Membangun DSS

Membangun DSS

- Pendahuluan
  - Membangun sebuah DSS, apalagi yang besar, merupakan proses yang rumit.
  - Melibatkan hal-hal : teknis (hardware, software) dan perilaku (interaksi manusia-mesin, dampak DSS pada individu.

Strategi Pengembangan

1. Tulis DSS dengan bahasa pemrograman umum : Pascal, Delphi, Java, C++ dll.
2. Menggunakan 4GL : financial-oriented language, data-oriented language.
3. Menggunakan DSS Generator : Excell.
4. Menggunakan DSS Generator khusus
5. Mengembangkan DSS dengan metodologi CASE

Level Teknologi

Kerangka kerja untuk memahami konstruksi DSS mengidentifikasi 3 level teknologi DSS : Specific DSS, DSS-generation, dan DSS tools.

- Specific DSS (DSS applications).

  “Final Product” atau aplikasi DSS yang nyata-nya menyelesaikan pekerjaan yang kita inginkan disebut dengan specific Decision (SDSS). Contoh : SDSS untuk menganalisis joint venture.

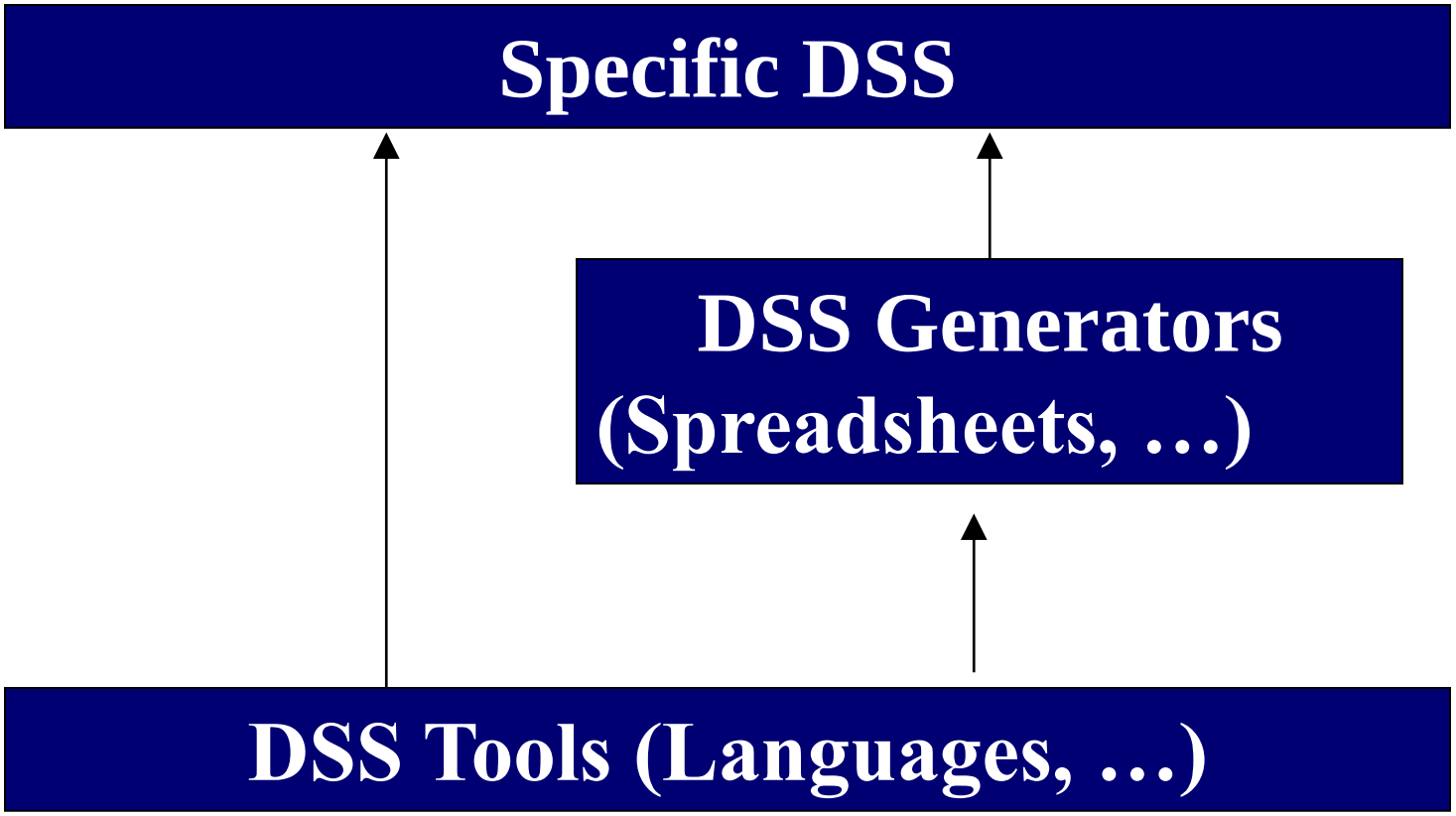
Level Teknologi (lanjutan) :

