schwaber & Beedle, 2012



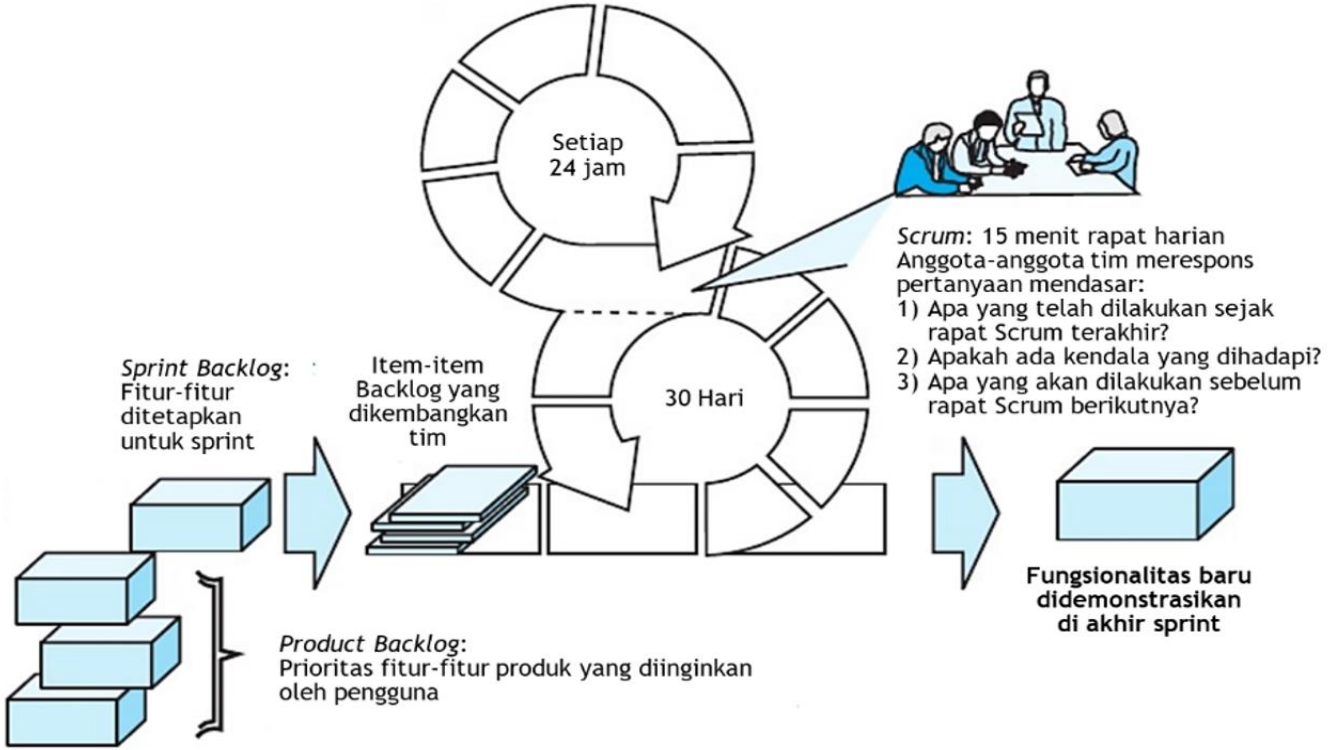
Product Owner membuat *product backlog* (daftar tugas yang perlu diprioritaskan dalam sebuah proyek). *Product owner* adalah klien, yaitu orang yang memesan perangkat lunak.

Product Backlog adalah daftar tugas yang diprioritaskan dalam sebuah proyek

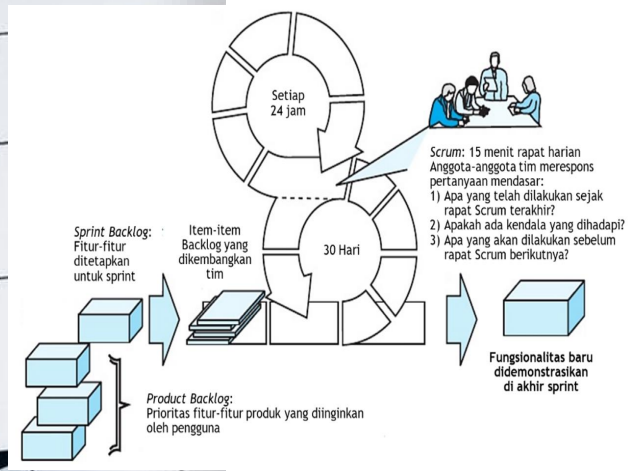
Scrum Master ialah menjadi fasilitator dan menjembatani komunikasi dan membuat pelaporan, namun tidak menentukan jadwal atau tugas.

Scrum Team adalah grup kecil para pengembang, biasanya berjumlah antara 5-9 orang yang saling bekerja sama untuk menghasilkan PL

Framework Scrum Pressman & Maxim 2020



3.4.1 Scrum Teams and Artifacts



- Scrum Team: Terdiri atas Product Owner, Scrum Master, 3-6 orang Development Team
- Scrum Artifact: Product Backlog, Sprint Backlog, dan Code Increment
- Pengembangan dilanjutkan dengan memecah proyek menjadi serangkaian periode pengembangan prototipe incremental selama 2 hingga 4 minggu yang disebut Sprint
- Product Backlog adalah daftar prioritas kebutuhans produk atau fitur yang memberikan nilai bisnis bagi pelanggan. Item dapat ditambahkan ke backlog kapan saja dengan persetujuan Product Owner dan persetujuan tim pengembang
- Sprint Backlog adalah subset dari item product backlog yang dipilih oleh tim produk untuk diselesaikan sebagai code increment pada sprint aktif (current sprint active)
- Code Increment adalah union dari semua item backlog produk yang diselesaikan di sprint sebelumnya dan semua item backlog yang harus diselesaikan di sprint saat ini
- Scrum Master: adalah fasilitator untuk seluruh anggota Scrum Team
  - Menjalankan Daily Scrum Meeting dan bertanggung jawab untuk menghilangkan hambatan yang diidentifikasi oleh anggota tim selama rapat
  - Melatih anggota tim pengembangan untuk saling membantu menyelesaikan Sprint Task ketika mereka memiliki waktu yang tertentu
  - Membantu Product Owner menemukan teknik untuk mengelola item product backlog dan membantu memastikan bahwa item backlog dinyatakan dalam istilah yang jelas dan ringkas

