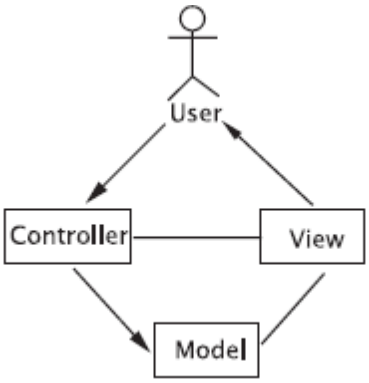


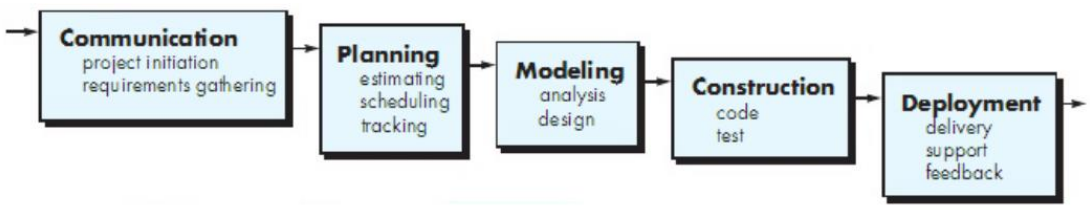
Gambar 2. 1 *Model-View-Controller architecture*

(Sumber: Sarnath Ramnath & Brahma Dathan, 2010:240)

Model, merupakan *object passive* yang relatif, yaitu menyimpan data. Setiap object memiliki aturan tersendiri dalam *model*. *View* menerjemahkan *model* kedalam bentuk format yang ditentukan, biasanya yang cocok untuk interaksi dengan *end user*. Misalnya jika *model* menyimpan informasi tentang rekening bank, pada *view* tertentu saja yang dapat menampilkan jumlah rekening dan total saldo rekening. Sedangkan, *controller* mendapatkan input dari penggunan dan jika diperlukan, lalu *method calls* pada *model* akan mengubah data yang tersimpan. Ketika *model* diubah, maka *view* akan respon dengan apa yang diubah lalu ditampilkan.

2.3.6 *Waterfall Model*

Menurut Pressman (2015:42), *waterfall model* adalah pendekatan sistematis dan sekuensial ke pengembangan *software* yang dimulai dengan spesifikasi *requirement* oleh pengguna dan berlanjut ke proses *planning*, *modeling*, *construction*, dan *deployment*, yang berpuncak dengan dukungan terus-menerus terhadap *software* yang telah diselesaikan.



Gambar 2. 15 *Waterfall Model*

(Sumber: Pressman, 2015:42)

2.3.7 *Eight Golden Rules*

Menurut Shneiderman (2005:74), *eight golden rules* adalah delapan prinsip yang bisa diaplikasikan ke sebagian besar sistem interaktif. Prinsip-prinsip tersebut erasal dari pengalaman dan dikembangkan selama lebih dari dua dekade, membutuhkan validasi dan pengaturan untuk tujuan desain spesifik. Berikut ini delapan prinsip tersebut:

- Konsistensi

  Urutan tindakan yang konsisten sebaiknya diharuskan pada situasi yang serupa, yaitu menggunakan istilah, warna, dan layout yang sama.
- Memenuhi kebutuhan universal

  Mengenali kebutuhan dari banyak pengguna dan mendesain fleksibilitas, untuk memfasilitasi transformasi konten bagi pengguna yang berbeda-beda.
- Memberikan umpan balik yang informatif

  Untuk setiap tindakan pengguna, sebaiknya terdapat umpan balik sistem. Untuk tindakan yang sering dilakukan dan berskala kecil, responsnya sebaiknya sederhana, sedangkan untuk tindakan yang jarang dilakukan dan berskala besar, responsnya sebaiknya lebih besar.
- *Dialog closure* (Desain dialog untuk menghasilkan penutupan)

  Urutan tindakan sebaiknya diorganisasikan menjadi satu kelompok dengan awal, pertengahan, dan akhir.
- Penanganan kesalahan yang sederhana

  Sebisa mungkin, desain sistem sehingga pengguna tidak bisa membuat kesalahan serius atau yang bisa merusak sistem.
- Memudahkan kembali ke tindakan sebelumnya

  Setiap tindakan sebaiknya bisa dikembalikan, sehingga mendorong mereka untuk mencoba fitur-fitur yang belum mereka ketahui.
- Mendukung tempat pengendali internal

  Pengguna yang ahli sangat ingin merasakan bahwa mereka memegang kendali sistem dan sistem merespon tindakan mereka.
- Mengurangi beban ingatan jangka pendek

  Batasan dari ingatan jangka pendek manusia mengharuskan sistem didesain sederhana sehingga tidak membebani mereka.