What do teachers teach in introductory programming?

Carsten Schulte, Jens Bennedsen

Introduction: Some statistics…

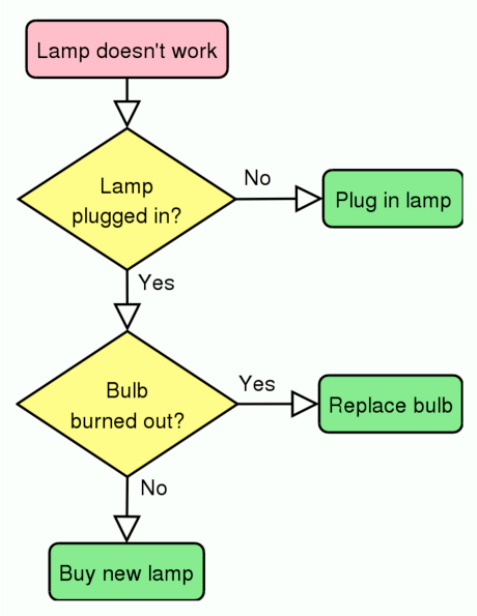
Freie Universität Berlin

- … on the statistics in the paper
  - 28 items in three dimension on four different groups, plus some other issues ~400 numbers,
    each with 3 digits = 1200 digits
- ~20 min presentation time
  - 20min*60 sec=1200 -> 1 sec per digit
- Let’s go ;-)

Focus of presentation

–topics–

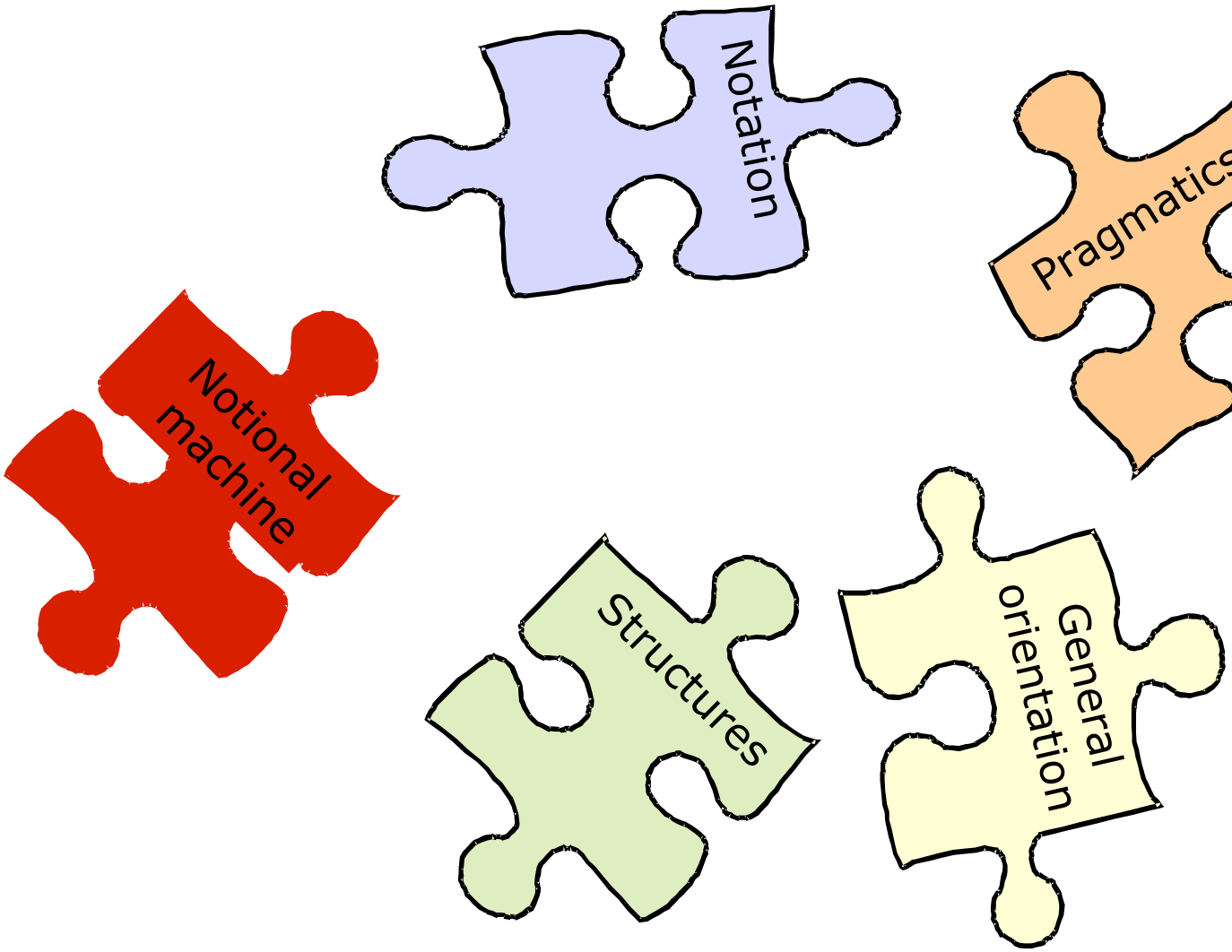
- Teachers' s opinions about what should be taught in CS1 (as an introductory programming course)
- What to teach besides topics



