Software Engineering

Software Engineering 05

SOFTWARE DESIGN

Software Design යනු Software Engineering හි ඉතාමත් වැදගත් කොටසකි. මෙහිදී සලකා බලනු ලබන්නේ Requirement Analysis Phase එකේදී හදුනාගත් Requirements , Software Tools උපයෝගී කරගනිමින් Implement කළ හැකි Software එකක නිරූපණයක් ලෙස ඉදිරිපත් කිරීමයි.
Design හි කාර්යයන

1. Sub-Systems හදුනාගැනීම
2. Software Components හදුනාගැනීම
3. Software Architecture හදුනාගැනීම
4. දත්ත Design කිරීම
5. මුහුණුවර DesignÂ කිරීම
6. ගණනය කිරීම් Design කිරීම ආදිය වේ.

සාර්ථක Design එකක් සාර්ථක Software එකකට හේතුවේ. සාර්ථක Design එකක් Maintenance පහසු කරයි.

වැදගත් Software Design මූලධර්ම

- Abstraction
- Modularity
- Information hiding
- Polymorphism

- Abstraction
  Abstraction යනු software component එකක ව්‍යුක්ත ආකෘතිය නිර්මාණය කිරීමයි. එහිදී ඒ components වල තිබිය යුතු මූලික අර්ථ දැක්වීම් සඳහන් කිරීමයි. ඒ අනුව තවත් කෙනෙක් එම component එකක් භාවිතා කිරීමේදී එම අර්ථ දැක්වීම් භාවිතා කිරීම සඳහා යොමු කිරීමක් සිදු වේ. බොහෝ විට මෙය inheritance දී භාවිතා කෙරේ.

- Modularity
  මෙමගින්Software එක වෙන් වෙන් වශයෙන් නම් කරන ලද කොටස් වලට බෙදීමක් සිදු කර වෙන වෙනම Develop කරයි. මෙමඑක් කොටසක් Component ලෙස හදුන්වයි.මෙසේ System එක කොටස් කිරීම මගින් Development Process එක වේගවත් වේ. Maintenance පහසු කරවයි.

Module Coupling යනු කුමක්ද?මෙයComponent අතර අන්තර් සම්බන්දතාව මනිනා මිනුමකි.එක Component එකක සිදු වන වෙනසක් තවත් Component එකකට බලපායිනම් එය Tight Coupling ලෙස හදුන්වයි. විශාල බලපෑමක් සිදු නොකරයි නම් LosseCoupling ලෙස හදුන්වයි. Parameters යොදා ගනිමින් Component අතර සන්නිවේදනයක් සිදුවේ නම් බොහෝවිට LosseCoupling ඇති වීමට ඉඩ ඇත.Component නිර්මාණය කිරීමේදී Coupling Low වන සේ නිමවීමට වග බල ගත යුතුය.
Module Cohesion යනු කුමක්ද?