- DSS Generators (atau Engines).

  adalah software pengembangan terintegrasi yang menyediakan sekumpulan kemampuan untuk membangun specific DSS secara cepat, tak mahal, dan mudah. Contoh : Microsoft Excel.
- DSS Tools.

  Level terendah dari teknologi DSS adalah software utility atau tools. Elemen ini membantu pengembangan baik DSS generators atau SDSS.Contoh : grafis (hardware dan software), editors, query systems, random number generator, dan spreadsheets.

Relasi diatara 3 level teknologi DSS



Proses Pengembangan DSS

1. Perencanaan.

   Merumuskan kerangka dan ruang lingkup SPK, persyaratan unjuk kerja , dan memilih konsep-konsep & menganalisis model pembuatan keputusan yang relevan dengan tujuan SPK. Langkah ini menentukan pemilihan jenis SPK yang akan dirancang dan metode pendekatan yang dipergunakan.

2. Penelitian.

   Berhubungan dengan pencarian data serta sumber daya yang tersedia

3. Analisis & Perancangan konsep.

   Penentuan teknik pendekatan yang akan dilakukan serta sumber daya yang dibutuhkan

4. Perancangan.

   Melakukan perancangan ketiga subsistem utama SPK yaitu subsistem database, model dan Dialog.

5. Konstruksi.

   Merupakan kelanjutan dari perancangan dimana ketiga subsistem yang telah dirancang digabungkan menjadi suatu SPK

6. Implementasi.

   Menerapkan SPk yang dibangun. Pada tahap dilakukan testing, evaluasi, penampilan, orientasi, pelatihan dan penyebaran

7. Pemeliharaan.

   Tahapan yang dilakukan terus menerus untuk mempertahankan keandalan sistem

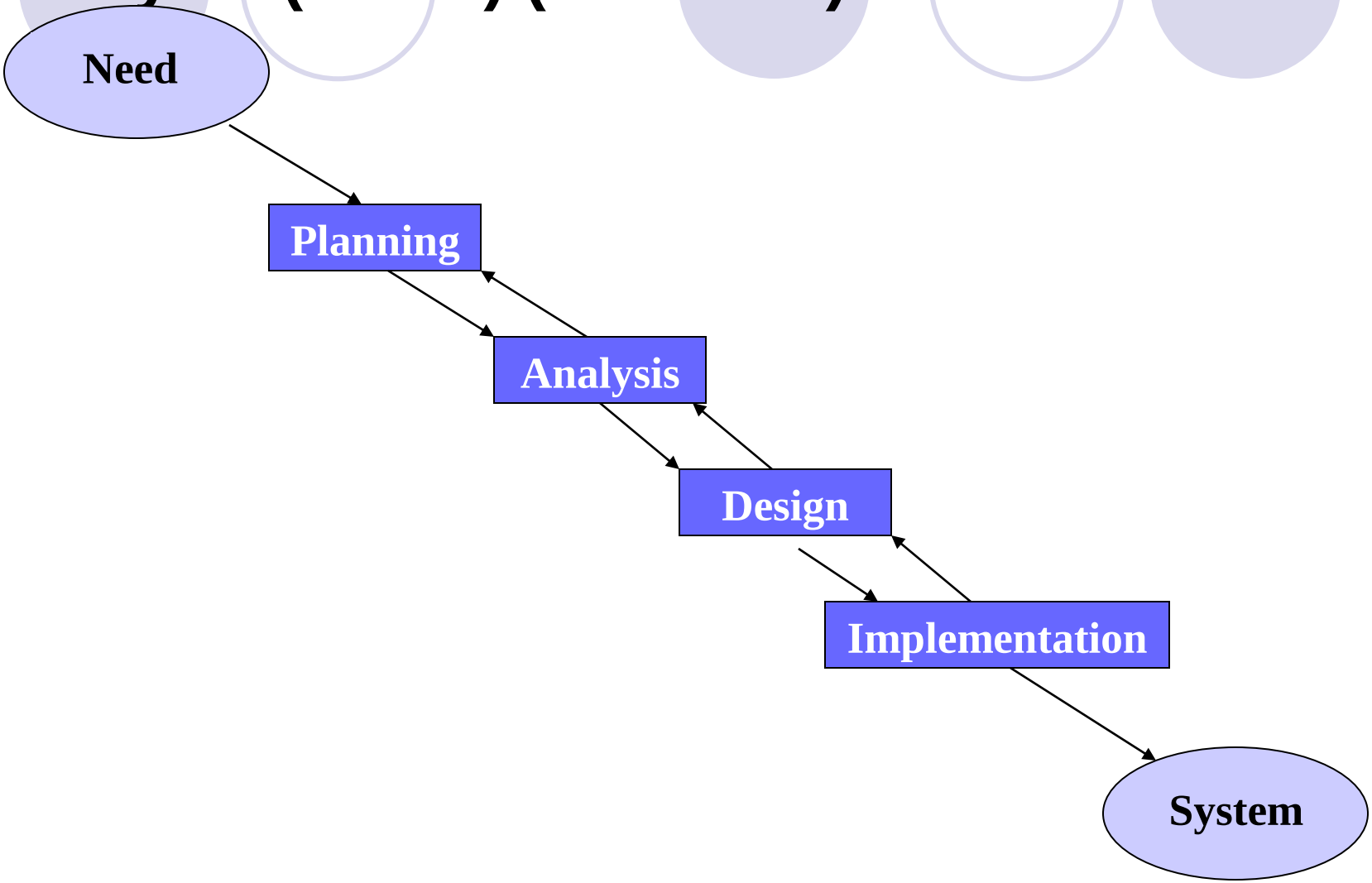
8. Adaptasi.

