university of notre dame
computer science and engineering

Experiences and Results from three Years of CSE 211 “Fundamentals of Computing I”

Matthias Scheutz

Department of Computer Science and Engineering

University of Notre Dame

http://www.nd.edu/~mscheutz/

Outline

- Pre-text of CSE at Notre Dame

Outline

- Pre-text of CSE at Notre Dame
- Structure of CSE 211 – Fundamentals of Computing I

Outline

- Pre-text of CSE at Notre Dame
- Structure of CSE 211 – Fundamentals of Computing I
- 3 Hypotheses about Student Performance in CSE 211

Outline

- Pre-text of CSE at Notre Dame
- Structure of CSE 211 – Fundamentals of Computing I
- 3 Hypotheses about Student Performance in CSE 211
- Results and Conclusions

Pre-text
CSE at Notre Dame

- CSE curriculum only 3 years, because all engineering students take “first year engineering sequence”

Pre-text
CSE at Notre Dame

- CSE curriculum only 3 years, because all engineering students take “first year engineering sequence”
- Hence, core CS/CE materials that are usually spread over two years at institutions with 4 year programs have to be covered **as quickly as possible**

Pre-text
CSE at Notre Dame

- CSE curriculum only 3 years, because all engineering students take “first year engineering sequence”
- Hence, core CS/CE materials that are usually spread over two years at institutions with 4 year programs have to be covered **as quickly as possible**
- The new CSE 2002 undergraduate curriculum at Notre Dame (ND02)

Pre-text
CSE at Notre Dame

- CSE curriculum only 3 years, because all engineering students take “first year engineering sequence”
- Hence, core CS/CE materials that are usually spread over two years at institutions with 4 year programs have to be covered **as quickly as possible**
- The new CSE 2002 undergraduate curriculum at Notre Dame (ND02)
- Modelled based on IEEE/ACM Computing Curricula 2001 (CC2001)

Pre-text
CSE at Notre Dame

- CSE curriculum only 3 years, because all engineering students take “first year engineering sequence”
- Hence, core CS/CE materials that are usually spread over two years at institutions with 4 year programs have to be covered **as quickly as possible**
- The new CSE 2002 undergraduate curriculum at Notre Dame (ND02)
- Modelled based on IEEE/ACM Computing Curricula 2001 (CC2001)
- No single CC2001 proposal fits without adaptation

Curriculum 2001 Y1&Y2

Semester 1	Semester 2
Year 1	
Programming Fundamentals Calculus	The Object-Oriented Paradigm Discrete Structures for CS Calculus II
Year 2	
Data Structures and Algorithms Science course I	Intr. to Computer Organization Science course II Probability and Statistics

university of notre dame
computer science and engineering

Curriculum 2001 Y1&Y2 vs. ND 02

Semester 1	Semester 2
Year 1	
Programming Fundamentals Calculus	The Object-Oriented Paradigm Discrete Structures for CS Calculus II
Year 2	
Data Structures and Algorithms Science course I	Intr. to Computer Organization Science course II Probability and Statistics

Semester 1		Semester 2	
MATH 125	Calculus I (4)	MATH 126	Calculus II (4)
CHEM 121	General Chemistry I (4)	CHEM 122	General Chemistry II (3)
EG 111	Intro to Engineering I (3)	EG 112	Intro to Engineering II (3)
Arts & Letters Elective (3)		PHYS 131	General Physics I (4)
Composition (3)		University Seminar (3)	
CSE211	**Fund Computing I (4)**	CSE212	Fund Computing II (4)
CSE210	Discrete Mathematics (3)	CSE221	Logic Design (4)
MATH 225	Calculus III (3.5)	MATH 228	Linear Algebra & Diff Eq (3.5)
PHYS 132	General Physics II (4)	Technical Elective (3)	
Philosophy/Theology (3)		Philosophy/Theology (3)	

Curriculum 2001 Y3&Y4

Semester 1	Semester 2
Year 3	
Algorithm Design and Analysis Computer Architecture Advanced mathematics elective	Operating Systems Social and Professional Issues CS elective Undergraduate research project
Year 4	
Net-centric Computing Information and Knowledge Man. Software Development Undergraduate research project	Capstone Project CS elective CS elective

university of notre dame
computer science and engineering

Curriculum 2001 Y3&Y4 vs. ND 02

Semester 1	Semester 2
Year 3	
Algorithm Design and Analysis Computer Architecture Advanced mathematics elective	Operating Systems Social and Professional Issues CS elective Undergraduate research project
Year 4	
Net-centric Computing Information and Knowledge Man. Software Development Undergraduate research project	Capstone Project CS elective CS elective