Flow chart -wikipedia

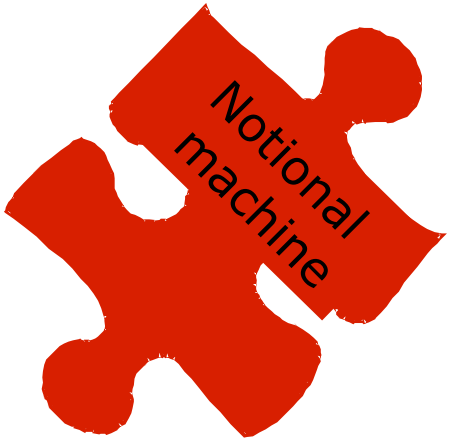
- Milne and Rowe (2002, p. 55): "inability [of students] to comprehend what is happening to their program in memory"
- Lathinen et al (2005, p. 15): "Students also have problems with understanding that each instruction is executed in the state that has been created by the previous instructions"
- Ragonis and Ben-Ari (2005, p. 214): "[high school] students find it hard to create a general picture of the execution"

Dimensions of learning programming



Du Boulay

Notional machine for OO?



1. *Interaction with objects*
   The student can understand simple forms of inter-actions between a couple of objects, such as method calls and creation of objects. The student is aware that the results of method calls depend on the identity and state of the object(s) involved.
2. *Interaction on object structures*
   The student is able to comprehend interaction on more than a couple of objects, including iteration through object structu-res and nested method calls. The structure is created and chang-ed explicitly via creations, additions and deletions.
3. *Interaction on dynamic object structures*
   The student knows the dynamic nature of object structures, understands the overall state of the structure and that interaction on the structure or elements from it can lead to side-effects (e.g. implicit changes in the structure).
4. *Interaction on dynamic polymorphic object structures*
   The student takes into account polymorphism in dynamic object structures and is able to understand the effects of inheritance and late binding on dynamic changes in the object structure. The student takes into account side-effects of late binding (different method-implementations, different actual objects referred to by the same variable).

Research Theme

- What do teachers teach in introductory programming?
- Topics
  - Imp. and/or OO
- Didactical perspective
  - Research from the eighties
  - Role of mental model for oo -> object interaction
- Descriptive (not Explanatory)

Population

Freie Universität Berlin

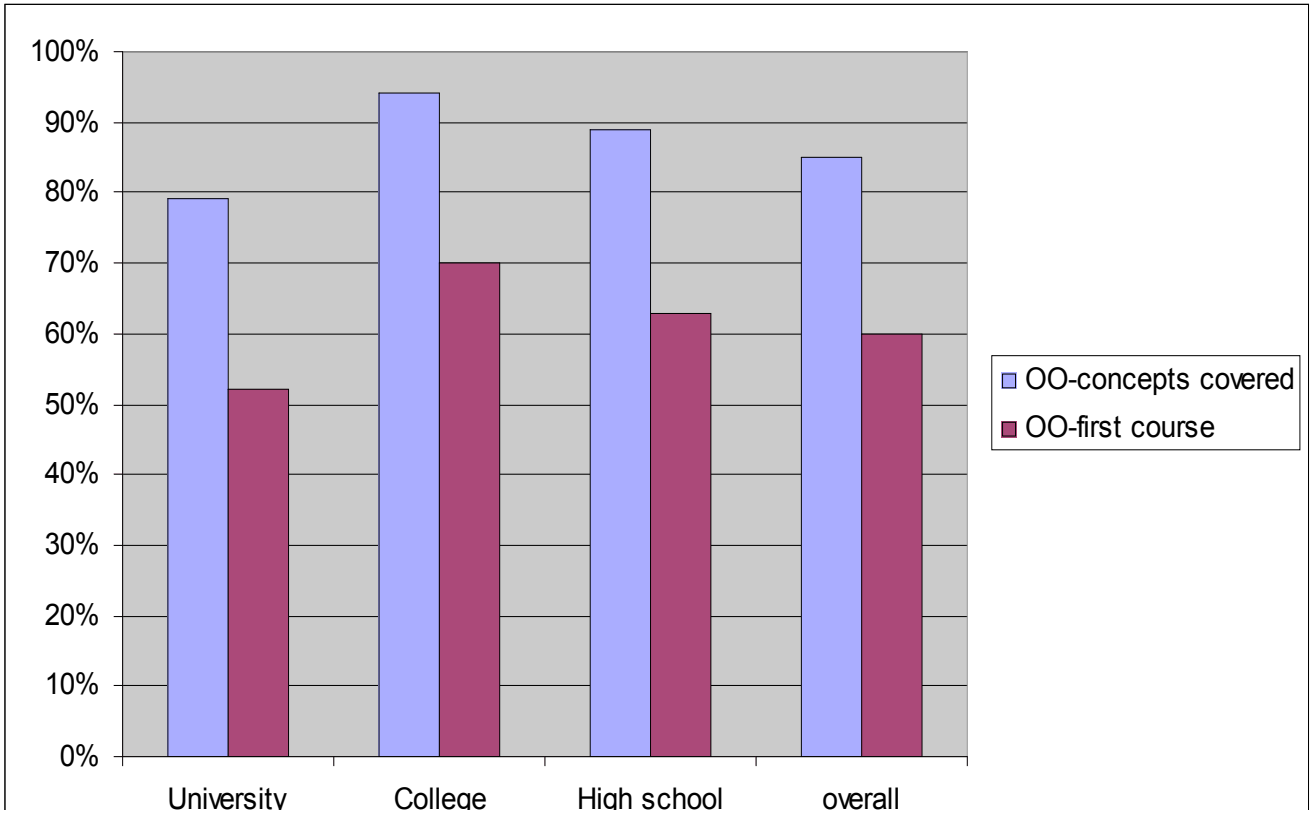
- Participants: teachers at university, college and high school, worldwide
- are attending educational workshops or conferences

