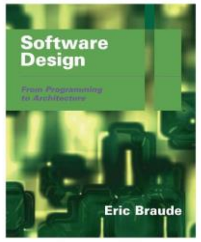
Lec 09 - Frameworks

Framework

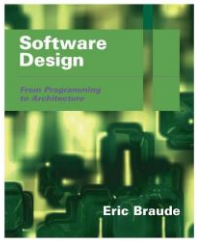
- ✧ Framework --- a collection of classes applicable to multiple applications
- ✧ Business logic: the parts that are particular to an application and are often not reused.
- ✧ Reusability can be at the architecture level, design level, and code level

The meaning and usage of frameworks



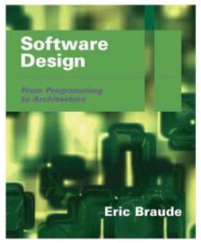
- A framework is a collection of software artifacts that is usable by several different applications
- Classes within a framework may be related and applications may use them by means of inheritance, aggregation or dependency.
- E.g.
  - Java API's (3D, 2D, Swing)
  - The Enterprise Java Bean (EJB)

Framework definitions



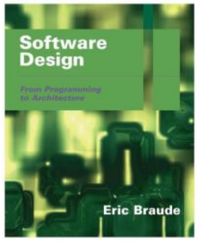
- ✧ A framework is a set of cooperating classes that comprise a reusable backbone for a specific application domain.
  - A framework provides an integrated set of domain-specific functionality
- ✧ A framework is a semi-complete application
  - A framework is to be customized to a particular application by deriving new application specific classes from abstract classes defined by the framework.

Why frameworks



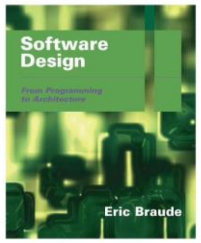
- Frameworks are the key to OO reuse
- Frameworks reuse analysis and design, not just code.
- Frameworks reuse overall architecture, not just components.

Framework challenges



- ✧ Developing high quality reusable extensible frameworks for application domains involves complicated tasks.
- ✧ It takes a substantial amount of time and effort to learn how to utilize OO application frameworks.
- ✧ Requirements of frameworks often change with changing application requirements.
- ✧ Frameworks need maintenance including both adaptation and modification.
- ✧ No current common standards exist for the design, implementation, documentation, and adaptation of frameworks.

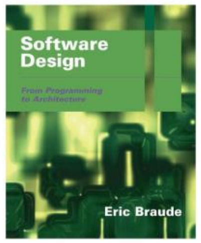
Frameworks vs. components



- ✧ Frameworks have lots of room for customization and they have interfaces that are more complex than those of many other components.

- ✧ Frameworks and other software components are different but cooperating technologies.
  - Frameworks are built from building blocks, which are their component classes.
  - At the same time, frameworks themselves are application building blocks and are used to build larger systems.

Framework Usages



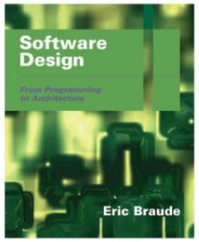
1- Toolkit

- "Bag of tools" with no architectural assumption, e.g. java.math
  - A collection of useful classes
  - Each usable in isolation
  - (Sometime not called a framework)

2- Abstract architecture

- Common denominator of architecture, with architectural assumption, e.g. java.awt and container class
  - An architecture in the abstract

The **Abstract Window Toolkit** (**AWT**) is Java's original platform-dependent windowing, graphics, and user-interface widget toolkit preceding Swing.



Selected Framework Goals

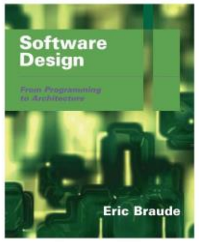
- *Persistence Service*
  - Store and retrieve instances between executions
- *Identity Service*
  - Identify objects sufficiently to retrieve them across executions
- *Pooling (Sharing)*
  - - of objects: Reusing objects at runtime to save time and space
  - - of threads
  - - of database connections
- *Security*

KEY CONCEPT

Goal : Reuse

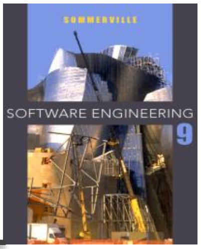
We Use Frameworks …to reuse classes, relationships among classes, or pre-programmed control.

