1 Redegør for ideerne bag komponentbaseret programudvikling, og tilhørende designprincipper

1.1 Komponenter

- Komponenter er udskiftelige.
- Kan interagere med hinanden.
- Kommer af "komponere"altså sætte sammen med andre komponenter.
- Forskellige producenter kan lave en enhed, sålænge de har det samme interface.
- På denne måde er det nemmere at lave nye ting.

De var oprindeligt tænkt som hardwarekomponenter, da man havde lavet hardware der kunne yde meget. Man manglede desværre software, der kunne udnytte hardwaren.
Man lavede derfor nogle små dele (komponenter), der kunne bruges til noget af hardwaren.

Eftersom komponenter kan genbruges, kan de sælges til andre. Dette kan også reducere udviklingstiden.
En grov regel siger, at den skal kunne genbruges 3 gange, før det er tidsbesparende - men dette er måske lidt outdatet.

Bruges gerne på store projekter, da det er nemmere for andre folk ikke at se på source kode, men på et interface og magi. Det kræver derfor, at man får lavet nogle ordenlige interfaces.

1.2 Load-time Dynamic Linking

Udviklingen

- Man skriver et program (gerne i flere sourcefiler)
- En compiler laver objekt-filer
- En linker laver binære .exe-filer og .dll-filer
- Ved at køre .exe-filen siger den "jeg skal bruge en .dll-fil"
- Den leder så efter en .dll med et bestemt navn - her kan man indsætte sin egen.

Fordele:

- Dynamic linking sparrer plads
- Skal ikke rekompileres

Run-time Dynamic Linking

- Her fortælles hvilken dll, der ønskes loaded.
- Når loaded har man en klump kode
- Så kaldes "GetProcAdrdress", hvilket giver os en funktions pointer"

DLL typer

Traditionelle C-Style Win32 DLL'er:

- Standard som er en del af Windows.
- Bruger lavet

COM DLL'er:

- Indeholder kun 4 funktioner

.NET DLL:

DLL'er og Memory Management

En dll kan bruges fra flere programmer og loades i hver sit program, med hver sin ram.

Klasser i DLL

Det skal helst undgås, da det KUN er C++, der kan forstå disse klasser