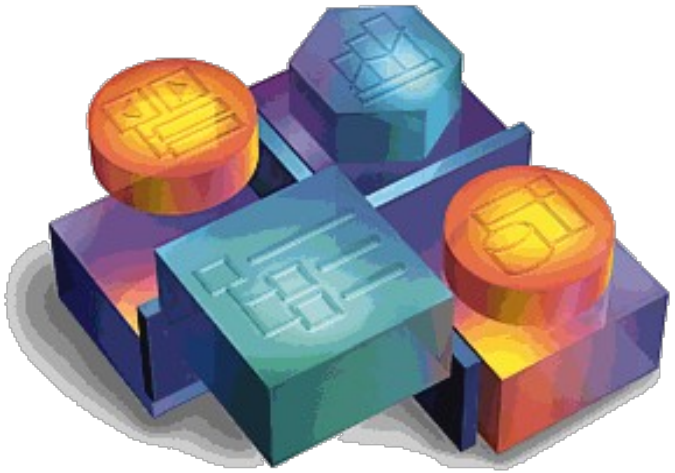
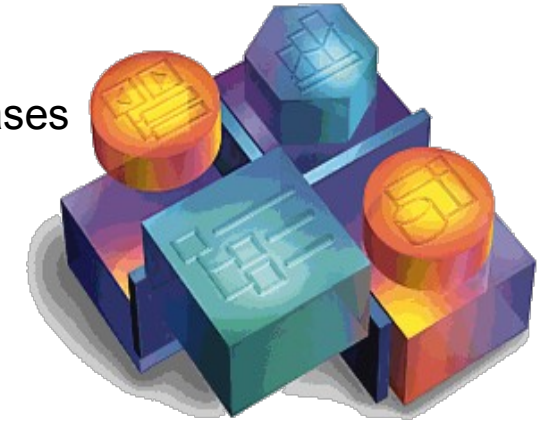
Object-Oriented Programming Design



1. Introduction to Object-Oriented Concepts
2. How to think in Terms of Objects
3. Advanced Object-Oriented Concepts
4. The anatomy of a Class
5. Class Design guidelines

6. Designing with objects

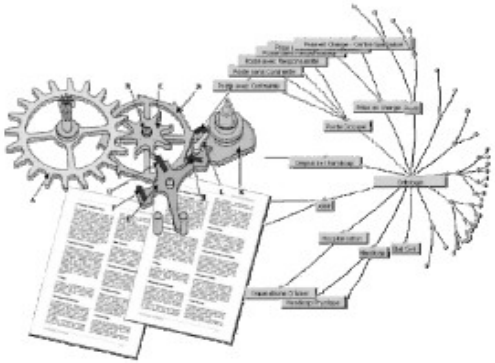
7. Mastering Inheritance and Composition
8. Frameworks and Reuse: Designing with interfaces and Abstract classes
9. Building objects
10. Creating Object Models with UML
11. Objects and Portable Data: XML
12. Persistent Objects: Serialization and Relational Databases
13. Objects and the Internet
14. Objects and Client/Server Aplications
15. Design Patterns



6. Designing with objects

Design Guidelines

- ►There is no right or wrong way to create a design. Many organizations do not follow a standard software development process.
- ►The most important factor in creating a good design is to find a process that you and your organization can feel confortable with.



6. Designing with objects

Design Guidelines

A solid OO design process will include the following steps:

1. Doing the proper analysis
2. Developing a statement of work that describes the system
3. Gathering the requirements from this statement of work
4. Developing a prototype for the user interface
5. Identifying the classes
6. Determining the responsibilities of each class
7. Determining how the various classes interact with each other
8. Creating a high-level model that describes the system to be built (UML)

6. Designing with objects

Design Guidelines, the Ongoing Design Process

- Despite the best intentions and planning, in all but the most trivial cases, the design is an **ongoing process**.

- Even after a product is in testing, design changes will pop up. It is up to the **project manager** to draw the line that says when to stop changing a product and adding features.

6. Designing with objects

Design Guidelines

- It's important to understand that many design methodologies are available.

  One early methodology, called the **waterfall model.** In this case, the design phase is completed before the testing phase. In practice, the waterfall model has been found to be unrealistic.

- Currently there are other design models, such as rapid prototyping, that promote a true iterative process.

