Software Development
Process

Software Engineering Text (chapters 1 and 3)
What is Software Process?

_activities:

- Specification (what)
- Design (how)
- Coding/development
- Testing/validation
- Maintenance (change!)
What is Software Process?

★ Activities:
★ Specification (what)
★ Design (how)
★ Coding/development
★ Testing/validation
★ Maintenance (change!)
Specification (What)

- Feasibility study
  Can we do it using the tools, time and budget constraints?
- Requirements Analysis
  Figuring out specifics of what we need to do
- Requirements Spec.
  Document everything you found out.
- Requirements Validation
  Show them to customers, do prototypes
Specification – Documents

- UI Spec
- Use Case Scenarios
- Prototype
- Test Results of prototypes
What is Software Process?

✧ Activities:
  ✧ Specification (what)
  ✧ Design (how)
  ✧ Coding/development
  ✧ Testing/validation
  ✧ Maintenance (change!)
Design (how)

- Architecture
  Subsystems, and their relationships
- Abstract spec.
  Subsystem services and constraints
- Interface Design
  Methods or messages between subsystems
- Component Design
- Data Structure and Class Design (if OOD)
- Algorithm Design
Design: Structured Models

- Data-flow model
- **Entity Relation model**
  (should have seen that in Dbase)
- **Object oriented methods**
  (inheritance, class/object interaction and relationship)
- State Transition models
What is Software Process?

Activities:
- Specification (what)
- Design (how)
- Coding/development
- Testing/validation
- Maintenance (change!)
What is Software Process?

Activities:
- Specification (what)
- Design (how)
- Coding/development
- Testing/validation
- Maintenance (change!)
Testing

- Unit Testing
- Module/object testing
- Subsystem testing
- System testing

Done by

- programmer
- testing group
Testing: Verification and Validation

- **Verification**: are we building the product right? *(test against specs)*
- **Validation**: are we building the right product? *(test with the user)*

How do you translate that for your game?
What is Software Process?

✦ Activities:
  ✦ Specification (what)
  ✦ Design (how)
  ✦ Coding/development
  ✦ Testing/validation
  ✦ Maintenance (change!)
What is Software Process?

 중요한 작업들:

- Specification (what)
- Design (how)
- Coding/development
- Testing/validation
- Maintenance (change!)

How do you translate this process to games?
Software Myths

“If we are behind the schedule, just add more programmers”

“Nine Women cannot make a baby in nine months”
Software Myths

“Once we write the program and get it to work, our job is done”

“60%-80% of all effort expended on software will be expended after it is delivered for the first time”
What is Software Process?

📍 Activities:

📍 Specification (what)
📍 Design (how)
📍 Coding/development
📍 Testing/validation
📍 Maintenance (change!)

In what order?
Challenges

- Wiked problems

- Customers don’t know what they want

- Hard to change code, esp. with evolution of programming languages
Methods the people developed

- The waterfall model
  - Separate and distinct phases of specification and development
- Evolutionary development
  - Specification and development are interleaved
- Formal systems development
  - A mathematical system model is formally transformed to an implementation
- Reuse-based development
  - The system is assembled from existing components
Waterfall Model (1970)

- Requirements definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance
Advantages and Disadvantages

❖ Good when requirements are understood and problem has been attached before

❖ Cannot adapt to rapidly changing requirements or ill-defined problems
Building a software is like driving

“Driving is not about getting the car going in the right direction. Driving is about constantly paying attention, making little correction this way, a little correction that way.” — from XP
Evolutionary Development

- **Exploratory development**
  - Objective is to work with customers and to evolve a final system from an initial outline specification

- **Throw-away prototyping**
  - Objective is to understand the system requirements. Should start with poorly understood requirements
Evolutionary Development

**Problems**
- Lack of process visibility
- Systems are often poorly structured
- Special skills (e.g. in languages for rapid prototyping) may be required

**Applicability**
- For small or medium-size interactive systems
- For parts of large systems (e.g. the user interface)
- For short-lifetime systems
Reuse Oriented Development

- Requirements Spec.
- Research Existing Components
- Rework Requirements
- Design
- Component Integration and testing
Process Iteration

🔹 System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems

🔹 Two (related) approaches
  ✹ Incremental development
  ✹ Spiral development
Incremental Method
Spiral Model

