cs121 - software development introduction

alexandre r.j. francois
visiting associate professor of computer science

© arjf 2010
outline

course objectives
topics covered
project
interactive systems
examples
project ideas
objective

introduction to
tory and practice
of software design and development...

... in a fun and practical way!

why it is worth studying:
- software is ubiquitous
- software failures can be costly and dangerous
- software design and implementation is hard!
challenge
key processes of software development

requirements
design
implementation (cs70)
testing
maintenance

software life cycle: how to organize the key processes of software development
waterfall does not work!

initial requirements are speculative
initial design is speculative

speculative decisions compound:
  build the wrong thing,
  which does not work anyways!
iterative models

e.g. Boehm's spiral model (1988)
each iteration is like a mini-waterfall
  identify the objectives of the iteration
  design a solution to achieve the objectives
  implement the solution
  test the implementation

other example: Rational Unified Process (IBM)

problem: iterative progress tends to assume monotonic convergence towards solution
agile principles

working software, delivered regularly, is the primary measure of progress
high standards of excellence: test regularly and re-factor/redesign when necessary
customer involvement is critical
simplicity: just-in-time design/development
adaptability: embrace change
small, cross-functional, self-organizing teams of professionals

example: SCRUM
at the end of the day...

software projects come in all shapes and sizes (and so do software professionals)
different methods work well for some types of projects and not at all for others

**cs121 objectives:**
understand the problems
understand various solutions to the problems
practice applying solutions to a particular problem
topics covered

**design** (for usability)
requirements
architecture (system, components, code)
code design
testing

teamwork, reviews, documentation, etc.

**people aspects of software!**
project

interactive system
classwide project
YOU (we) define the project
  we are our own customers!
methodology

“design” approach
- divergence / convergence phases
- prototyping
- document history (non linear) - portfolio

collaborative enterprise
- synergy
- diversity of expertise and interests

agile iterative approach (with design twist)
- fluid structure
tools – how familiar are you with...

Google Docs
Trac: Wiki, version control, tickets
UML tools(?)
Doxygen
SAI/MFSM
C++
Emacs + gcc?
Xcode?
Google Test?
...

© arj 2010
interactive systems

involve humans
complex (no matter how complicated...)
situated
embodied
dynamic
desirable behavior
robust
adaptive
context-dependent
challenges

by humans for humans
complex task
beyond straightforward computation

system design
cross-disciplinary teams

system implementation, testing and maintenance
distributed in time and space

this is mostly about humans
programming models and languages are machine-centric
all concepts exist in time
programming models and languages abstract time
applications
- vision -

virtual mirror (2000-2002) gra: e.kang
[siggraph2002]

video painting (1999)
[racquetball tracking (2001)]

[cviu2007]

people detection and tracking (2004)
funding: arda

stevi (2005-2006) pi: g.medioni, funding: etri

© arf 2010
applications
-music-

music computation and cognition laboratory at usc

esp (2004-) co-pi: e.chew, gra: j.liu

mimi (2006-) co-pi: e.chew d.thurmond

musa.rt (2002-) co-pi: e.chew

[cie2005]
[nime2005]
[nime2006]
interactive games!

interactive systems
  specific requirements
  rich and fun history and industry
  source of innovation
  fun!

evolution of interaction
  input devices and modalities
  graphics
  sound/music
  haptics
project ideas – first round

list here: ...