BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

LECTURE NOTES

Software Engineering

Minggu 10

Software Risk Management

And Reengineering

LEARNING OUTCOMES

Setelah menyelesaikan pembelajaran ini, mahasiswa akan mampu:

- ☑ LO 4 – Menganalisa proyek *management software*

Outline Materi (Sub-Topic) :

1. *Reactive versus Proactive Risk Strategies*
2. *Software Risk*
3. *Software Maintenance*
4. *Business Process Reengineering*
5. *Software Reengineering*
6. *Reverse Engineering*
7. *Forward Engineering*
8. *The Economics of Reengineering*
9. Studi Kasus

ISI MATERI

1. *Project Risk Management*

Risiko memilik arti sesuatu yang bersifat tidak pasti (uncertainty) dan memiliki kemungkinan terjadi pada masa yang akan dating. Risiko sendiri yang sering dipahami adala sesuatu yang bersifat negative dan berdampak pada proyek, dapat secara Teknik atau pun non teknis.

Risiko yang bersifat teknis misalnya:

- Kemungkinan terjadi nya kehilangan data karena tidak ada backup system
- Kemungkinan adanya penyusup karena penerapan system security yang kurang memadai
- Kemungkinan terjadinya issue pada basis data

Adapun risiko yang sifatnya non teknis lebih pada aspek komponen dari manajemen proyek, seperti:

- Ada risiko mengenai keterlambatan proyek
- Risiko pengeluaran biaya yang melebihin anggaran
- Risiko mengenai adanya scope yang tidak dapat diimplementasikan
- Risiko tidak diterimanya *software* oleh pelanggaran

Risiko-risiko tersebut perlu diidentifikasikan di awal untuk mengurangi dampak dan kemungkinan terjadi, sehingga diperlukan pengetahuan dan ketrampilan mengenai pengelolaan manajemen risiko oleh semua tim proyek.

2. ***Reactive versus Proactive Risk Strategies***

Rective risk management:

- Tim proyek bereaksi terhadap resiko ketika resiko itu muncul
- Mitigasi – rencana untuk sumber daya tambahan sebagai bentuk antisipasi
- Memperbaiki kegagalan, sumber daya ditemukan dan diimplementasikan ketika resiko terjadi
- Manajemen krisis, kegagalan tidak merespon untuk diterapkan untuk sumber daya dan proyek dalam bahaya

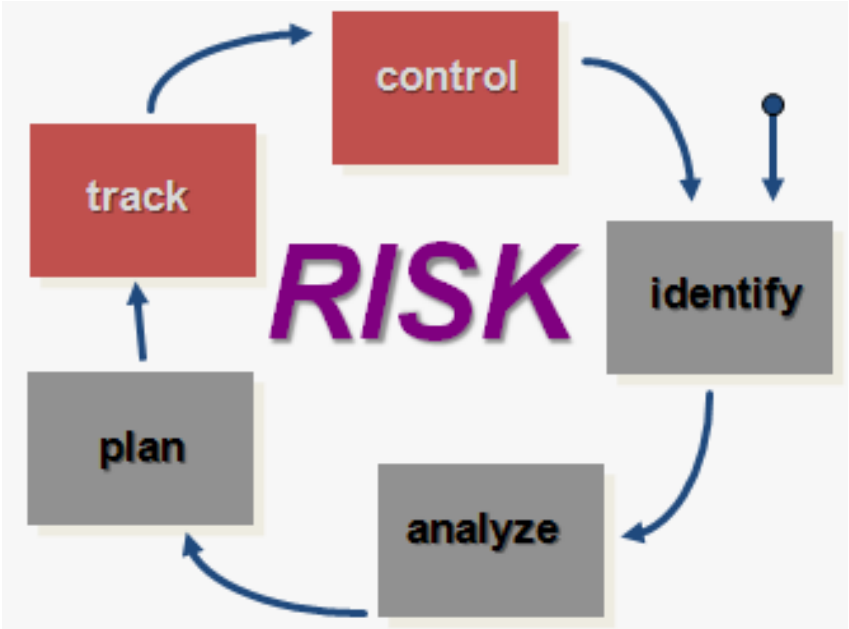
Proactive risk management:

- Analisa resiko secara formal dijalankan
- Melakukan perbaikan terhadap akar masalah dari resiko, hal ini dapat menggunakan konsep TQM dan analisis SQA, serta memeriksa sumber dari resiko.

