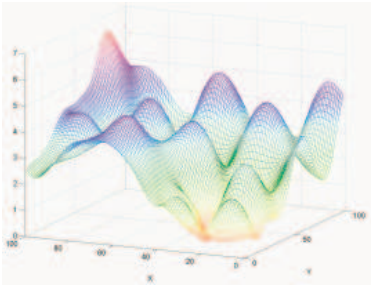
Node Placement and Monitoring in a Wireless Sensor Network

Group 5:
David Bangerter
Matt Laroche
Melissa Ludowise
Ben McCann

Clients:
Andreas Krause
Professor Carlos Guestrin
(SELECT Lab)



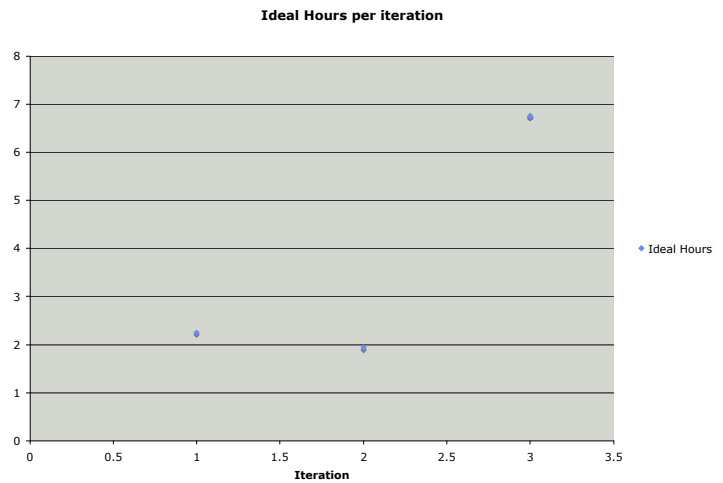
Project Overview

- For a given floor plan, what is the optimal placement of sensors to measure light (or temperature, or even something else)?
  - Problem is NP-complete
- Work done by our clients: use current sensor data and machine learning algorithms to learn a probabilistic model to optimize the placement.
- Our job: develop a system that would do the following…

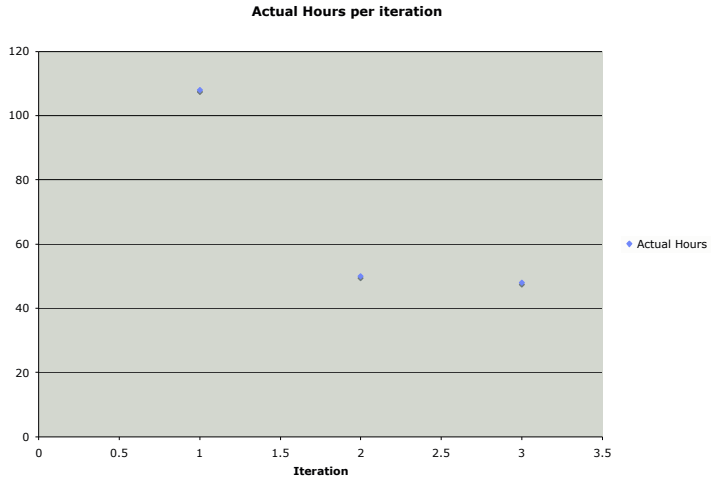
Our Requirements

- Read in data from the sensors (measurements, sensor location, etc) and store this information in a database.
- Utilize existing machine learning algorithms (written by our clients in MATLAB) which optimize sensor placement.
- Develop a GUI to visualize the floor plan, current sensor placements and readings, as well as suggested placements from the learned model.

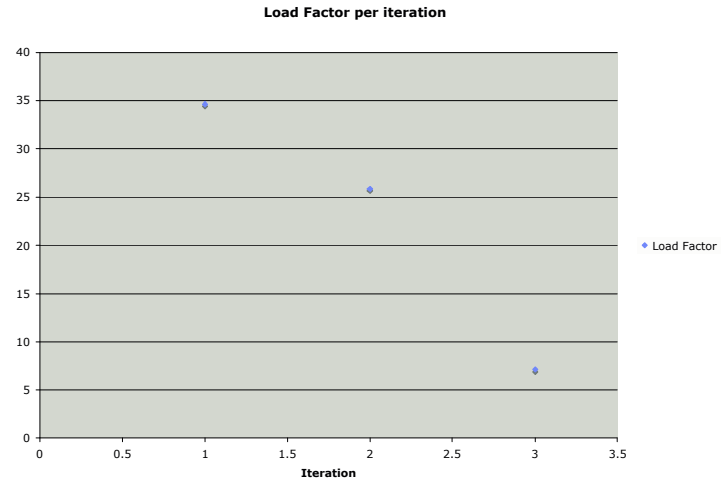
Engineering Metrics
Ideal hours per iteration



Engineering Metrics
Actual hours per iteration



Engineering Metrics
Load factor per iteration



Project status versus initial picture of success

- We wanted to be done with everything by start of finals
- We wanted the client to find our tool more usable than executing the old scripts manually
- We wanted to have clean code that would be easily extendable and professional, so it could eventually be open sourced, released, and updated by the community
- Due to changing requirements, this picture changed drastically

Changes

- Realized many of our initial ideas for interfacing Matlab and Java were infeasible
- Client requested the development of a “Test GUI”, a GUI separate from the rest of the project that would allow the user to synthesize light patterns on a projector in order to test sensors.
- Test GUI was not initially a requirement and then became very important
  - Was a factor in our inability to successfully complete application GUI
    - Should have application GUI as a “high risk story” and reserved the right to do it first under XP
  - The addition of the test GUI caused us to deliver something very different from our original goals

Successes

- Have successfully created ability to call Matlab from Java
- Created strong backend architecture and code to handle persistence
  - Hibernate Object-Relational persistence layer backed by MySQL
- Code is clean, robust, and should be able to be handed off to our clients
- Have built a functioning Test GUI

Test GUI Demo

Room for Improvement

- Will not be able to complete all requirements
  - Might have realized this earlier and possibly completed the GUI instead of some of the backend code
  - Some problems were unavoidable as team members' priorities changed, travel and conflicts arose, and scheduling group meetings became difficult

Room for Improvement – 2

- We made the mistake of having one person do all the GUI code in parallel with the backend
  - Coupled with infrequent check-ins, this lead to a late realization of the low probability of integration

Status vs. Picture of Success

- Have successfully created ability run clients' Matlab scripts from Java
- Code is clean, robust, and should be able to be handed off to our clients
- Application GUI will not be finished
  - But we've delivered something not originally required

Lessons Learned

- Pair programming is not easily executed in a college project
  - It is hard to find long blocks of free time for 2
  - Good alternative: code alone and have every check in peer reviewed
    - Give every reviewer the right to refuse check-in until code passes guidelines

Lessons Learned - 2

- Other XP procedures are excellent however
  - Frequent check-ins led towards increased shared code ownership
  - Frequent check-ins show progress
- Portions requiring significant client input such as GUI should be handled earlier in the project