Software Frameworks

Gerson Sunyé
University of Nantes
gerson.sunye@univ-nantes.fr
http://sunye.free.fr

Agenda

- Introduction.
- Properties
- Template method pattern
- A toy example
- Correlated concepts
- Real-world examples
- Conclusion

Introduction

Software Framework

- A framework is a reusable, "semi-complete" application that can be specialized to produce custom applications [Johnson and Foote, 1988].

In Short, a Framework is

- A set of classes, abstract classes and interfaces.
- A set of behaviors, spread over these classes.
- An incomplete application for a family of products.
- A set of hooks, where subclasses can insert their specialized behavior.
- The expectations placed upon the subclasses.
- A logic decomposition of a problem.
- Represented by its code.

Creating Real-World Applications by

- Modifying working examples.
- Creating subclasses.
- Configuring objects.
- Writing configuration files.
- Writing programs/scripts for a domain-specific language.
- Modifying a model@run.time.

Framework Goals

- Reuse: code, design, domain analysis, and documentation.
- Simplify software development.
- Reduce code writing.
- Allow inexperienced designers and programmers to develop good software.
- Extract the knowledge of experimented designers and programmers.

Properties

Basic Properties

1. Modularity
2. Reusability
3. Extensibility
4. Inversion of control
5. Non-modifiable code

Modularity

- Abstract classes have a stable interface that encapsulates volatile implementation details.
- They provide hotspots or "points of planned variability", where the behavior can be extended.
- Design and implementation changes are limited to these points, reducing the effort to understand and maintain the software.

Reusability

- The stable interfaces define generic components that can be extended to create new applications.
- Reuse of framework components improves developer productivity, as well as software performance, reliability, and interoperability.

Extensibility

- A framework enhances extensibility by providing explicit hook methods for planned variability.
- Extensibility is essential to ensure rapid customization of new application features.

Inversion of Control

- In a framework, the flow of control is not dictated by the callers, but by the framework itself (the abstract classes).
- The inversion of control enables canonical application processing steps to be customized by hotspots.
- Also called the Hollywood principle: "Don't call us, we'll call you" [Richard E. Sweet 1985].

Non-Modifiable Code

- The framework source code is supposed to be extended, not modified.

Template Method Pattern

Hook and Template Methods

- Hook and template methods are the building blocks of software frameworks.
- They allow the implementation of commonality and variability.