- are Experts for Teaching

Responders

	Uni-versity	College	High school	Other	Overall
Denmark	5	49	1	3	*16.7%*
Germany	16	4	40	6	*19.0%*
USA	79	34	4	1	*33.9%*
Other	98	4	4	0	*30.5%*
Overall	*56,9%*	*26,1%*	*14,1%*	*2,9%*	*100%* *348*

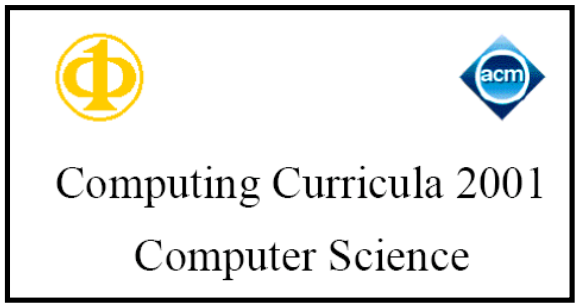
OO included

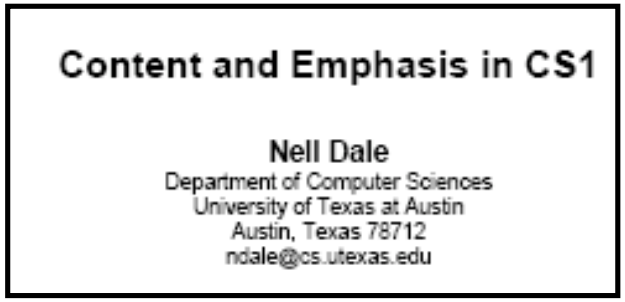

100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
University
College
High school
overall
OO-concepts covered
OO-first course