3.4.2 Sprint Planning Meeting

Prior to Beginning

Setiap tim pengembang bekerjasama dengan Product Owner dan seluruh stakeholder lainnya untuk mengembangkan item-item pada product backlog

Product Backlog Diranking

- Product backlog diranking berdasarkan kepentingan kebutuhan bisnis pemilik dan kompleksitas tugas RPL (pemrograman dan pengujian) yang diperlukan untuk menyelesaikannya
- Terkadang dihasilkan identifikasi fitur yang hilang padahal diperlukan untuk memberikan fungsionalitas yang diperlukan kepada pengguna akhir

Sebelum Memulai Sprint

- Product Owner menyatakan tujuan pengembangan increment yang akan diselesaikan di Sprint mendatang
- Scrum Master dan tim pengembangan memilih item yang akan dipindahkan ke sprint backlog
- Tim pengembangan menentukan apa yang dapat disampaikan pada increment dalam batasan waktu yang dialokasikan untuk sprint dan, dengan Scrum Master, pekerjaan apa yang diperlukan untuk mengirimkan increment
- Tim pengembangan memutuskan peran mana yang dibutuhkan dan bagaimana peran tersebut perlu diisi

3.4.3 Daily Scrum Meeting

- Daily Scrum Meeting ialah event 15-menit yang dijadwalkan setiap awal hari kerja agar anggota tim melakukan sinkronisasi aktivitas dan membuat rencana untuk 24 jam
- Daily Scrum Meeting dihadiri oleh Scrum Master dan terkadang Product Owner
- Scrum master memimpin rapat dan menilai tanggapan dari setiap orang

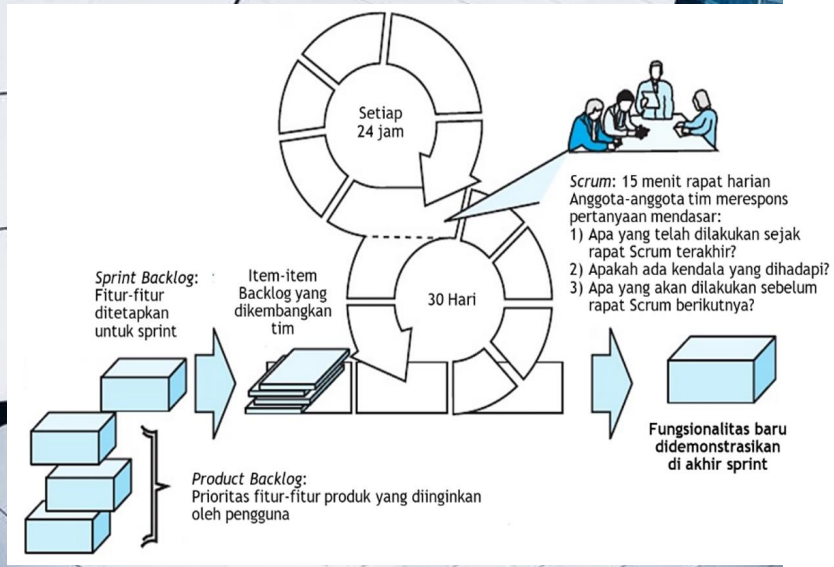
Tiga pertanyaan kunci diajukan dan dijawab oleh semua anggota tim:

1) Apa yang Anda lakukan sejak pertemuan tim terakhir?
2) Apa kendala yang Anda hadapi?
3) Apa yang Anda rencanakan untuk dicapai pada pertemuan tim berikutnya?

- Daily Scrum Meeting membantu tim untuk mengungkap potensi problem sedini mungkin
- Daily Scrum Meeting bukan pertemuan pemecahan problem (untuk pemecahan problem dilakukan secara off-line dan hanya melibatkan pihak-pihak yang terkena dampak)
- Daily Scrum Meeting mengarah pada "sosialisasi pengetahuan" , jadi mempromosikan struktur tim yang mengatur diri sendiri

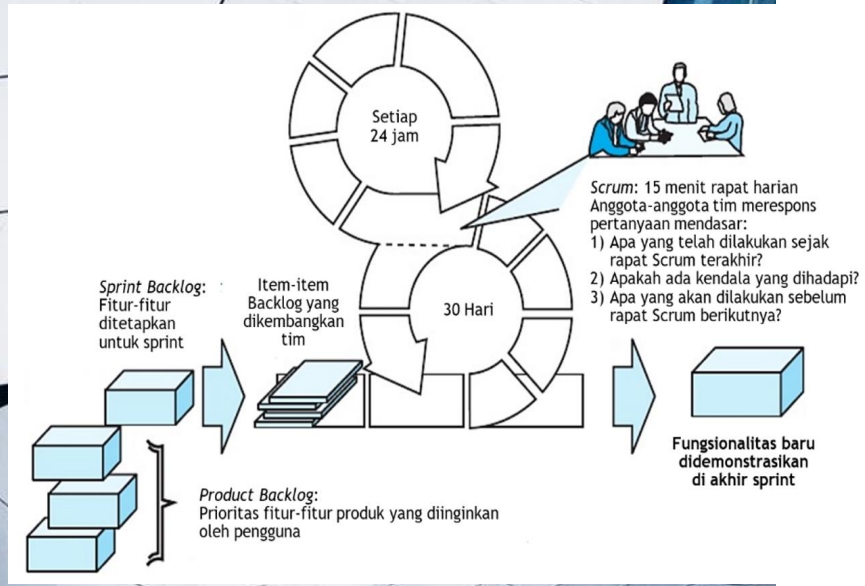
- Beberapa tim menggunakan pertemuan ini untuk menyatakan item sprint backlog Complete atau Done
- Ketika tim menganggap semua item sprint backlog Complete, tim dapat memutuskan untuk menjadwalkan demo dan meninjau increment yang diselesaikan dengan Product Owner

3.4.4 Sprint Review Meeting



- Terjadi pada akhir sprint ketika tim pengembang telah menilai increment selesai.
- Seringkali dibatasi sebagai 4 jam meeting untuk sprint 4 minggu.
- Scrum Master, tim pengembangan, Product Owner, dan Stakeholder terpilih menghadiri tinjauan ini.
- Aktivitas utama adalah demo increment PL yang diselesaikan dari suatu sprint.