Semester 1	Semester 2
CSE331 Data Structures (3) CSE321 Computer Architecture I (4) Technical Elective (3) CSE Elective (3) Philosophy/Theology (3)	CSE341 Operating Systems (3) CSE351 Theory of Computing (3) MATH 328 Probability & Stats (3) CSE Elective (3) Philosophy/Theology (3)
CSE413 Algorithms (3) CSE Elective (3) CSE Elective (3) Technical Elective (3)	CSE475 Ethical and Social Issues (3) CSE Elective (3) CSE Elective (3) Arts & Letter Elective (3)

US4Y $\subset$ ND02

- The US4Y curriculum requires 15 CSE courses, 6 math and science courses, 2 undergrad project courses and one capstone

US4Y $\subset$ ND02

- The US4Y curriculum requires 15 CSE courses, 6 math and science courses, 2 undergrad project courses and one capstone
- ND02 requires 16 CSE, 9 math and science courses, 3 tech electives (for undergrad research and capstone), and 2 course introduction to engineering sequence

US4Y $\subset$ ND02

- The US4Y curriculum requires 15 CSE courses, 6 math and science courses, 2 undergrad project courses and one capstone
- ND02 requires 16 CSE, 9 math and science courses, 3 tech electives (for undergrad research and capstone), and 2 course introduction to engineering sequence
- Thus, ND02 meets the CC01 requirement for US4Y

CSE 211 "Functional First"

- CSE 211 (3 cr) based on CSE 233 *Functional Programming* plus new mandatory lab section (1cr)

CSE 211 "Functional First"

- CSE 211 (3 cr) based on CSE 233 *Functional Programming* plus new mandatory lab section (1cr)
- CSE 233 was essentially based on MIT's 6.001 *Structure and Interpretation of Computer Programs*

CSE 211 "Functional First"

- CSE 211 (3 cr) based on CSE 233 *Functional Programming* plus new mandatory lab section (1cr)
- CSE 233 was essentially based on MIT's 6.001 *Structure and Interpretation of Computer Programs*
- Appealing was the book's quick focus on procedural and data abstraction and its methodological goal of keeping new syntactic constructs to a minimum, which is facilitated by using SCHEME

CSE 211 "Functional First"

- CSE 211 (3 cr) based on CSE 233 *Functional Programming* plus new mandatory lab section (1cr)
- CSE 233 was essentially based on MIT's 6.001 *Structure and Interpretation of Computer Programs*
- Appealing was the book's quick focus on procedural and data abstraction and its methodological goal of keeping new syntactic constructs to a minimum, which is facilitated by using SCHEME
- Typically, about half of the incoming CS and CE students have very limited or no programming background

Why SCHEME?

- SCHEME is a syntactically simple language

Why SCHEME?

- SCHEME is a syntactically simple language
- SCHEME is easy and quick to learn

Why SCHEME?

- SCHEME is a syntactically simple language
- SCHEME is easy and quick to learn
- SCHEME allows students to focus on programming concepts right away (rather than having to spend a significant time on learning syntactic constructs as is the case with syntactically complex languages like C++)

Why SCHEME?

- SCHEME is a syntactically simple language
- SCHEME is easy and quick to learn
- SCHEME allows students to focus on programming concepts right away (rather than having to spend a significant time on learning syntactic constructs as is the case with syntactically complex languages like C++)
- SCHEME does not advantage students with prior programming background in imperative languages likes C/C++

Topics in CSE 211

- Procedural abstraction, Recursion, Data abstraction

Topics in CSE 211

- Procedural abstraction, Recursion, Data abstraction
- Algorithms and problem-solving, Object-oriented paradigm

Topics in CSE 211

- Procedural abstraction, Recursion, Data abstraction
- Algorithms and problem-solving, Object-oriented paradigm
- Basic computability theory, Basic computational complexity

Topics in CSE 211

- Procedural abstraction, Recursion, Data abstraction
- Algorithms and problem-solving, Object-oriented paradigm
- Basic computability theory, Basic computational complexity
- Overview of programming languages, Fundamental programming constructs

Topics in CSE 211

- Procedural abstraction, Recursion, Data abstraction
- Algorithms and problem-solving, Object-oriented paradigm
- Basic computability theory, Basic computational complexity
- Overview of programming languages, Fundamental programming constructs
- Evaluation strategies, Software development methodology

Topics in CSE 211

- Procedural abstraction, Recursion, Data abstraction
- Algorithms and problem-solving, Object-oriented paradigm
- Basic computability theory, Basic computational complexity
- Overview of programming languages, Fundamental programming constructs
- Evaluation strategies, Software development methodology
- Machine level representation of data

Topics in CSE 211 Breakdown by lectures

Topics and Time Spent on Topics in CSE211 (out 41)