3. ***Software Risk***

Tujuh prinsip resiko *software*:

a. Menjaga prespektif global
b. Mengambil pandangan jauh ke depan
c. Menghimbau komunikasi terbuka
d. Integrasi
e. Menekankan proses yang berlanjut
f. Mengembangkan visi produk yang sama
g. Mendorong kerja tim



Mengidentifikasikan resiko sebuah software dapat dilakukan dengan mempertimbangkan hal berikut:

- Ukuran produk, resiko diasosiasikan dengan ukuran secara keseluruhan dari software yang akan dibuat/dimodifikasi.
- Dampak bisnis, resiko diasosiasikan dengan batasan yang dimiliki oleh pihak *management* atau pasar.
- Karakteristik *customer*, resiko diasosiasikan dengan kecanggihan dari customer dan keahlian dari developer dalam berkomunikasi dengan customer dengan cara yang tepat.
- Defisi proses, resiko diasosiasikan dengan ukruang dimana proses software telah didefinisikan dan diikuti oleh perusahaan.
- Lingkungan pengembangan, resiko diasosiasikan dengan ketersediaan dan kualitas dari tool yang digunakan untuk membangun produk.
- Teknologi yang akan dibuat, resiko diasosiasikan dengan kompleksitas sistem yang akan dibuat dan keterbaruan teknologi yang disatukan dengan sistem tersebut.

- Pengalaman dan jumlah staff, resiko diasosiasikan dengan pengalaman proyek dan teknikal dari para software engineer yang akan melakukan pengembangan.

4. *Software Maintenance*

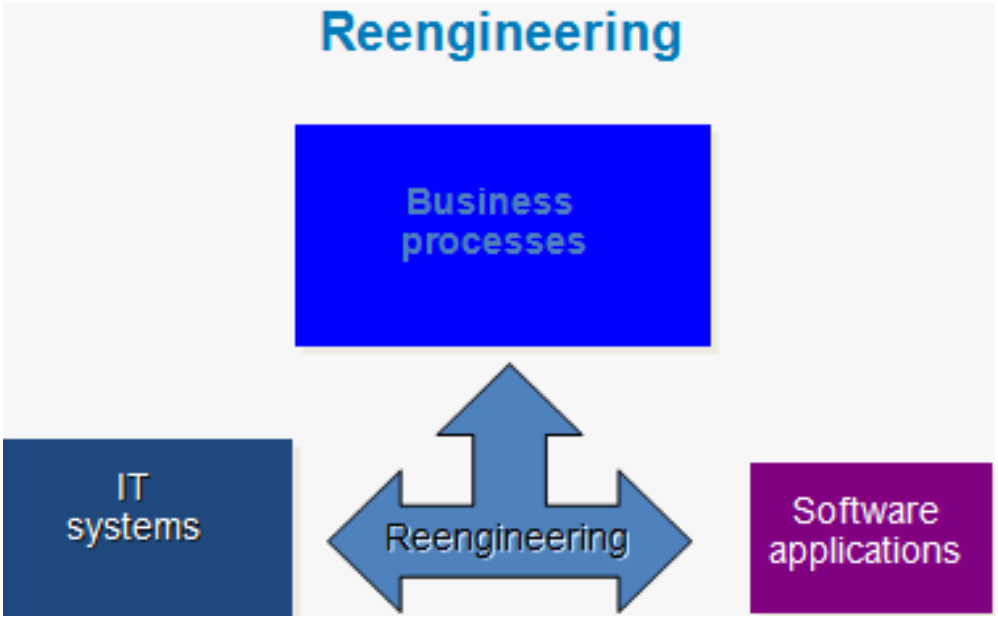
Ketika *software* telah direlease, maka kegiatan setelahnya disebut sebagai software maintenance proses.

- Memperbaiki laporan *bug* yang diterima
- Permintaan terhadap perubahan *software* sehingga dapat memenuhi kebutuhan khusus.
- Permintaan pengembangan *softwar* dari perusahaan untuk meningkatkan keuntungan.