3.4.5 Sprint Retrospective

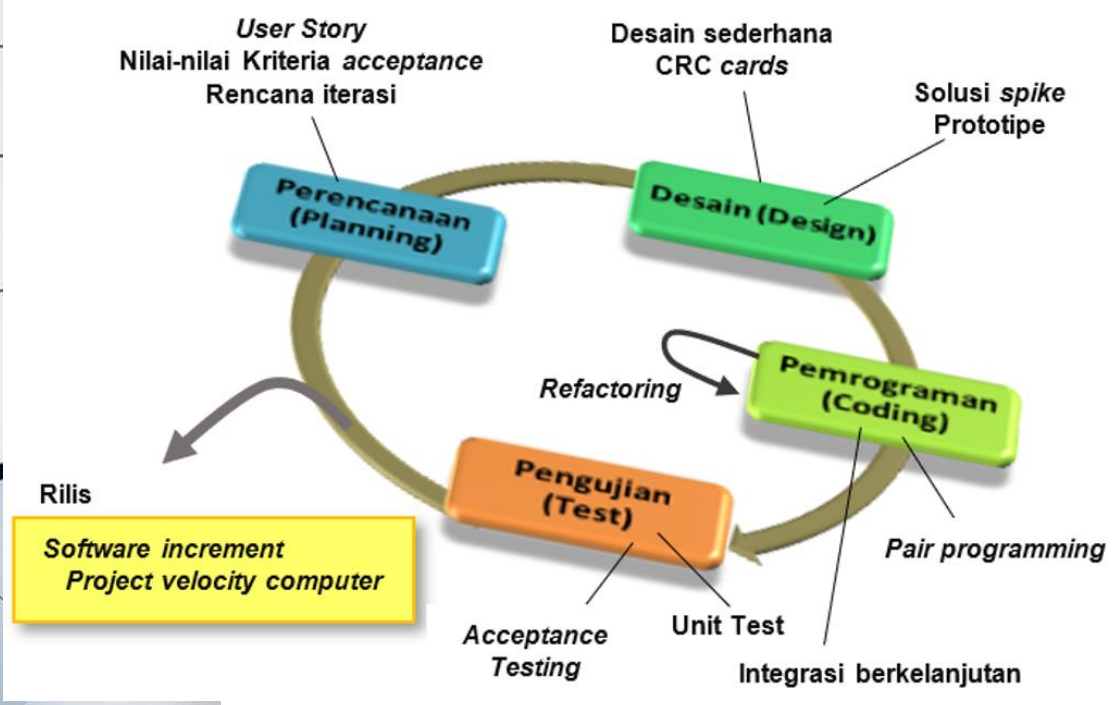


- Scrum Master memimpin 3 jam meeting (utk 4-minggu sprint) dengan Tim Pengembang:
  - Apa yang berjalan baik di sprint?
  - Apa yang bisa ditingkatkan?
  - Apa yang akan dilakukan tim untuk ditingkatkan di sprint berikutnya
- Tim merencanakan cara untuk meningkatkan kualitas produk dengan mengadaptasi definisi "selesai".

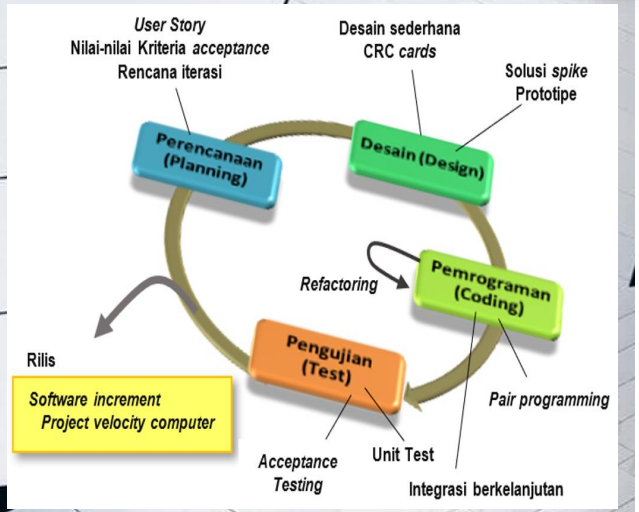
3.5 Framework Agile Lainnya

3.5.1 Framework XP
3.5.2 Kanban
3.5.3 DevOps

3.5.1 Extreme Programming (XP) Framework



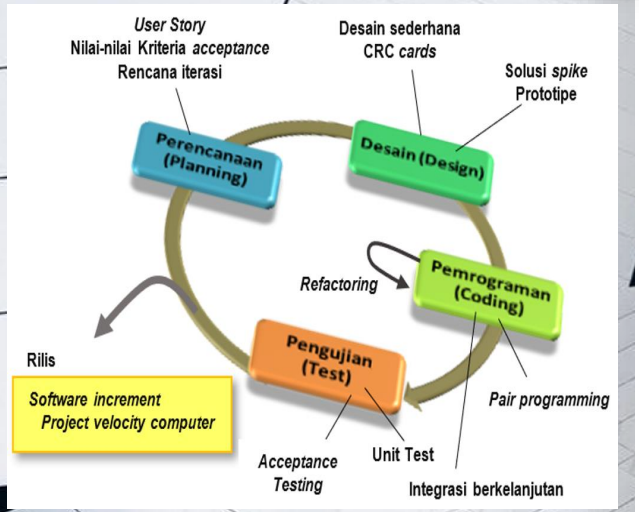
- *Extreme Programming* (XP) mengutamakan kepuasan konsumen dan kerja tim dengan komunikasi, kesederhanaan, dan keberanian sebagai nilai-nilai utama
- Pengembang berkomunikasi dengan pengguna dan programmer
- Desain dijaga tetap sederhana dan bebas kesalahan
- Testing yang dilakukan di awal seringkali menyediakan umpan balik sehingga pengembang berani mengubah kebutuhan dan teknologi
- Tim proyek dijaga berukuran kecil, sekitar 10 orang



