EECE 262 Course Overview

Jules White
Distributed Object Computing (DOC) Group
Institute for Software Integrated Systems
Vanderbilt University

What is EECE 262 About?

- This class is about picking a cool project that involves networking and building it

EECE 262 Course Information

- EECE 262 class web page
  - www.dre.vanderbilt.edu/ ~jules/eece262/
- My office hours in Featheringill Hall room 226 are
  - Tues/Thurs 10:50am to 11:20am or by appointment

- Required textbooks

None!!!

Please send all questions to jules@dre.vanderbilt.edu

I'll send the answers to the class mailing list

EECE 262 Goal

- The goal of this class is to develop innovative new smartphone applications that involve networking.
- Every team should be run like a startup. Design a revolutionary application that people need and build it.
- Startups don't build "me too" applications. Startups build highly novel revolutionary apps.
  - Any Vanderbilt-specific app is fine too
- Just like in a startup, there are lots of roles for the team members – but everyone works hard. You can choose any role you want but you must produce results.

- My Expectations:
  - I am interested in seeing really creative thinking.
  - I want everyone to work hard.
  - Every project will be highly novel or have some aspect that sets it apart from its competition.
  - A valiant attempt on a revolutionary idea that fails will get just as good of a grade as a successful implementation of a vanilla idea
  - Hard work, participation in class, and creative thinking will guarantee an A.

EECE 262 Goal (Grad Students)

- The goal of this class for grad students is to help you produce a research paper.
- Grad student projects do not have to be focused on smartphones and do not require implementing some type of software product.
- Grad students should replace all instances of “project” with the word “research project.”
- Grad students will deliver experimental results and/or pages of their paper each week.
- I encourage ALL groups to write papers on their work.

- My Expectations:
  - I am interested in seeing creative research projects.
  - I want to see incremental progress EVERY week.
  - Every grad student must turn in 1pg on their paper each week.
  - All papers must be written using the latex template I provide. No MS Word.
  - If you turn in all of your weekly page assignments, you will be guaranteed an A.

EECE 262 Research Papers

- Publishing papers in academic venues is very valuable.
- I will work with any undergrad that is interested on publishing a paper.
- I can't guarantee that papers will get accepted, but we have a pretty good track record in the DOC group ;-)
  - Published undergrads that worked with me:
    - Chris Thompson
    - Hamilton Turner
    - Scott Campbell (graduated)
    - Harrison Strowd (graduated)
    - Sean Mulligan (graduated)
  - This list should soon include Ben Gotow, Krzysztof Z.
- I encourage ALL groups to write papers on their work.

EECE 262 Ground Rules

- Build cycles must be completed on time
- ~~Work ***must*** be your own*~~ All group projects!!!
- ***Bring your laptops every day (just in case)***
- You will be called upon every day to answer questions
- You'll get out of this course what you put into it, so be prepared to work hard
- Be prepared for occasional guest lectures
- No quizzes, no tests, no exam → instead: weekly demos, code reviews, and a final demo
- Make sure to avail yourself of available help, e.g., office hours, TAs, mailing list, etc.

EECE 262 Course Contents

- Focus on developing large-scale smartphone networking projects in a team setting:
  - Code must be turned in every build cycle
  - Agile software development practices must be followed
  - Demos of projects every 3rd class
- Everyone must be a member of a team working on a smartphone networking project
- The course will completely revolve around topics that will aid the projects.
- There will be lectures and in-class exercises on topics related to the design and development of the projects
- I assume that people will have a wide range of coding skills. Don't worry if you can't code, there will be lots of different roles for project team members.
- Feel free to ask me questions via email/class/office hours related to:
  - Eclipse
  - Java
  - Framework XYZ
  - Patterns
  - Development practices
  - Promoting your open source project
  - Etc…

EECE 262 Course Contents

- My main goal of the class is to facilitate and guide everyone through the implementation of a complex smartphone networking project
- You will learn by *doing*
- Feel free to suggest advanced topics that you would like to cover in class:
  - Integrating smartphone apps and cloud computing
  - Ad-hoc networks for smartphones
  - Smartphone software platforms
  - Advanced web apis/protocols
  - Etc.
- I am also free to help outside of class with any questions you have

- Every member of each team must contribute
- Although I will be focused on groups as a whole, I will also pay attention to each team member's individual effort
  - I will look at SVN to see who committed what code
  - I will look at the bug tracking system to see who was reporting errors
  - I will look at project wikis to see who posted what
  - I will pay attention in class to who is contributing to the discussion

EECE 262 Course Work

- There will be ~10 build cycles
  - All projects must be implemented for Android or iPhone
  - Can be done on Windows, Linux, Mac, etc.
  - **Must be done as a team***
- Your grade will be based on:
  - 40% bi-weekly build cycle execution
  - 20% final project demo/presentation
  - 40% in-class participation
- Waiting until the end of the course and trying to code everything (regardless if it works) will produce a poor grade
- A key part of the course is staying on the development schedule, following the development guidelines, and contributing each class period
- Feel free to use any open source code that you want (as long as you aren't just ripping it off or writing a wrapper around it)

