INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

OO Analys och Design

Programming Style

Ulf Seigerroth
Internationella Handelshögskolan
Jönköping

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Objekt Orienterad stil

- Bra program tillfredsställer mer än de funktionella kraven
- Program som utvecklas under bestämda riktlinjer under design fasen blir mer korrekta, återanvändningsbara, utbyggbara och snabbare avlusade
- Det som går att applicera på traditionell programmeringsstil går även att tillämpa på OO stil
- De största skillnaderna i det OO konceptet är arv, starkare tanke på inkapsling, gränssnitt, fokus, återanvändbarhet och kommunikation

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

I OMT ges riktlinjer för följande kategorier:

- Återanvändbarhet
- Utbyggbarhet
- Robusthet
- Programmering i team

Återanvändbarhet

Genom att återanvända delar som redan blivit utvecklade räknar man med följande vinster

- Stabilare system (färre fel)
- Kortare utvecklingstider -> lägre kostnader
- Bättre förståelse (standardutveckling)

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Det talas om två typer av återanvändning

- Nyligen skriven kod inom ett pågående projekt
- Kod som skrivits för ett tidigare projekt

Baksidan av återanvändning är att den som skall återanvända tidigare skriven kod måste tillbringa mycket tid för att lära sig hitta önskade moduler

Det tar också lång tid att förstå hur modulerna fungerar

För att minimera de negativa sidorna med återanvändning så bör stilen på implementationerna följa ett visst mönster

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Stilregler för återanvändning

- *Håll metoder strikta*: en metod skall göra en och endast ensak. Om en metod gör två skilda saker så bör den delas i två metoder
- *Håll metoder små*: en metod blir då lättare att förstå och återanvända
- *Håll metoder konsistenta*: liknande metoder skall ha samma namn, villkor, parameterordning, datatyper, returvärden och felhantering
- *Separera policy och implementation*
- *Tillhandahåll metoder med generell täckning*: om en metod hämtar sista elementet i en lista så skriv även en metod för att hämta det första

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- *Generalisera metoder så mycket som möjligt*
- *Undvik global information*: det är ett brott mot inkapsling
- *Undvik olika mode*: metoder skall inte ändra beteende beroende på i vilket sammanhang de används. Gör då i stället två olika metoder

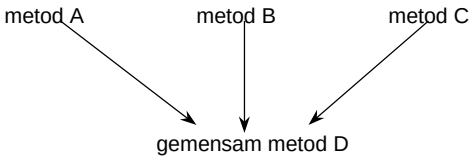
INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Användande av arv

Uppdelning av metoder för att ärva kod:

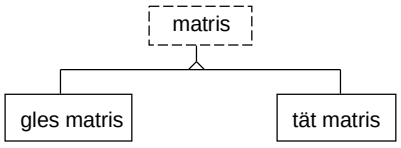
- Subrutiner: plocka ut gemensam kod ur fler metoder och lägg detta i en ny metod som sedan anropas av övriga



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Factoring: Behåll gemensamma delar i en abstrakt superklass och gör nya klasser för det som skiljer
- Arbetssätt:
  - Skapa superklassen
  - Lägg in gränssnittet i superklassen
  - Flytta upp gemensamma attribut i superklassen
  - Flytta upp gemensam kod i superklassen



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Managing reuse: Applying the law of grawity (Tim Korson)

- Tim introduserar 4 axiom för åter användning
  - **Axiom 1:** Arbete med återanvändning som är begränsat till utveckling och utnyttjande av klassbibliotek kommer inte fundamentalt att påverka produktiviteten i processen för systemutveckling.
  - **Axiom 2:** En komponent är inte färdig för generell återanvändning förrän den på ett tillfredställande sätt återanvändts i tre av varandra oberoende system.

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- **Axiom 3:** Sannolikheten att en systemutvecklare skall återanvända en komponent är omvänt proportionell mot kvadraten på avståndet mellan utvecklaren och komponenten.
- **Axiom 4:** En organisation skall inte sträva efter att uppnå ett möjligt maximum av återanvända komponenter i organisationen.

