[CS3704] Software Engineering

Shawal Khalid
Virginia Tech
03/20/2024

Announcements

- **HW3 due March 22nd at 11:59pm**

High-Level Design II

Package Diagrams
Database Design
Intro to Design Patterns

Learning Outcomes

By the end of the course, students should be able to:

- **Understand software engineering processes, methods, and tools used in the software development life cycle (SDLC)**
- Use techniques and processes to create and analyze requirements for an application
- **Use techniques and processes to design a software system**
- Identify processes, methods, and tools related to phases of the SDLC
- Explain the differences between software engineering processes
- Discuss research questions and current topics related to software engineering
- Create and communicate about the requirements and design of a software application

Warm-Up

TODO: Complete a stand-up meeting!

- **What I did.**
- **What I need to do next.**
- **What is blocking me.**

**** Share about progress since last standup meeting, standing is optional.***

How to Do Architecture Design?

When decomposing a system into subsystems, take into consideration:

- how subsystems share data
  – data-centric or data-distributed
- how control flows between subsystems
  – as scheduled or event-driven
- how they interact with each other
  – via data or via method calls

Architecture Modeling

- To organize architectural elements and diagrams into groups

UML Package Diagrams

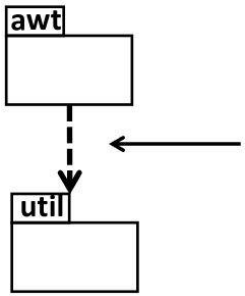
- To show packages and dependencies between the packages
- Can illustrate layered architecture
  – A layer, such as UI layer, can be modeled as a package named UI
  – Depicts relations between packages that make up a model

Example with JDK Packages

Package: a general purpose mechanism to group together semantically related elements.

Class: a member of the package. It can be represented as a brief, detailed class diagram, or simply text.

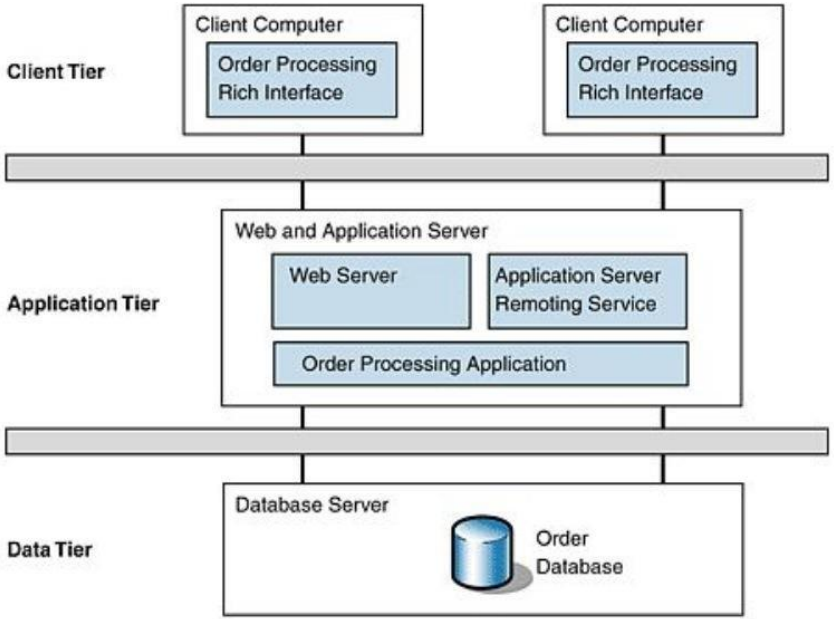
- *Members of a package can be classes or other packages.*



