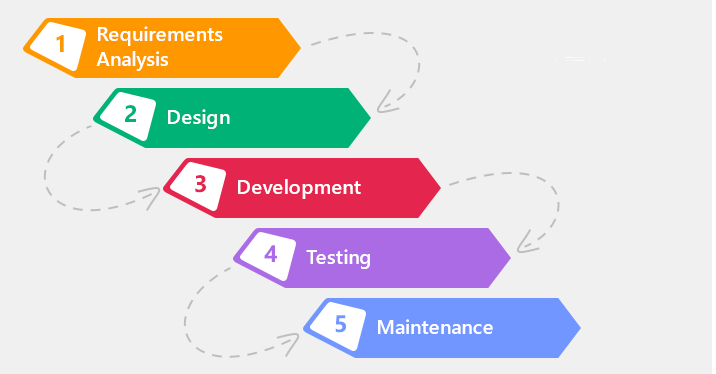
Saat ini ada beberapa model yang berkembang terkait dengan System Development Life Cycle (SDLC). Namun terdapat beberapa model yang populer dalam dunia pengembangan perangkat lunak.

1. **Waterfall**



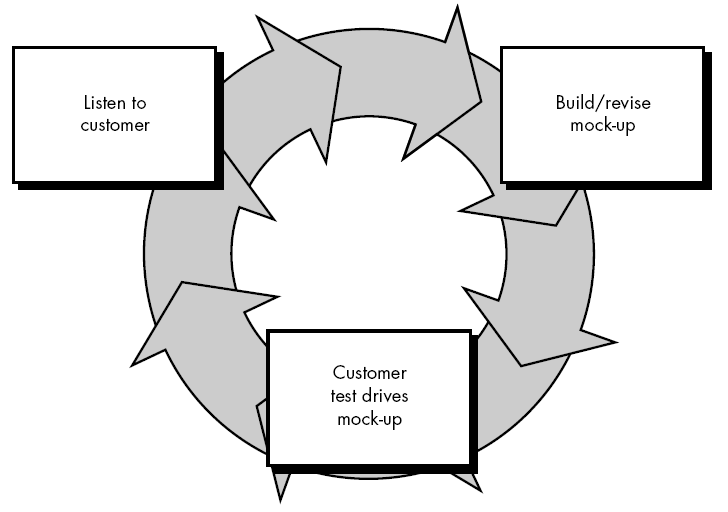
Model ini menggunakan pendekatan secara sistematis dan sekuensial yang mulai pada tingkat requirment sampai tingkat maintenance.

## Kelebihan

1. Proses menjadi teratur
2. Jadwal menjadi lebih menentu
3. Proses mudah dipahami dan jelas
4. Mudah dalam pengelolaan proyek
5. Kondisi requirment jelas.

## Kelemahan

1. **Sifatnya kaku**, karena tidak mudah tanggap dalam menanggapi segala perubahan.
2. Membutuhkan daftar kebutuhan yang lengkap diawal
3. Proses pengembangan relative lama, dikarenakan tahap selanjutnya bisa dilakukan jika tahap sebelumnya selesai dikerjakan.
4. **Prototype**



Model prototype merupakan model pengembangan software yang mengijinkan pengguna/user memiliki gambaran awal tentang program yang akan dikembangkan serta melakukan pengujian awal.

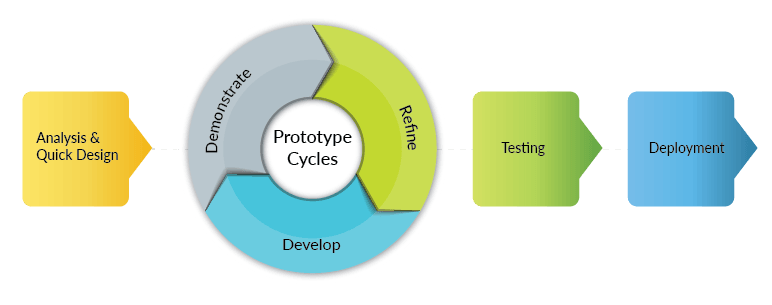
Model prototype juga memberi fasilitas bagi pengembangn dan user untuk saling terkait dan berinteraksi.

## Kelebihan

1. Fleksibel
2. Bersifat aktif, sehingga user dapat melihat, merasakan, dan mengalami proses pengembangan.
3. Perbaikan kesalahan relative cepat.