6. Designing with objects

Design Guidelines, the waterfall method:

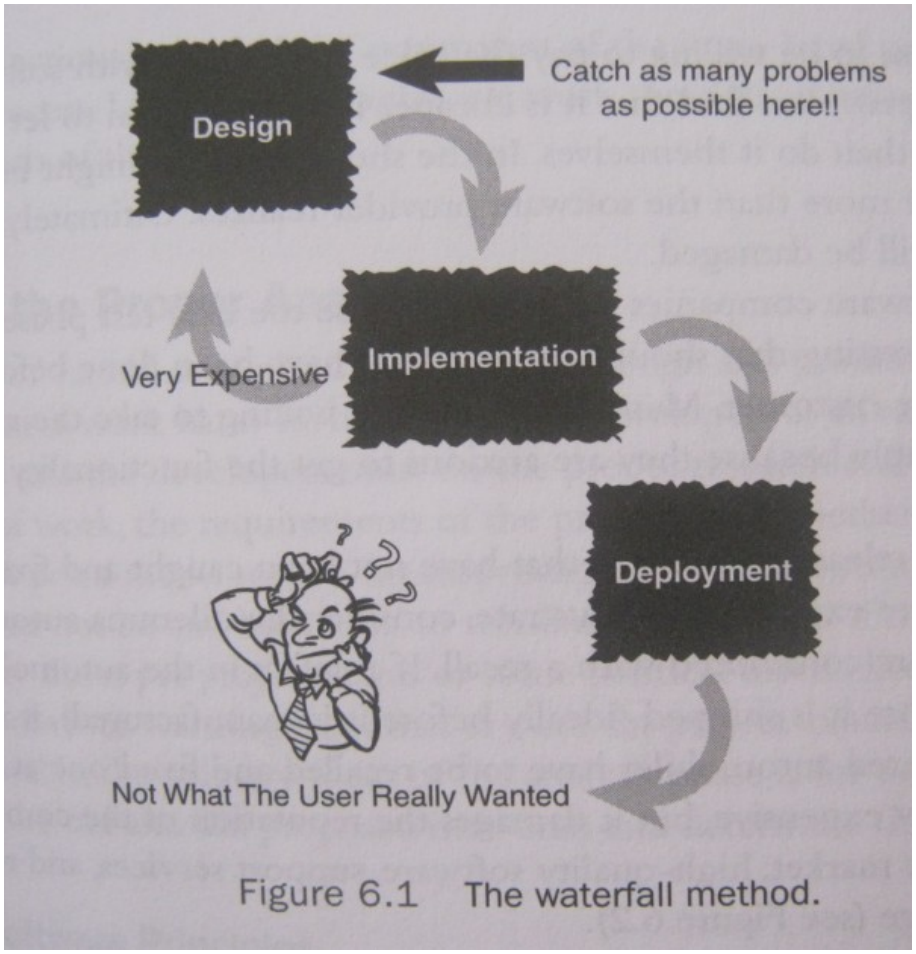


Figure 6.1 The waterfall method.

►The cost of a requirement/ design change in the design phase is relatively small

►The cost of a design change in the implementation phase is significantly higher

►The cost of a design change after the deployment phase is astronomical when compared to first item

6. Designing with objects

Design Guidelines

- Some software providers find that is cheaper in the long run to let the customers test the product rather than do it themselves. In the short term this might be true, but in the long run it costs far more than the software provider realizes. Ultimately, the software provider's reputation will be damaged.

6. Designing with objects

Design Guidelines

- Some computer software companies are willing to use the beta test phase to let the customers do testing. Many customers are willing to take the risk of using pre-release software simply because they are anxious to get the functionality the product promises.

6. Designing with objects

Design Guidelines

- After the software is released, problems that have not been caught and fixed prior to release become much more expensive.
- To illustrate, consider the dilemma automobile companies face when they are confronted with a recall. If a defect in the automobile is identfied and fixed before it is manufactured, it is much cheaper than if all delivered automobiles have to be recalled and fixed one at a time.

  Not only is this scenario very expensive, but it damages the reputation of the company.

6. Designing with objects

Design Guidelines

►In a increasingly competitive market, high-quality software, support services, and reputation are the **competitive advantage**.

