Software Design
13. Reuse

Natsuko Noda
nnoda@shibaura-it.ac.jp

ソフトウェア設計論
13. 再利用

Natsuko Noda
nnoda@shibaura-it.ac.jp

Today's Topics

- Reuse
  - Reuse of code
  - Reuse of design
- Summary of the course

本日のお題

- 再利用
  - コードの再利用
  - 設計の再利用
- まとめ

Software reuse

ソフトウェア再利用

cf. SE-J, 9.3

Software reuse (1/2)

- Technique of using components and the structure of other software in order to develop new software
  - application system reuse
    - an entire application system is reused, either by incorporating it without change into other systems (COTS reuse) or by developing application families.
  - component reuse
    - components of an application may be reused.
  - object and function reuse
    - software components that implement a single well-defined object or function may be reused.

ソフトウェア再利用 (1/2)

- 他のソフトウェアのコンポーネントや構造をソフトウェアの開発に使う技術
  - アプリケーションシステムの再利用
    - 他のシステムに変更なく組み込む(COTS再利用)もしくはアプリケーションファミリーを開発することで、アプリケーション全体を再利用
  - コンポーネントの再利用
    - サブシステムから単一のオブジェクトまで、アプリケーションを構成するコンポーネントを再利用
  - オブジェクトや機能の再利用
    - 1つの明確に定義されたオブジェクトや機能を実装したソフトウェアコンポーネントを再利用

Software reuse (2/2)

- Reusable asset
  - a component and/or a structure that is reused to develop other new software.
- Software reuse has a long history.
- No development without reuse
  - it is very rare to develop newly all of the software.
  - but still, software reuse is difficult.

ソフトウェア再利用 (2/2)

- 再利用資産
  - 他の新しいソフトウェアを開発するために再利用されるコンポーネントや構造
- ソフトウェア再利用には長い歴史あり
- 再利用しない開発はない
  - 現実のソフトウェア開発では、すべてを新規に開発することは極めて少ない
  - しかし再利用は依然として難しい

Benefits of software reuse

Benefit	Explanation
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its design and implementation faults should have been found and fixed.
Reduced process risk	The cost of existing software is already known, whereas the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as subsystems are reused.
Effective use of specialists	Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.

再利用の利点

利点	説明
信頼性の向上	実績があり安定したソフトウェアを使うことで、新たに作るよりも信頼性が高まる
開発上のリスクの軽減	これから新規に開発するよりも、既存のものを使うほうがリスクが小さい
開発リソースの有効利用	専門家の知識や労力の結果である既存資産を活用 （毎回新規開発すれば専門家を毎回投入する必要）

cf. SE-J, p.202

Benefits of software reuse

Benefit	Explanation
Standards compliance	Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users make fewer mistakes when presented with a familiar interface.
Accelerated development	Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.

再利用の利点

利点	説明
標準化への適合	標準に適合化のために標準的な部品を再利用する（標準的なGUI部品を使うなど）
開発期間の短縮	すべてを作らずに再利用することで開発期間を短縮し、製品化までの時間(time to market)を短縮する

Problems with reuse

Problem	Explanation
Increased maintenance costs	If the source code of a reused software system or component is not available then maintenance costs may be higher because the reused elements of the system may become increasingly incompatible with system changes.
Lack of tool support	Some software tools do not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse into account. This is particularly true for tools that support embedded systems engineering, less so for object-oriented development tools.
Not-invented-here syndrome	Some software engineers prefer to rewrite components because they believe they can improve on them. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.

再利用の課題

問題	説明
保守コストの増加	再利用部分のソースコードなどが利用できない場合には、保守のコストが増加する
ツール支援の欠如	多くの開発ツールは、新規開発を支援するが、再利用の支援が不十分である
NIHシンドローム	ソフトウェア技術者は自分で新たに開発することを好む(保守がやりやすい、新たに作る方がチャレンジング)