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- **Axiom 1**
  - Klassbibliotek är ekvivalent med funktionsbibliotek
    - Återanvändning av funktionalitet
    - Återanvändning av kod
  - Inkapling ger dock
    - Möjlighet till högre abstraktion
    - Bättre nivådefinitioner (generallitet) men fortfarande ett begränsat fokus
      - Ingen fundamental förändring i systemutvecklings-processen

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

  - Målet med klassbibliotek
    - Bygga lego, men
    - Klasser kan inte sättas ihop hur som helst, såvida inte
    - Klasserna är designade att fungera inom ett visst ramverk
  - Självstående klasser som
    - string, date, account, customer, sensor, etc, kan
    - Öka produktiviteten inom ett paradigm, men
    - De förändrar knappast systemutvecklingsparadigmet
  - För att ändra detta krävs en högre nivå av återanvändning
  - Komponenter kan PnP när regler och åtaganden klart är definierade
  - Dessa regler skall vara definierade inom arkitekturen och realiserade i ett ramverk

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Detta kan ge en portfölj för återanvändning som inte bara innehåller klassbibliotek utan även
  - Mönster,
  - Ramverk och
  - Olika nivåer av abstraktion

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- **Axiom 2**
  - Komponenten måste exponeras i olika fokus för att mogna till en återanvändbar generallitet
  - För tidig exponering kan medföra
    - Komponenten har för svag generallitet, vilket medför
    - Fel vid användning, vilket medför
    - Minskat förtroende och användande, vilket kan leda till
    - Ingen plats på återanvändningsmarknaden
  - Moment 22 i axiom 2

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- **Axiom 3**
  - Betydelselöst om avståndet är fysiskt eller konceptuellt
  - Fysikt avstånd
    - Utvecklaren är mer benägen att återanvända något som en kollega inom samma väggar har gjort, än
    - Att hämta något från en annan plats
    - En lösning på detta skulle kunna vara företagsnätverk, men
  - Konseptuellt avstånd
    - Inom nätverket upplevs ändå skilda organisationer som barriärer för obehindrad kommunikation och utbyte

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

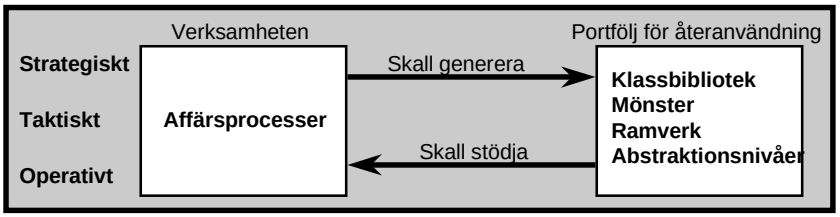
- **Axiom 4**
  - Vid maximering av återanvändning kommer projekten att saktas ner, beroende på
  - För att uppnå maximal återanvändning kommer utvecklarna vara tvungna att ägna en oproportionerlig tid till att definiera, specifisera och skriva
    - Klasser, mönster, arkitekturer och ramverk

    som är så generella att de kan maximera återanvändningen

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Slutsats
  - Fokus bör inte vara på återanvändning i en teknisk mening
  - Fokus bör istället vara på affärsprocesserna eftersom det är genom dessa som det finns en direkt koppling till portföljen för återanvändning som skall stödja verksamheten på alla nivåer



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

A Pattern for Reuse, Let architectural reuse guide component reuse (John D McGregor, Jim Doble och Asha Keddy)

- Det finns ett behov av tekniker för att kunna återanvända befintlig arkitektur, design och kod vid applikationsutveckling
- Trots många försök har man varit begränsad till ett fåtal områden för återanvändning
  - Datastrukturer
  - GUI, etc.