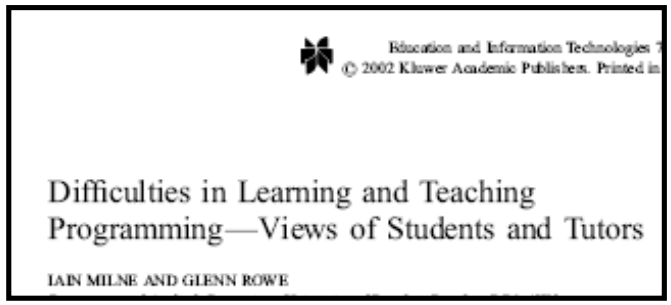
Important teaching topics

Freie Universität Berlin

What topics to ask for…



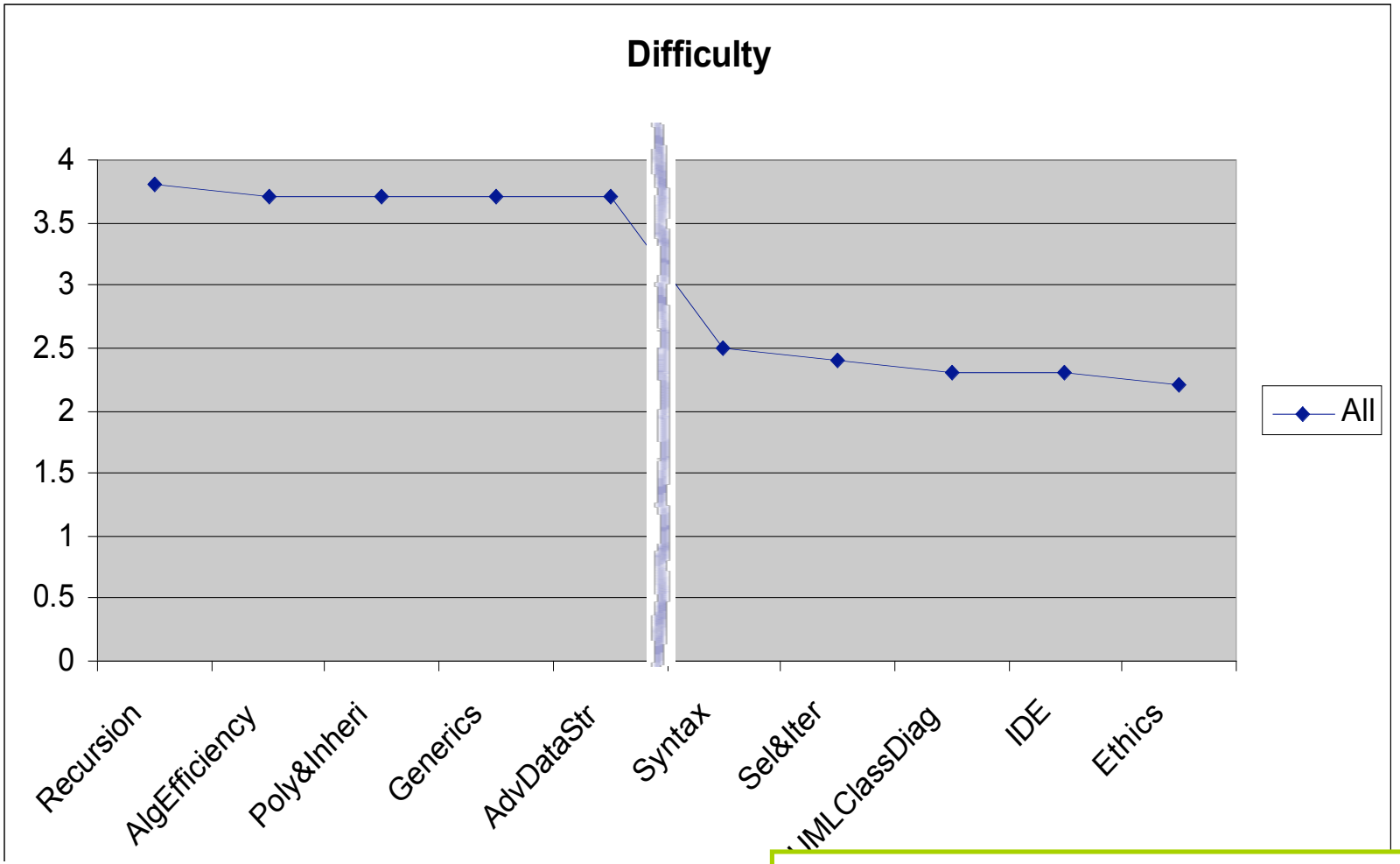




- Result: list of 28 topics

- Difficulty
- Relevance
- Level

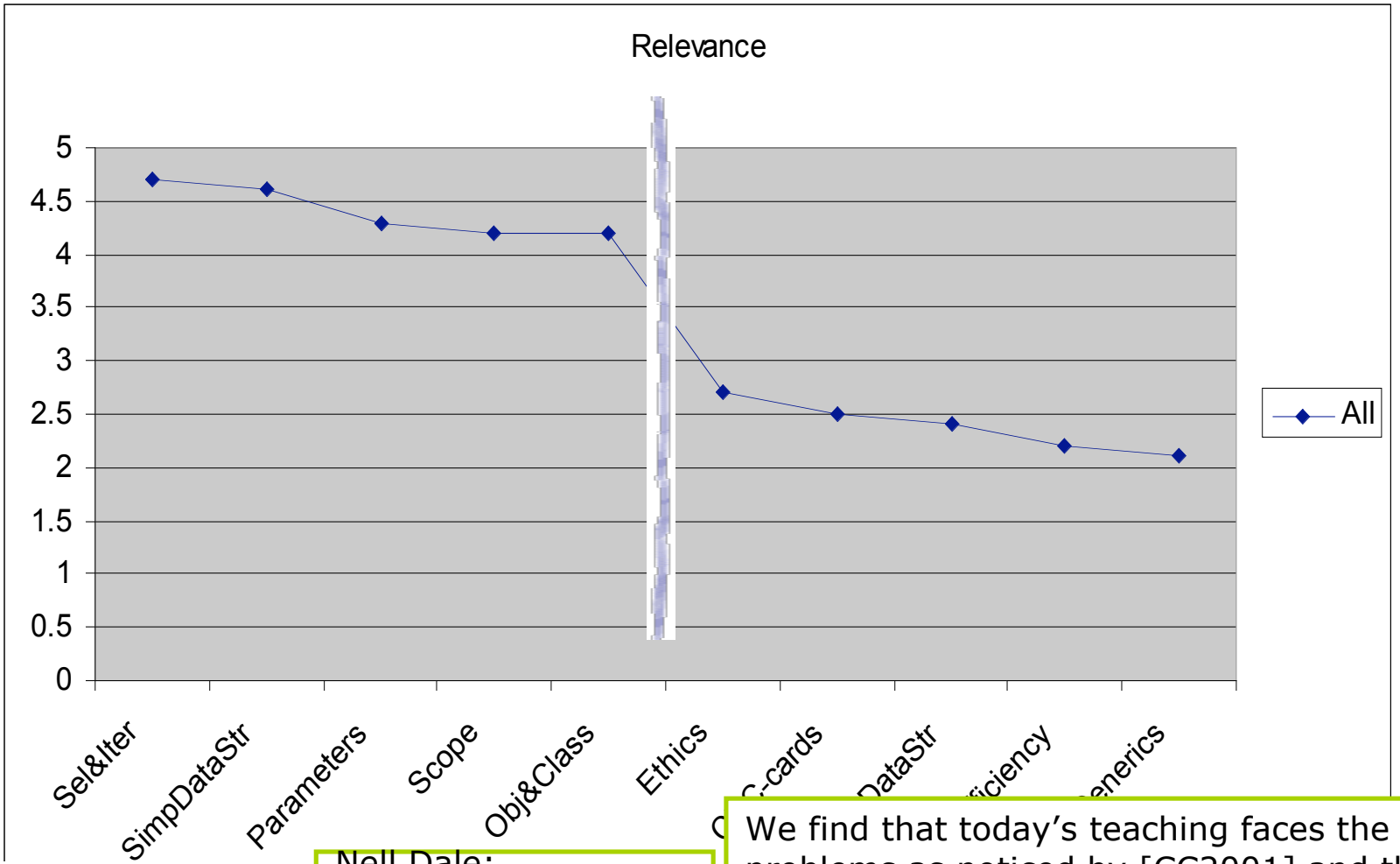
Difficulty of 28 topics, all teachers



Scale: 1 – 5, left side: most; right side: least

Milne and Rowe: polymorphism, recursion, pointers
CC2001: Recursion, AlgEfficiency, Generics, Adv-Data-Str, Poly are part of CS2