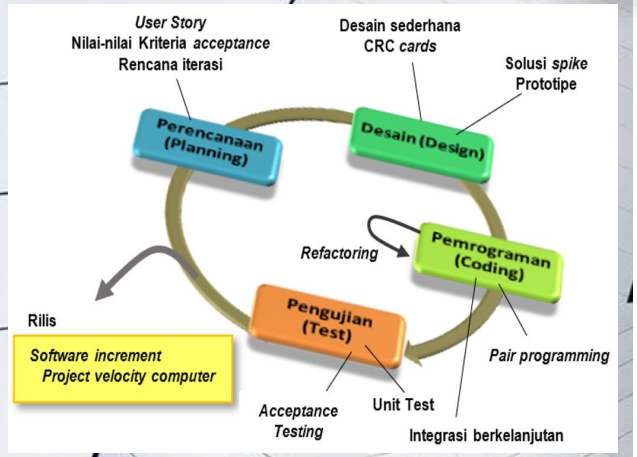
Perencanaan

- Disebut juga Planning Game, diawali aktivitas kebutuhan listening menghasilkan user stories
- User Stories: output yang diperlukan, fitur, dan fungsionalitas.
- Setiap user story dibuat pada Index Card, diberi value (prioritas setiap user story) serta cost (ukuran minggu pengembangan, bila > 3 minggu user story di-*split* (dipecah menjadi beberapa user story)
- Komitmen: user story yang dicakup, delivery date, info lain terkait proyek) dalam 3 cara:
  i. Seluruh user story diimplementasi dalam beberapa minggu,
  ii. Prioritas tinggi didahulukan,
  iii. User story paling berisiko diimplementasikan pertama kali
- Customer dan developer menentukan jumlah Release (rilis), yaitu increment PL berikutnya yang disertai dengan penghitungan Project Velocity

Desain



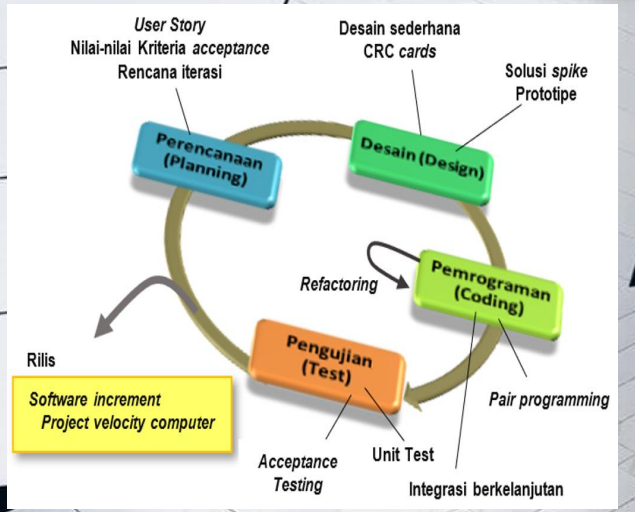
- Mengikuti prinsip KIS, yaitu *Keep It Simple* (tetap sederhana).
- Menyediakan panduan untuk implementasi sesuai yang tertulis pada *story*.
  - Desain dengan penambahan fungsionalitas ekstra (karena pengembang berasumsi nantinya akan diperlukan) tidak disarankan untuk dilakukan.
- XP menganjurkan penggunaan kartu *class responsibility collaborator* (CRC) sebagai mekanisme efektif terkait PL dalam konteks berbasis obyek.
- Kartu CRC mengidentifikasi dan menata class berbasis obyek yang relevan dengan *software increment*. Kartu CRC ialah satu-satunya produk desain yang dihasilkan sebagai bagian dari proses XP.
- Jika kesulitan mendesain suatu *story*, maka langsung saja dibuat prototipe operasionalnya, atau disebut sebagai *Spike Solution*, yaitu prototipe desain yang langsung diimplementasikan dan dievaluasi.
  - Tujuan: untuk memperkecil resiko saat implementasi dan untuk memvalidasi estimasi orisinil dari *story* yang sulit didesain.