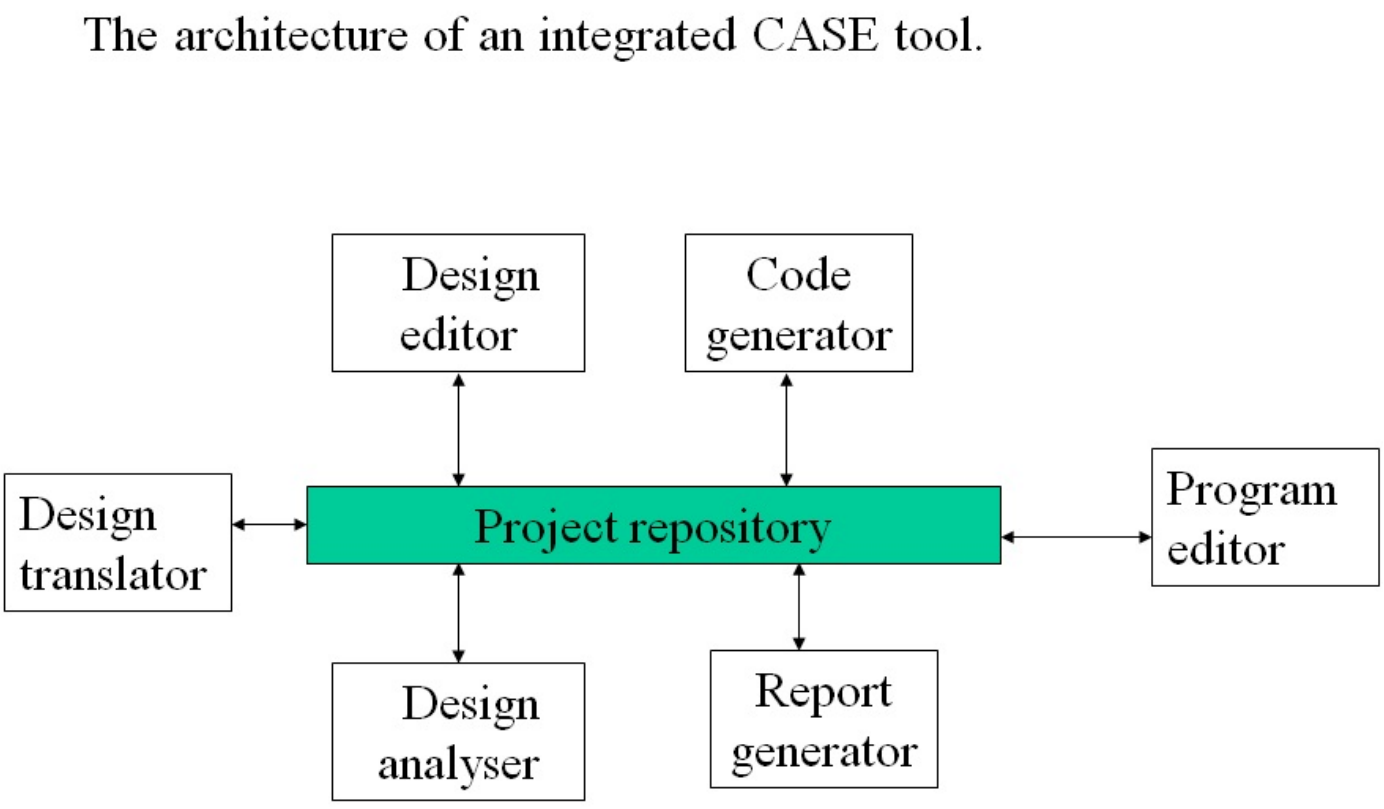
Component 1ක් ඇතුළත අන්තර් ක්‍රියාවන්ය. මෙය Component 1ක් කොතරම් හොඳින් එකට බැඳී තිබෙද යන්න මනින මිනුමකි. Component නිර්මාණය කිරීමේදීCohesion High වන සේ නිමවීමට වග බල ගත යුතුය. Component 1ක් එක් කාර්යයක් සඳහා පමණක් නිර්මාණය කිරීමෙන් High Cohesion ළඟා කර ගත හැක.

- Information Hiding

මෙමගින් කියවෙන්නේ Component Design කළ යුත්තේ Component එකක Information (Procedure and Data) සෘජුව අනෙක් Component වලට Access කල නොහැකි වන අයුරින් බවයි. මෙසේ Informationඅනෙක් Component වලට සෘජුවම Access කිරීමට නොදී Well Defined Interface මගින් කිරීමට දීම Encapsulation ලෙස හඳුන්වයි.

