Spring-11: PDC in CS1/2 and a mobile/cloud intermediate software design course

J. Kaylor • K. Läufer • C. Sekharan • G. K. Thiruvathukal
Dept. of Computer Science • Loyola University Chicago • USA
www.cs.luc.edu • laufcr@cs.luc.edu • edupar.cs.luc.edu

Abstract/Current Focus
AY 2012-13: We are moving PDC topics further down into CS1 and CS2, fleshing out PDC coverage in our intermediate object-oriented development course (CS 313), and stepping up evaluation.

Prior Outcomes
Spring 2011: We implemented three three-week PDC course modules (20% of our 15-week semester) targeting three required courses usually taken in the second year.
AY 2011-12: We implemented four three-week advanced PDC course modules in programming and distributed computing targeting electives typically offered every three semesters.

Institutional Profile
• Urban, private, Jesuit, liberal arts, ~16k students
• College of Arts and Sciences, ~8k students
• Dept. of Computer Science, ~250 students
• 10 FT faculty: 9 CS, 1 bioinf (1/2 FTE), 1 algorithmist (1/4 FTE)
• ~140 undergrad majors in CS, SE, IT, Networks & Security, bioinformatics
• ~110 master’s students in CS, SE, IT
• External funding: NSF S-STEM, NSF BPC lead institution, NSF research grants, industry grants and donations
• Very early adopters: ‘97 concurrency course, OOPSLA ‘98 edu symp paper, HPJC book

Carnegie Classification
• Level: 4-year or above
• Control: Private not-for-profit
• Undergraduate Instructional Program: BAI/HGC
• Graduate Instructional Program: CompSci/MedVent
• Enrollment Profile: MU
• Undergraduate Profile: FT4/MS/HTI
• Size and Setting: L4/R
• Basic: RU/H: Research Universities (high research activity)
• Community Engagement: Curricular, Outreach, Partnerships
*CS/SE/IT: up to masters’ level

Where our Graduates Go
• Industry ~ 80%
  • midwest, coasts, international
  • consulting, finance, software, telecom, ...
• Academia and Government ~ 15%
  • Argonne, county admin, local universities
• Graduate School ~ 3%
  • local, national
• Professional Schools ~ 2%
  • business, law, medical

CS1
• Spring 2013: separate section aimed at majors in physics and other hard sciences
• includes some material on numerical methods at the K and C levels
• about 9 class hours were dedicated to sequential and parallel versions of these algorithms and the possible resulting speedup using data parallelism in C#
• example: threads for speeding up trapezoidal rule integration

Position Statements
[for discussion]
• To teach PDC topics effectively, they should not be taught in isolation. Instead, they should be taught in the context of relevant software engineering best practices.
• Mobile applications backed by RESTful cloud services are rapidly emerging as the mainstream paradigm of computing [Christensen, OOPSLA ’09]. PDC curricula should embrace it.

Eval: Learning Assessment
Sample concurrency questions:
Suppose we have two philosophers. The first one, Kant, repeatedly behaves like so:
think for 10 minutes
wait for any available fork and grab it when available
think for 2 minutes
wait for any available fork and grab it when available
release both forks
The second one, Heidegger, repeatedly behaves like so:
think for 15 minutes
wait for any available fork and grab it when available
think for 5 minutes
wait for any available fork and grab it when available
release both forks
Suppose Kant and Heidegger sit at the same table with two forks available and start their respective behaviors at the same time. Give an event trace such as
When type of unavoidable situation does this scenario illustrate?
• A. lots of thread safety
• B. run-time type error
• C. memory leak
• D. deadlock
What are possible ways of avoiding this kind of unavoidable situation?
• use an explicit locking mechanism to enforce mutually exclusive access to each fork
• update only the fork(s) last used or both forks
• some fork would be a single resource bundle that must be acquired together at the same time
• provide a fork and a knife instead of two forks and reverse the behavior such that each philosopher must acquire the fork first

Eval: Course Effectiveness
Sample questions (Likert scale): Rate BEFORE/AFTER taking this course
• your event-based programming expertise
• your thread-based concurrency expertise
• your cloud computing expertise
• your feeling of preparedness for the job market
• the effectiveness of Android for learning event-based programming
Results available at laufcr.ca.luc.edu/teaching/edupar (QR code above).

Intermediate Object-Oriented Development
• Loyola course number: COMP 313
• since fall 2011: C#, emphasis on PDC topics
• since fall 2012: Java with Android as highly effective context for studying concurrency and distributed computing topics (C and A levels)
• double 18-hour PDC module covers
  • concurrency and coordination
  • design forces: safety, liveliness, performance
  • external events and internal timers
  • background threads
  • progress reporting and cancellation
  • offloading computation from mobile device to cloud
  • throughout-latency tradeoff
• example to illustrate these topics (see screenshot)
• Android app: brute-force prime checker
• remote task returns quickly, local one is still churning

Future Plans
Computer Systems/CS3
• Loyola course number: COMP 364
• offered every spring
• envisioning PDC module with suitable architecture, programming, and cross-cutting topics

Algorithms/CS7
• Loyola course number: COMP 363
• offered every fall
• developing PDC module: models of computation and complexity, basic algorithmic paradigms, and specific problems and their algorithmic solutions

Evaluation
Once our course modules have stabilized, we will need to measure their effectiveness longitudinally over a three- to five-year period.
• Refine our current evaluation approach by working with Loyola’s Center for Science & Math Education, as well as the TCPP and fellow early adopters.

Dissemination:
workshops for subsequent adopters in the Midwest