EECE 262 & CS 279 Integration

- This year, the CS 279 class is available to you as a development resource
- Each project should propose a server-side component that could be developed by the students in CS 279
- If your project is selected by one or more 279 students:
  - You will serve as their customer
  - Your user stories will include ones for the server-side team
  - You will be responsible for developing the design documentation for them
  - If you are not an experienced Java coder and do not plan on doing any Java coding for your project, you are required to manage the design docs and coordination of your team with CS 279

EECE 262 Assignment 1

- Your first assignment is to come up with a list of 10 potential projects.
  - Please put thought into these ideas
  - Make sure you explain what the problem is that you are solving and why your solution is going to be different than what already exists
  - Graduate students must produce research ideas for their papers
- Each idea should be summed up in a 1-2 paragraph description.
- Even if you already know (or think you know) what you are going to work on, you must complete this assignment. You may come up with a new idea, an idea that changes how you think about your current plan, or an idea that inspires someone else.
- Email the ideas to me by Sunday at midnight (subject: “eece262 ideas”)
- We will discuss these ideas on Tuesday and form teams in class
- Your server-side ideas will be pitched to CS 279 next Thursday

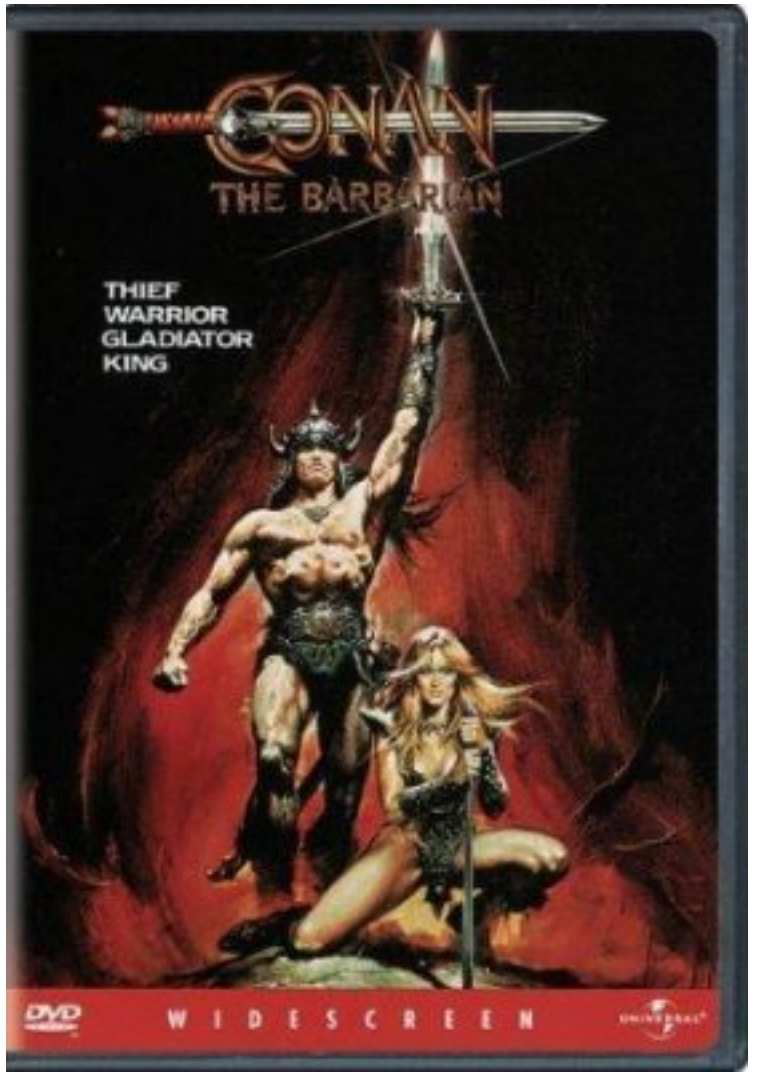
EECE 262 Assignment 2

- Your second assignment is to:
  - Write a 1-page description of your project and list your team members
  - Setup a Google Code website
  - Setup a Google Group for your project to serve as a mailing list (add me to the group)
  - Create a wiki page and enter your first set of user stories
- Email me a link to your Google Code site by midnight next Wed
- Be prepared to discuss your project and user stories next Thurs. in class