Dependency: to show “use” relationship

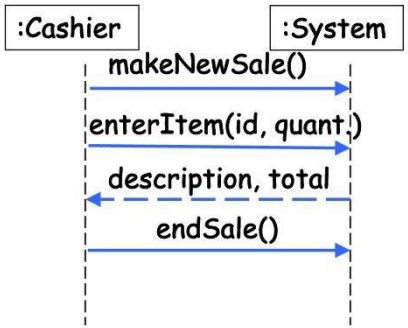
Case Study: Ordering System

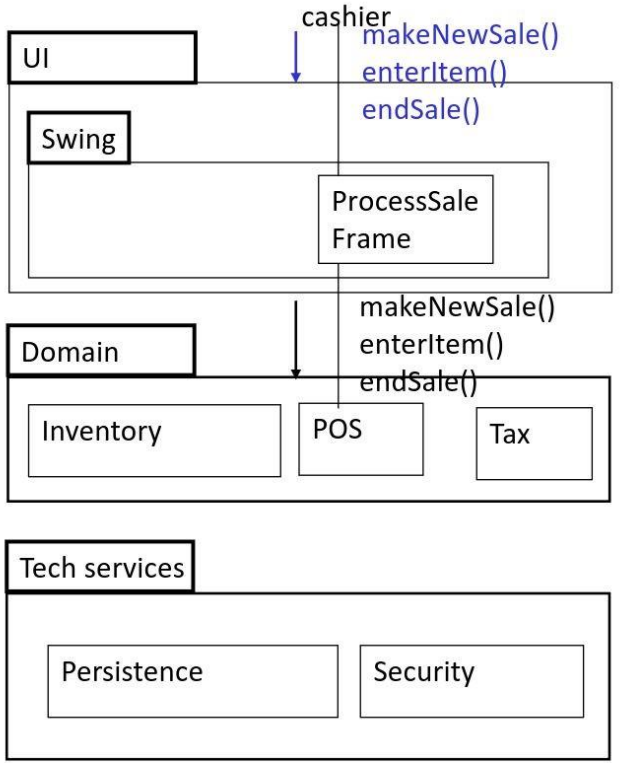
- 3-layer architecture
  - User Interface
  - Application logic
    - Software objects representing domain-specific concepts (i.e. Sale)
  - Technical Services
    - General-purpose objects and subsystems that provide supporting services, such as interfacing with database or error logging
    - Usually application-independent and reusable across systems
    - Does ***not*** include the modeling of data!



Case Study: Ordering System (cont.)

Example: Messages illustrated in system sequence diagrams can correspond to messages sent from the UI layer to the domain layer.





Database Design

Modern software is collecting and processing increasing amounts of data (data-centric).

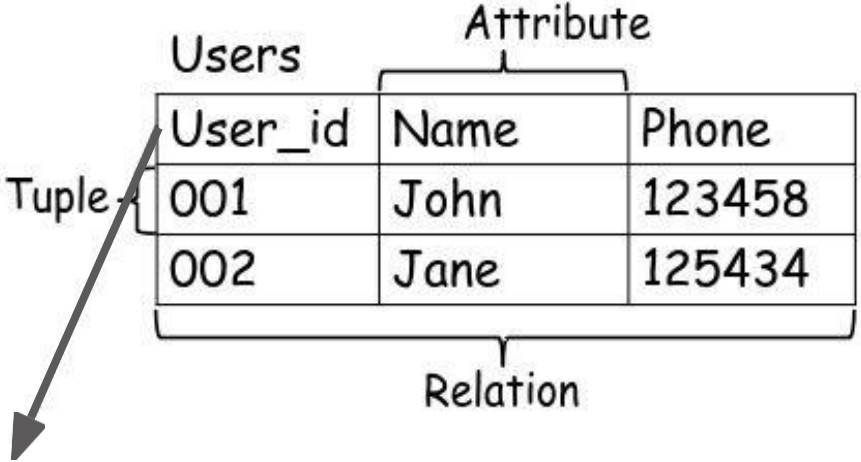
- What is a database?
  - A system that stores data, and lets you create, read, update, and delete the data
    - Ex) files, spreadsheets, XML, relational, noSQL,...
- Why use databases?
  - Every non-trivial application uses databases to keep program states and to store manipulate, and retrieve data
  - Databases plays a critical role in applications
    – Corrupted data => execution failure
    – Poor data organization => poor performance
  - A poorly designed database allows developers and users to put in arbitrary data (i.e. “none” as a phone number) *or access data without authorization!*

Relational Databases

A digital database with a collection of tables.

- Each table contains rows and columns, with a unique key for each row
- Each entity type described in a database has its own table
  - E.g., “Employee”, “Item”, “Order”
- Each row represents an instance of the entity
  - E.g., “John Jenny”, “Soap”
- Each column represents an attribute
  - E.g., “phone number”, “price”

Relational Databases (cont.)