Selected Framework Goals



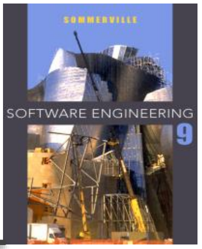
- Persistence Service
  - objects that are created have to be persisted (persevered) in a data store so that these objects and their associated properties can be retrieved for future reference and manipulated with ease. Web NMS Persistence Services allow Java objects to be stored in a relational database, maintain the relationship between these objects and provide a mapping between these objects and their relational counterparts. This service uses **Hibernate**, a powerful, high performance object/relational persistence and query service.

Application frameworks



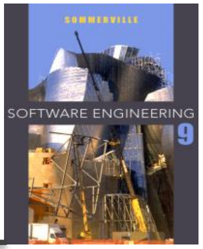
- ✧ Frameworks are moderately large entities that can be reused. They are somewhere between system and component reuse.
- ✧ Frameworks are a sub-system design made up of a collection of abstract and concrete classes and the interfaces between them.
- ✧ The sub-system is implemented by adding components to fill in parts of the design and by instantiating the abstract classes in the framework.

Framework classes



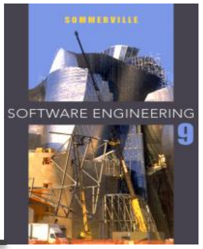
- System infrastructure frameworks
  - Support the development of system infrastructures such as communications, user interfaces and compilers.
- Middleware integration frameworks
  - Standards and classes that support component communication and information exchange.
- Enterprise application frameworks
  - Support the development of specific types of application such as telecommunications or financial systems.

Web application frameworks



- ✧ Support the construction of dynamic websites as a front-end for web applications.
- ✧ WAFs are now available for all of the commonly used web programming languages e.g. Java, Python, Ruby, etc.
- ✧ Interaction model is based on the Model-View-Controller composite pattern.

