ASPECT ORIENTED SOFTWARE DEVELOPMENT
CS 791Z Special Topics on Software Engineering
Javier Garcia
April 23, 2013
A Concern is a particular set of information that has an effect on code

Examples of concerns

- Security
- Transactional behavior
- Logging
- Others
Cross-cutting concerns are parts of a program that affect other concerns.

Examples of cross-cutting concerns:
- Synchronization, caching, logging, etc.

Often cannot be cleanly decomposed from the rest of the system.
- Results in scattering or tangling.

Cross-cutting concerns form the basis for aspects.

Cross-cutting concerns do not fit cleanly into object-oriented programming or procedural programming.
ASPECT-ORIENTED SOFTWARE DEVELOPMENT

- Approach to software development that addresses modularization of software systems
- Separates out or isolates supporting functions from the program’s main business logic
- Concerns are addressed independently
- The dominant programming paradigm is object-oriented software development (OOSD)
  - AOSD is not meant to replace OOSD but instead to build upon and improve
  - AOSD supports separation of concerns that OOSD handles poorly
Example

```java
void transfer(Account fromAcc, Account toAcc, int amount) throws Exception {
    if (fromAcc.getBalance() < amount) {
        throw new InsufficientFundsException();
    }

    fromAcc.withdraw(amount);
    toAcc.deposit(amount);
}
```
Example Cont’d

```java
void transfer(Account fromAcc, Account toAcc, int amount, User user, Logger logger)
    throws Exception {
        logger.info("Transferring money... ");
        if (!checkUserPermission(user)) {
            logger.info("User has no permission.");
            throw new UnauthorizedUserException();
        }
        if (fromAcc.getBalance() < amount) {
            logger.info("Insufficient funds.");
            throw new InsufficientFundsException();
        }
        fromAcc.withdraw(amount);
        toAcc.deposit(amount);
        // get database connection
        // save transactions
        logger.info("Successful transaction.");
    }
```
EXAMPLE CONT’D

```java
aspect Logger {

    void Bank.transfer(Account fromAcc, Account toAcc, int amount, User user, Logger logger) {
        logger.info("Transferring money...";
    }

    void Bank.getMoneyBack(User user, int transactionId, Logger logger) {
        logger.info("User requested money back.");
    }

    // other crosscutting code...
}
```
RESOURCES

- **AOSD.net** contains much information on the topic, it is also a great starting point because it links to separate resources.

- AspectJ is an AOP extension for Java.

- Since its release in 2001, it quickly became the de facto standard for AOP.

- Integrated into the Eclipse IDE
  - Displays the cross-cutting structure.

- Many programming languages have implemented AOP either within the language or as a separate library.
CHALLENGES

- Visualizing cross-cutting concerns
  - Difficult even for developers with training on the subject
  - Several tools have been created to attempt to alleviate the issue

- Logical errors
  - Possible to make a mistake in cross-cutting that can have widespread errors
  - Can happen if the developer makes a change that the aspect writer did not anticipate
CONCLUSION

- AOSD has both strengths and weaknesses
- Not meant to replace other programming paradigms
TEST QUESTIONS

- Describe the concepts and ideas behind aspect oriented software development.

- Describe a simple application that you would write using AOSD and briefly explain why you chose this particular application.
REFERENCES


REFERENCES CONT’D