Untuk memenuhi tuntutan dalam proses maintenance, *software* harus dapat di *maintain* (*maintainable*):

- *Software* yang dapat dimaintain menunjutkan modularitas yang efektif
- Menggunakan *design pattern* yang memudahkan pemahaman
- Dikembangkan dengan menggunakan konvesi dan standar kode yang bagus, sehingga menghasilkan *source code* yang dapat mendokumentasikan dirinya sendiri serta dapat mudah dipahami.
- Telah melewati berbagai teknik penjaminan kualitas yang membuka masalah maintaince yang potensial sebelum software direlease.
- Dibuat oleh software engineer yang memahami bahwa mereka mungkin tidak berada ditempat ketika terjadi masalah.

5. *Business Process Reengineering*



Pada gambar dibawah ini, dapat dilihat proses-proses yang ada dalam BPR *(Business Process Reengineering):*

- **Pendefinisian bisnis**

  Tujuan bisnis diidentifikasikan dalan konteks 4 driver kunci: mengurangi biaya, mengurangi waktu, meningkatkan kualitas, dan pengembangan serta pemberdayaan personel.

- **Identifikasi proses**

  Identifikasi proses-proses yang penting dalam mencapai tujuan yang didefinisikan dalan bisnis.

- **Evaluasi proses**

  Proses yang digunakan untuk melakukan pengukuran dan analisa.

- **Perancangan dan spesifikasi proses**

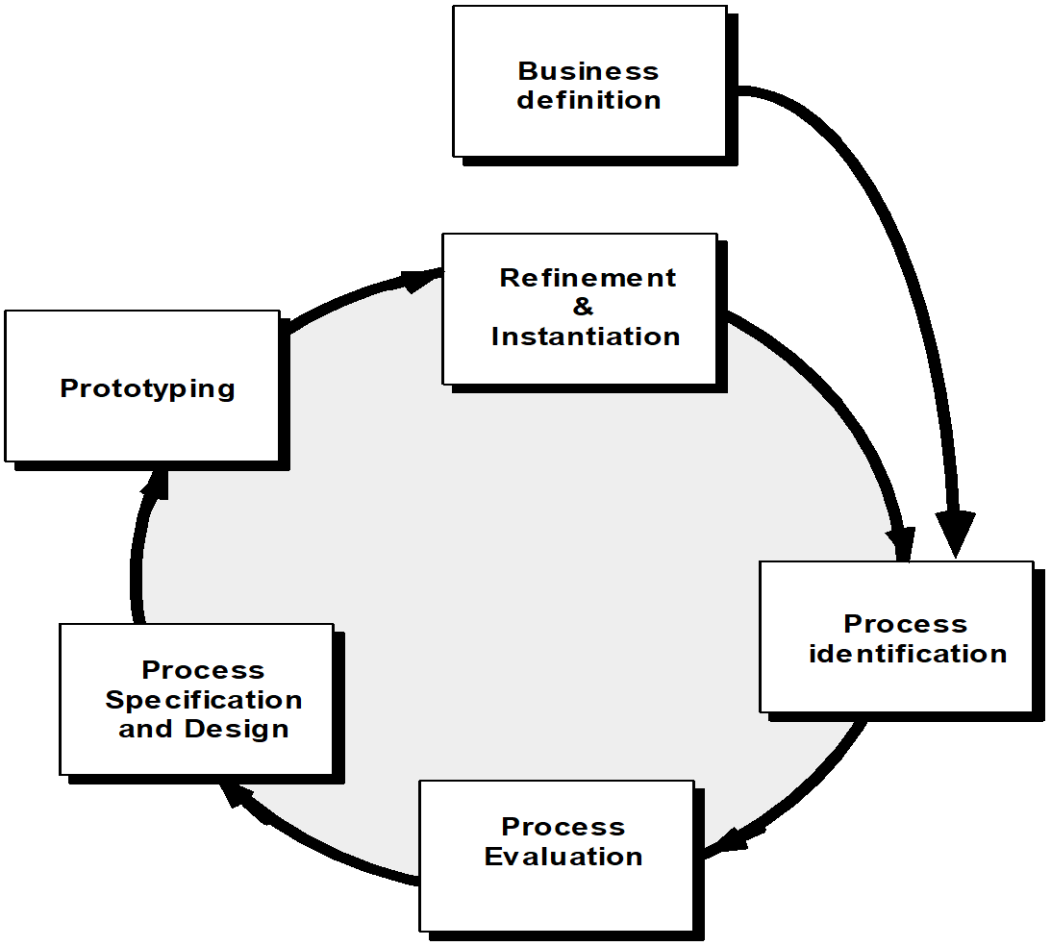
  Berdasarkan informasi yang didapat selama 3 fase pertama BPR, use case disiapkan untuk setiap proses yang dirancang kembali

