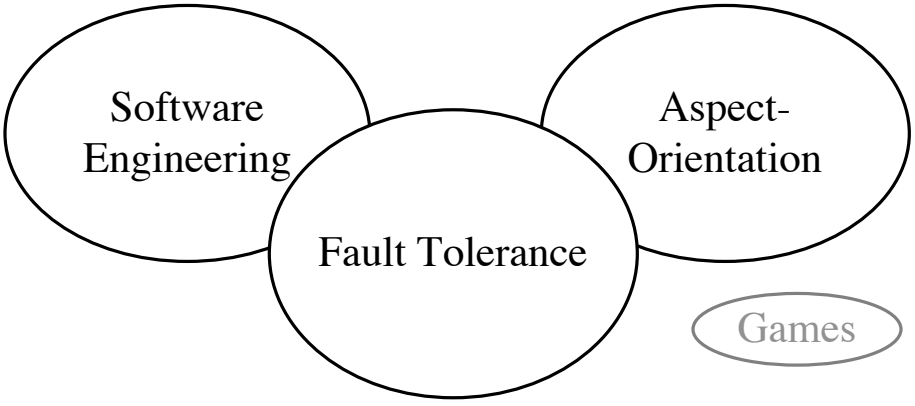
SAP Mini-Presentation

Software Engineering Lab
Research Overview

Jörg Kienzle
McGill University
Montreal, Canada

McGill

Research Areas (1)



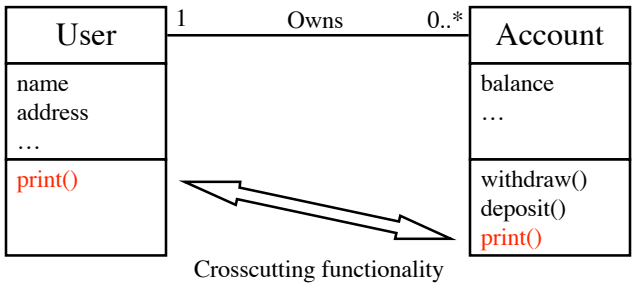
McGill

Aspect-Orientation (1)

- Object-Orientation
  - Decompose problem into a set of abstractions
  - Objects
    - Encapsulate *state* and *behavior*
    - Are assigned responsibilities
- "Tyranny of the dominant decomposition" [T+99]
- Result (at the programming language level):
  - Similar / identical code-fragments, all implementing some common functionality, are often scattered through the code

McGill

Aspect-Orientation (2)



Scattering and Tangling

McGill

Aspect-Oriented Programming (1)

- Modularize crosscutting concerns at the programming language level
- Decompose problem into *aspects*, encapsulating different *concerns* of the application [K+97]
- *Weave* aspects together for final product
- Weaving happens at so-called *joinpoints*
- Benefits: Simpler structure, improve readability, customizability and reuse

McGill

Aspect-Oriented Programming (2)

- Logging (orthogonal concern) without AOP

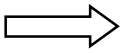
```
class Account {
   public Account() {};
   private int balance = 0;

   public void deposit(int amount) {
      balance += amount;
   }

   public void withdraw(int amount) {
      balance -= amount;
   }

   public int getBalance() {
      return balance;
   }
}
```

With Logging



```
class Account {
   public Account() {};
   private int balance = 0;

   public void deposit(int amount) {
      System.out.println("Call to deposit");
      balance += amount;
   }

   public void withdraw(int amount) {
      System.out.println("Call to withdraw");
      balance -= amount;
   }

   public int getBalance() {
      System.out.println("Call to getBalance");
      return balance;
   }
}
```

McGill

Aspect-Oriented Programming (3)

- Logging (orthogonal concern) with AOP

```
class Account {
   public Account() {};
   private int balance = 0;

   public void deposit(int amount) {
      balance += amount;
   }

   public void withdraw(int amount) {
      balance -= amount;
   }

   public int getBalance() {
      return balance;
   }
}
```

```
aspect Logging {

   pointcut AccountMethods() :
      call(public * Account+.*(..));

   before AccountMethods {
      System.out.println("Call to " + …);
   }
}
```

McGill

Aspect-Orientation & SE

- Aspect-Oriented Analysis
  Can AO ideas be applied to specifications written in UML and OCL?
- AOP & Reuse
  What features of an aspect-oriented language affect code reusability?
- Aspect-Oriented Modeling
- Aspect-Oriented Software Development
  - Is there a pure aspect-oriented way to go from requirements to implementation?

McGill

My Expertise in Aspect-Orientation

- Aspect-Orientation
  - 5 years of expertise (started in 2000)
    - First AOP paper 1997, first international conference 2002
  - Using AOP to modularize non-functional concerns like concurrency, distribution, fault-tolerance, security at the code level
  - Use AOP to build safe and secure APIs to (fault tolerance) libraries
  - Use of AOP to enforce programming conventions
  - Aspect-oriented modeling
    - Organizer of 7 (!) workshops on aspect-oriented modeling at AOSD and MoDELS conference

McGill

Fault Tolerance (1)

- How to develop applications that continue to provide service in spite of design faults or faults of the environment

McGill

Fault Tolerance & SE

- Integrating the concern of fault tolerance into the software development cycle
  - Determine the need for fault tolerance at the requirements level
  - Specify the computational model, the assumed faults and what parts of your system are critical at the analysis level
  - Choose an appropriate architecture and fault tolerance model during design
- Providing fault tolerance to the programmer (frameworks, aspect-orientation)
- Implementing fault tolerance models on top of COTS middleware

McGill

My Expertise in Fault Tolerance

- Fault Tolerance Implementation
  - Started 1997
  - Ph.D. on integrating transactions and concurrent object-oriented programming languages
    - Definition of a new transaction model *Open Multithreaded Transactions*, that can handle cooperative and competitive concurrency
    - Integrate transactions and exceptions -> forward and backward recovery
    - Control transaction participants -> provide consistency and detect deserters
    - Design of an object-oriented framework OPTIMA that provides support for Open Multithreaded Transactions
    - Definition of easy-to-use and safe APIs for programmers
    - Implementation of OPTIMA for Ada 95 and Java
- Reliable Software Development
  - Participated in the definition of a UML-based, model-driven development process (1999 - present)
  - Extension of use-case driven requirements elicitation to focus on exceptional behavior

McGill

Mondriaan Project

- (Meta) Modeling tool
- Describe your modeling needs and instantly get a tool that supports your formalism
- Enables experimentation
  - Domain-specific
  - Extensions to UML
  - Model Transformations (MDA)
  - Aspect-Oriented Modeling

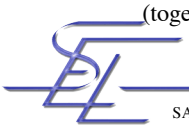
(together with Greg Dudek and Hans Vangheluwe)

McGill

Mammoth Project

- Massively Multiplayer Game Development Framework
- Scaling to 1000s of players
- Fault Tolerance
- Cheating
- Graphics / AI / Networking / Simulation / Content Creation / Software Engineering / Aspect-Orientation
- http://mammoth.cs.mcgill.ca

(together with Clark Verbrugge, Bettina Kemme and Hans Vangheluwe)



McGill

Potential Collaboration with SAP

- Aspect-Orientation
  - Can aspect-oriented ideas be applied to models of business workflows?
  - Can aspect-oriented ideas help for configuration / deployment of large software systems
- Meta-modeling (together with Hans Vangheluwe)
  - Can meta-modeling help producing domain-specific tools for SAP developers / users?
- Fault Tolerance
  - Investigate possible ways to incorporate fault tolerance into SAP applications

McGill