- For Grad students, write a 1 page outline of your paper

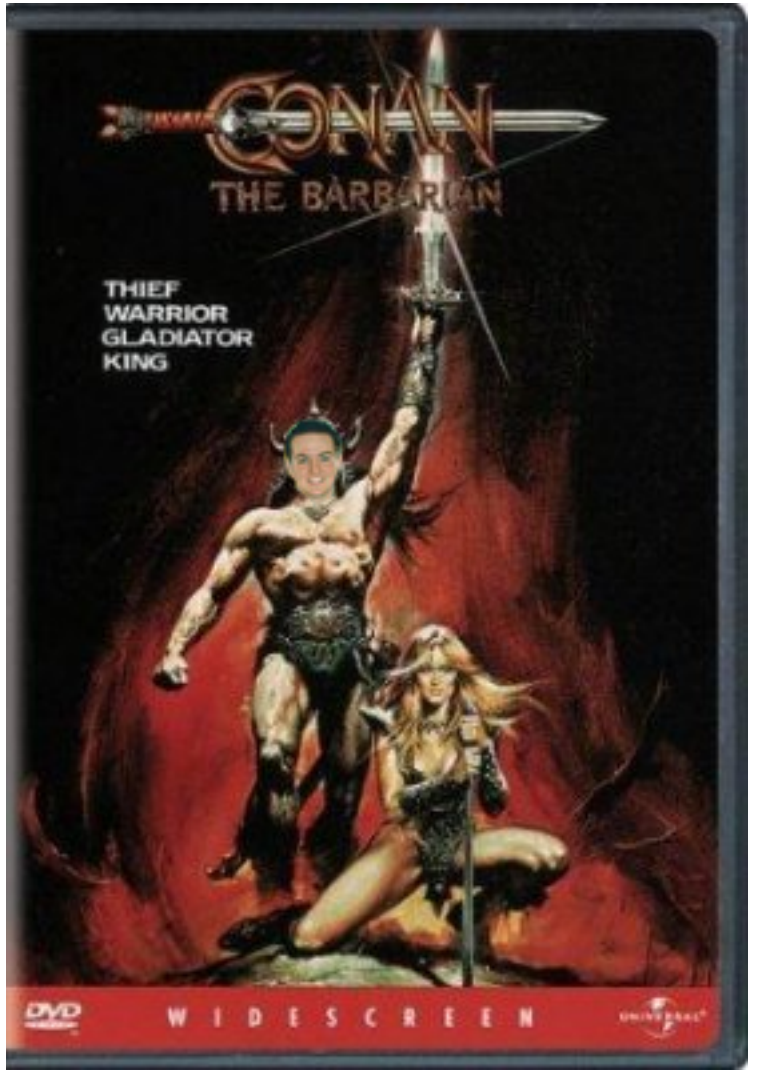
Lessons from Conan

- The secret of steel has always carried with it a mystery. You must learn its riddle, Conan. You must learn its discipline. For no one - no one in this world can you trust. Not men, not women, not beasts. **Steel you can trust**



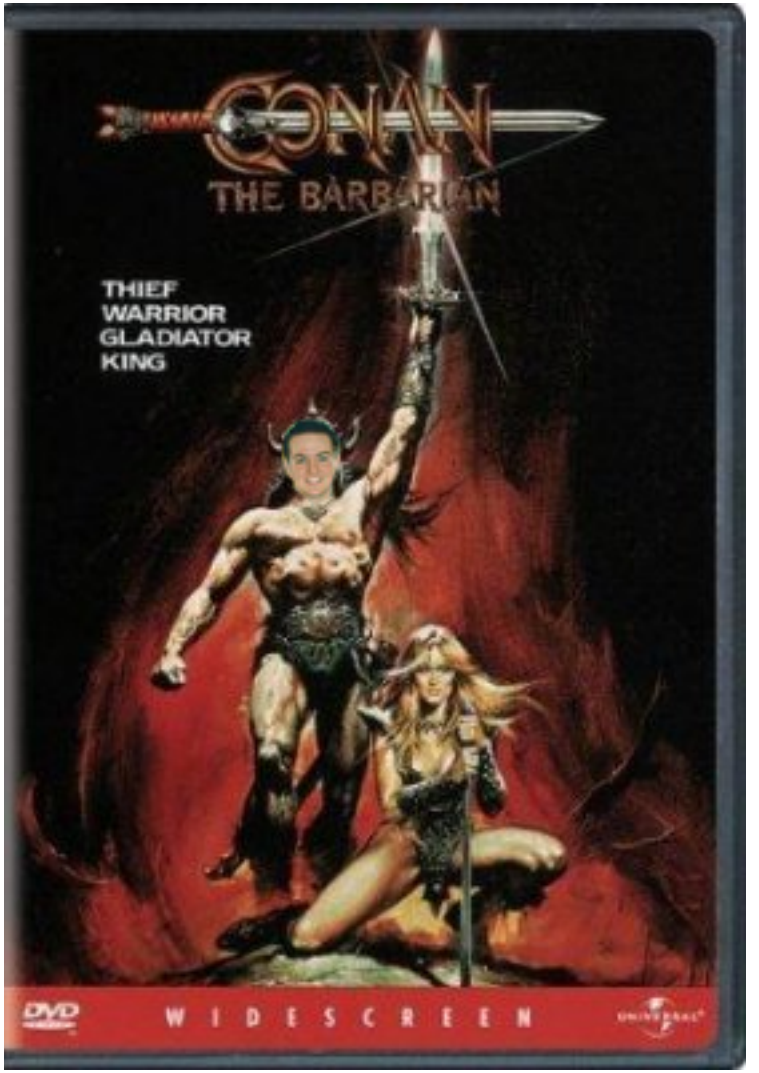
Lessons from Agile Development

- The secret of *code* has always carried with it a mystery. You must learn its riddle, Conan. You must learn its discipline. For no project manager - no developer in this world can you trust. Not UML diagram, not test plan, not architect hype. **Code you can trust**
- **(if it is thoroughly tested)**



Lessons from Agile Development

- We will be using an Agile development process in EECE 262
- Short concentrated build cycles that focus on working code
- Client-focused, we will be demoing each others' software at the end of each build cycle