NIH: Not-invented-here

Problems with reuse

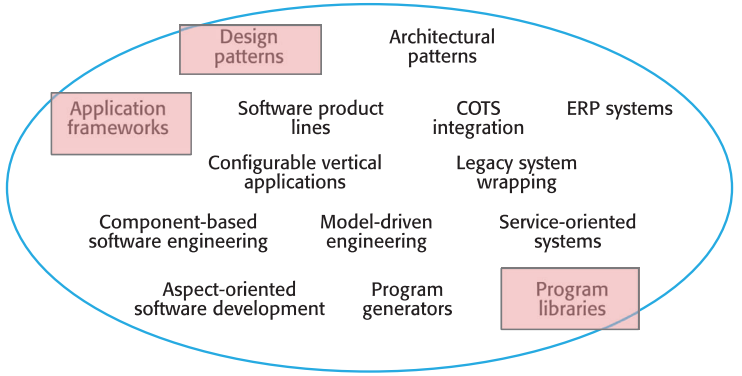
Problem	Explanation
Creating, maintaining, and using a component library	Populating a reusable component library and ensuring that the software developers can use this library can be expensive. Development processes have to be adapted to ensure that the library is used.
Finding, understanding, and adapting reusable components	Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they include a component search as part of their normal development process.

再利用の課題

問題	説明
再利用資産の維持	部品を広め、使わせ、維持することは困難
検索、理解、適用	利用できる部品を探し、その部品を理解し、自分のソフトウェア開発に適用することはいずれも困難な作業

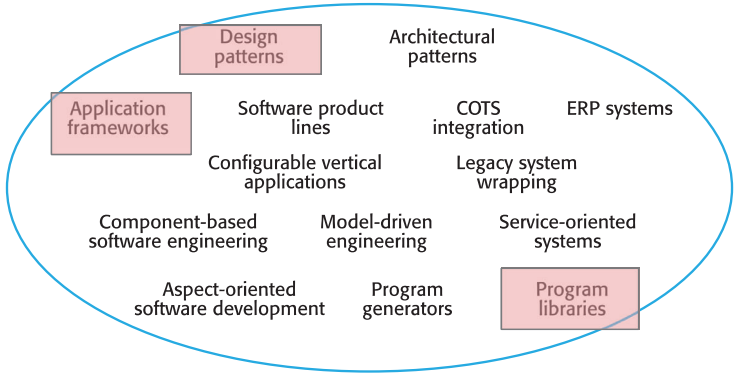
NIH: Not-invented-here

Possible reuse techniques



再利用技術

(日本語訳略)



Program libraries and framework

プログラムライブラリとフレームワーク

Program libraries

- Collections of programs that are prepared to be used in program development.
- Libraries of C programming language
  - The C programming language itself does not have basic functions of string operations and I/O.
  - These functions are provided by libraries (archive of functions).
  - Ex. ANSI C standard libraries
    - C libraries standardized by ANSI

ANSI: American National Standard Institute

Operation	Header	Functions
Character operations	ctype.h	isdigit(), isupper(), tolower()
Mathematical calculations	math.h	exp(), log(), pow(), sin()
I/O	stdio.h	fprintf(), fopen(), fputc()
Utilities	stlib.h	atof(), getenv(), malloc()
String operations	string.h	strcmp(), strcp(), strlen()
Date and time	time.h	clock(), ctime(), time()

プログラムライブラリ

cf. SE-J, p.203

- プログラム開発に使用するために用意されたプログラム集合
- C言語のライブラリ
  - 文字列操作や入出力など基本機能を言語として持たない
  - ライブラリ（関数のアーカイブ）として提供される
  - 例：ANSI C標準ライブラリ
    - ANSIによって標準化されたC言語でのライブラリ

ANSI: American National Standard Institute

ライブラリの内容例

内容	ヘッダ	関数例
文字操作	ctype.h	isdigit(), isupper(), tolower()
数学	math.h	exp(), log(), pow(), sin()
入出力	stdio.h	fprintf(), fopen(), fputc()
ユーティリティ	stlib.h	atof(), getenv(), malloc()
文字列操作	string.h	strcmp(), strcp(), strlen()
日付・時間	time.h	clock(), ctime(), time()

Class libraries

- Program libraries for object-oriented programming languages.
  - Collection of classes as reusable components
  - Examples
    - Java class library
    - JFC (Java Foundation classes)
      - provides classes for GUIs development

クラスライブラリ

- オブジェクト指向プログラミング言語用のプログラムライブラリ
- 再利用可能なコンポーネントとしてのクラスの集合
  - 例
    - Java クラスライブラリ
    - JFC (Java Foundation classes)
      - GUI開発のために用意されたクラス

cf. SE-J, p.203

Application frameworks

- Collections of modules (abstract and concrete classes) are adapted and extended to create application systems.
  - ex. GUI framework, Web application framework
- The developer chooses a framework that meets the goal, and develops an application for customizing the framework.