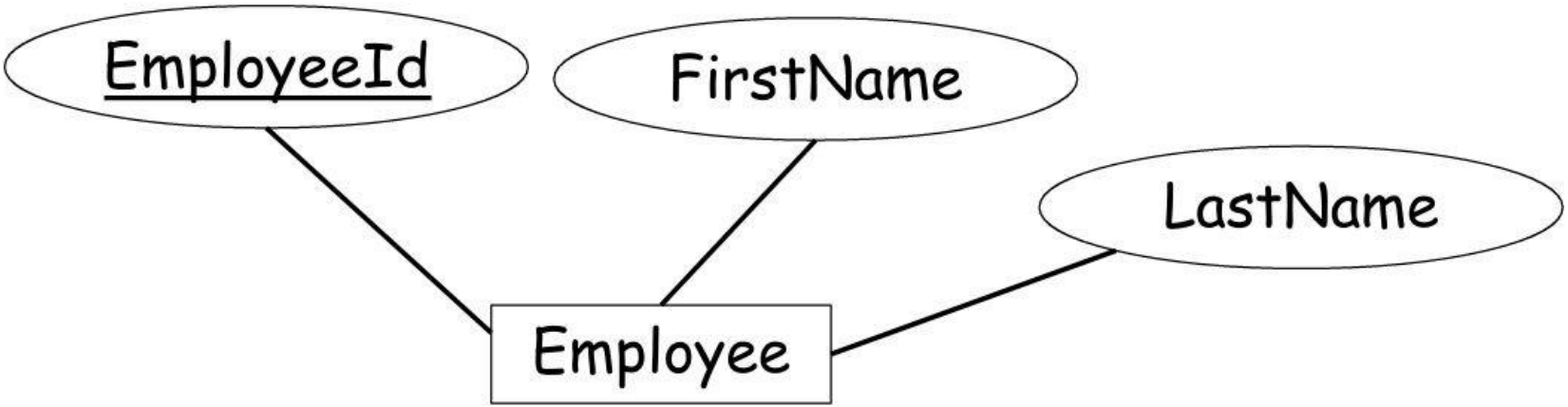


Primary Key/Unique Key: to uniquely specify a tuple in a table

Foreign Key: an attribute in a relational table that matches the primary key column of another table. It can be used to cross-reference tables.

Entities and Attributes

- An entity is similar to a semantic object
- It includes attributes that describe the object

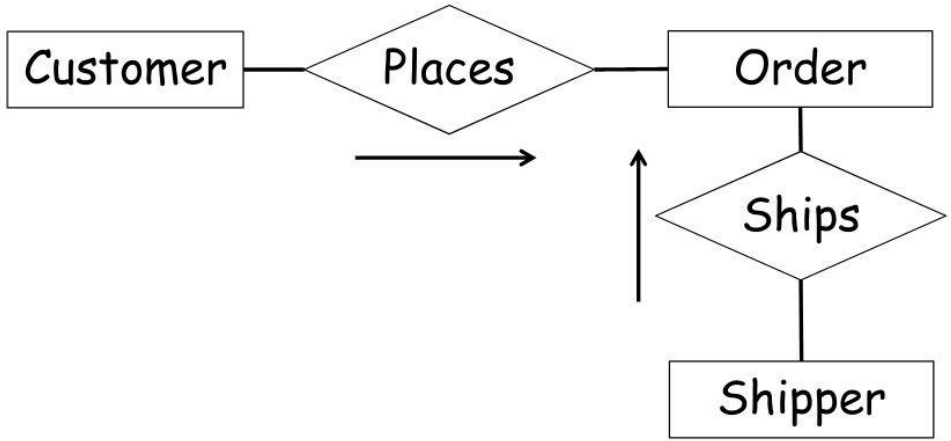


Entity-Relationship Models

- Entity-relationship (ER) diagrams are similar to semantic object modelings (i.e., class diagrams)
- They use different notations
- Focus is more on relations and less on class structure

Relationships

- An ER diagram indicates a relationship between entities with a diamond
- Sometimes arrows are added to indicate direction of relationship