## Kelemahan

1. Mengurangi kreatifitas perancangan
2. Cakupan pengembangannya sistem dapat lebih luas
3. **Rapid Application Development**



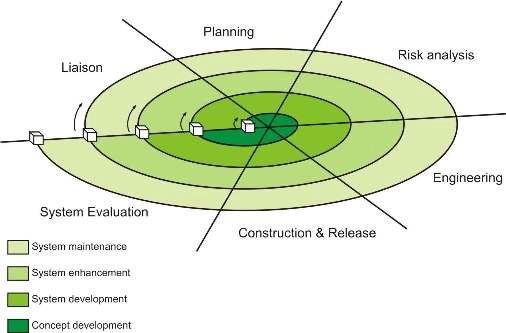
Model pengembangan yang menggunakan pendekatan orientasi komponen terhadap pengembangan perangkat lunak. Model ini bertujuan mempersingkat waktu yang biasanya diperlukan dalam siklus hidup pengembangan konvensional.

## Kelebihan

1. Pengembangan yang cepat
2. Adanya prototype
3. Pengurangan penulisan kode yang kompleks, dikarenakan reuse code yang sudah ada.

## Kelemahan

1. Tidak relevan untuk proyek skala besar
2. memerlukan komitmen yang kuat antara pengembang dengan konsumer
3. membutuhkan sumber daya yang besar untuk proyek skala besar.
4. **Spirall**



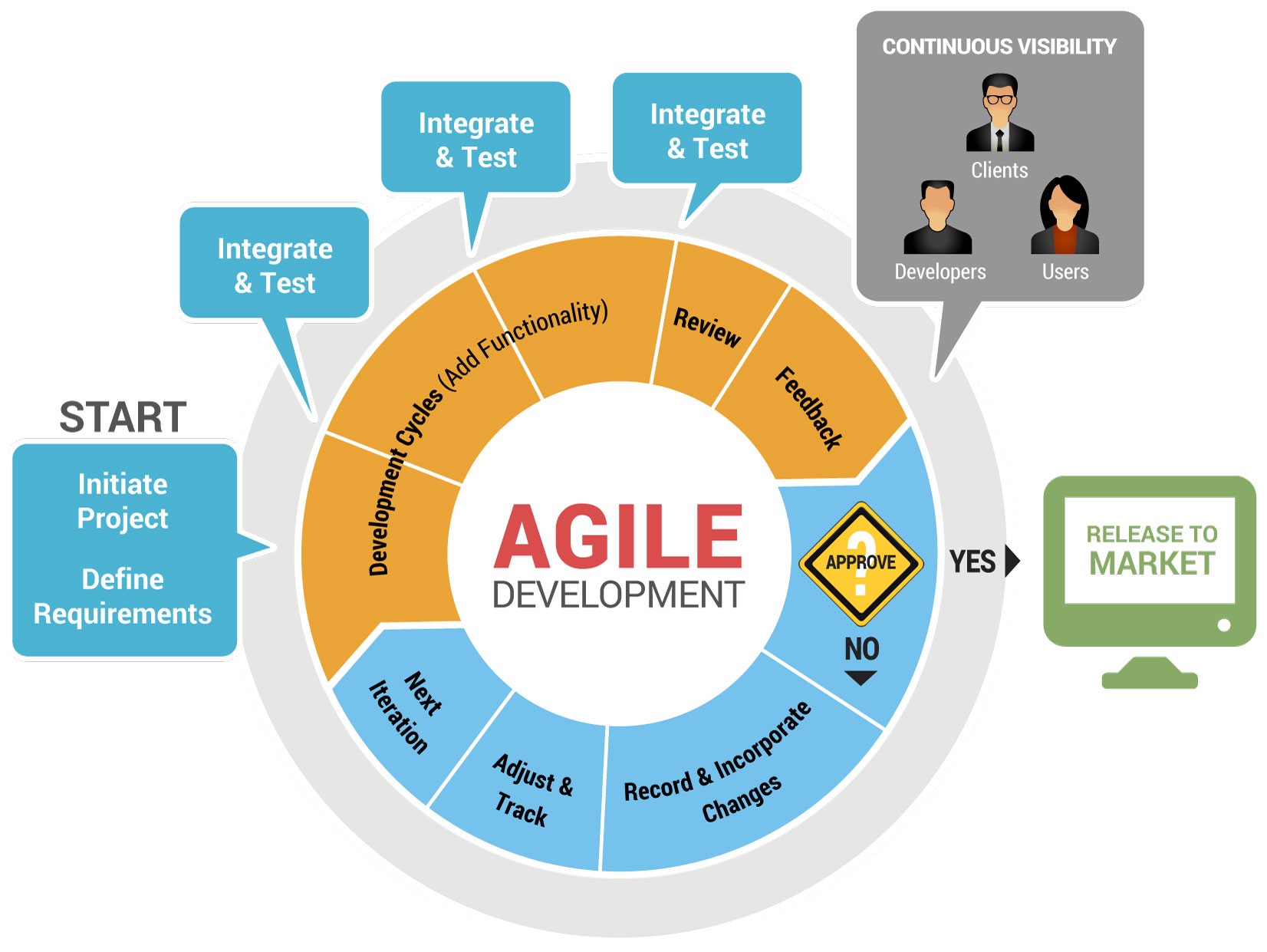
Model Spirall merupakan model pengembangan perangkat lunak yang evolusioner yang memadukan sifat iteratif model protype dan aspek sisematis dari mode sekuensial. **Version Release** meningkat setiap iterasi terjadi.