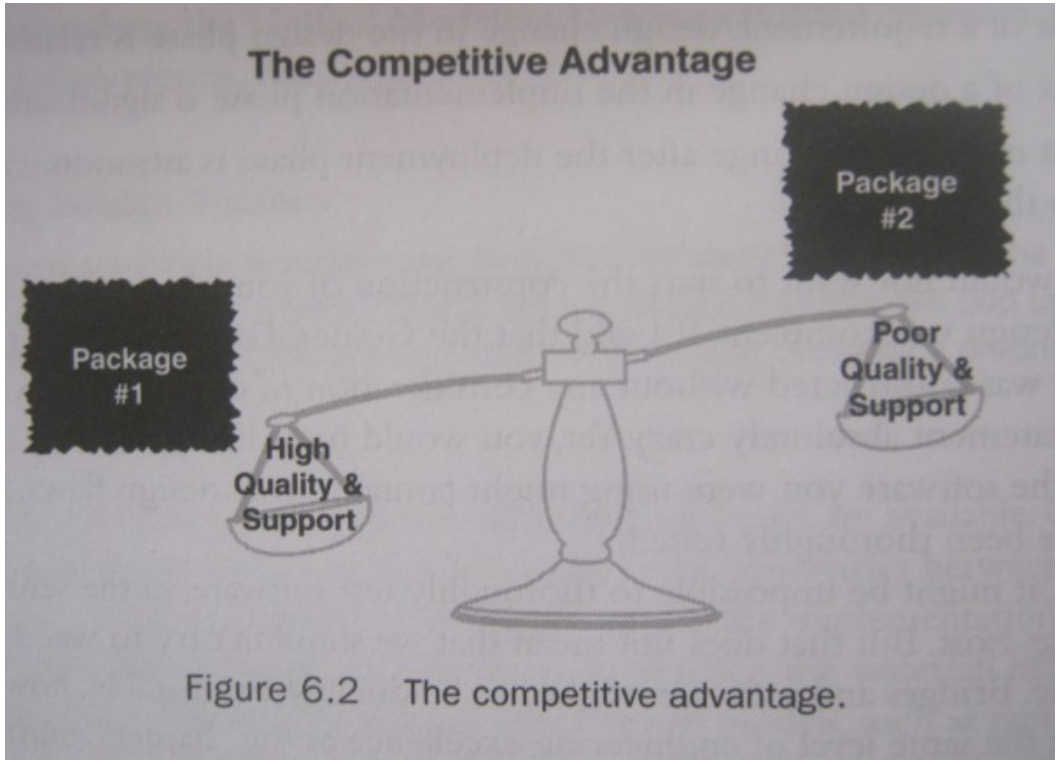


Figure 6.2 The competitive advantage.

6. Designing with objects

Design Guidelines

A solid OO design process will include the following steps:

1. Doing the proper analysis
2. Developing a statement of work that describes the system
3. Gathering the requirements from this statement of work
4. Developing a prototype for the user interface
5. Identifying the classes
6. Determining the responsibilities of each class
7. Determining how the various classes interact with each other
8. Creating a high-level model that describes the system to be built (UML)

6. Designing with objects

Design Guidelines

1. Performing the proper analysis

The users must work hand-in-hand with the developers at all stages. In the analysis phase, the users and developers must do proper research and analysis to determine the statement of work, the requirements of the project.

►The primary focus of the analysis phase is for everyone to learn the systems and determine the system requirements.

6. Designing with objects

Design Guidelines

2. Developing a Statement of Work

The statement of work (SOW) is a document that describes the system. The SOW should give anyone who reads it a complete understanding of the system. The SOW must reprensent the complete system and be clear about how the system will look and feel.

6. Designing with objects

Design Guidelines

2. Developing a Statement of Work

The SOW contains everything that must be known about the system. Many customers create a request-for-proposal (RFP) for distribution, which is similar to the SOW. A customer creates a RFP that completely describes the system they want built.

6. Designing with objects

Design Guidelines

3. Gathering the Requirements

The requirements document describes what the users want the system do.

- ►The level detail of the requirements document does not need to be of highly technical nature.
- ►The requirements must be specific enough to reprensent the true nature of the user's needs for the end product.

6. Designing with objects

Design Guidelines

3. Gathering the Requirements

- ►The requirements are usually represented as a list of items.
- ►Each item represents one specific requirement of the system.
- ►The requirements are the final representation of the system must be implemented.
- ►All future documents in the software development process will be based on the requirements.

6. Designing with objects

Design Guidelines

4. Developing a Prototype of the User Interface

One the best way ways to make sure users and developers understand the system is to create a ***prototype***.

► By creating actual screens and screen flows, it is easier for people to get an idea of what they will be working with and the system will fill like.

►Traditionnaly, Visual Basic.Net is a good envirionnement for prototyping.

►Having a good prototype help a lot when identifying classes.