Graphs and trees (3)	Fundamental programming constructs (3)
Algorithms and problem-solving (2)	Fundamental data structures (6)
Recursion (5)	Basic algorithmic analysis (2)
Algorithmic strategies (2)	Fundamental computing algorithms (4)
Basic computability (1)	Overview of programming languages (1)
Declarations and types (1)	Abstraction mechanisms (2)
Functional programming (4)	Concurrency (2)
Software design (1)	Software tools and environments (1)
History of computing (1)	

CSE 211 Course Organization

- 50% for assignments (25% for weekly individual and 25% for five large group assignments)

CSE 211 Course Organization

- 50% for assignments (25% for weekly individual and 25% for five large group assignments)
- 5% for class participation and attendance (which is mandatory in both sections)

CSE 211 Course Organization

- 50% for assignments (25% for weekly individual and 25% for five large group assignments)
- 5% for class participation and attendance (which is mandatory in both sections)
- 15% for three short examinations after the first three chapters in SICP (5% each)

CSE 211 Course Organization

- 50% for assignments (25% for weekly individual and 25% for five large group assignments)
- 5% for class participation and attendance (which is mandatory in both sections)
- 15% for three short examinations after the first three chapters in SICP (5% each)
- 30% for the comprehensive final exam

Topics
Group Assignments

- Recursion and basic functions

Topics
Group Assignments

- Recursion and basic functions
- Recursion and data structures, multiple representations of data types

Topics
Group Assignments

- Recursion and basic functions
- Recursion and data structures, multiple representations of data types
- Pattern matching and tree search

Topics
Group Assignments

- Recursion and basic functions
- Recursion and data structures, multiple representations of data types
- Pattern matching and tree search
- Object-oriented programming

Topics
Group Assignments

- Recursion and basic functions
- Recursion and data structures, multiple representations of data types
- Pattern matching and tree search
- Object-oriented programming
- Program interpretation and register machines

Computing Infrastructure

- 2002: KAWA, XEmacs (on Sun UNIX workstations only), execution on the command line, editor with synatx highlighting

Computing Infrastructure

- 2002: KAWA, XEmacs (on Sun UNIX workstations only), execution on the command line, editor with synatx highlighting
- 2003/2004: DrSCHEME (available for UNIX, Mac OS X, Windows, Linux), integrated programming environment, synatx highlighting, different language extensions selectable, error highlighting, debugger, etc.

3 Hypotheses underwriting CSE 211

- CSE 211 is feasible (i.e., to include materials in the first introductory course that are intended for the second according CC01 without sacrificing the students' level of understanding of other materials)

3 Hypotheses underwriting CSE 211

- CSE 211 is feasible (i.e., to include materials in the first introductory course that are intended for the second according CC01 without sacrificing the students' level of understanding of other materials)
- SCHEME as a programming language eliminates advantages and/or disadvantages of students based on high school programming background

3 Hypotheses underwriting CSE 211

- CSE 211 is feasible (i.e., to include materials in the first introductory course that are intended for the second according CC01 without sacrificing the students' level of understanding of other materials)
- SCHEME as a programming language eliminates advantages and/or disadvantages of students based on high school programming background
- An appropriate programming environment (plus syntactically simple programming language) is critical to students' learning and perception of the course

Results

- No significant difference with respect to students' perception of overall teaching quality in three offerings

Results

- No significant difference with respect to students' perception of overall teaching quality in three offerings
- No significant difference in students' perception of what they learned in CSE211

Results

- No significant difference with respect to students' perception of overall teaching quality in three offerings
- No significant difference in students' perception of what they learned in CSE211
- Most students thought that they learned how to solve problems well (16/28 in 2002, 20/30 in 2003, and 25/38 in 2004), followed by very good skill development (11/28 in 2002, 5/30 in 2003, and 8/38 in 2004)

Results

- No significant difference with respect to students' perception of overall teaching quality in three offerings
- No significant difference in students' perception of what they learned in CSE211
- Most students thought that they learned how to solve problems well (16/28 in 2002, 20/30 in 2003, and 25/38 in 2004), followed by very good skill development (11/28 in 2002, 5/30 in 2003, and 8/38 in 2004)
- Attrition rates in CSE211 are very low: 5/36 (2002), 3/40 (2003), and 4/48 (2004)

H1: Feasibility

- Results based on student survey (4=disagree, 3.2=agree, 2.4=indifferent, 1.6=disagree, 0.8=strongly disagree

H1: Feasibility

- Results based on student survey (4=disagree, 3.2=agree, 2.4=indifferent, 1.6=disagree, 0.8=strongly disagree
- “I feel I have a good overview of different aspects of CS”: 2.9 (2002) vs. 3.1 2003

H1: Feasibility

- Results based on student survey (4=disagree, 3.2=agree, 2.4=indifferent, 1.6=disagree, 0.8=strongly disagree
- “I feel I have a good overview of different aspects of CS”: 2.9 (2002) vs. 3.1 2003
- “Have good idea of CS topics coming up later in ND02 Curr.:” 2.7 (2002) vs. 3.1 (2003)