EECE 262 Development Cycle

- **We will use a 3 class development cycle that will start next Thursday**
- **1st Class:**
  - Discuss/select user stories in class (rough drafts prepared before class)
  - Discuss why the features are important and how they compare/contrast to any existing features from competitors
  - Rough sketches of any user interfaces are presented
  - The class as a whole discusses the features and their implementation
  - Groups present architecture for how the stories will be implemented
  - Other groups play devil's advocate and critique architecture/features

EECE 262 Development Cycle

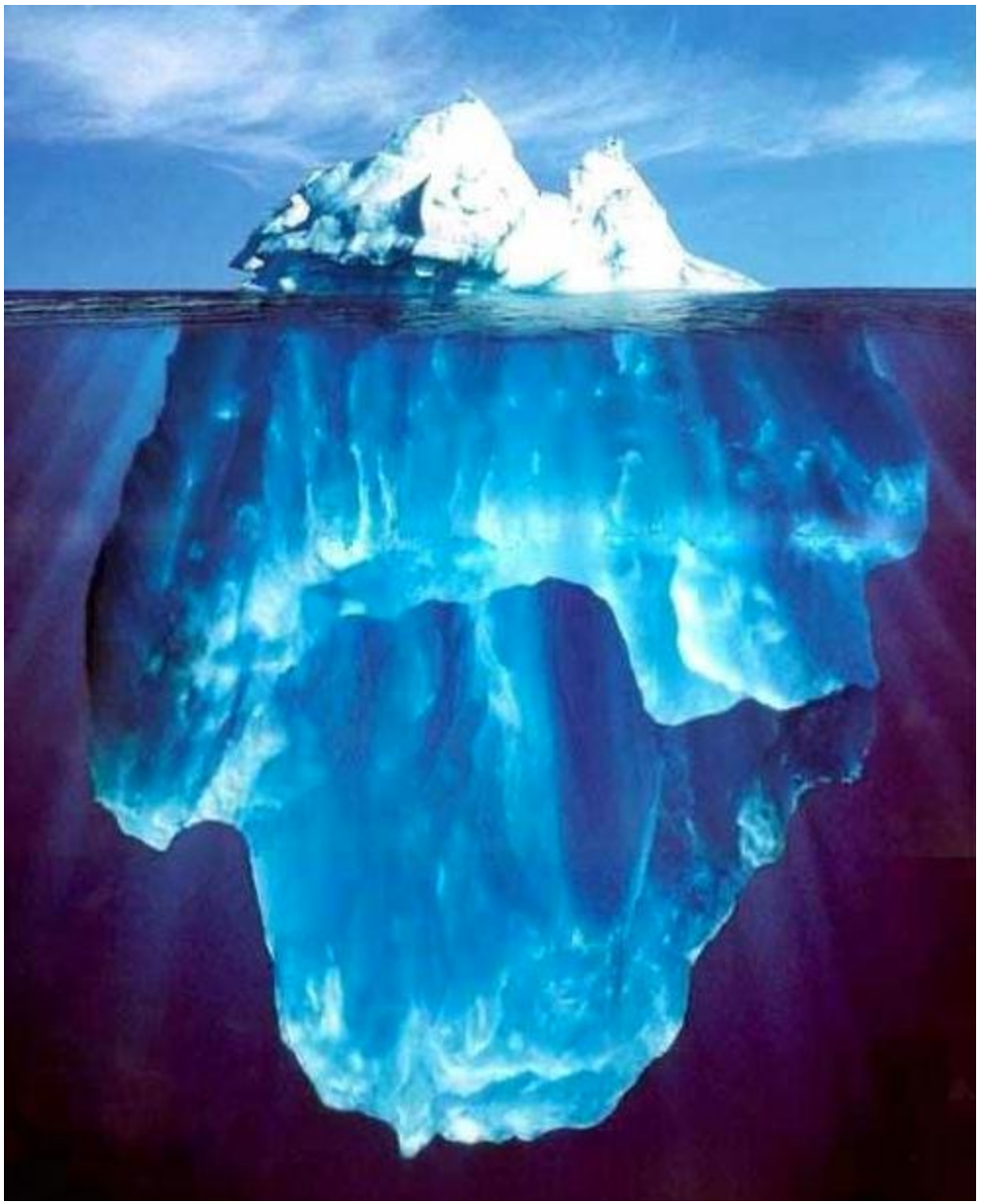
- **2nd Class:**
  - Discussion of any issues with user stories for current build cycle
  - Topic related to the projects will be presented
  - In-class exercises on topic

EECE 262 Development Cycle

- **3rd Class:**
  - New user stories are demoed in class
  - Each team tests each of the other teams' new application release
  - Each team reports bugs that they find in the other teams' applications
    - Bugs are reported in the team's project's bug tracker
  - Each team brainstorms ideas to improve the other teams' applications
    - Ideas for improvement are reported in the project wiki
  - Each team brainstorms ways of changing the architecture or approach of the other teams
    - Crazy ideas are reported in the project wiki