アプリケーションフレームワーク

- モジュール群(抽象もしくは具象クラス)を採用し拡張してアプリケーションシステムを開発
  - 例: GUIフレームワーク、Webアプリケーションフレームワーク
- 開発者は目的にあったフレームワークを選び、カスタマイズして、要求を満たすアプリケーションを開発

cf. SE-J, 9.3.3

Object-oriented frameworks

- 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.
- The outline of the control is determined by the framework.

オブジェクト指向フレームワーク

- フレームワークは、抽象クラスと具象クラスの集合体と、それらの間のインターフェイスからなるサブシステムの設計
- サブシステムは、デザインの一部を埋めるためにコンポーネントを追加し、フレームワークの抽象クラスをインスタンス化することで実装
- コントロールの概要は、フレームワークによって決定される

Structure of frameworks (1/2)

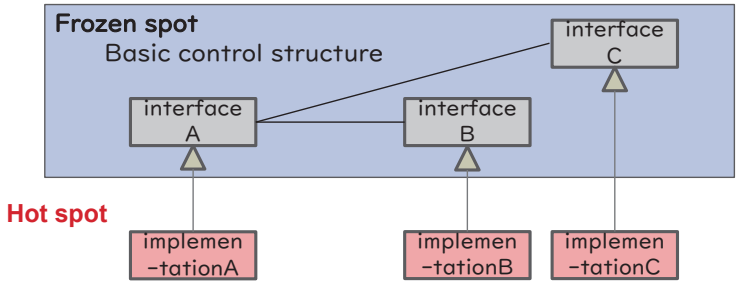
- Frozen spots
  - fixed parts of the framework
  - describes in what condition and in which order the instances are called.
- Hot spots
  - variable parts of the framework. They can be customized.
    - classes and parts of them that the user develops.
  - in hot spots, programs, which are developed by the user and are called in some specific conditions, are embedded.

フレームワークの構造 (1/2)

- フローズンスポット
  - 固定部分
  - インスタンスがどのような条件でどのような順序で呼び出されるかが記述される
- ホットスポット
  - 可変部分。カスタマイズが可能
    - 自分で作成したクラスやその一部
  - ある条件下で呼び出されるプログラムをユーザは作成し、埋め込む

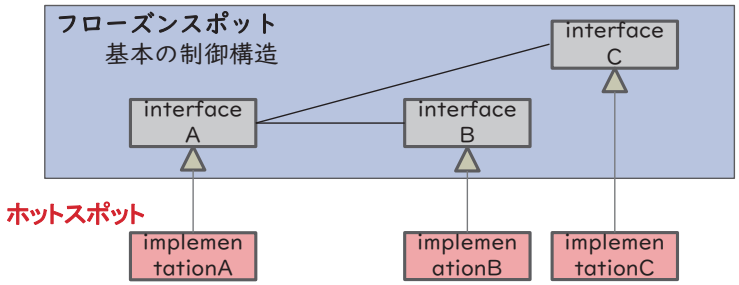
Structure of frameworks (2/2)

- Example：GUI framework
  - standard functions, e.g. detecting mouse clicks and selecting menus, are described in the framework.
  - concrete actions at the time of mouse clicking or selecting a menu have to be implemented by the user (programmer).



フレームワークの構造 (2/2)

- 例：GUIフレームワーク
  - マウスクリックの検出やメニューの選択等、標準的な機能はフレームワークに記述されている
  - マウスクリック時やメニュー選択時の具体的な動作をフレームワークの利用者(プログラマ)が作成