H1: Feasibility

- Results based on student survey (4=disagree, 3.2=agree, 2.4=indifferent, 1.6=disagree, 0.8=strongly disagree
- “I feel I have a good overview of different aspects of CS”: 2.9 (2002) vs. 3.1 2003
- “Have good idea of CS topics coming up later in ND02 Curr.:” 2.7 (2002) vs. 3.1 (2003)
- “I feel my programming skills improved significantly”: 2.9 (2002) vs. 3.2 (2003)

H1: Feasibility

- Results based on student survey (4=disagree, 3.2=agree, 2.4=indifferent, 1.6=disagree, 0.8=strongly disagree
- “I feel I have a good overview of different aspects of CS”: 2.9 (2002) vs. 3.1 2003
- “Have good idea of CS topics coming up later in ND02 Curr.:” 2.7 (2002) vs. 3.1 (2003)
- “I feel my programming skills improved significantly”: 2.9 (2002) vs. 3.2 (2003)
- “I learned to decompose complex problems into simpler ones”: 2.9 (2002) vs. 3.1 (2002)

H2: Computing Background

- In 2002 and 2003 about 50% (15 out of 31 and 19 out of 37) had prior exposure to C++

H2: Computing Background

- In 2002 and 2003 about 50% (15 out of 31 and 19 out of 37) had prior exposure to C++
- Only 40% (18 out of 45) in 2004 had C++ experience (and 25/48 had no programming experience whatsoever)

H2: Computing Background

- In 2002 and 2003 about 50% (15 out of 31 and 19 out of 37) had prior exposure to C++
- Only 40% (18 out of 45) in 2004 had C++ experience (and 25/48 had no programming experience whatsoever)
- None of the students in all three offerings had prior experience with SCHEME

H2: Computing Background

- In 2002 and 2003 about 50% (15 out of 31 and 19 out of 37) had prior exposure to C++
- Only 40% (18 out of 45) in 2004 had C++ experience (and 25/48 had no programming experience whatsoever)
- None of the students in all three offerings had prior experience with SCHEME
- Correlations between prior programming experience and final grade (2004): $r = .11$

H2: Computing Background

- In 2002 and 2003 about 50% (15 out of 31 and 19 out of 37) had prior exposure to C++
- Only 40% (18 out of 45) in 2004 had C++ experience (and 25/48 had no programming experience whatsoever)
- None of the students in all three offerings had prior experience with SCHEME
- Correlations between prior programming experience and final grade (2004): $r = .11$
- Correlating C++ and final grade (2004): $r = .18$

H3: Programming Environment

- Most differences in student ratings between 2002 & 2003

H3: Programming Environment

- Most differences in student ratings between 2002 & 2003
- Since all course materials were almost identical (including assignments and exams), the difference must have come from the programming environment

H3: Programming Environment

- Most differences in student ratings between 2002 & 2003
- Since all course materials were almost identical (including assignments and exams), the difference must have come from the programming environment
- “The SCHEME implementation worked well for me:” 2.4 (2002) vs. 2.9 (2003) p=.005

H3: Programming Environment

- Most differences in student ratings between 2002 & 2003
- Since all course materials were almost identical (including assignments and exams), the difference must have come from the programming environment
- “The SCHEME implementation worked well for me:” 2.4 (2002) vs. 2.9 (2003) p=.005
- “The programming environment worked well for me:” 2.5 (2002) vs. 3.2 (2003) p=.004

H3: Programming Environment

- Most differences in student ratings between 2002 & 2003
- Since all course materials were almost identical (including assignments and exams), the difference must have come from the programming environment
- “The SCHEME implementation worked well for me:” 2.4 (2002) vs. 2.9 (2003) p=.005
- “The programming environment worked well for me:” 2.5 (2002) vs. 3.2 (2003) p=.004
- “I see the utility of SCHEME as an instructional language”: 2.3 (2002) vs. 2.8 (2003)

Conclusion

- Functional-first plus additional topics (e.g., OO-programming) feasible in first CSE course

Conclusion

- Functional-first plus additional topics (e.g., OO-programming) feasible in first CSE course
- No effect of prior programming background on course grades (given the use of SCHEME)

Conclusion

- Functional-first plus additional topics (e.g., OO-programming) feasible in first CSE course
- No effect of prior programming background on course grades (given the use of SCHEME)
- Programming environments make a big difference in student's perceptions of the course and leads to higher time commitments

Conclusion

- Functional-first plus additional topics (e.g., OO-programming) feasible in first CSE course
- No effect of prior programming background on course grades (given the use of SCHEME)
- Programming environments make a big difference in student's perceptions of the course and leads to higher time commitments
- BUT: no significant difference between students' performance on the finals (64.6/100 in 2002 and 64.8/100 in 2003)