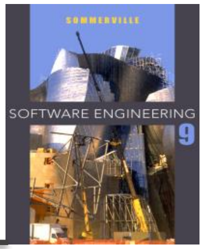
Model-view controller

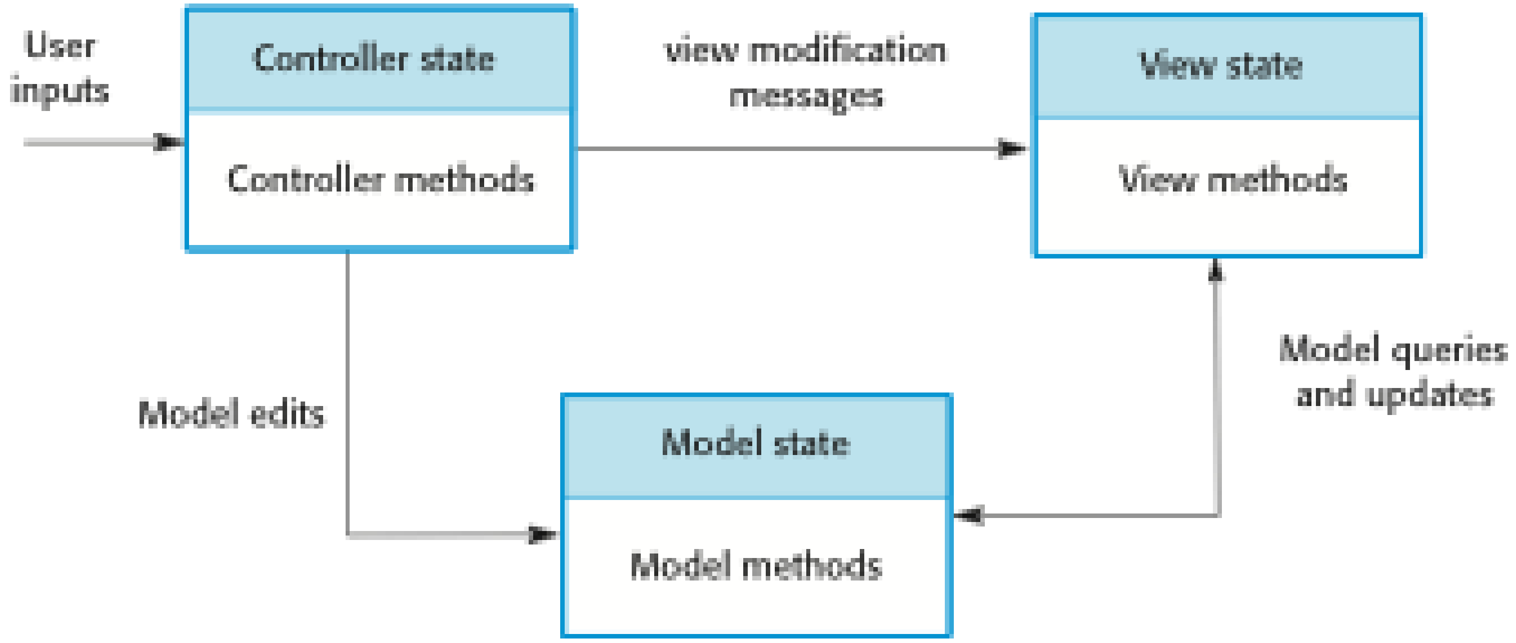


✧ System infrastructure framework for GUI design.

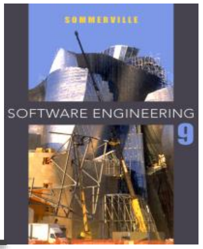
✧ Allows for multiple presentations of an object and separate interactions with these presentations.

The Model-View-Controller pattern



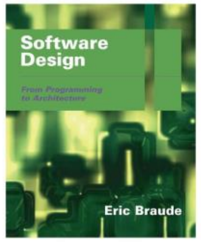


WAF features



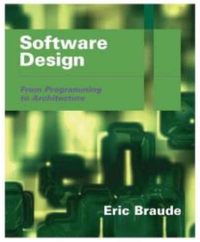
- *Security*
  - WAFs may include classes to help implement user authentication (login) and access.
- *Dynamic web pages*
  - Classes are provided to help you define web page templates and to populate these dynamically from the system database.
- *Database support*
  - The framework may provide classes that provide an abstract interface to different databases.
- *Session management*
  - Classes to create and manage sessions (a number of interactions with the system by a user) are usually part of a WAF.
- *User interaction*
  - Most web frameworks now provide AJAX support (Holdener, 2008), which allows more interactive web pages to be created.