- Polymorphism

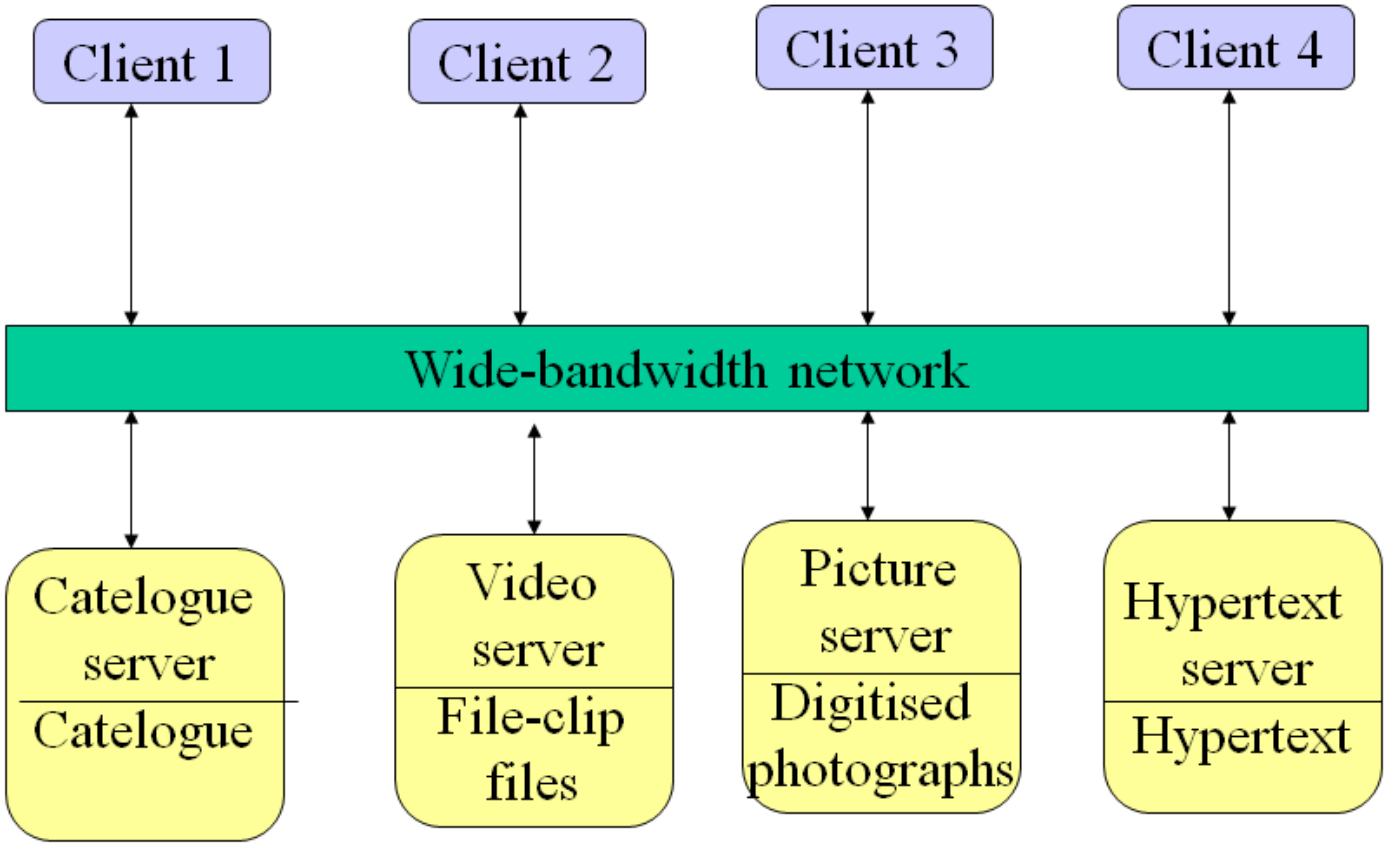
Polymorphism ලෙස හඳුන්වන්නේ සරලව කිවහොත් බොහෝ ආකාර තිබීම යන්නයි.Software Design Phase එකේ කාර්යයක් ලෙස Architecture Design හැදින්විය හැක.මෙමගින් සිදුවන්නේ System එකේ මූලික Component හඳුනාගැනීම සහ එම Component අතර සන්නිවේදනය සිදුවන ආකාරය හඳුනාගැනීමයි.
මේ යටතේ අපි Models 2ක් පිළිබඳ ඉගන ගමු.ඒවා නම් Repositary Model සහ Client Server Model ය.
Repositary Model
Sub Systems එකතු වී Systems එක තැනීමේදී ඒවා අතර තොරතුරු හුවමාරු කරගැනීම කළ යුතුය. මෙය කළ හැකි එක් ආකාරයක් වනුයේCentralDatabase එකක් තබා ගනිමින්Data Share කරගැනීමයි. මෙසේ CentralDatabase මගින්DataShare කරන System Model එක Repositary Model ලෙස හඳුන්වයි. මෙය වඩාත් සුදුසු වන්නේ එක් Sub System එකකින්Data Generate කර වෙනත් Sub Systems එකකින් එමData Use කරන Application වලටය.



Client Server Model
මෙම Modelඑකෙහි Major Components කිහිපයක් ඇත.

- Sub System වලට සේවා සපයන Stand Alone Servers කිහිපයක්. උදා: Web Servers, Print Servers
- සේවාවන් ඉල්ලුම් කරන Clients

- Clients ලට සේවා Access කළ හැකි වන සේ ඇති Network



User Interface Design

හොඳ Interface එකක් සාර්ථක System එකකට උපකාරී වේ. Interface එක පැහැදිලි හා සරල වීම වැදගත් වේ. UserInterface එකක් නිර්මාණය මගින් පරිගණක දැනුමක් නැති අයෙකුට වුවද Application එක පහසුවෙන් හේරුම් ගත හැක.

User Interface Design මූලධර්ම

- User Familiarity (Application එක නිතරම භාවිත කරන Users ලට හේරුම් ගත හැකි යෙදුම් සහ සංකල්ප යෙදීම වැදගත්)
- Recoverability (Error එකක් පැමිණිය හොත් Recover කර ගත හැකි ආකාරයක් නිර්මාණය යුතුය.)
- User Guidance (User Manual ආදියෙන් Application එක පිළිබඳ දැනුවත් කළ හැකි ආකාරයක් නිර්මාණය යුතුය.)
- User Diversity (විවිධ පුද්ගල කොට්ඨාස වලට භාවිත කළ හැකි විය යුතුය.)