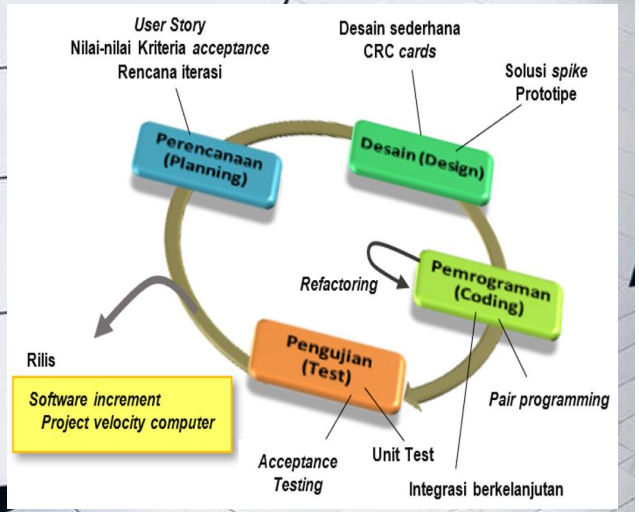
Koding

- Konsep kunci dari aktivitas koding ialah *pair programming*, yaitu dua orang bekerja sama pada satu komputer untuk membuat koding dari suatu *story*.
- Hal ini dilakukan agar terjadi mekanisme pemecahan problem dan jaminan mutu secara *real time* (saat koding dibuat langsung direviu) dan agar pengembang fokus pada permasalahannya.
  - Konsep pemrograman parallel - dua programer bekerja untuk tugas yang sama pada komputer yang sama; salah seorang programer membuat program sementara programer satunya lagi melakukan navigasi/menjadi pengamat.
  - Programer pengamat memeriksa koding secara strategis dalam garis besar PL sementara programer yang satunya lagi (*the driver*) lebih menangani tugas individual.
  - Pada dasarnya setiap orang memiliki peran berbeda. Misalnya satu orang memikirkan detail koding dari suatu desain sementara satu orang lainnya memastikan standar-standar koding dipenuhi.
- Hasil koding *pair programmer* diintegrasikan dengan hasil koding dari *pair programmer* lainnya.
  - Integrasi dilakukan secara harian oleh tim integrase atau *pair programmer* yang bertanggung jawab melakukan integrasi.
  - Strategi integrasi secara kontinyu dilakukan agar terhindar dari problem kesesuaian (*compatibility*) dan antar muka (*interfacing*) dan menyediakan lingkungan testing untuk menemukan error di awal

Refactoring



- *Refactoring* ialah teknik konstruksi yang merupakan metode untuk optimisasi desain.
  - *Refactoring* adalah proses mengubah sistem PL yaitu tidak mengubah perilaku eksternal namun memperbaiki struktur internalnya sedemikian rupa.
  - *Refactoring* adalah suatu disiplin untuk merapikan koding (dan modifikasi/ menyederhanakan desain internal) yang meminimisasi peluang adanya kesalahan (*bug*).
- Pada dasarnya *refactoring* sebenarnya memperbaiki desain dari koding setelah koding tersebut dibuat.
- Selain kartu CRC dan *spike solution*, desain XP secara virtual tidak menggunakan notasi dan jumlah desainnya sedikit. Oleh karenanya desain XP sering dianggap sebagai artefak sementara (*transient artifact*) sehingga saat konstruksi berlangsung harus secara kontinyu dimodifikasi.
- Tujuan dari *refactoring* ialah mengendalikan modifikasi dengan cara menganjurkan perubahan kecil pada desain yang pada akhirnya secara radikal akan memperbaiki desain itu sendiri. Namun penting diperhatikan bahwa upaya yang dibutuhkan untuk *refactoring* akan bertambah secara dramatis sesuai penambahan ukuran dari aplikasi.
- Pada XP, desain berlangsung baik sebelum maupun sesudah koding dilakukan. *Refactoring* artinya ialah bahwa desain terjadi secara kontinyu selama sistem dikonstruksikan. Dalam kenyataannya, aktivitas konstruksi akan memberikan panduan bagi tim XP untuk memperbaiki desain.



Pengujian

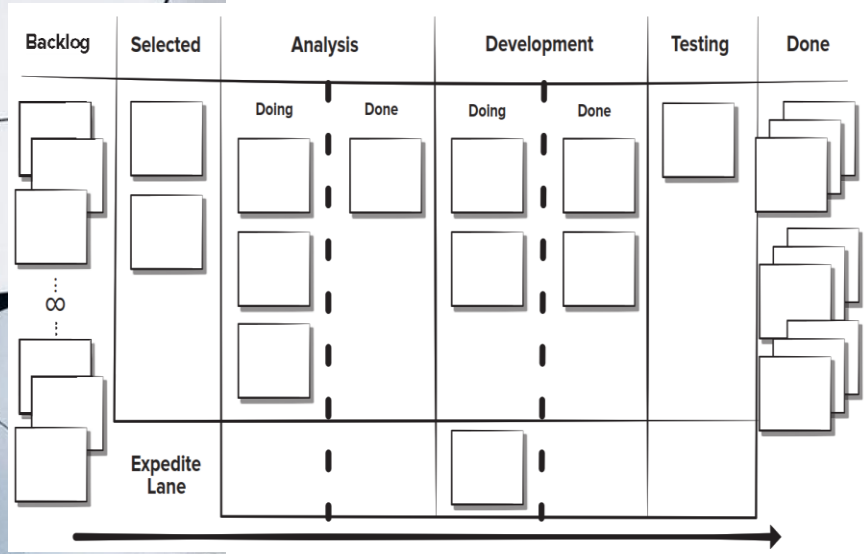
- Pembuatan unit test sebelum koding dilakukan merupakan elemen kunci dari pendekatan XP.
- *Unit test* yang dibuat harus diimplementasikan dengan *framework* yang memungkinkan otomasi (sehingga bisa dieksekusi dengan mudah dan berulang kali).
- Setelah setiap individual *unit test* ditata ke dalam "*universal testing suite*", maka testing integrasi dan validasi dilakukan secara harian. Dengan demikian, tim XP bisa mengetahui indikasi kontinyu terkait kemajuan pengembangan dan bisa segera mengetahui apabila ada problem yang terjadi.
- *Acceptance test* pada XP, disebut juga *customer test*, dilakukan oleh konsumen dengan fokus pada keseluruhan fitur dan fungsionalitas sistem yang *visible* dan bisa direviu oleh konsumen.
- *Acceptance test* dikembangkan dari berbagai *user story* yang telah diimplementasikan sebagai bagian dari rilis perangkat lunak.

Rilis

- Setelah rilis pertama (disebut juga *software increment*) diserahkan, maka tim XP menghitung *project velocity*, yaitu jumlah *story* yang selesai diimplementasikan pada rilis yang pertama.
- *Project Velocity* digunakan untuk:
  i. Estimasi tanggal penyerahan dan jadwal rilis berikutnya dan
  ii. Menentukan apakah terjadi janji yang berlebih-lebihan (*overcommitment*) untuk seluruh *story* pada keseluruhan proyek pengembangan.