- **Prototyping**

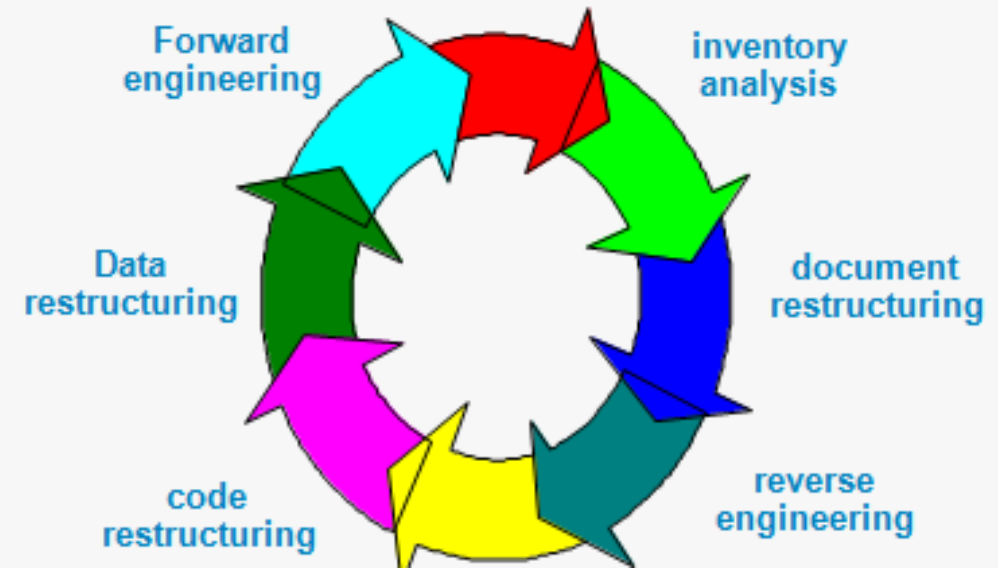
  Perancangan kembali proses bisnis harus dibuat prototypenya sebelum benar-benar diintegrasikan ke dalam bisnis.

- **Perbaikan dan instansiasi**

  Berdasarkan masukkan dari proses prototype, proses bisnis diperbaiki dan kemudia diinstansiasikan ke dalam sistem bisnis.



6. *Software Reengineering*



Inventory Engineering:

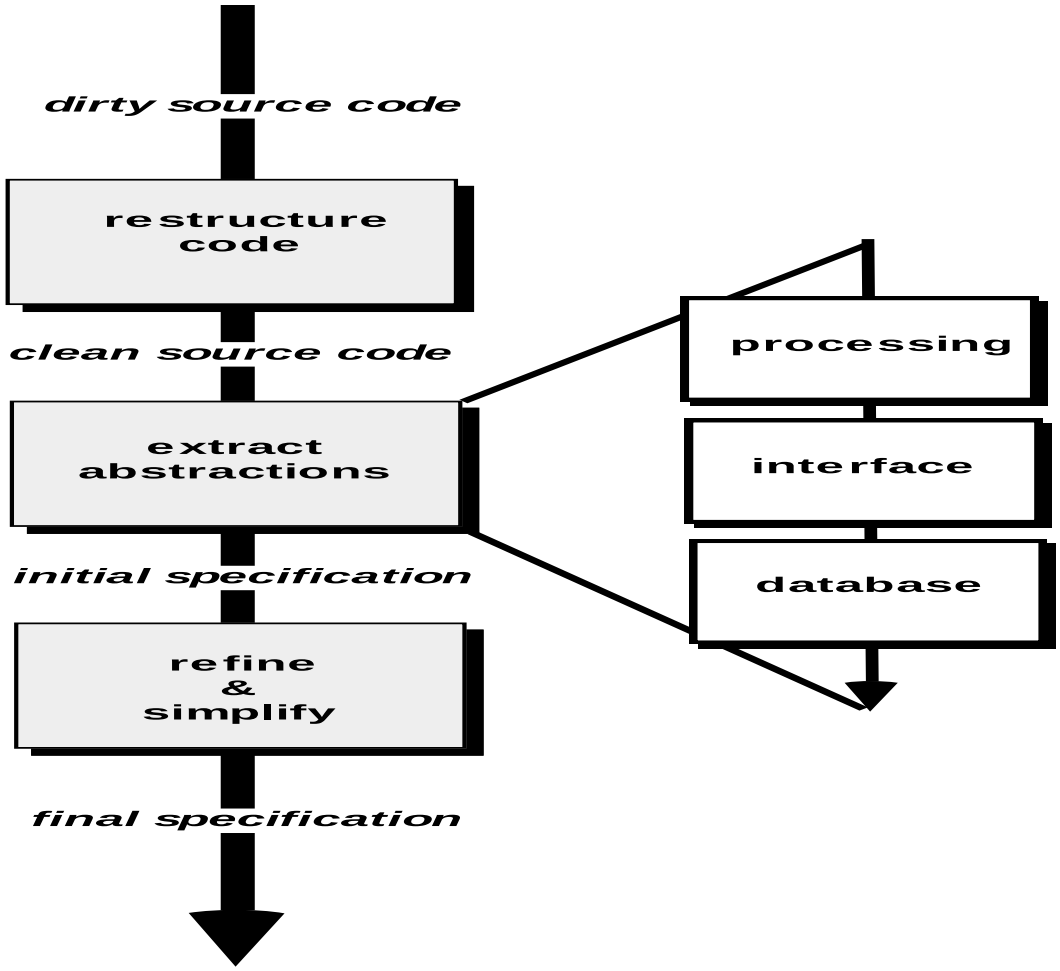
Software engineering dimulai dengan tahapan *inventory engineering* dengan membuat sebuah tabel yang berisi semua aplikasi yang dimiliki. Kemudian menganalisa dan memprioritaskan kandidat aplikasi yang akan di-*reengineeing.*