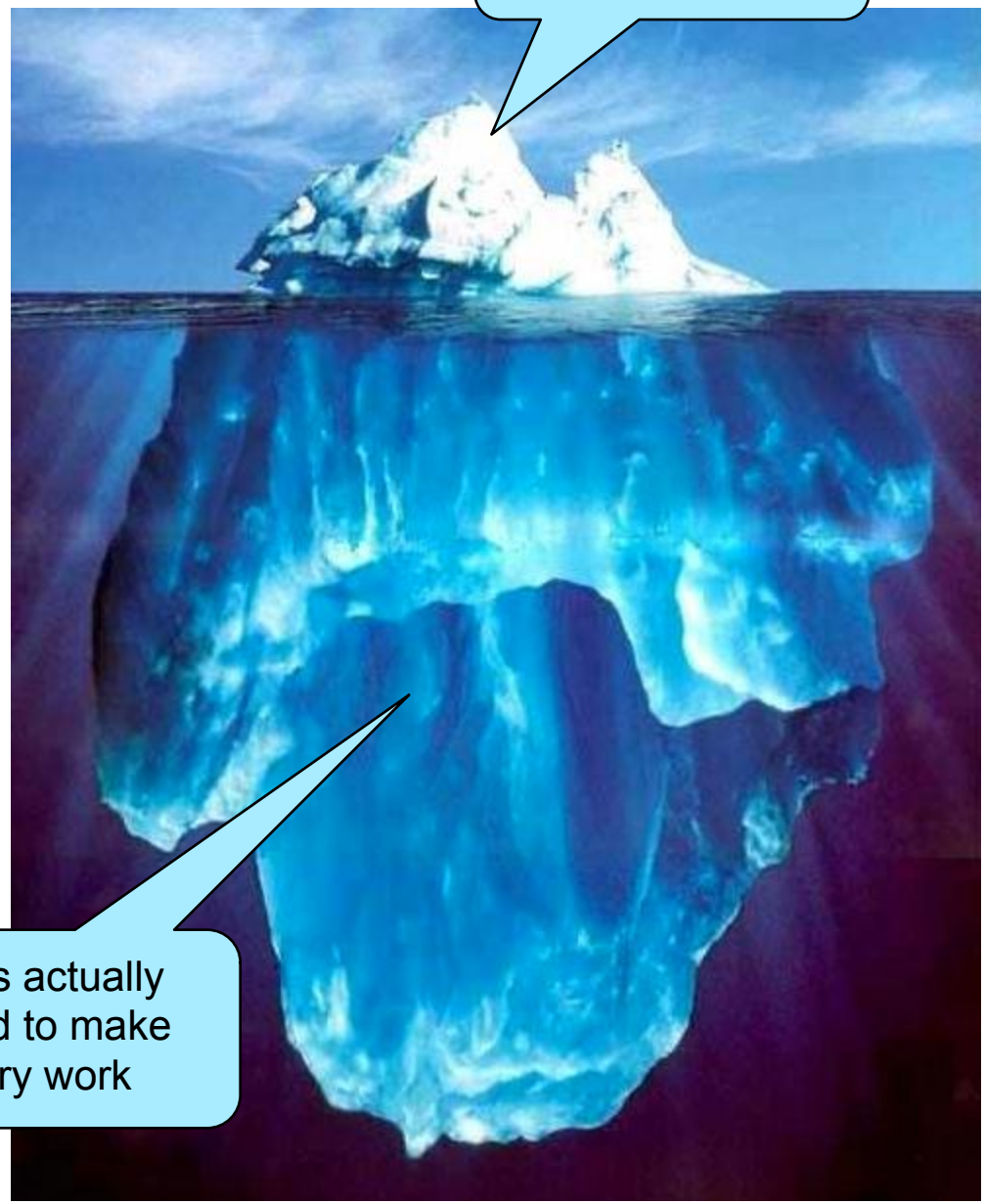
User Stories

- **What is a “user story”?**
- **A user story should be a short 1-2 sentence explanation of something that a user can do with the application:**
  - A student can add a new course to his/her schedule
  - A player can view the results of a match
- **User stories must be assigned to team members**
- **Team members will be graded on their assigned user stories & integrated functionality**