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Återanvändning av individuella kodavsnitt är varken effektivt eller lyckat i en vetenskaplig mening av återanvändning
- Återanvändning bör i stället
  - Börja tidigare i systemlivscykeln
    - Svårare att relatera en viss del av designen till kodsegment, pga
    - Tidiga modeller är ej precisa
  - Vara större konceptuella områden
- Krueger Skrev i en artikel 1995

  ”*For software reuse technique to be effective, it must reduce the cognitive distance between the initial concept of a system and its final executable implementation*”

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Definitioner

- Mjukvaruarkitektur
  - ”The structure of the components of a program/system, their inter-relationships, and principles and guidelines governing their design and evolution process” (Garland, Perry)
  - Liksom arkitekturen hos en byggnad tillhandahåller relationer mellan komponenter och begränsningar hos dessa relationer

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Mönster
  - Ett designmönster är en lösning till ett problem i ett visst kontext
  - Mönster skall fånga standardlösningar till gemensamma problem och där igenom
    - Hjälpa noviser att dra fördel av vad andra gjort
- Komponenter
  - En enhet som tillhandahåller en relativt oberoende del som används ihop andra delar i en konfiguration

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Krafter som påverkar
  - Att lyckas samla ett antal olika komponenter till en större generisk del bidrar till svårigheten att lokalisera och plocka fram den komponent som sökes
    - Det måste gå snabbare att hitta en komponent än det gör att skriva en ny
  - Noviser behöver riktlinjer för att konstruera designbaserade återanvändbara komponenter
    - Det räcker inte med att tillhandahålla en User Guide över återanvändbara komponenter
    - Det krävs förståelse för arkitektuella concept bakom komponenterna

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Återanvändning av mer sofistikerade komponenter som ramverk kommer att ge en hävstångseffekt
  - Längre tid att lära sig
  - Tid att hitta + tid att lära sig < tid att skriva en ny
- För att återanvända sofistikerade komponenter måste arkitekturen i applikationen konceptuellt vara i linje med arkitekturen i underliggande ramverk
  - Borlands Object Window Library har en arkitektur inom produkten (ramverket)

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Utbyggbarhet

Riktlinjer för att underlätta utbyggbarhet

- Inkapsling i klasser
  - Den interna strukturen skall vara osynlig för andra klasser
  - Bara klassens metoder skall komma åt implementationen
- Göm datastrukturer
  - Metoder skall ej exportera runt interna strukturer
- Undvik traversering av multipla länkar
  - Metoder bör endast ha kunskap om strukturen i ett steg
- Undvik case satser på objekt typer
  - Välj inte beteende beroende på typ av objekt

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Bestäm publika och privata operationer
  - Publika = gränssnitt
  - Varför denna klassificering
    - Användare skall ej behöva veta om den privata delen
    - Privata metoder beror på den interna implementationen. Detta medför att metoden kan ändras lokalt
    - Privata metoder kan endast anropas av klassens egna metoder och är beroende av dem
    - Privata metoder ökar modulariteten. Interna detaljer påverkar bara strukturer inom klassen

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Robusthet

En metod är robust om den inte kraschar då den får fel parametrar

Det är en avvägning mellan robusthet och effektivitet

Ge robusthet mot användare

- Skydd mot error
  - Felaktiga input får aldrig orsaka en krasch
- Optimera efter att programmet fungerar
- Validera argument
  - Publika metoder måste vara bättre på validering av argument då externa användare har större benägenhet att bryta mot restriktioner

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

- Undvik fördefinierade utrymmen
  - Använd dynamisk minnes allokering för alla datastrukturer och förekomster
- Utrusta programmet för debugging och prestanda test
  - Lägg in kontrollpunkter i programstrukturen
  - Kontrollpunkter bör ha väldefinierade tillstånd

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Programming Style

Programmering i team

Börja inte programmera förrän designfasen är klar

- Metoderna skall kunna förstås av andra
- Gör metoderna läsbara
- Använd samma namn som i objektmodellen
- Välj väl beskrivande namn
- Använd programmeringsriktlinjer inom org om de finns
- Paketera i moduler
- Publicera klasspecifikationerna