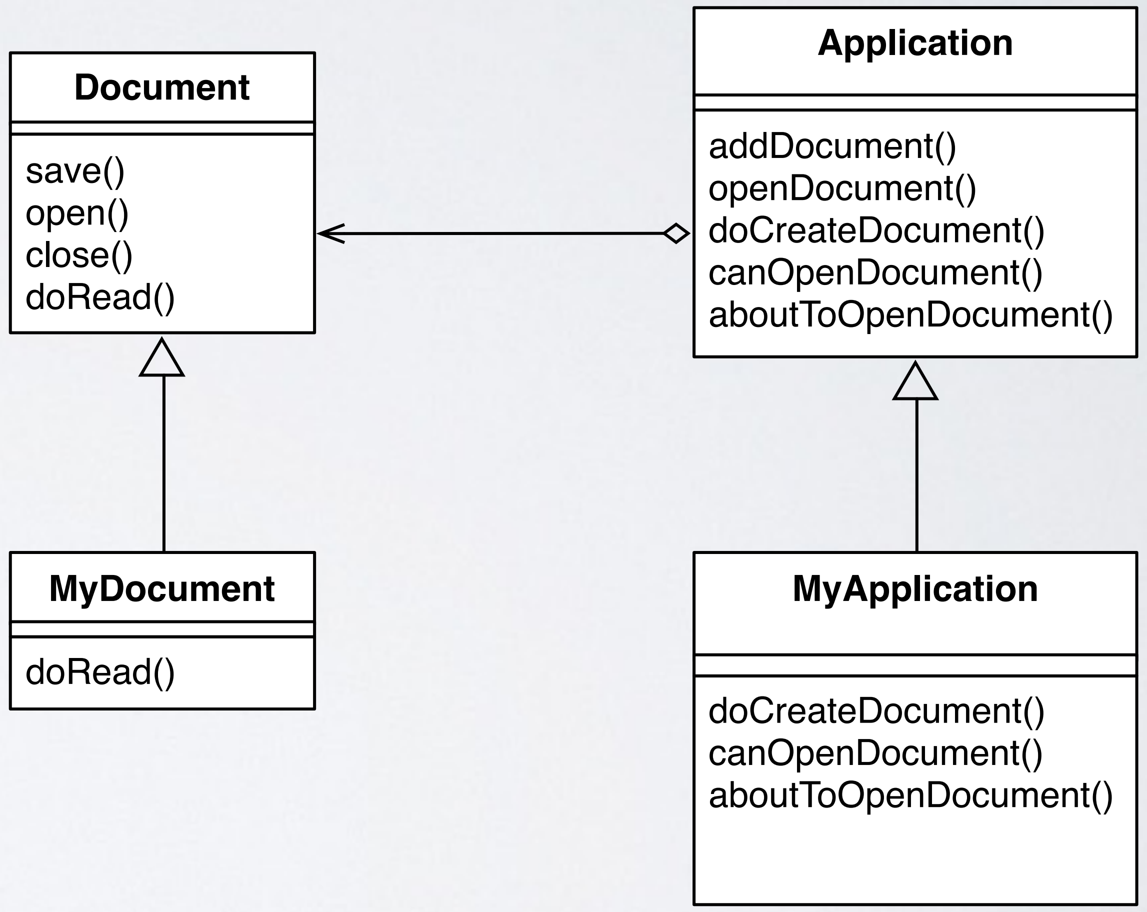
Template Method

- A template method defines the skeleton of an algorithm, deferring some steps to subclasses.
- Subclasses can redefine some steps without changing the algorithm’s structure.

Template Method Example

```
public void openDocument(String
name) {
 if (!canOpenDocument(name)) {
     // cannot handle this
document
    return;
 }

 Document doc =
 doCreateDocument();
 if (doc != null) {
  docs.add(doc);
  aboutToOpenDocument(doc);
  doc.open();
  doc.doRead();
 }
}
```



Template Method Behavior

- A template method usually calls the following kinds of operations:
    - concrete Client operations.
    - concrete AbstractClass operations (i.e., operations that are generally useful to subclasses).
    - concrete operations..
    - abstract operations.
    - factory methods (see Factory Method Pattern).
    - hook operations, which provide default behavior that subclasses can extend if necessary. A hook operation often does nothing by default.