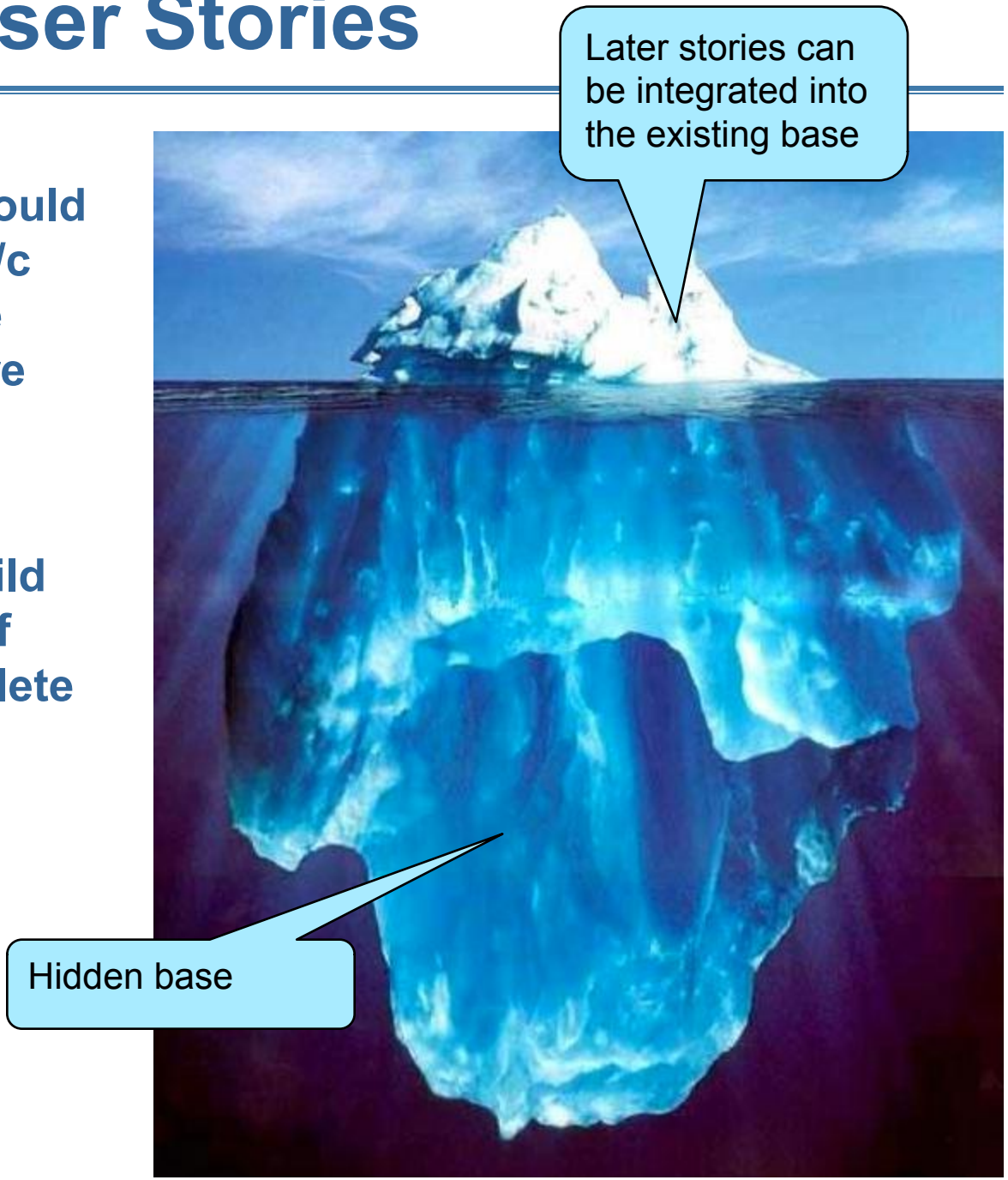
User Stories

- Each user story will be simple but will require a lot of things to work under the hood
- User stories emphasize working fully integrated software rather than large bodies of un-integrated code
- At the end of the build cycle, if a user can't complete the story, it isn't finished



User Stories

- At the beginning, you should pick fewer user stories b/c you will need to build the "hidden base" of software beneath it
- Later, you can pump out more user stories per build cycle because the bulk of your application is complete



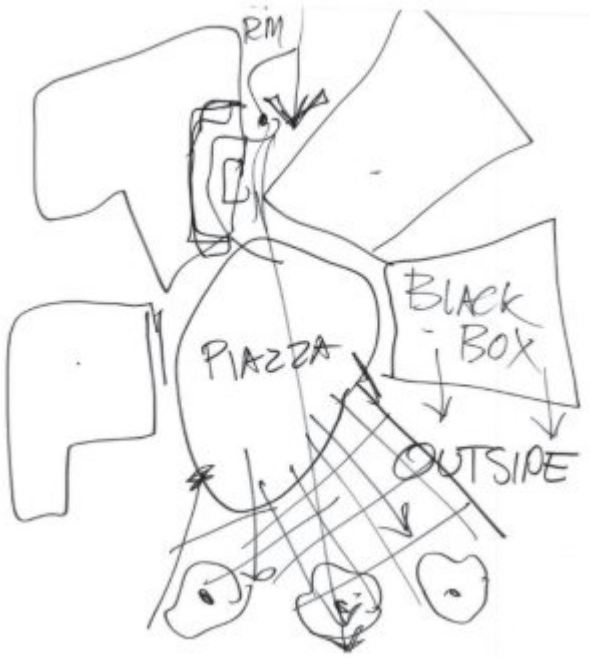
All Projects are Open Source

All code should be released under the Apache 2 license:

```
/*************************************************************************
* Copyright 2008 Jules White                                             *
*                                                                        *
* Licensed under the Apache License, Version 2.0 (the "License");        *
* you may not use this file except in compliance with the License.       *
* You may obtain a copy of the License at                                *
*                                                                        *
* http://www.apache.org/licenses/LICENSE-2.0                             *
*                                                                        *
* Unless required by applicable law or agreed to in writing, software    *
* distributed under the License is distributed on an "AS IS" BASIS,      *
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.*
* See the License for the specific language governing permissions and    *
* limitations under the License.                                         *
*************************************************************************/
```

Project Requirements

- Every group must maintain their project in Google code
- You must use SVN
- You must maintain a wiki that provides detailed instructions on how to build, run, and test your code
- You must actively use the bug tracker and wiki
- You must produce a binary distribution at the end of each build cycle