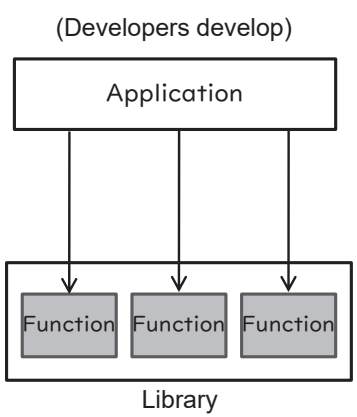
Library and framework

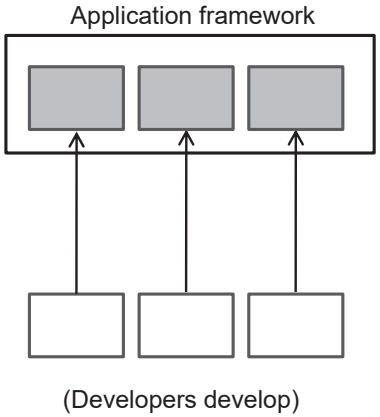
- Program libraries
  - can be used by any application
  - for general usage
  - reuse assets in small size
- Frameworks
  - semi-finished products of specific applications
  - not applicable for all applications
  - reuse assets in large size

ライブラリとフレームワーク

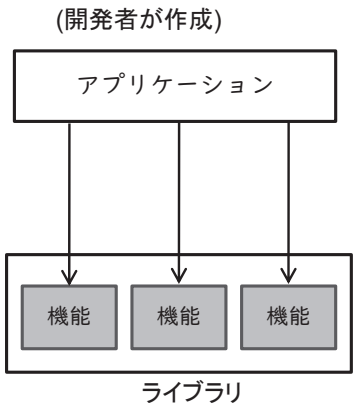
- プログラムライブラリ
  - どんなアプリケーションでも使える
  - 一般ユーザ向き
  - 再利用資産は小規模
- フレームワーク
  - 特定のアプリケーションのための半製品
  - 全てのアプリケーションに使えるわけではない
  - 大規模な再利用資産

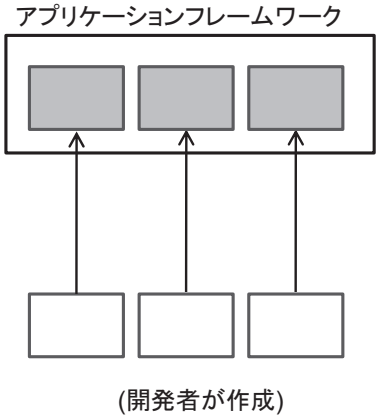
Library and framework (CONT.)





Library and framework (CONT.)





cf. SE-J, p.207

Software pattern

ソフトウェアパターン

cf. SE-J, 9.4

Patterns in software development

- A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.
  - Dirk Riehle and Heinz Zullighoven, "Understanding and Using Patterns in Software Development”
- A pattern is a named nugget of insight that conveys the essence of a proven solution to a recurring problem within a certain context amidst competing concerns.
  - by B. Appleton

ソフトウェア開発におけるパターン

- パターンとは、恣意的ではない特定の文脈で繰り返される具体的な形を抽象化したもの
  - Dirk Riehle and Heinz Zullighoven, "Understanding and Using Patterns in Software Development”
- パターンとは、競合する問題の中で、ある状況下で繰り返される問題に対する実証済みの解決策の本質を伝える、名づけられた洞察の塊のこと
  - by B. Appleton

Patterns

- Various types of patterns in software development
  - idiom, design patterns, architectural patterns
- Purpose
  - giving concrete and practical knowledge
  - solving *trade-off* problems in design
  - giving vocabularies of design

パターン

- ソフトウェア開発には様々なタイプのパターンがある
  - イデオム、デザインパターン、アーキテクチャパターン
- 意義
  - 具体的・実践的な知識を与える
  - 設計におけるトレードオフ問題を解決する
  - 設計ボキャブラリを提供する

Design patterns (1/2)

- Software patterns for issues in software design.
  - reuse in design level
- A catalog of design patterns by Gamma et al. (also called as "GoF") is famous.

GoF: Gang of Four

  - format of the catalog
    - pattern name： a descriptive and unique name
    - motivation (Forces) ： a scenario consisting of a problem and the context in which this pattern can be used.
    - structure ： a graphical representation of the pattern. Class diagrams may be used.
    - collaboration ： a description of how classes and objects used in the pattern interact with each other.
    - implementation ： a description of an implementation of the pattern; important reminder for implementation
    - etc. ...