## Kelebihan

1. Cocok untuk proyek skala besar
2. Manajemen kesalahan baik
3. Menggunakan prototype sebagai mekanisme pengurangan resiko dan pada setiap keadaan didalam evolusi produk.

## Kelemahan

1. Waktu pengembangan cukup lama
2. Dibutuhkan ahli dalam penanganan resiko
3. Kesulitan dalam meyakinkan konsumen.
4. **Agile**



Model Agile merupakan model pengembangan jangka pendek yang memerlukan adaptasi cepat dan pengembangan terhadap perubahan dalam bentuk apapun. Dalam agile terdapat beberapa poin penting diantaranya sebagai berikut:

1. **Interaksi antar personal** lebih penting daripada **proses dan alat**.
2. **Software** yang berfungsi lebih penting daripada **dokumentasi** yang lengkap
3. **Kolaborasi** dengan klien lebih penting daripada **negoisasi kontrak**.
4. **Sikap tanggap** lebih penting daripada **mengikuti rencana/plan**.

## Kelebihan

1. Functional dapat dibuat dengan cepat dan dilakukan testing
2. Perubahan dengan cepat ditangani

## Kelemahan

1. Analisis, desain, dan pengembangan sulit diprediksi
2. Dapat memunculkan permasalahan dari arsitektur maupun desain.

**Contoh Studi Kasus WaterFall**

**Contoh studi kasus:**

Sulitnya petugas bagian administrasi dalam mengolah data perpustakaan yang mengakomodasi peminjaman, buku, pengembalian dan membuat laporan yang membutuhkan banyak waktu. Adapun tujuan dari model sistem ini adalah

memodelkan sebuah sistem informasi Perpustakaan yang berbasis komputer dengan menggunakan metode waterfall dan sistem informasi perpustakaan ini, untuk membantu petugas dalam menghadapi kendala yang dihadapi dalam melakukan transaksi, sehingga dengan adanya sistem informasi tersebut diharapkan dapat menyelesaikan permasalahan yang berhubungan dengan Perpustakaan.

**Contoh studi kasus:**

Seorang pelanggan mendefinisikan serangkaian sasaran umum bagi perangkat lunak, tetapi tidak melakukan mengidentifikasi kebutuhan output, pemrosesan, atupun input detail. Pada kasus yang lain, pengembang mungkin tidak memiliki kepastian terhadap efisiensi algoritme, kemampuan penyesuaian dari sebuah sistem operasi,atau bentuk-bentuk yang harus dilakukan oleh interaksi manusia dengan mesin. Dalam hal ini, serta pada banyak situasi yang lain, prototyping paradigma mungkin menawarkan pendekatan yang terbaik.  
Prototyping paradigma dimulai dengan pengumpulan kebutuhan. Pengembang dan pelanggan bertemu dan mendefinisikan obyektif keseluruhan dari software, mengidentifikasi segala kebutuhan yang diketahui, dan area garis besar diman definisi lebih jauh merupakan keharusan kemudian dilakukan “perancangan kilat”. Perancangan kilat berfokus pada penyajian dari aspek-aspek software tersebut yang akan nampak bagi pelanggan atau pemakai (contohnya pendekatan input dan format output). Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan software. Iterasi terjadi pada saat prototipe disetel untuk memenuhi kebutuhan pelanggan, dan pada saat yang sama memungkinkan pengembang untuk secara lebih baik memahami apa yang harus dilakukannya.

**Perbedaan SCRUM dan KANBAN**

Bagi sebagian orang yang telah memutuskan untuk menjalankan [metode kerja Agile](https://medium.com/codexstories/learning-agile-by-doing-agile-c4cc68a5a718?source=collection_detail----c44a95080695-----2---------------------" \t "_blank), hal pertama yang harus dilakukan adalah memilih kerangka kerja. Mengapa demikian? Sebab sebuah kerangka kerja akan memandu dan memudahkan kita untuk kerja dengan Agile.

Di artikel sebelumnya, saya menyebutkan beberapa pilihan [kerangka kerja](https://medium.com/codexstories/learning-agile-by-doing-agile-c4cc68a5a718" \t "_blank) yang berkaitan dengan metode Agile. Dan dalam praktiknya, saat ini Codex sedang **mempraktikkan Scrum dan Kanban**.