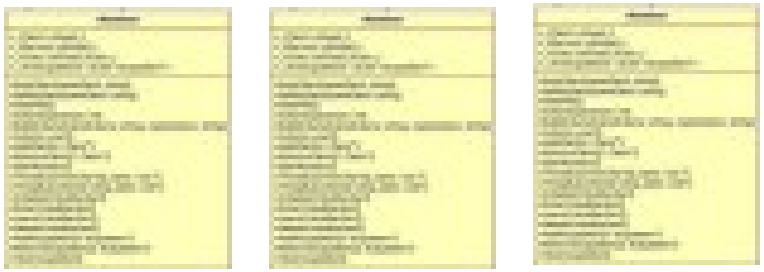
6. Designing with objects

Design Guidelines

5. Identifying the Classes

After the requirements are documented,
the process of identifying classes can begin.

►Identify Classes from the requirements.

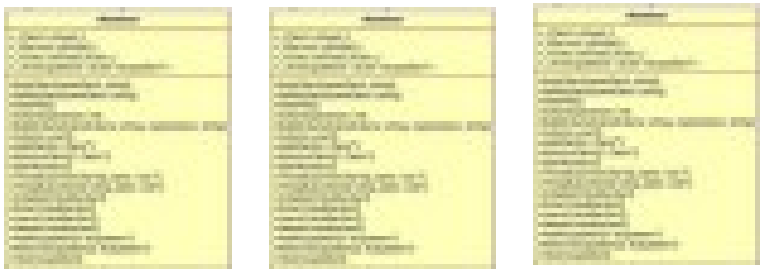


6. Designing with objects

Design Guidelines

6. Determining the Responsibilities of Each Class

You need to determine the responsibilities of each class you identified. This includes the data that the class must store and what operations the class must perform.

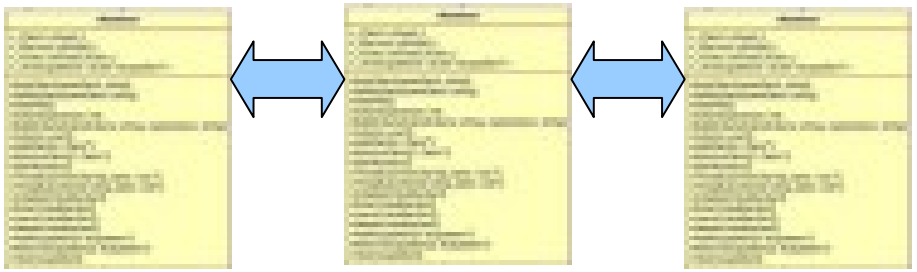


6. Designing with objects

Design Guidelines

7. Determining How the Classes Collaborate with each Other

Most classes do not exist in isolation. One class can send a message to another class when it needs information from that class, or if it wants the other class to do something for it.



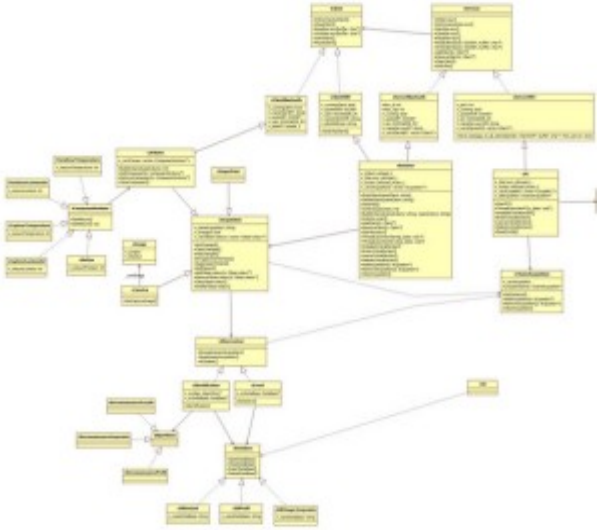
6. Designing with objects

Design Guidelines

8. Creating a Class Model to describe the System

When all classes are determined and the class responsibilities and collaborations are listed, a class model that represents the complete system can be constructed.

►We can use UML to model the system



1. Introduction to Object-Oriented Concepts
2. How to think in Terms of Objects
3. Advanced Object-Oriented Concepts
4. The anatomy of a Class
5. Class Design guidelines
6. Designing with objects

7. Mastering Inheritance and Composition

8. Frameworks and Reuse: Designing with interfaces and Abstract classes
9. Building objects
10. Creating Object Models with UML
11. Objects and Portable Data: XML
12. Persistent Objects: Serialization and Relational Databases
13. Objects and the Internet
14. Objects and Client/Server Aplications
15. Design Patterns