   Melakukan pengulangan terhadap tahapan diatas sebagai tanggapan terhadap perubahan kebutuhan “pemakai”.

System Development Issues

- System development life cycle (SDLC)
- Prototyping
- Forming the development team

Traditional Systems Development Life Cycle (SDLC) (Waterfall)



Fundamental SDLC Phases

- **Planning**
- **Analysis**
- **Design**
- **Implementation**

Steps and deliverables follow

Planning

Why Build the System?

Minor Step	Deliverable
1. Identify business value	System request
2. Analyze feasibility	Feasibility study
3. Develop work plan	Work plan
4. Staff project	Staffing plan, Project charter
5. Control and direct project	Project management tools CASE tool Standards list Project binders / files Risk assessment

Analysis

Who, What, When, Where?

Minor Step	Deliverable
6. Analyze problem	Analysis plan
7. Gather information	Information
8. Model process(es)	Process model
9. Model data	Data model

Design

How Will the System Work?

Minor Step	Deliverable
10. Design physical system	Design plan
11. Design architecture	Architecture design, Infrastructure design
12. Design interface	Interface design
13. Design database and files	Data storage design
14. Design program(s)	Program design

Implementation

System Delivery

Minor Step	Deliverable
15. Construction	Test plan, Programs, Documentation
16. Installation	Conversion plan, Training plan

2. Prototipe

- Suatu metode dalam pengembangan sistem yang menggunakan pendekatan untuk membuat sesuatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai
- Hal ini berbeda dengan pendekatan SDLC tradisional (konvensional) yang lebih banyak menghabiskan waktu untuk menghasilkan spesifikasi yang sangat rinci sebelum pemakai dapat mengevaluasi sistem
- Mengingat kebanyakan pemakai mengalami kesulitan dalam memahami spesifikasi sistem berakibat bahwa pemakai tidak begitu paham sampai pengujian dilakukan

