Application logic design
What to do?

• Detailed class model
  – class model completion
  – class model refinement
• Fitting application logic to system architecture
• Joining classes, interfaces and components
• Component organization
Class model completion

• Container class addition
  – entity management (adding, ordering, remove)
  – cross-entity data validation
  – aggregation

• Class hierarchy complement (find missing classes)
  – generalization
  – specialization
Class model refinement

• Entity classes
• Entity identification
  – int or GUID?
• Properties or fields (attributes)
  – readonly or read-write?
  – write-only? write-once?
  – const?
  – derived properties – expressions
• Feature visibility
  – private, protected, public
  – internal? friend?
  – class, static
Properties refinement

- General types (integer, float)
- Specific types (int32/int64, single/double)
- Fixed-point real numbers (currency, decimal)
- Date and time?
- Collections (multi-value properties)
  - multiplicity: [*], [n..*]
  - capacity – does not guarantee minimal item count
  - list, dictionary (indexed access)
  - vector list or linked list (access / modification efficiency)
Operations refinement

• Operations vs. functions (<<realize>> relationship)

• Function parameters
  – in, out, inout (ref)
  – default values
  – overloaded functions

• Result types
  – two or more result values – out parameters needed
Operation kind

- regular
- virtual
- abstract
- class, static
- new, override
Application logic & system architecture

• Single application logic model?
• Dividing logic to layers
• Complexity management (packages)
Interfaces (as data type)

• Multiple inheritance and interfaces
  – multiple inheritance only in few languages (e.g. C++)
  – multiple class parents implemented as interfaces
  – each interface must be implemented in each class separately
  – implementation via delegation
Classes – interfaces – components

- Classes at the lower infrastructure layer
- Components at the higher layer
- Implemented component = library (DLL)
- Components joined through interfaces
- Interface implementation in a class
- At least one implementing class in a component
Assigning classes to components

• Component “Orders”
  – class “Order Manager”
    • implement interface “Orders”
  – class “Order”
  – class “Article”  →  to component “Storage”
  – class “Customer”  →  to component “Contractors”
    • a new component is needed!
Better component diagram

- Orders
- Sales
- Accounting
- Contractors
- Storage
- Invoices
Bibliography

• Roger S. Pressman: Software Engineering. A Practitioner's Approach (book, PDF)
• Dennis A., Wixom B.H., Tegarden D., Systems Analysis & Design. An Object-Oriented Approach with UML, John Wiley and Sons, USA, 2002