デザインパターン (1/2)

- 設計上の課題に対する既存のパターン
  - 設計レベルでの再利用
- Gammaら("GoF")によるカタログが有名

GoF: Gang of Four

  - カタログ形式
    - パターン名： 名称
    - 目的： どのような課題に対するものか
    - 構造： パターンの構造（クラス図的）
    - 協調関係： 構成要素間の協調関係
    - 実装： パターンを実装する際の留意点等
    - その他...

Design patterns (2/2)

- 23 patterns by GoF

creational pattern	Abstract Factory Builder Factory Method Prototype Singleton	behavioral pattern	Chain of Responsibility Command Interpreter Iterator Mediator Memento Observer State Strategy Template Method Visitor
structural pattern	Adapter Bridge Composite Decorator Façade Flyweight Proxy		

"Design Patterns: Elements of Reusable Object-Oriented Software", Erich Helm, Richard Johnson, Ralph Vlissides, John Gamma, Addison-Wesley, 1994

Design patterns (2/2)

- GoFによる23のパターン

生成のパターン	Abstract Factory Builder Factory Method Prototype Singleton	振舞いのパターン	Chain of Responsibility Command Interpreter Iterator Mediator Memento Observer State Strategy Template Method Visitor
構造のパターン	Adapter Bridge Composite Decorator Façade Flyweight Proxy		

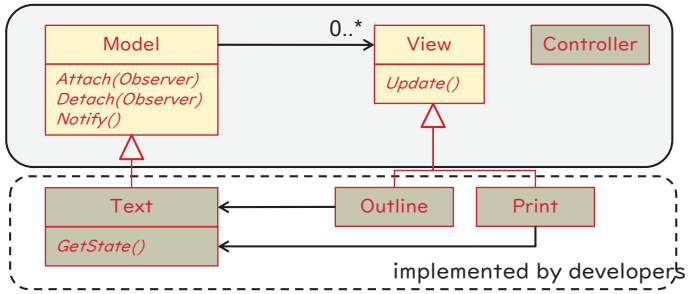
"Design Patterns: Elements of Reusable Object-Oriented Software", Erich Helm, Richard Johnson, Ralph Vlissides, John Gamma, Addison-Wesley, 1994

(翻訳) "オブジェクト指向における再利用のためのデザインパターン", ソフトバンククリエイティブ, 1999

Design patterns and frameworks

- Design patterns
  - is a catalog of “guidelines of the design”.
  - do not provide concrete implementations.
- Framework
  - customizable application framework.
  - includes concrete implementations.
  - is implemented based on various design patterns.

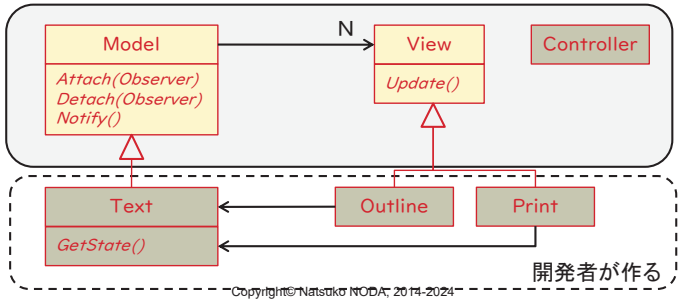
MVC framework



デザインパターンとフレームワーク

- デザインパターン
  - 設計の指針を記述したカタログ
  - 具体的な実装は規定されていない
- フレームワーク
  - 開発者がカスタマイズできるアプリケーションの骨組み
  - 具体的な実装を含む
  - 様々なデザインパターンに基づいて実装されている

MVCのフレームワーク



Reuse and structure

- In software development, making the base structure of the software is important.
  - just combining ad hoc independent objects and modules doesn’t result in any software that works well.
- Each class has a meaning only in a specific context.
  - reusable classes have assumptions and contexts of usages.