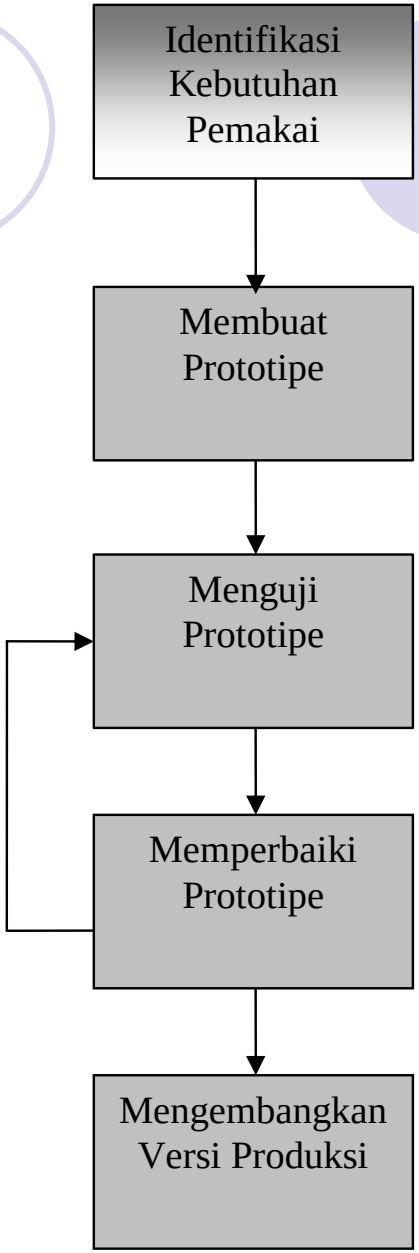
Prototipe (Lanjutan…)

- Selain itu, prototipe membuat proses pengembangan sistem informasi menjadi lebih cepat dan lebih mudah, terutama pada kedaaan kebutuhan pemakai sulit untuk diidentifikasi.
- Prototipe dapat dibuat dengan menggunakan perangkat-perangkat, misalnya Visual BASIC dan PowerBuilder, ataupun DBMS (*Database Management System*) seperti Microsoft Access, sehingga pembuatan program dapat dilakukan dengan cepat

Sasaran Prototipe :

1. Mengurangi waktu sebelum pemakai melihat sesuatu yang konkret dari usaha pengembangan sistem
2. Menyediakan umpan balik yang cepat dari pemakai kepada pengembang
3. Membantu menggambarkan kebutuhan pemakai dengan kesalahan yang lebih sedikit
4. Meningkatkan pemahaman pengembang dan pemakai terhadap sasaran yang seharusnya dicapai oleh sistem
5. Menjadikan keterlibatan pemakai sangat berarti dalam analisis dan desain sistem

Pendekatan Prototipe



- Pengembang dan pemakai bertemu
- Pemakai menjelaskan kebutuhan sistem

- Pengembang mulai membuat prototipe

- Pemakai menguji prototipe dan memberikan kritikan atau saran

- Pengembang melakukan modifikasi sesuai dengan masukan pemakai

- Pengembang merampungkan sistem sesuai dengan masukan terakhir dari pemakai

Kelebihan Prototipe

- Pendefinisian kebutuhan pemakai menjadi lebih baik karena keterlibatan pemakai yang lebih intensif
- Meningkatkan kepuasan pemakai dan mengurangi risiko pemakai tidak menggunakan sistem mengingat keterlibatan mereka yang sangat tinggi sehingga sistem memenuhi kebutuhan mereka dengan lebih baik
- Mempersingkat waktu pengembangan
- Memperkecil kesalahan disebabkan pada setiap versi prototipe, kesalahan segera terdeteksi oleh pemakai
- Pemakai memiliki kesempatan yang lebih banyak dalam meminta perubahan-perubahan
- Menghemat biaya (menurut penelitian, biaya pengembangan dapat mencapai 10% hingga 20% dibandingkan kalau menggunakan SDLC tradisional)

Kelemahan Prototipe

- Prototipe hanya bisa berhasil jika pemakai bersungguh-sungguh dalam menyediakan waktu dan pikiran untuk menggarap prototipe
- Kemungkinan dokumentasi terabaikan karena pengembang lebih berkonsentrasi pada pengujian dan pembuatan prototipe
- Mengingat target waktu yang pendek, ada kemungkinan sistem yang dibuat tidak lengkap dan bahkan sistem kurang teruji
- Jika terlalu banyak proses pengulangan dalam membuat prototipe, ada kemungkinan pemakai menjadi jenuh dan memberikan reaksi yang negatif
- Apabila tidak terkelola dengan baik, prototipe menjadi tak pernah berakhir. Hal ini disebabkan permintaan terhadap perubahan terlalu mudah untuk dipenuhi

Prototipe baik dipakai pada keadaan

- Sistem mempunyai resiko tinggi
  - Tidak jelas permasalahannya
  - Tidak jelas kebutuhan & keinginan
  - Tidak pasti ada yang ingin dilakukan
- Perancangan Dialog User - Komputer
  - Bagiamana membuat dialog yg. baik, ramah, mudah ?
- Sistem diminati oleh banyak pemakai
  - Mencari kesepakatan
  - Basis untuk menyamakan persepsi
- User ingin cepat selesai
  - User tidak sabar menunggu
  - Prototipe segera memperlihatkan bentuk kerja sistem