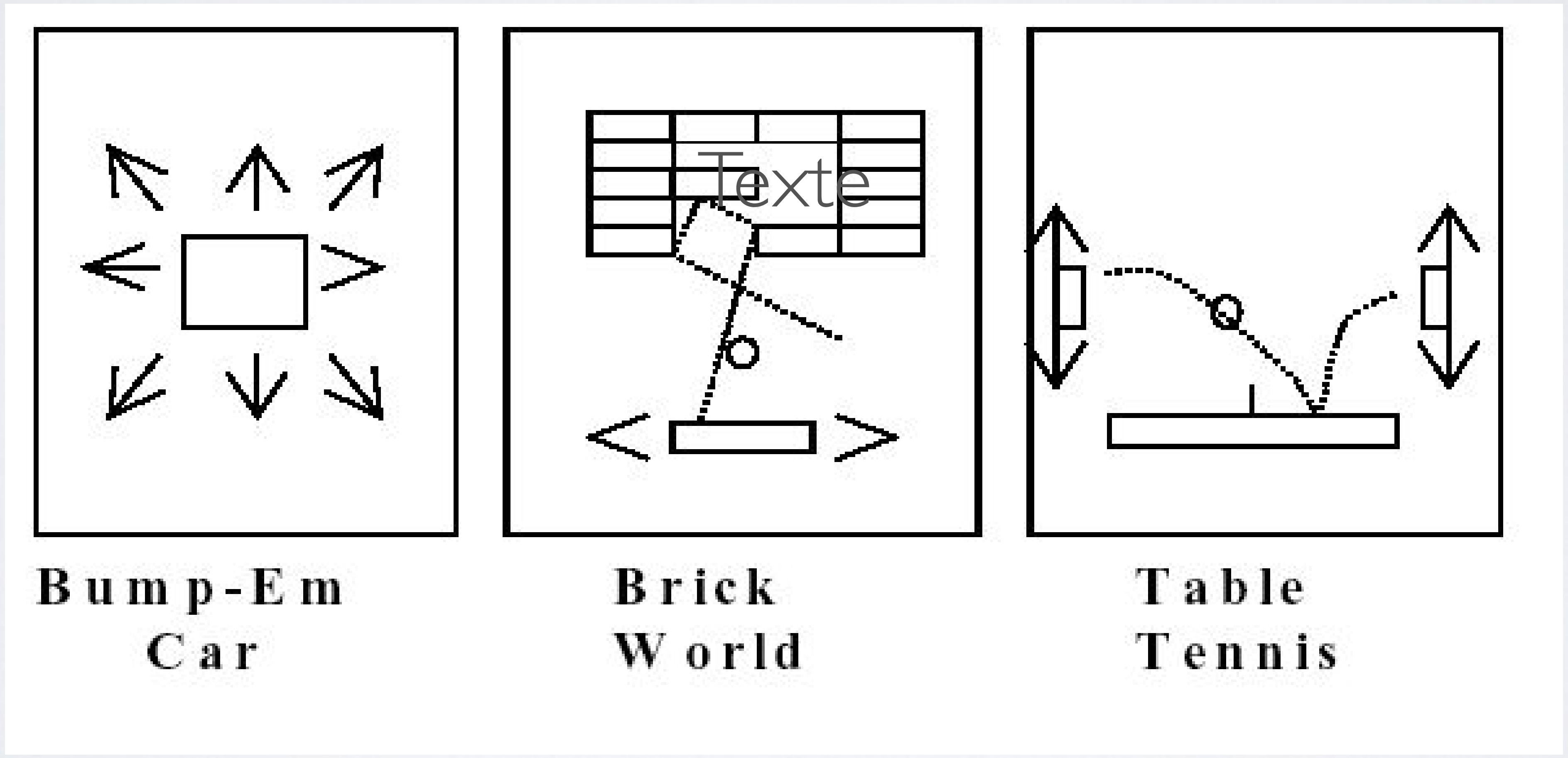
Hook Methods

- A hook method represents a point of variability by providing the calling interface to a variable behavior.
- Each implementation of a hook method provides a variant of that behavior.

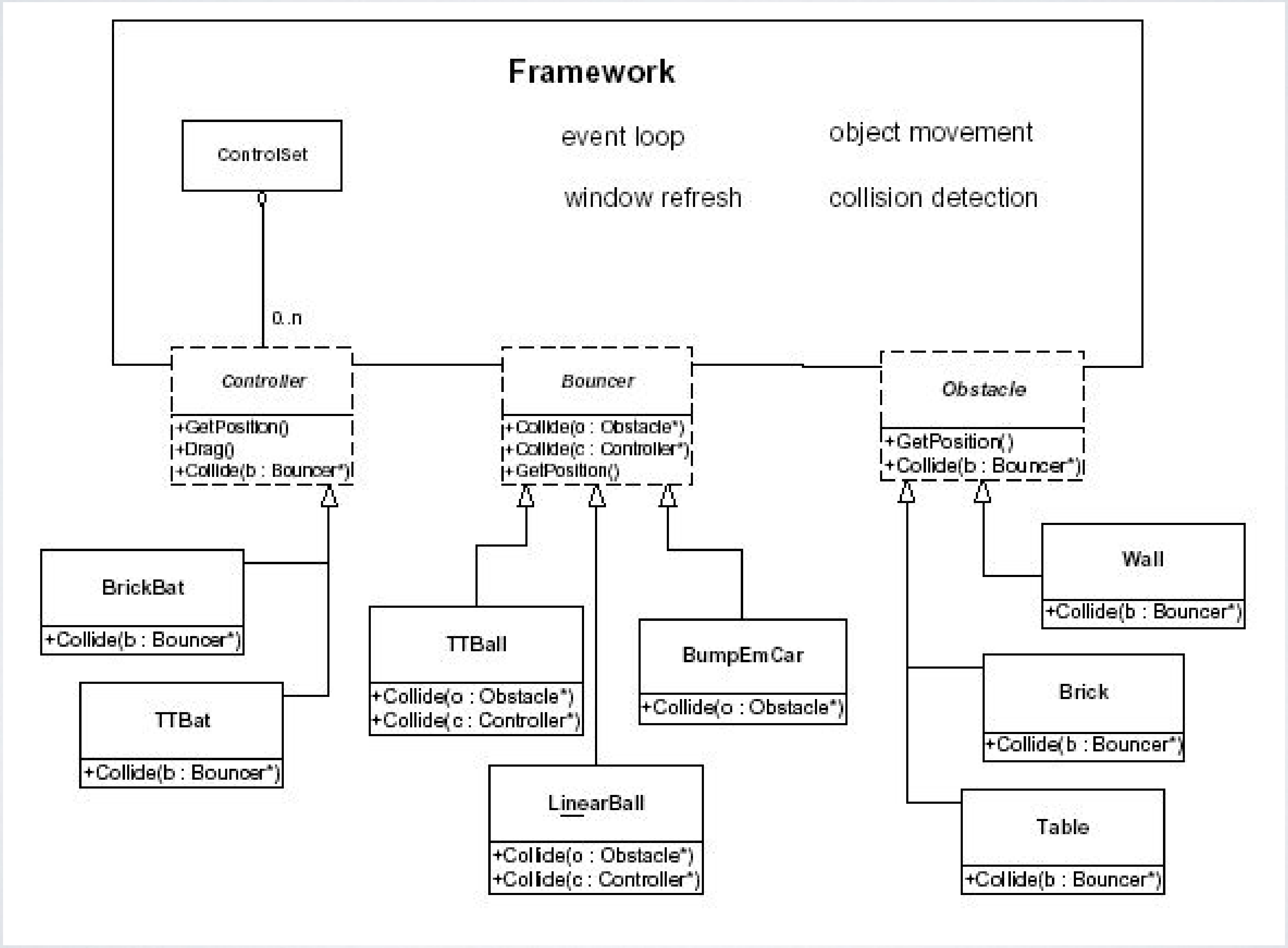
A Toy Example
[Greg Butler]