Framework design Issues

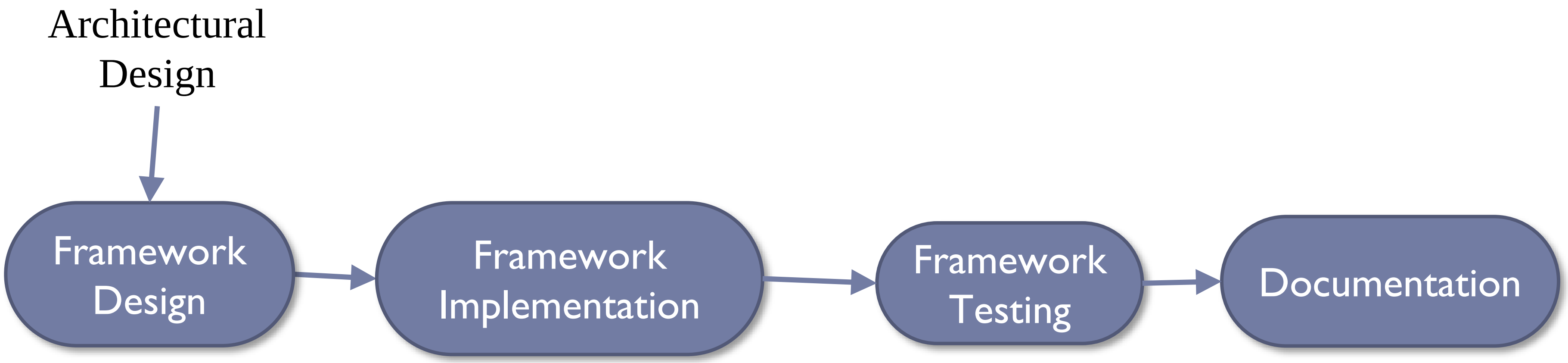


- Keep it simple
- Be consistent
- Don’t make things almost the same
- Design to prevent user errors
- Have default behavior (good when way out in exceptional cases)
- Have documentation especially for interactions

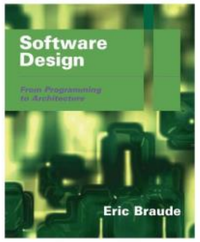
Framework development



- ✧ Framework Design
- ✧ Framework Implementation
- ✧ Framework Testing
- ✧ Documentation

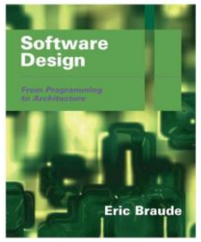


Principles of framework design

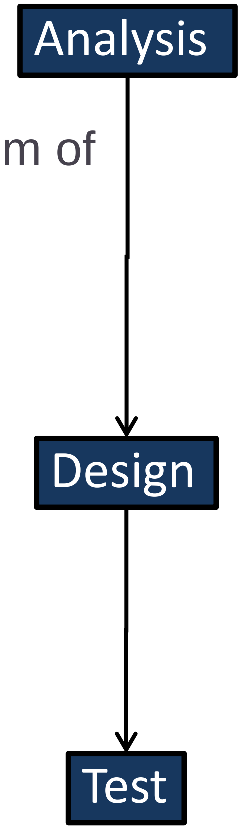


- First rule:
  - Don't design, buy one, instead.
- Designing reusable code requires iteration
  - Reusable code requires many iterations.
  - Basic law of software engineering: “if it hasn't been tested, it doesn't work”, thus software that hasn't been reused is not reusable.
- Frameworks encode domain knowledge
  - Frameworks solve a particular set of problems.
  - Not always application-domain specific
- **Customer of framework is application programmer**
- Frameworks are abstractions.

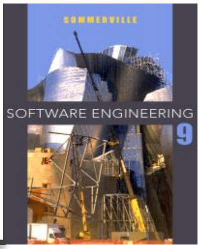
Ideal way to develop framework



- Analyze problem domain
  - Learn well-known abstractions.
  - Collect examples of programs to be built from framework. (Minimum of 4 or 5).
- Design abstraction that covers examples.
  - look for commonalities, represent each idea once.
- Test framework by using it to solve the examples.
  - Each example is a separate application.
  - Performing a test means writing software.

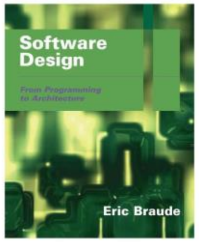


Extending frameworks



- ✧ Frameworks are generic and are extended to create a more specific application or sub-system. They provide a skeleton architecture for the system.
- ✧ Extending the framework involves
  - Adding concrete classes that inherit operations from abstract classes in the framework;
  - Adding methods that are called in response to events that are recognised by the framework.
- ✧ Problem with frameworks is their complexity which means that it takes a long time to use them effectively.

Summary



- ✧ A framework is a collection of software artifacts that is usable by several different applications
- ✧ Frameworks reuse overall architecture, not just components.