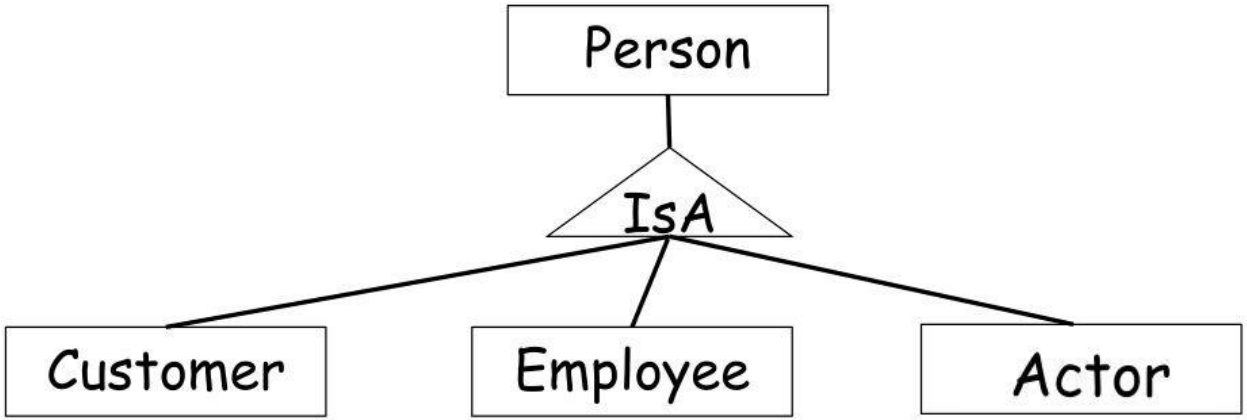


Cardinality

- Numbers used to describe relationship quantitatively.

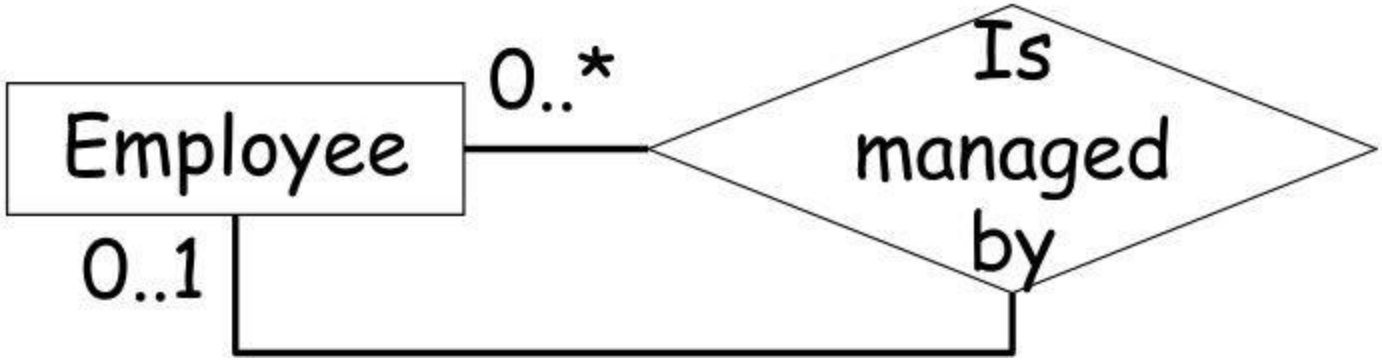
Inheritance

- A triangle named “IsA” represents the inheritance relationship.



Reflexive Associations

- An object refers to an object of the same class.



Mapping Class Diagrams to Tables

Can often map content of class diagrams to ER diagrams to show relationships between data.

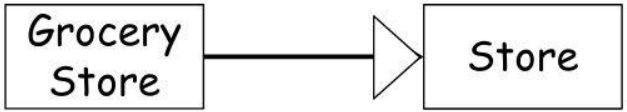
- Does not work for other classes
- Sometimes you need to explicitly add a primary key to distinguish data in tables
- Database management systems (DBMSs) usually provides functionality to automatically increment primary key