Relevance of 28 topics, all teachers

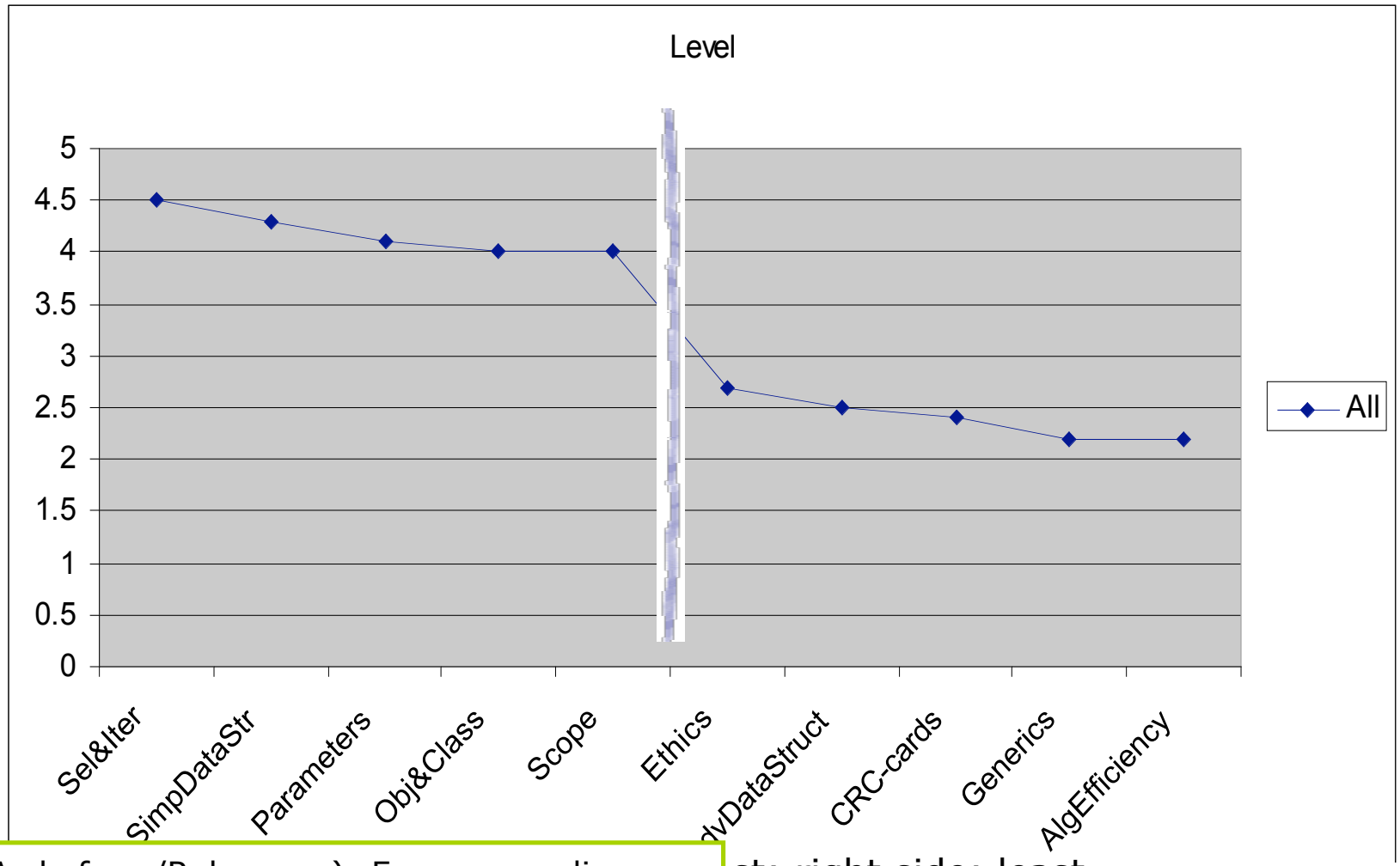


Scale: 1 – 5, left side: most; right side: least

Nell Dale: Repetition, Selection, Information Hiding

We find that today's teaching faces the same problems as noticed by [CC2001] and therefore students still associate programming with coding and not with more abstract, design-oriented and intellectual challenging activities.