SVN Commit Rules

- **Rule #1: Never ever ever commit code that doesn't compile**

SVN Commit Rules

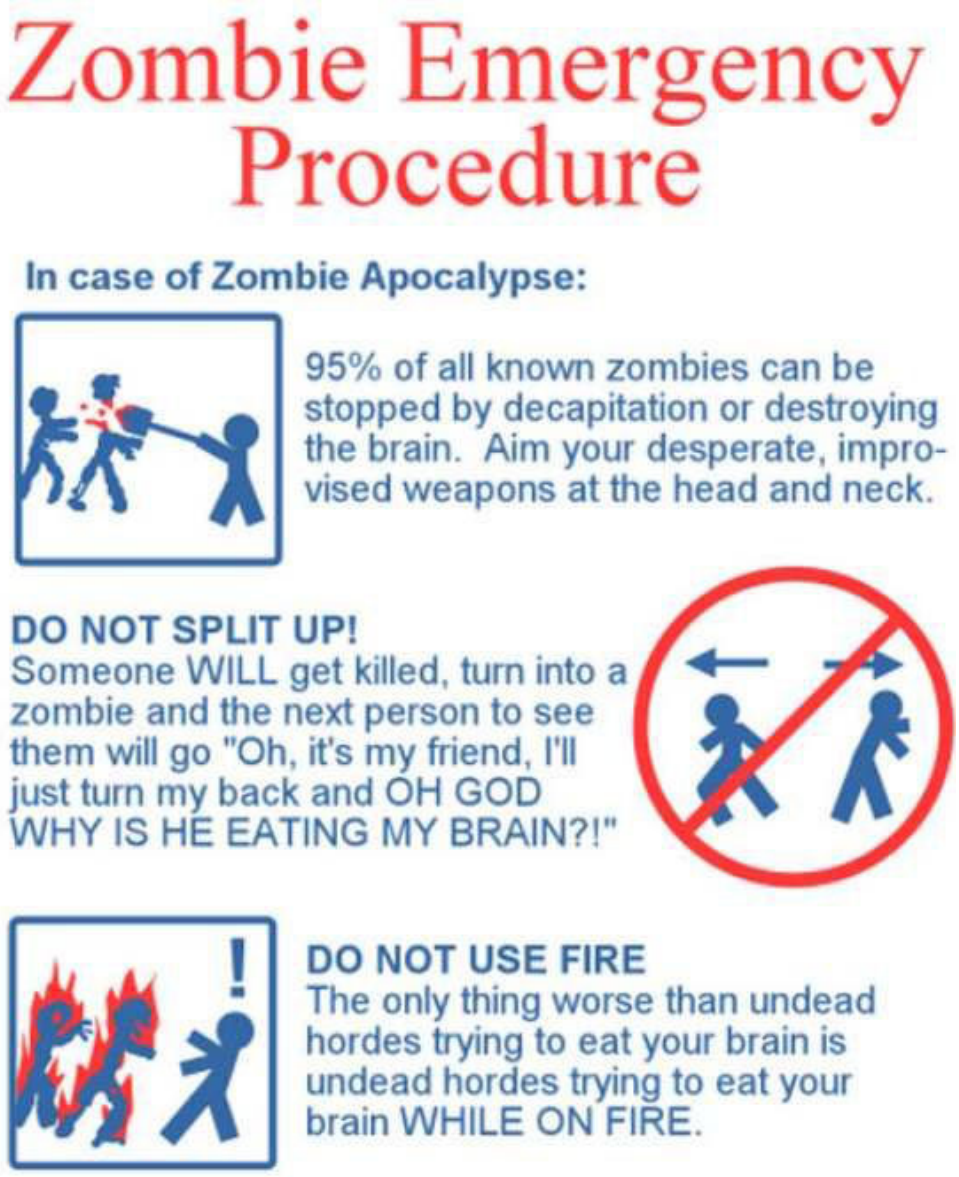
- **Rule #2: Always include a commit comment that briefly summarizes what changes you are checking in**

SVN Commit Rules

- **Rule #3: Always try to make sure your code passes the unit tests before checking it in**

Bugs

- If I check in code and you notice that it breaks something, you must report it as a bug in the bug tracker (issues in google code)
- Make sure that you provide sufficient information to reproduce the bug
- All bugs should be cleaned up by the end of the build cycle or used as a rational for rescoping a user story
- When you commit a fix for a bug, reference the bug tracking ID in the SVN commit comment



