

# Software Engineering via Service-Learning



## A "Center for Multicultural Cooperation" Case Study



<http://zimmer.csufresno.edu/~shliu>

**Instructors: Shih-Hsi "Alex" Liu and Yu Cao**

Department of Computer Science, California State University, Fresno  
{shliu, yuca}@CSUFresno.edu

**Students: Tim Alvord, Josh Pena, Thell Smith, Christopher Weatherby, and Brandon Wilson**

Department of Computer Science, California State University, Fresno  
{scool007, gomer555, thell, cweatherby, radioactiveman}@CSUFresno.edu

### Problem Statements

- Software Engineering is one of the most profound realms in Computer Science:
  - Communication and comprehensibility with clients
  - Design and development of software
  - Team organization and communication
  - Management of budget and deliverables
- "The hard thing about building software is deciding what one wants to say, not saying it" by Fred Brooks Jr. is one of the most essential difficulties to be tackled.
- Conventional course settings do not really exercise this essential difficulty. Instead, an instructor usually assigns projects along with software requirements properly defined for students to implement.
  - Students lose opportunities to learn about how to extract and reify software requirements.
  - Students lose opportunities to solve the real-world problems in computing.

### Requirements and Analysis with Service-Learning



- How service-learning components are included in the software development process?
  - Nonprofit organization (NPO): Provide students an opportunity to develop a real-world software product that meets NPO's needs
  - Students:
    - Learn Software Engineering principles, concepts, and tools in class and other techniques off class to accommodate NPO's product needs
    - Dedicate service-learning hours by developing the product for NPO
  - Similarity: Like traditional service-learning, it is "learning through service"
  - Difference: Unlike traditional service-learning, our service candidates "are" NPOs (not "through" NPOs)
  - Realization: Meet with NPO during requirements and analysis workflows. Demonstrate and deploy the software product to NPO by the end of semester
- Students interview with Center for Multicultural Cooperation (CMC) three times for software requirements extraction and analysis
  - First time:
    - Self introduction among students and CMC members
    - Find out the current hardware and software configurations of CMC's computing devices
    - Get familiar with the domain of CMC.
    - Discuss existing product and directions of future product
  - Second Time:
    - Clear ambiguities from last meeting
    - A rapid prototype gives CMC an insight of the product and provides students an insight of what/how to build
    - Discuss features to be added and changed
  - Third time:
    - Clear ambiguities from last meeting
    - Finalize features and data needed



A rapid prototype

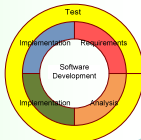
### Deliverable

- After a semester-long effort, including three interviews, several email and phone communications and numerous hours on designing, coding, testing, students delivered a Web-based volunteer management system that exactly meets CMC's needs:
  - Allow Web access all over the world
  - Manages over 3000 volunteers
  - Add/Edit/Delete volunteers, donors, programs, affiliations...
  - Provide search results with given criteria
  - Generate reports
- "This really is an amazing tool that would have cost us a lot of money, and it's going to bring us to a whole new level of organization/communication." from Brandon



### Software Development

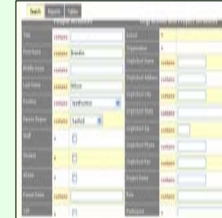
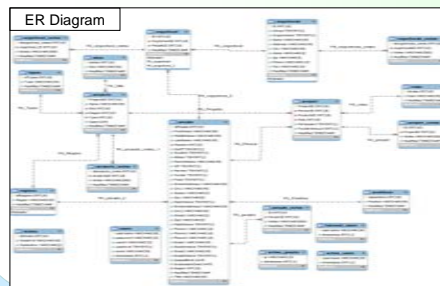
- When developing a software product, there is a set of major steps to follow, each of which focuses on accomplishing specific objectives:
  - Requirements: Acquire basic understanding of the application domain; Extract and elicit what clients need to computerize; Find out from clients what constraints exist.
  - Analysis: Analyze and refine the requirements to achieve detailed understanding of "what" the software product does. The artifacts can be used as a contract between client and development team and as a documentation for the next step.
  - Design: Refine the analysis artifacts by explicitly expressing how to develop the software product including its architecture and algorithms
  - Implementation: Implement the software product
  - Test: methodologically perform executable and non-executable testing to find out defects in artifacts listed above
- The steps are iteratively and incrementally performed (in parallel) until each artifact is satisfactory



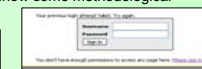
### Design, Implementation and Test with Service-Learning



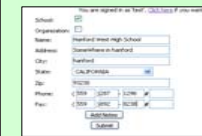
- After extracting, reifying and analyzing software requirements from CMC, students design, implement, and test the software product:
  - Design:
    - A database ER diagram is designed
    - A statechart diagram is designed
    - A workflow diagram is designed
    - The above diagrams complementarily show the software product's static structure and interrelationship as well as dynamic use-case driven execution flows
  - Implementation and Integration:
    - Based on the artifacts generated from requirements, analysis, and design, students break the software product to be built into smaller modules. Each student concentrates on realization of specific modules and user interfaces
    - Since CMC needs a Web-based volunteer management system, students choose PHP and MySQL as their implementation language and database management system, respectively.
    - After each module is done, students gather together to integrate all the pieces together
  - Test: The following diagrams show some methodological tests of modules after integration.



Edit contact



Login with incorrect username/password



Add New School Info

### Conclusion and Credits

#### Conclusion

- The advantages of Service-Learning projects introduced to CSCI are three-folds:
  - CSCI Dept: A synergistic means to convey discipline knowledge to students, to conduct software studios/labs w/ real-world projects, and to convince students the usefulness of the knowledge and skills learnt in CSCI
  - Students:
    - Experience with developing real-world software projects within academic settings;
    - Exercise interview and negotiation techniques by asking NPOs questions to find out their needs, scope, and constraints
    - Envision design, implementation, testing earlier
    - Self-learn latest industry techniques that may not be covered in academia
    - Real-world deliverable that can be shown to future employer
  - NPO: A free customized software specific for NPO

#### Credits

- CMC Supervisors: Brandon (Assistant Director), Mary Jane (Technical Lead), Nyeland (Director),
- Service-Learning Mentors: Chris Fiorentino, Trish Studt, Sally Tannenbaum