Reminder: Class Diagram Syntax

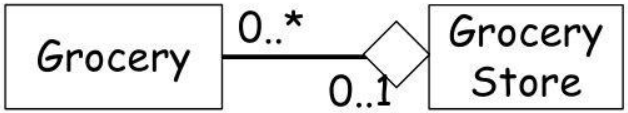
Class name: abstract concept

Attributes: properties relevant to the use case

Operation (Method signatures): behaviors of the class

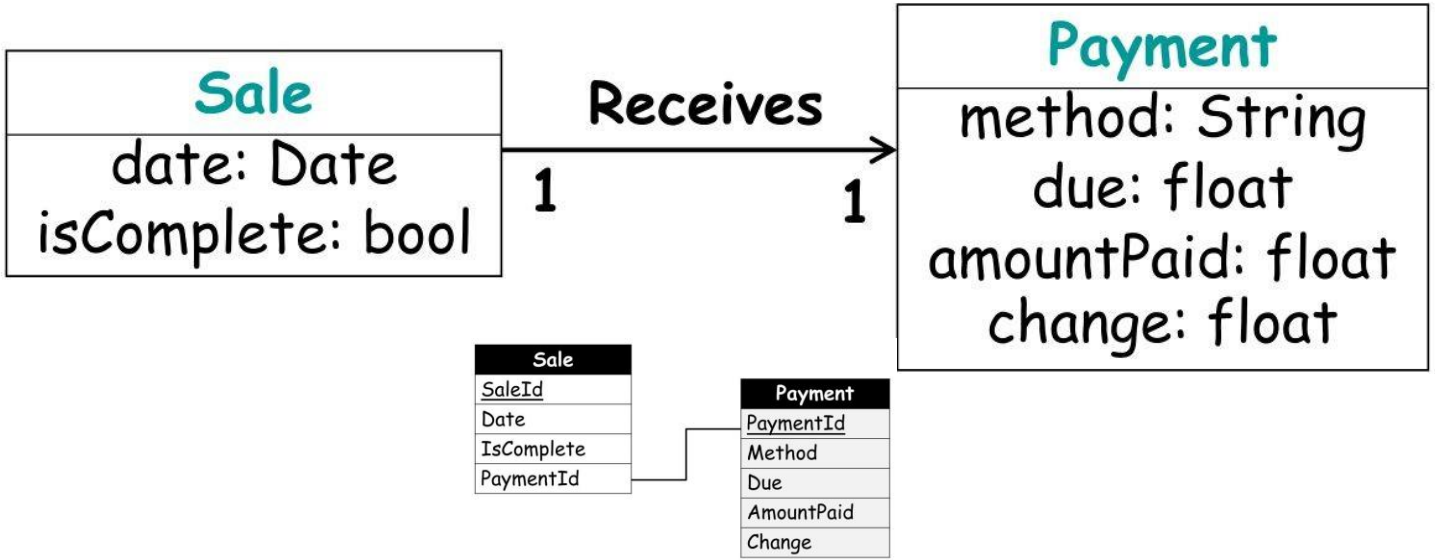


Generalization: "is-a" relationship. A sub-class inherits all attributes and operations of its super class.

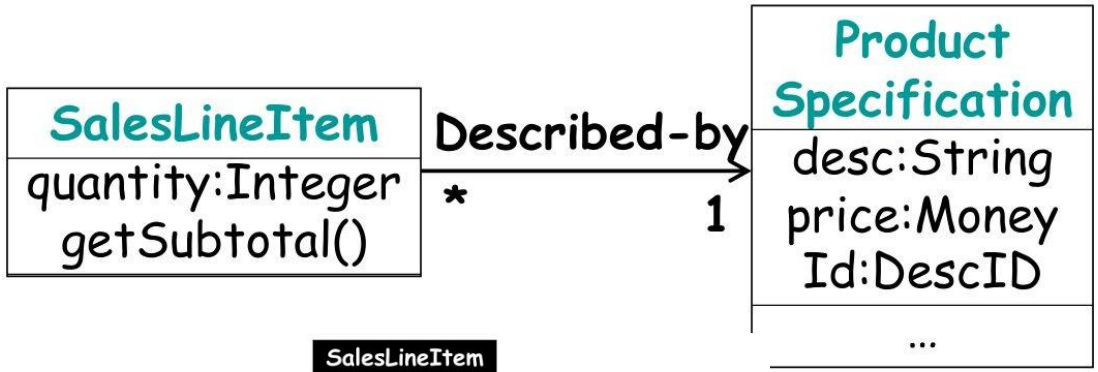


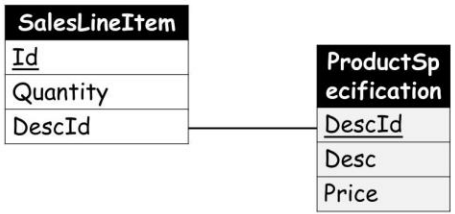
Aggregation: "has-a" relationship. The container and elements can exist independently from each other