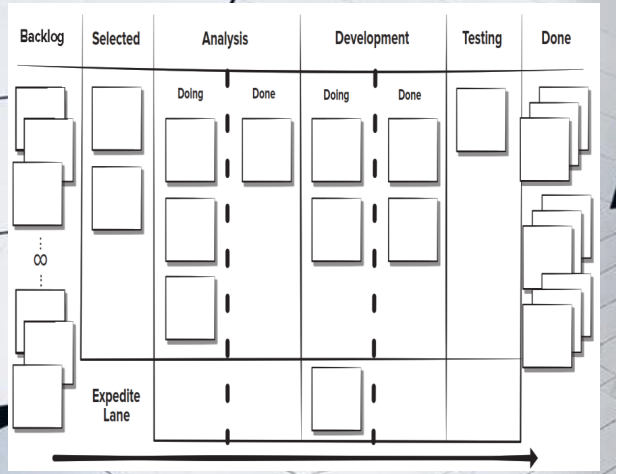
3.5.2 Metode Kanban

Kanban Board



- Merupakan *lean methodology*, yaitu metode untuk meningkatkan setiap proses atau workflow
- Berfokus pada *change management* dan *service delivery*
- *Change management* mendefinisikan proses di mana perubahan yang diminta diintegrasikan ke dalam sistem berbasis perangkat lunak.
- *Service Delivery* didorong dengan berfokus pada pemahaman kebutuhan dan harapan pelanggan
- Anggota tim mengelola pekerjaan dan diberi kebebasan untuk mengatur diri sendiri untuk menyelesaikannya.
- Kebijakan akan berkembang sesuai kebutuhan untuk meningkatkan hasil
- Kanban berasal dari Toyota yang merupakan praktik-praktik rekayasan industrial yang diadaptasikan ke pengembangan PL

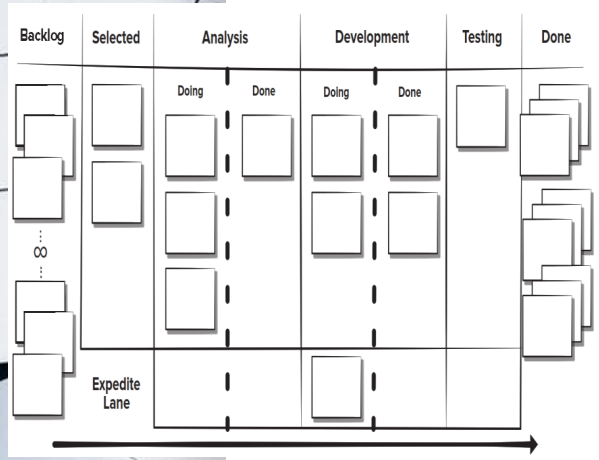
Kanban Board



- *Team meetings* untuk Kanban mirip seperti yang ada dalam framework Scrum. Jika Kanban digunakan untuk proyek yang sudah ada, maka tidak semua item akan masuk di kolom backlog.
- Pengembang perlu menempatkan kartu (cards) di kolom proses tim dengan bertanya pada diri sendiri:
  - Di mana mereka sekarang? Dari mana mereka berasal? Kemana mereka pergi?
- Dasar dari Kanban Standup Meeting ialah task yang disebut 'Walking The Board'
  - Pemimpin meeting dirotasi secara harian
  - Anggota tim mengidentifikasi item yang hilang dari Kanban Board yang sedang dikerjakan dan menambahkannya ke Kanban Board
  - Setiap item dikerjakan sampai 'Done'
  - Tujuannya adalah untuk mengerjakan item bernilai bisnis tinggi terlebih dahulu
  - Tim melihat alur dan mencoba mengidentifikasi hambatan pada penyelesaian dengan melihat beban kerja dan risiko
- Saat Restrospective Meeting Mingguan dilakukan process measurement.
  - Tim mempertimbangkan di mana perbaikan proses mungkin diperlukan dan mengusulkan perubahan untuk diterapkan.
  - Kanban dapat dengan mudah digabungkan dengan praktik pengembangan agile lainnya untuk memperkaya disiplin proses

Six Core Practice Metode Kanban

Kanban Board



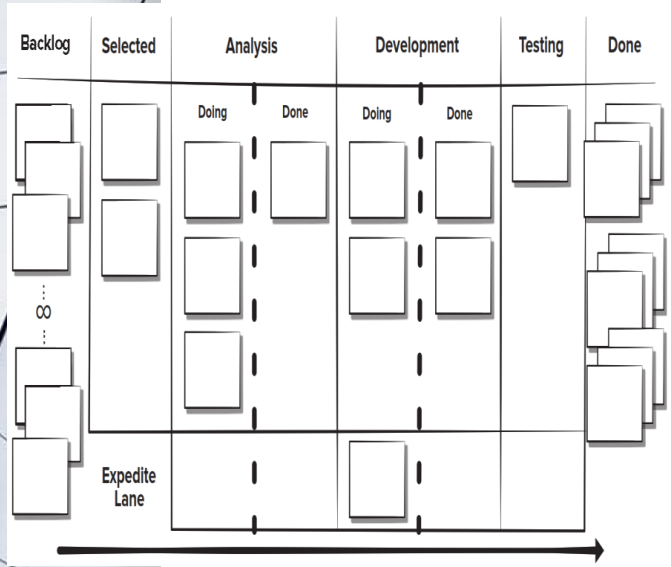
1) Visualisi Workflow menggunakan Kanban Board.
   Kanban Board terdiri atas kolom-kolom yang merepresentasikan tahap pengembangan dari setiap elemen fungsionalitas PL. Kartu-kartu (Cards) pada Kanban Board dapat berisi 1 user story atau defect yang ditemukan. Tim bergerak dari 'To Do' ke 'Doing', dan 'Done'

2) Jumlah *Work in Progress* (WIP) pada suatu waktu tertentu dibatasi
   Pengembang didorong untuk menyelesaikan suatu task sebelum bergerak ke task berikutnya. Hal ini untuk mengurangi lead time, meningkatkan kualitas kerja, dan meningkatkan kemampuan tim untuk lebih sering menghasilkan fungsionalitas PL kepada stakeholder