再利用と構造

- ソフトウェア開発において、骨格となる構造を考えることが重要
  - 独立した個別のオブジェクト・モジュールを単に組み合わせても、意味のあるソフトウェアとして機能しない
- 個々のクラスは、あるコンテキストのもとで意味を持つ
  - 再利用されるクラスも、使われ方の前提・コンテキストがある

Model driven engineering

モデル駆動工学

cf. SE-J, p.205, 10章

Model driven engineering (MDE)

- Development that utilize software models and model transformations.
- Model transformation
  - Model to model
  - Model to program
- Not programming, but modelling
  - Higher abstraction of programming languages

モデル駆動工学 (MDE)

- ソフトウェアモデルやモデル変換を利用した開発
- モデル変換
  - モデルからモデル
  - モデルからプログラム
- プログラミングではなく、モデリング
  - プログラミング言語の高度な抽象化

Techniques of MDE

- Domain specific model
  - Profile
  - Extension of UML
- Model transformation
  - Mapping
- Domain specific language (DSL)
  - Programming language specialized for a specific domain
- Tools and environment

MDEの諸技術

- ドメインに特化したモデル
  - プロファイル
  - UMLの拡張
- モデル変換
  - マッピング
- ドメイン特化言語 (DSL)
  - 特定のドメインに特化したプログラミング言語
- ツールや環境

Quiz

1. What is the difference between program libraries and application frameworks?
2. List the types of software patterns.

Quiz (日本語訳略)

1. What is the difference between program libraries and application frameworks?
2. List the types of software patterns.

Summary of the course

まとめ

Modeling using UML

- UML is a standardized graphical language and notation for describing object-oriented models.
- For modeling of static structure :
  - Class diagram
  - Object diagram
- For modeling of behavior :
  - Activity diagram
  - Sequence diagram
  - State machine diagram
  - Use case diagram

What kind of diagrams?
For what purpose?

UMLを用いたモデリング

- UMLは、オブジェクト指向モデルを記述するための図式の言語及び記法
- 静的構造のモデル化：
  - クラス図
  - オブジェクト図
- 振舞いのモデル化：
  - アクティビティ図
  - シーケンス図
  - ステートマシン図
  - ユースケース図

どんな図?
どんな目的のための図?

Techniques for software development

- Requirements engineering
- Validation and verification
- Software process
- Reuse

ソフトウェア開発のための諸技術

- 要求工学
- 検証 (V&V)
- ソフトウェアプロセス
- 再利用

For good software design

- "No silver bullets."
- Which method/technique/way is the best is dependent on
  - domains
  - scale of the target software
  - developers' experience
  - …

良いソフトウェア設計のために

- 「銀の弾丸はない」
- どの方法/技術/方法が最善かは以下に依存
  - ドメイン
  - 対象ソフトウェアの規模
  - 開発者の経験
  - …

For good software design (Cont.)

- Learning from the wisdom and experience of our predecessors
  - Software patterns
  - Design principles ("Design guideline" in 9th lecture)
    - "Agile Software Development, Principles, Patterns, and Practices" by Robert Martin
  - Best practices
    - Examples of practices that worked

良いソフトウェア設計のために (Cont.)

- 先人の知恵・経験に学ぶ
  - ソフトウェアパターン
  - 設計原則 (第9回資料中の「設計ガイドライン」)
    - 「アジャイルソフトウェア開発の奥義 第2版 オブジェクト指向開発の神髄と匠の技」ロバート・マーティン
  - ベストプラクティス
    - うまくいった実践例

Information about UML

- UML specification
  - Available from the OMG website
  - https://www.omg.org/spec/UML/
- Reference book
  - UML Distilled: A Brief Guide to the Standard Object Modeling Language
    - by Martin Fowler. Addison-Wesley

UMLに関する情報

- UML仕様書
  - OMGのWebサイトから入手可能
  - https://www.omg.org/spec/UML/
- 参考書
  - UML Distilled: A Brief Guide to the Standard Object Modeling Language
    和訳：UMLモデリングのエッセンス 第3版 (翔泳社)

Final exam

- Answer quizzes and an assignment on SocmbZ
  - In PC lecture room 1 and 2
    - at an assigned seat
    - The seat map will be posted in the rooms on the day of the exam.

期末試験

- ScombZに公開されるテストと課題に答える
  - 会場：PC講義室1、PC講義室2
    - 指定席
    - 座席表は、当日教室に掲示します

Misc.

- Minute paper
  - If you didn't submit any of the minute papers, please submit them by this Friday 23:00 (JST).
- Quiz
  - If you missed some quizzes, don't worry too much. (No chance to retry.)

その他

- ミニットペーパー
  - 未提出のものがあれば(日本時間)今週金曜日23時までに提出
- Quiz
  - 未受験のものがあっても気にしすぎなくて良いです(再受験することはできません)