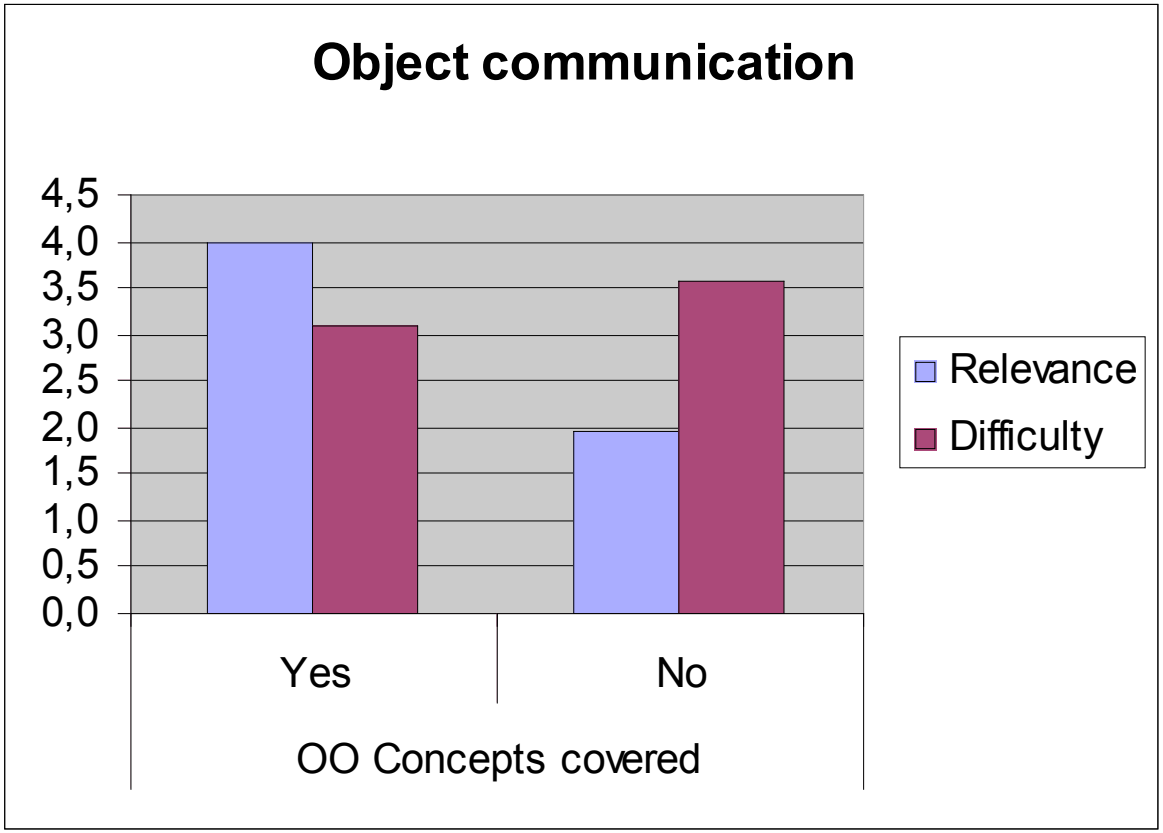
Level of 28 topics, all teachers



As before /Relevance): Focus on coding

ost; right side: least

Correlations between Difficulty, Relevance and Level



Interpretation: Role of 'time spend' to teach a topic?
Typical for OO-topics, not as typical for Non-OO (Selection..)

Subgroups: "OO Yes" vs "OO No"

Freie Universität Berlin

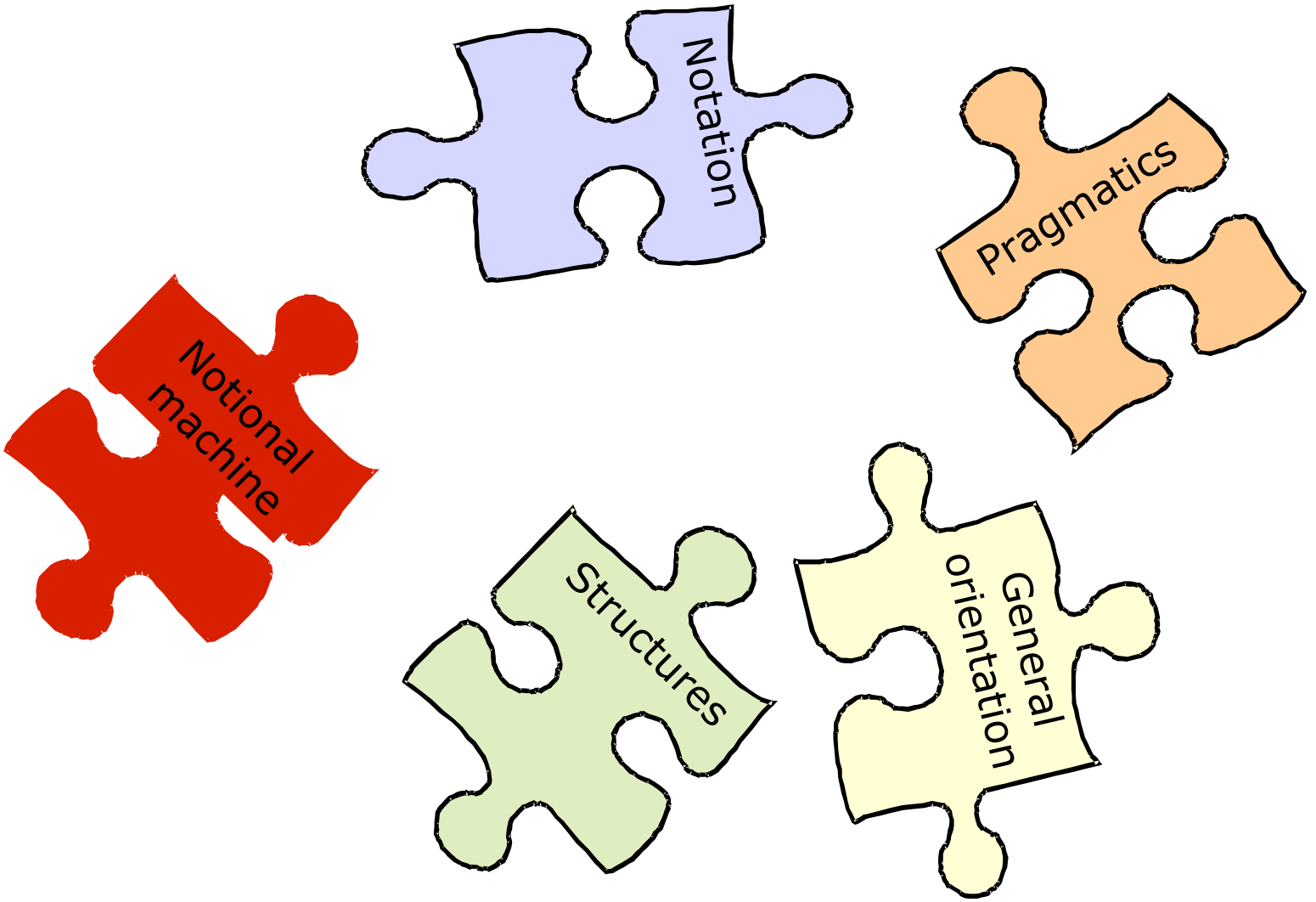
OO concepts covered	**Topics that are seen as statistically significant more *difficult***
No	Poly&Inheri, Obj&Class, **ptr&refs**, **recursion**, DesignClasses, **parameters** and Encapsulation
Yes	