3) Pengelolaan Workflow
   Mengelola workflow untuk mengurangi pemborosan dengan memahami nilai workflow saat ini, menganalisis tempat yang terhenti, menentukan perubahan, dan kemudian menerapkan perubahan

Six Core Practice Metode Kanban

Kanban Board

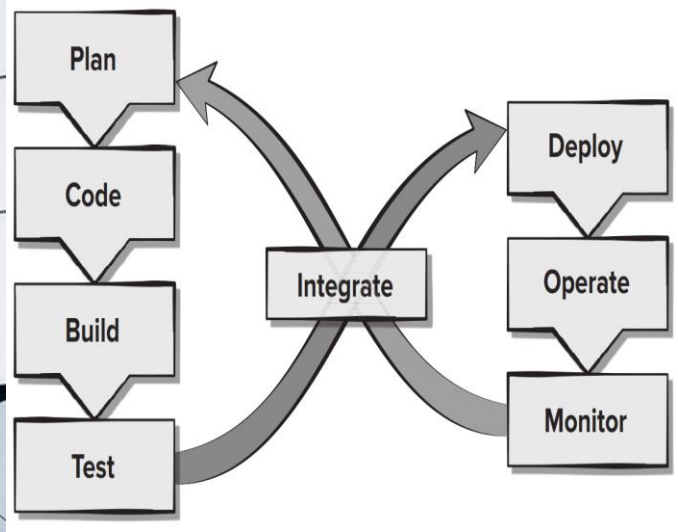


4) Membuat kebijakan proses yang eksplisit
   Misal, menuliskan alasan mendasar mengapa suatu item dipilih untuk dikerjakan dan kriteria yang digunakan untuk definisi 'Done'

5) Berfokus pada peningkatan terus menerus (continuous improvement)
   dengan membuat loop umpan balik di mana perubahan diperkenalkan berdasarkan data proses dan efek dari perubahan pada proses diukur setelah perubahan dibuat.

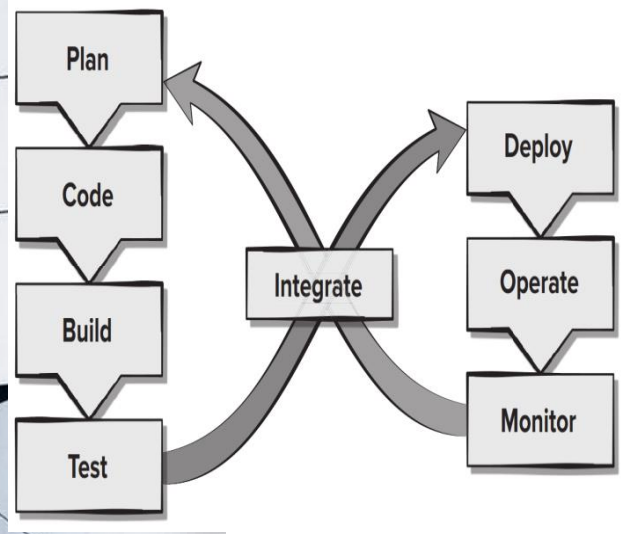
6) Melakukan perubahan proses secara kolaboratif dan melibatkan semua anggota tim dan stakeholder lainnya sesuai kebutuhan

3.5.3 DevOps

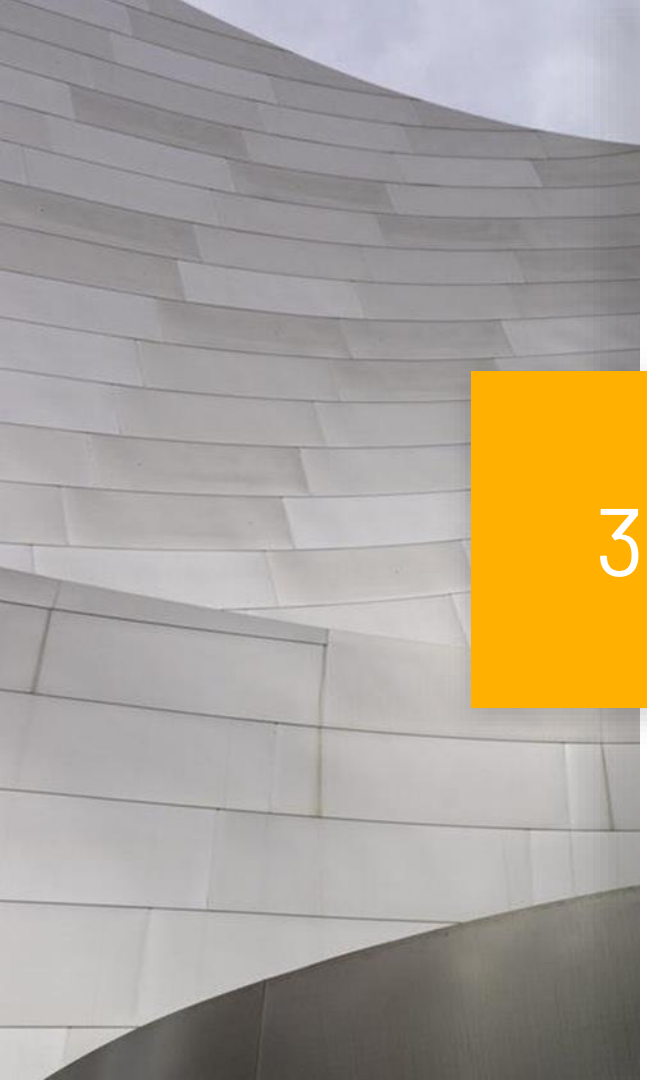


- Diciptakan oleh Patrick DeBois untuk mengkombinasikan Development dan Operations
- Menerapkan prinsip-prinsip agile dan lean development di seluruh software supply chain
- Workflow DevOps (gambar di kiri) melibatkan 5 tahapan dalam loop terus menerus sampai dihasilkan produk yang diinginkan
- DevOps meningkatkan pengalaman pelanggan dengan bereaksi cepat terhadap perubahan kebutuhan atau keinginan mereka
- Hal ini dapat meningkatkan loyalitas merek dan meningkatkan pangsa pasar.
- Pendekatan ramping (*lean*) seperti DevOps dapat memberi organisasi peningkatan kapasitas untuk berinovasi dengan mengurangi pengerjaan ulang dan memungkinkan peralihan ke aktivitas nilai bisnis yang lebih tinggi
- Produk tidak menghasilkan uang sebelum konsumen memiliki akses ke produk tersebut, dan DevOps dapat memberikan waktu deployment yang lebih cepat ke platform produksi