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.
Spiral Method

- Determine objectives, alternatives, and constraints
- Evaluate alternatives, identify, resolve risks
- Risk analysis
- Prototype 1
- Prototype 2
- Prototype 3
- Operational prototype
- Simulations, models, benchmarks
- Detailed design
- Product design
- S/W requirements
- Concept of Operation
- Requirements plan, life-cycle plan
- Development plan
- Integration and test plan
- Plan next phase
- Requirement validation
- Design, V&V
- Acceptance test
- Integration test
- Unit test
- Develop, verify next-level product
- Service
Project

- Pitch (Presented Oct 29)
- High Level Design document (Nov 7)
- Prototype 1 (Nov 14)
- Prototype 2 (Nov 26)
- Prototype 3 (Dec 3)
- Final Game Show Case (Dec 5), video and presentation video
UML

All materials (including images) are taken from: http://edn.embarcadero.com/article/31863 and http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/diagrams.htm
UML (what is it?)

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - Class diagrams
  - Object diagrams
  - Sequence diagrams
  - State chart diagrams
  - Component diagrams
  - Activity diagrams

Why do you think you need this?
Use Case

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - Class diagrams
  - Object diagrams
  - Sequence diagrams
  - State chart diagrams
  - Component diagrams
  - Activity diagrams
Use Case Diagrams

- Patient
- Make Appointment
- Cancel Appointment
- Request Medication
- Pay Bill
- Scheduler
- Doctor
- Clerk
Customer

Browse Catalog and Select Items

- Call Sales Person
- Give Shipping Info
- Give Payment Info

Get Confirmation #
Use cases in Game Development

- In games use cases are important for:
  - Testers to ensure all user interactions are satisfied
  - Developers to determine the features for the interaction
Use Case

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - **Class diagrams**
  - Object diagrams
  - Sequence diagrams
  - State chart diagrams
  - Component diagrams
  - Activity diagrams
Class Diagrams
In games developers use class diagrams to:

- Document the systems involved
- Share and discuss among other developers
- Code review and assessment to increase maintainability of the code especially for engine code or franchise games
Use Case

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - Class diagrams
  - **Object diagrams**
  - Sequence diagrams
  - State chart diagrams
  - Component diagrams
  - Activity diagrams
Object Diagrams

- Just like the class diagram but with objects instead. This is useful to show the relationships between objects not just classes.
Use Case

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - Class diagrams
  - Object diagrams
  - **Sequence diagrams**
  - State chart diagrams
  - Component diagrams
  - Activity diagrams
Sequence Diagram

.shows the dynamics of the systems. Specifically the sequence of operations between objects.

- **Green bar**: when object has control
- **Red box**: object destroyed
- **Blue bar**: object active
Sequence Diagrams in Game Development

- In games developers use sequence diagrams to:
  - Document the dynamics of the systems
  - Share and discuss among other developers
  - Code review and assessment to increase maintainability of the code especially for engine code or franchise games
Use Case

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - Class diagrams
  - Object diagrams
  - Sequence diagrams
  - State chart diagrams
  - Component diagrams
  - Activity diagrams
State Diagrams

- State diagrams describe the behavior of the system. They describe the state of the object when an event occurs.
Checking
  do / check items
  [all items available]

[an item is not available]

Canceled

Dispatching
  do / initiate delivery

Delivered
In games developers use state diagrams to:

- Document the behavior of objects, specifically AI behaviors
- Share and discuss among other developers
- Code review and assessment to increase maintainability of the code especially for engine code or franchise games
Use Case

- Unified Model Language
- Standardized representations or models for explaining software design and architecture (like the blueprints of the software)
  - Use case diagrams
  - Class diagrams
  - Object diagrams
  - Sequence diagrams
  - State chart diagrams
  - Component diagrams
  - Activity diagrams
Use Case

 כעת ננתח את המושג "Use Case".

1. **Unified Model Language**
2. **Standardized representations or models for explaining software design and architecture (like the blueprints of the software)**
   - Use case diagrams
   - Class diagrams
   - Object diagrams
   - Sequence diagrams
   - State chart diagrams
   - Component diagrams
   - Activity diagrams
Activity Diagrams

- Used to describe more complex dynamic behavior, adding to sequence diagram.

- This is especially important for threading or multi-process algorithms