OO concepts covered	**Topics that are seen as statistically significant more *relevant***
No	Sel&Iter
Yes	StatVsNonStat, AdvDataStr, ObjComm, Poly&Inheri, Generics, Obj&Class, UMLClassDiag, VarTypes, DesignClasses, DesignSglClass, CRC-cards, MethodDesign and Encapsulation

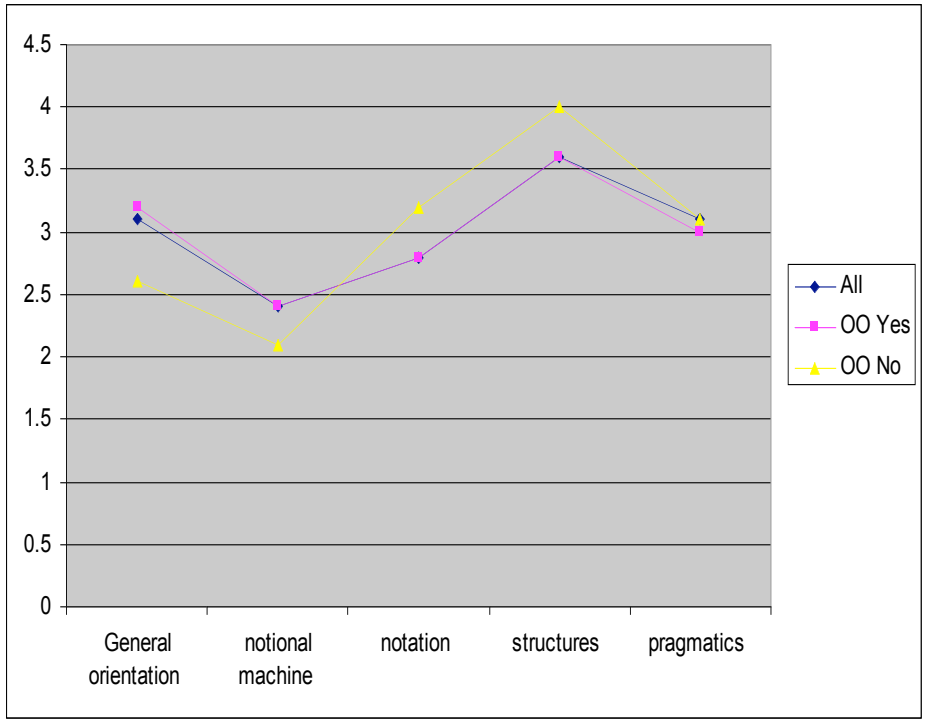
OO concepts covered	**Topics that are seen as statistically significant to be taught on a higher *level***
No	-
Yes	AdvDataStr, ObjComm, Poly&Inheri, Obj&Class, UMLClassDiag, DesignClasses, DesignSglClass, CRC-Cards and Encapsulation