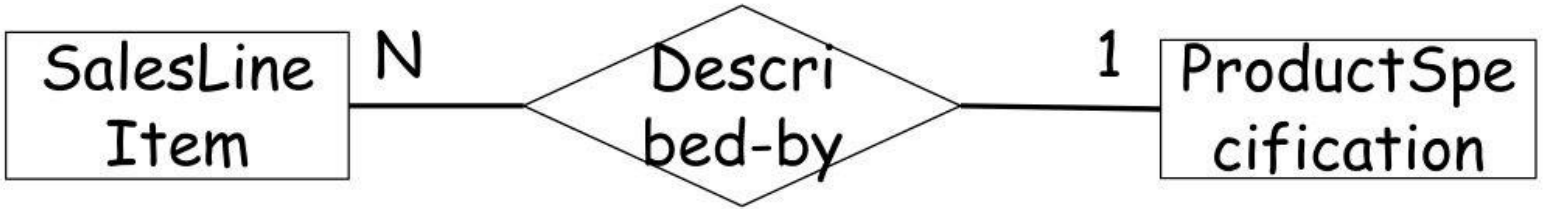
One-to-One Associations



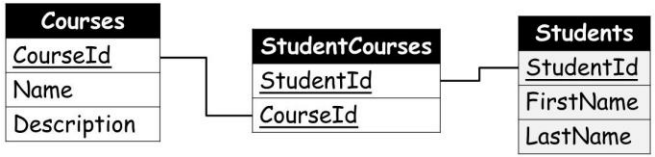
One-to-Many Associations





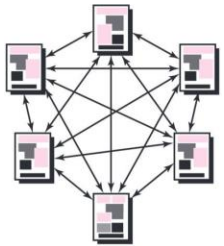


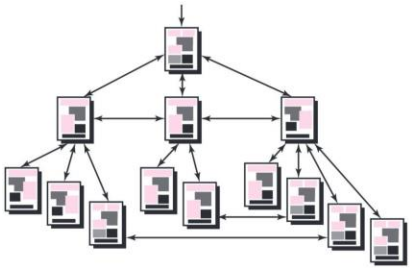
Many-to-Many Associations

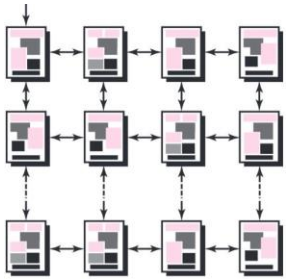


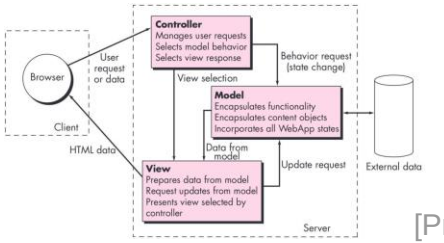
A brief digression on web app design

- What is a web app?
  - A program that uses a web browser to perform specific functions.
- *“There are essentially two basic approaches to [web] design: the artistic ideal of expressing yourself and the engineering ideal of solving a problem for a customer”* [Nielsen]
- Aesthetics, layout, graphic design, content, navigation,…









[Pressman]

…mobile app design

- What is a mobile app?
  - A program that uses a mobile device to perform specific functions.
- Still concerned with aesthetics, layout, graphic design, content, navigation,…
- And multiple hardware and software platforms!
  - Smartphones, tablets, wearable devices, etc.
  - Android, iOS, Blackberry, Windows, etc.
  - App stores have different rules
  - More complex interactions
  - Power and space/storage management
  - Security and privacy

Next Time…

- Design Pattern Workshop on Friday (03/22)

- **HW3 due March 22 at 11:59pm)**

References

- RS Pressman. *“Software engineering: a practitioner's approach”.*
- Cast Software *“What is Software Architecture?”*. <https://www.castsoftware.com/glossary/what-is-software-architecture-tools-design-definition-explanation-best>
- K.D. Cooper, L. Torczon, *“Engineering a Compiler”.* Theo Mandel. *“Golden Rules of User Interface Design”.* <https://theomandel.com/resources/golden-rules-of-user-interface-design/>
- <https://medium.com/swlh/ordering-food-and-the-mvc-architecture-d5cbf3859d60>
- Na Meng and Barbara Ryder
- Chris Parnin
- Sarah Heckman