*Di salah satu event komunitas, saya dapat mengetahui bahwa keduanya merupakan kerangka kerja yang sangat populer di kalangan penggiat Agile.*

Namun tahukah kalian bahwa Scrum dan Kanban merupakan dua kerangka yang memiliki karakteristik berbeda? Akan tetapi mengapa di kebanyakan implementasi, keduanya selalu digabung?

*Pernah mendengar istilah ScrumBan (perpaduan antara Scrum dan Kanban)?*

## **Karakteristik Scrum**

1. Mendefinisikan peran-peran tertentu dalam sebuah tim, yakni Product Owner, Development Team, dan Scrum Master. Perbedaan signifikan dengan Kanban, adalah dengan **adanya peran Scrum Master.**  
   Mereka bertanggung jawab untuk memastikan seluruh anggota tim memahami dan melaksanakan teori, praktik, dan aturan main Scrum.
2. Adanya empat acara formal yang wajib diikuti oleh seluruh anggota tim biasa disebut dengan Sprint. Aktivitas ini bertujuan untuk proses inspeksi (memeriksa dan mengamati) serta adaptasi (penyesuaian terhadap pekerjaan), yang terdiri atas Sprint Planning; Daily Standup; Sprint Review; dan Sprint Retrospective.
3. Scrum terlihat lebih disiplin karena pekerjaan dibatasi oleh waktu. Setiap anggota tim harus menyelesaikan semua backlog dalam jangka waktu yang telah ditetapkan di awal (yakni pada saat Sprint Planning).
4. Sebuah [artikel](https://agileweboperations.com/2015/07/27/scrum-vs-kanban/" \t "_blank) menyebutkan bahwa Scrum cocok digunakan pada organisasi yang benar-benar membutuhkan perubahan mendasar menuju proses yang lebih efisien. Dan juga cocok digunakan untuk membuat Minimum Viable Product (MVP).
5. Beberapa pekerjaan yang cocok menggunakan Scrum, yaitu pengembangan aplikasi baru, pengembangan merek, dan kampanye pemasaran.

## **Sementara itu, Kanban memiliki karakteristik…**

1. Peran di dalam tim hanya dibagi menjadi dua, yakni Product Owner dan anggota.
2. Tidak memiliki acara yang dibentuk secara seremonial.
3. Jika waktu menjadi concern utama pada Scrum, maka pada Kanban fokus utamanya adalah jumlah pekerjaan. Pekerjaan yang sudah dikomitmenkan harus terus mengalir dari status ke status lainnya sehingga hal ini dapat dicapai ketika jumlahnya tidak terlalu banyak.
4. Kanban cocok digunakan pada organisasi yang sudah memiliki proses kerja yang ingin ditingkatkan dari waktu ke waktu tanpa mengganggu seluruh sistem.  
   Selain itu, Kanban juga biasa digunakan ketika hanya dikerjakan oleh satu orang atau tim kecil. Untuk pekerjaan yang sifatnya berkelanjutan dan proyek pemeliharaan, Kanban menjadi pilihan yang cocok.
5. Pekerjaan yang cocok menggunakan Kanban antara lain desain dan produksi seni, desain UX, marketing collateral, dan pembuatan siaran pers.

## Pengalaman Codex dalam Praktik Scrum dan Kanban

Di masa-masa awal, Codex menggunakan Scrum untuk semua jenis pekerjaan. Akan tetapi, dalam perjalanannya tim mengalami hambatan ketika ada pekerjaan yang sifatnya rutin (berulang) dan pekerjaan yang sangat bergantung pada pihak ketiga.

Saat itu, tim tidak pernah mampu untuk menyelesaikan pekerjaannya tepat waktu dan hal ini tentu saja tidak cocok dengan Scrum yang mengutamakan waktu. Tidak hanya itu, dari pekerjaan rutin juga banyak memunculkan backlog baru di tengah-tengah sprint sehingga mengganggu jalannya sprint.

Dengan kondisi seperti itu, tim memutuskan untuk beralih ke Kanban untuk pekerjaan yang tidak melibatkan dev-team. Sementara untuk pekerjaan pengembangan produk, Codex tetap menggunakan metode Scrumban. Di sini biasanya Kanban dimanfaatkan untuk pekerjaan yang terkait dengan bug fixing.

Ketika memilih Scrum atau Kanban, maka yang perlu diperhatikan adalah bagaimana kita mampu mendefinisikan kebutuhan tim dan memiliki komitmen yang tinggi dalam menerapkan cara kerja Agile.

**Pustaka**

<https://medium.com/@purwanto.dev/metodologi-system-development-life-cycle-sdlc-2f0349df1364>

<http://rizkyahf.blogspot.com/2014/08/model-pengembangan-perangkat-lunak.html>

<https://medium.com/codexstories/scrum-dan-kanban-mana-yang-lebih-baik-c79f7c391a2b>