Document Restructuring:

Penstrukturan kembali dokumen dapat dilakukan:

- Menciptakan dokumentasi yang tidak memakan waktu.
- Dokumentasi harus selalu diperbarui, namun hal ini memiliki keterbatasan sumber daya.
- Sistem adalah kritikal bisnis dan harus selalu diredokumentasikan secara penuh.

7. *Reverse Engineering*



Code Restructuring:

- *Source code* dianalisa menggunakan tool restructuring
- Bagian kode yang dirancang dengan kurang baik akan dirancang ulang.
- Pelanggaran terhadap struktur pemrograman dicatat dan distruktur ulang
- *Code* yang telah distruktur ulang direview dan diuji untuk memastikan bahwa tidak ada anomaly yang disebabkan
- Dokumentasi untuk *internal code* diperbarui

8. Forward Engineering

a. Biaya untuk memaintain satu baris dari *source code* mungkin 20-40 kali lipat dari biaya awal yang dikeluarkan pada proses pengembangan untuk baris tersebut.

b. Merancang ulang arsitektur *software* menggunakan konsep perancangan *modern* dapat memfasilitasi proses maintenance di masa depan

c. Karena *prototype* dari aplikasi sudah ada, produktifitas pengembangan harus lebih tinggi dari rata-rata

d. *User* sudah memiliki pengalaman terhadap *software*, oleh karena itu, requirement baru dan arah perubahan dapat dihadapi dengan lebih mudah

e. Tool *CASE* untuk *reengineering* akan mengautomasi beberapa bagian pekerjaan.

f. Konfigurasi *software* yang lengkap akan dihasilkan dari penyelesaian proses

9. Studi kasus

Di dalam pendefinisian manajemen risiko, banyak terjadi kesalah pahaman antara risiko dan problem. Problem adalah sesuatu masalah yang saat ini sedang terjadi, sedangkan risiko adalah kemungkinan terjadinya masalah pada waktu yang akan datang. Problem sedang atau sudah terjadi, sedangkan risiko belum terjadi. Problem diselesaikan dengan issue atau problem management, sedangkan risiko dikelola melalui manajemen risiko.

Contoh problem: user tidak dapat masuk ke dalam system karena system down

Contoh risiko: karena tidak ada backup server, terjadi kemungkinan system tidak dapat diakses Ketika terjadi crash pada system yang ada sekarang

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

DAFTAR PUSTAKA

- Software engineering : a practitioners approach : Chapter 35/Pages 777, Chapter 36/Pages 795
- Sw maintenance and Reengineering, http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/13.25.SWEng4/html/text.html
- Risk Management & operational Risk, http://www.grafp.com/products/risk-manage.html