- Masa pakai singkat
  - Sistem hanya dipakai beberapa kali saja
- Ingin menunjukkan inovasi
  - Pengembangan dapat menunjukkan kecanggihan
  - Sistem cepat terlihat (mungkin juga cepat selesai)
- Kebutuhan berubah-ubah
  - User sulit menjelaskan kebutuhan
  - Menjadi keadaan yang paling umum untuk memakai prototyping

Hardware Selection

- PC
- Unix workstations
- Network of Unix workstations
- Web servers
- Mainframes
- Typically use existing hardware

Software Selection

Complex because

- At start, information requirements, etc. are unknown
- Hundreds of packages
- Software updated rapidly
- Price changes
- Many people involved in decision
- Language capability problems

(More)

- **Different tools might be needed**
- **Many criteria**
- **Technical, functional, end-user, and managerial issues**
- **Inaccurate published software reviews**
- **Might prefer a single vendor**
- **Maybe use the AHP!!!**

Pengembangan DSS Berbasis User

Pengembangan DSS berbasis user adalah pengembangan dan penggunaan sistem informasi berbasis komputer oleh orang-orang di luar wilayah sistem informasi formal.

- Keuntungan bila user sendiri yang membangun DSS :
  - Waktu penyelesaian singkat
  - Syarat-syarat spesifikasi kebutuhan sistem tak diperlukan
  - Masalah implementasi DSS dapat dikurangi
  - Biayanya sangat rendah
- Resikony adalah :
  - Kualitasnya bisa tak terjaga.
  - Resiko potensial kualitas dapat diklasifikasi dalam 3 katagori: (a) tool dan fasilitas dibawah standar (b) resiko yang berhubungan dengan proses pengembangan (contoh : pengembangan sistem yang mnghasilkan hasil yang salah), dan (c) resiko manajemen data (misal : kehilangan data).

Fleksibilitas dalam DSS

Hal-hal yang menyebabkan kebutuhan akan fleksibilitas dalam DSS :

- Tak seorang pun, baik user maupun pembangun DSS, yang mampu untuk menentukan kebutuhan fungsional seluruhnya.
- User tak tahu, atau tak dapat mengungkapkan, apa yang mereka mau dan butuhkan
- Konsep user mengenai tugas, dan persepsi dari sifat dasar masalah, berubah pada saat sistem dipakai.
- Penggunaan DSS secara aktual hampir pasti berbeda dari yang diinginkan semula.
- Solusi yang diturunkan melaluii DSS bersifat subyektif.
- Terdapat berbagai vriasi diantara orang-orang dalam hal bagaimana mereka menggunakan DSS

Ringkasnya ada 2 alasan utama adanya fleksibilitas dalam DSS :

- DSS harus berevolusi atau berkembang untuk mencapai desain operasional, sebab tak seorangpun yang bisa memperkirakan atau mengantisipasi apa yang dibutuhkan secara lengkap.
- Sistem jarang mencapai hasil final; ia harus sering diubah untuk mengantisipasi perubahan dalam hal : masalah, user dan lingkungan. Faktor-faktor ini memang sering berubah-ubah. Perubahan yang terjadi haruslah mudah untuk dilakukan.

Jenis Fleksibilitas dalam DSS

1. Fleksibilitas menyelesaikan.

   Kemampuan fleksibilitas dalam menampilkan aktivitas intelligence, design, dan choice dan dalam menjelajah perbagai alternatif memandang atau menyelesaikan suatu masalah. Contoh : kemampuan “what-if”.

2. Fleksibilitas memodifikasi.

   Kemampuan memodifikasi konfigurasi DSS tertentu sehingga dapat menangani berbagai masalah yang berbeda, atau pada perluasan masalah. Fleksibilitas ini diatur oleh user / pengembang DSS.

3. Fleksibilitas mengadaptasi.

   Level ketiga dalam hal mengadaptasi prubahan yang harus dilakukan pada berbagai DSS tertentu. Ini diatur oleh pembangun DSS.

4. Fleksibilitas berevolusi.

   Adalah kemampuan dari DSS dan DSS Generator dalam berevolusi untuk merespon perubahan sifat dasar teknologi dimana DSS berbasis disitu.