Domain

Bouncing-Bumping Games



Class Diagram



Template Methods

```
Game::makeWorld() {
  makeBouncer();
  makeControllers();
  makeObstacles();
  makeEventHandlerTable();
}
Game::run(){
  loop over event e in eventQueue {
    ehTable[e]-> handleEvent(e);
    refreshDisplay(); }
}
```

Correlated Concepts

Frameworks and Libraries

- Library use case:
  - The developer designs the application, decomposes the problem and specifies the flow of control.
  - The application calls the library.

Frameworks and Libraries

- Framework use case:
  - The developer extends the framework. The framework defines the flow of control and design decomposition.
  - The framework call the extension code.

Frameworks and Patterns

- Design Patterns:
  - describe micro-architectures
  - are abstract
- Software Frameworks:
  - have concrete architecture and code.
  - incorporate design patterns, often to provide extension points (variability).
  - are a rich field for design pattern mining.

Frameworks and Software Architectures

- Software architectures:
  - describe abstract macro-architectures.
  - target whole systems, but can be used for a product line and single applications.
  - are design artifacts.
  - are designed to ensure software quality.
  - describe the guiding principles behind a given application.

Frameworks and Software Architectures

- Software Frameworks:
  - are concrete implementation of abstract architectures.
  - are designed to be specialized/customized.
  - focus on reusability.

Real-World Examples

Technology Frameworks

- Provide a standard and generic software foundation.
- Examples: COM, CORBA, Java J2EE, ACE (Adaptive Communication Environment, Doug Schmidt et al).

Application Frameworks

- Implement the standard structure of an application.
  - MVC (Model View Controller) — Smalltalk [1980].
  - MFC, Microsoft Foundation Classes.
  - MacApp/ACS — Objective Pascal, C++ [1986]
  - NeXTStep/OpenStep/Cocoa/GNUStep — Objective C, Java.

Business Frameworks

- Domain-specific, business solution that can be extended into an organization.
  - Baan: Enterprise Resource Planning (ERP) software written in Java.
  - San Francisco Business Objects (Taligent/IBM).
  - The Oracle Enterprise Architecture Framework.

Web Application Frameworks

- Designed to support the development of dynamic websites, web applications, web services and web resources.
  - Zope (Zope Corporation) — Python.
  - Apache Struts — Java.
  - Django, Ruby on Rails, Symfony, Yii, Spring MVC, Stripes, Play, CodeIgniter, etc.

Conclusion

Framework Main Characteristics

- Hotspots: planned extension points.
- Inversion of control: the framework controls the application and not the opposite.

Main Issues

- Learning curve.
- Important initial investment.
- Framework developers must be domain experts.
- Framework evolution is complex.

Framework Benefits

- Code and design reuse.
- Perspective shift: programmers are forced to write reusable software.
- Improvement of software quality and developer productivity.

References

- M.E. Fayad, D.C. Schmidt, R.E. Johnson, "Building Application Frameworks", Addison-Wesley, 1999.
- Object Oriented Frameworks. Greg Butler. Ecoop 2001 Workshops.

Software Frameworks