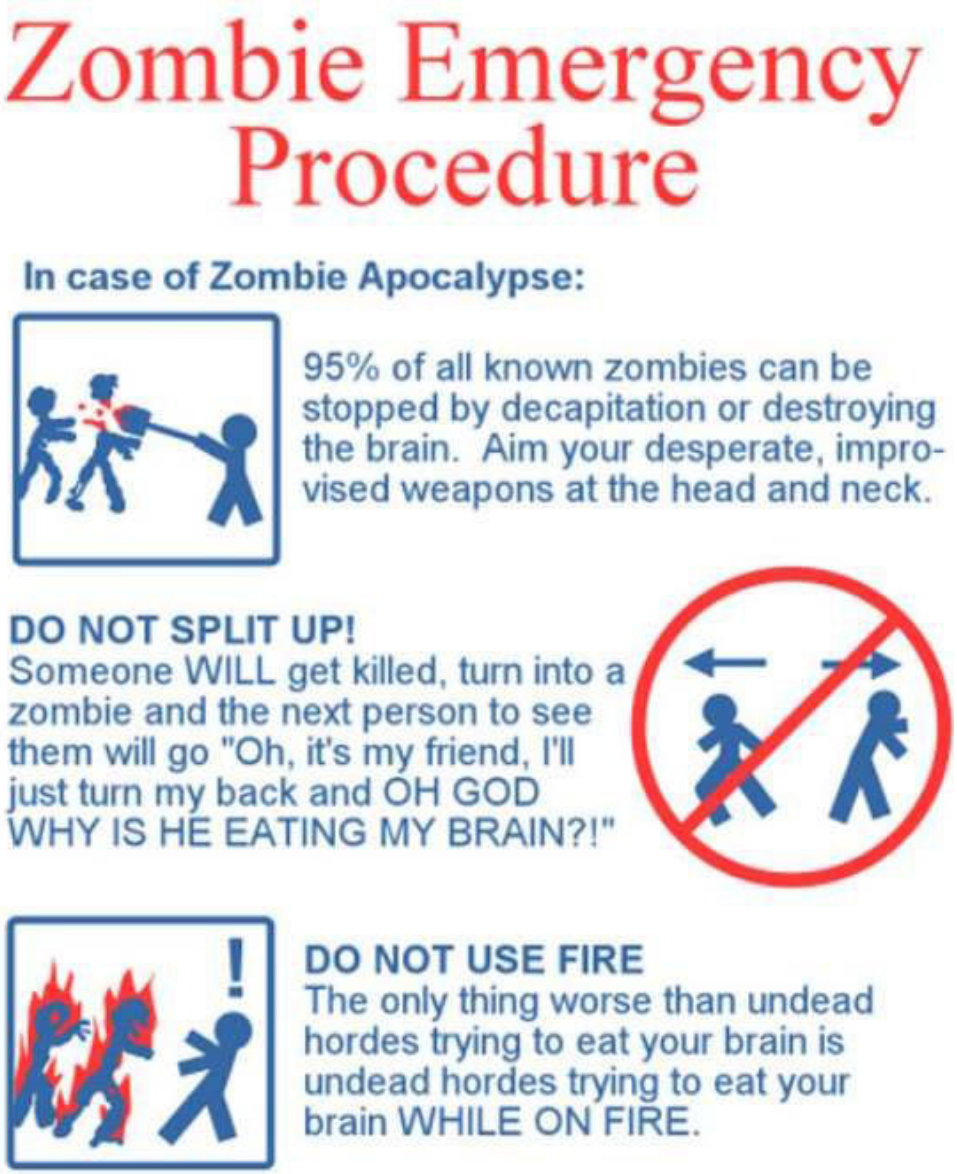
Bugs

- When you commit a fix for a bug, reference the bug tracking ID in the SVN commit comment
- Correct format:

Fixes for: 1878, 213, 7657

I removed the go to statements

that were causing these bugs.



Implementing User Stories

- **Only build the minimum of what is needed to realize the user story**
- **All code created during the build cycle should be directly traceable back to a user story**
- **On the 2nd Tuesday, we will do in class code reviews**
  - I will do code reviews for anyone who doesn't have their code reviewed in class
- **Code will need to be refactored by the following Thursday per the code review recommendations**

Implementing User Stories

- At the beginning, it is ok to “fake” or use mock objects for parts of the implementation
- For example, you may want to fake the communication with a remote server by creating a mock object that automatically returns the expected answers or stock data

What if I Just Can't Get X to Work?

- **If you realize that a user story is much harder than expected to implement, don't panic**
  - Discuss the issue with your group and send me email saying that you are going to postpone the user story until the next build cycle
  - Prioritize your other user stories and finish them
  - At the latest, you must notify me by the start of the 2nd class
- **Start early so that you can predict if you aren't going to finish a user story**
- **If you have a midterm, etc. during a build cycle, go easy on yourself and pick easy/fewer user stories**

In-class User Acceptance Testing

- On the last class of a build cycle, we will first let each team demo their working user story implementations
- Groups will then test each others' user story implementations
  - Every group will be required to have a binary distribution that other groups can download to test
  - Groups must have all usage directions posted on their project wiki (no hand holding)
  - Groups can bring in user surveys to get feedback from users(optional)

Binary Distributions

- **A binary distribution should be a compiled version of the code that can be run fairly easily by a user**
- **Examples:**
  - An Android APK file
  - A jar file, launch script, and instructions (always include a license file too)
  - A Java launcher, such as launch4j
  - An Eclipse plugin distribution
  - A set of project binaries and an ANT file to run them

Build Cycle Grading

- (40pts) Were all of the student's user stories completed or properly postponed?

OR

- (40pts) (for non-coders) Did the student complete all of their design and documentation activities?

AND

- (20pts) Did the student properly report and address bugs?
- (20pts) Did the student test and report bugs for the other projects?
- (20pts) Did the student report ideas/refinements for other projects in their wiki?
- (10pts Bonus) Did you bring up a cool new topic in class and provide examples for it?
- ***I reserve the right to change the weighting/grading criteria during the semester