LimaTahapan DevOps



1) **Continuous Development**. Deliverable dari PL dibagi dan dikembangkan menjadi beberapa sprint dengan increment yang dikirimkan ke anggota jaminan kualitas tim pengembangan untuk pengujian
2) **Continuous Testing**. Tools automasi testing digunakan untuk membantu anggota tim menguji beberapa increment koding secara bersamaan untuk memastikan bebas dari cacat sebelum integrase dilakukan
3) **Continuous Integration**. Potongan kode dengan fungsionalitas baru ditambahkan ke kode yang ada dan ke lingkungan run-time dan kemudian diperiksa untuk memastikan tidak ada kesalahan setelah penerapan
4) **Continuous Deployment**. Kode terintegrasi disebarkan (diinstall) ke lingkungan produksi, yang mungkin mencakup beberapa situs secara global yang perlu disiapkan untuk menerima fungsionalitas baru
5) **Continuous Monitoring**. Staf operasi yang merupakan anggota tim pengembangan membantu meningkatkan kualitas PL dengan memantau kinerjanya di lingkungan produksi dan secara proaktif mencari kemungkinan problem sebelum pengguna menemukannya



3.6 Rangkuman

Model Proses	PROS	CONS
SRUM	❖ Product Owner menetapkan prioritas ❖ Tim memiliki pengambilan keputusan ❖ Dokumentasinya ringan ❖ Mendukung pembaruan yang sering	❑ Sulit untuk mengendalikan biaya perubahan ❑ Mungkin tidak cocok untuk tim besar ❑ Membutuhkan anggota tim yang ahli
EXTREME PROGRAMMING (XP)	❖ Menekankan keterlibatan pelanggan ❖ Menetapkan rencana dan jadwal yang rasional ❖ Ada komitmen pengembang yang tinggi untuk proyek tersebut ❖ Kemungkinan penolakan produk berkurang	❑ Ada godaan untuk "mengirimkan" prototype ❑ Membutuhkan pertemuan yang sering tentang biaya yang meningkat ❑ Memungkinkan untuk perubahan yang berlebihan ❑ Ada ketergantungan pada anggota tim yang sangat terampil

Model Proses	PROS	CONS
KANBAN	❖ Memiliki kebutuhan anggaran dan waktu yang lebih rendah ❖ memungkinkan pengiriman produk lebih awal ❖ Kebijakan proses dituliskan ❖ Ada perbaikan proses yang berkelanjutan	❑ Keterampilan kolaborasi tim menentukan kesuksesan ❑ Analisis bisnis yang buruk dapat merusak proyek ❑ Fleksibilitas dapat menyebabkan pengembang kehilangan focus ❑ Keengganan pengembang untuk menggunakan pengukuran
DEVOPS	❑ Ada pengurangan waktu untuk deployment koding ❑ Tim terdiri atas pengembang dan staf operasi ❑ Tim memiliki kepemilikan proyek ujung ke ujung (end-to-end) ❑ Ada pemantauan proaktif dari produk yang dilakukan deploy	❑ Ada tekanan untuk mengerjakan koding lama dan baru ❑ Ada ketergantungan besar pada tools automasi agar efektif ❑ Deployment dapat mempengaruhi lingkungan produksi ❑ Membutuhkan tim pengembang yang ahli

Rangkuman

Model Proses Agile

Dalam ekonomi modern, kondisi pasar berubah dengan cepat, kebutuhan pelanggan dan pengguna akhir berkembang, dan ancaman persaingan baru muncul tanpa peringatan.

Praktisi harus mendekati RPL dengan cara yang memungkinkan mereka untuk tetap agile — untuk menentukan proses *lean* yang dapat bermanuver, adaptif, yang dapat mengakomodasi kebutuhan bisnis modern

Filosofi agile untuk RPL menekankan empat problem utama:

i. Pentingnya tim yang mengatur diri sendiri yang memiliki kendali atas pekerjaan yang mereka lakukan,
ii. Komunikasi dan kolaborasi antara anggota tim dan antara praktisi dan pelanggan mereka,
iii. Pengakuan bahwa perubahan mewakili peluang, dan
iv. Penekanan pada pengiriman cepat PL yang memuaskan pelanggan

Kenyataannya adalah tidak ada metode agile yang sempurna untuk setiap proyek. Pengembang agile bekerja dalam tim mandiri dan diberdayakan untuk membuat model proses mereka sendiri

Rangkuman

SCRUM

- Scrum menekankan penggunaan seperangkat pola proses PL yang telah terbukti efektif untuk proyek dengan garis waktu yang ketat, kebutuhan yang berubah, dan kekritisan bisnis
- Tidak ada alasan mengapa tim Scrum tidak dapat mengadopsi penggunaan bagan Kanban untuk membantu mengatur rapat perencanaan hariannya

Extreme Programming

- Extreme programming (XP) diatur di sekitar empat framework aktivitas —perencanaan, desain, pengkodean, dan pengujian—XP menyarankan sejumlah teknik inovatif dan kuat yang memungkinkan tim agile membuat rilis PL yang sering menghadirkan fitur dan fungsionalitas yang telah dijelaskan dan kemudian diprioritaskan oleh stakeholder
- Tidak ada yang mencegah tim XP untuk menggunakan teknik DevOps untuk mengurangi waktu penerapannya

Pertemuan 3:

Agility & Process

Meuthia Rachmaniah
Departemen Ilmu Komputer, FMIPA IPB

Software Engineering: A Practitioner's Approach, Ninth Ed.
Roger S. Pressman dan Bruce R. Maxim