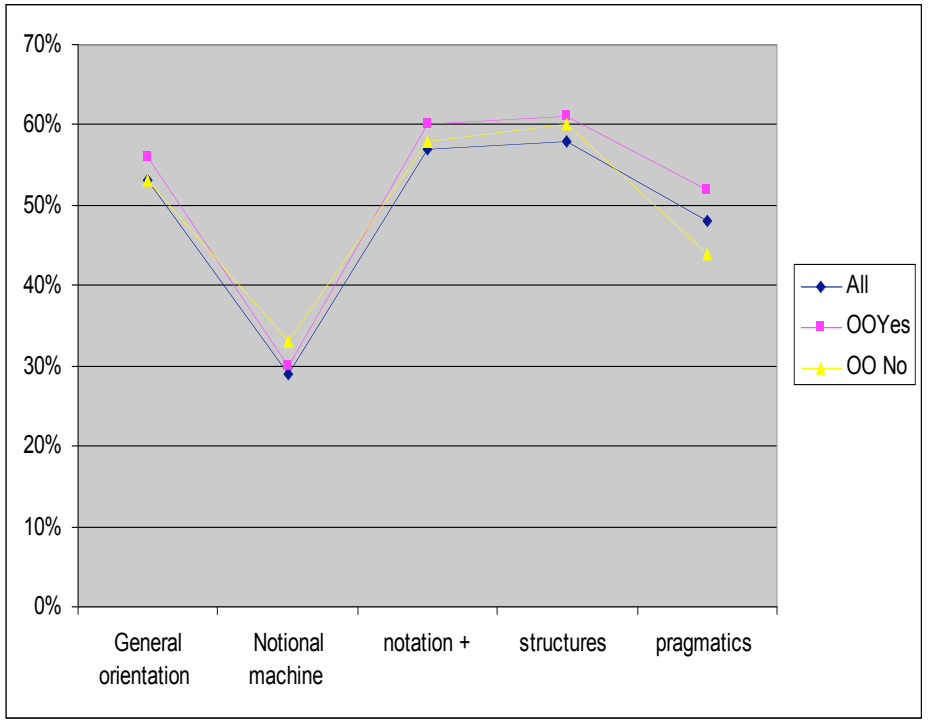
b

Areas of learning programming



Role of Areas

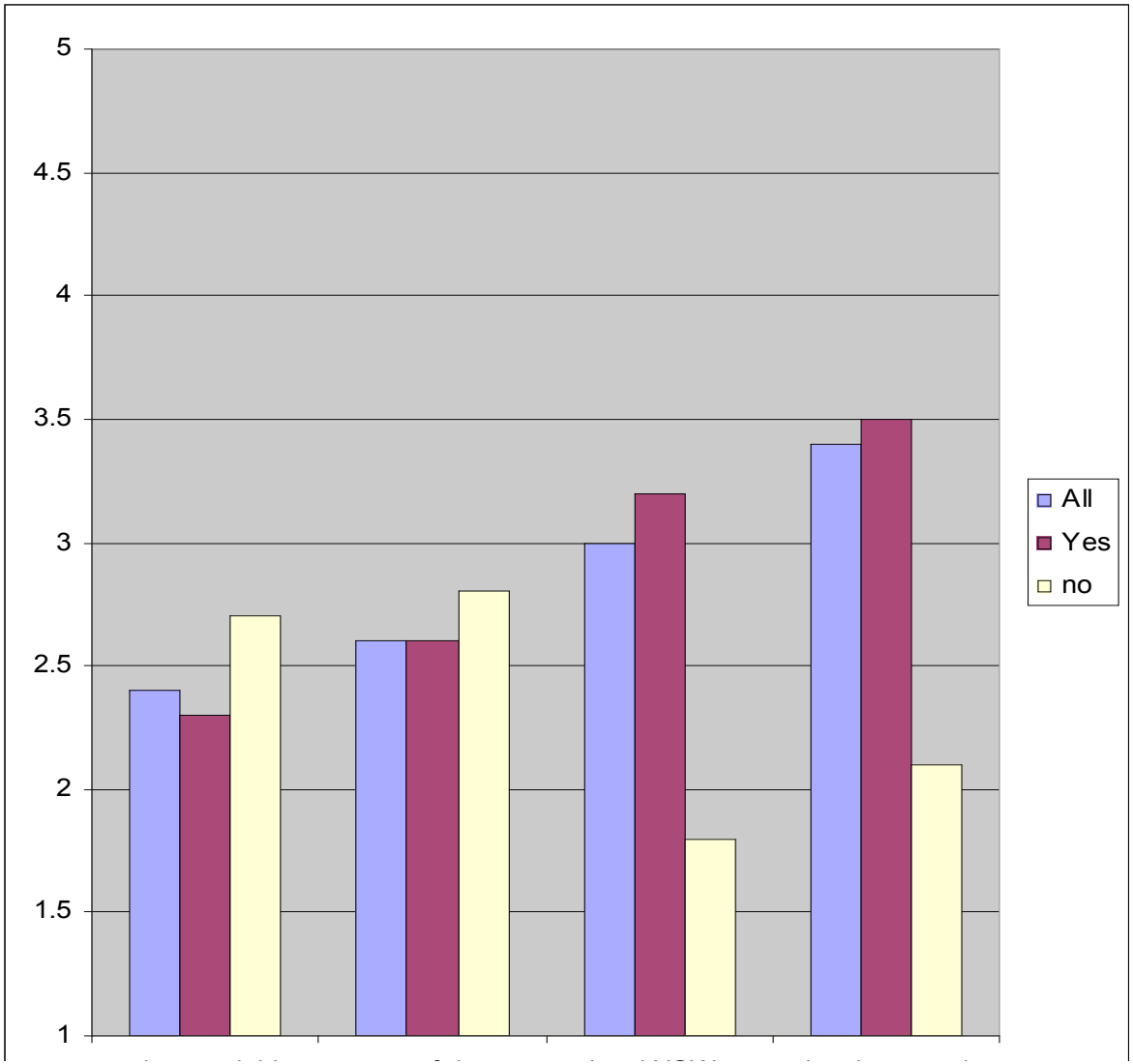




importance

taught

Hierarchy of Object Interaction



All Scales:
1-5
Underst. & Use:
Lower=better
Level: Higher=better

Differences between paradigms

Freie Universität Berlin

Teaching OO		Teaching PROC
👍	procedural topics	👍
👍	OO topics	👎
👍	Areas	👍
👍	Object Interaction	👍

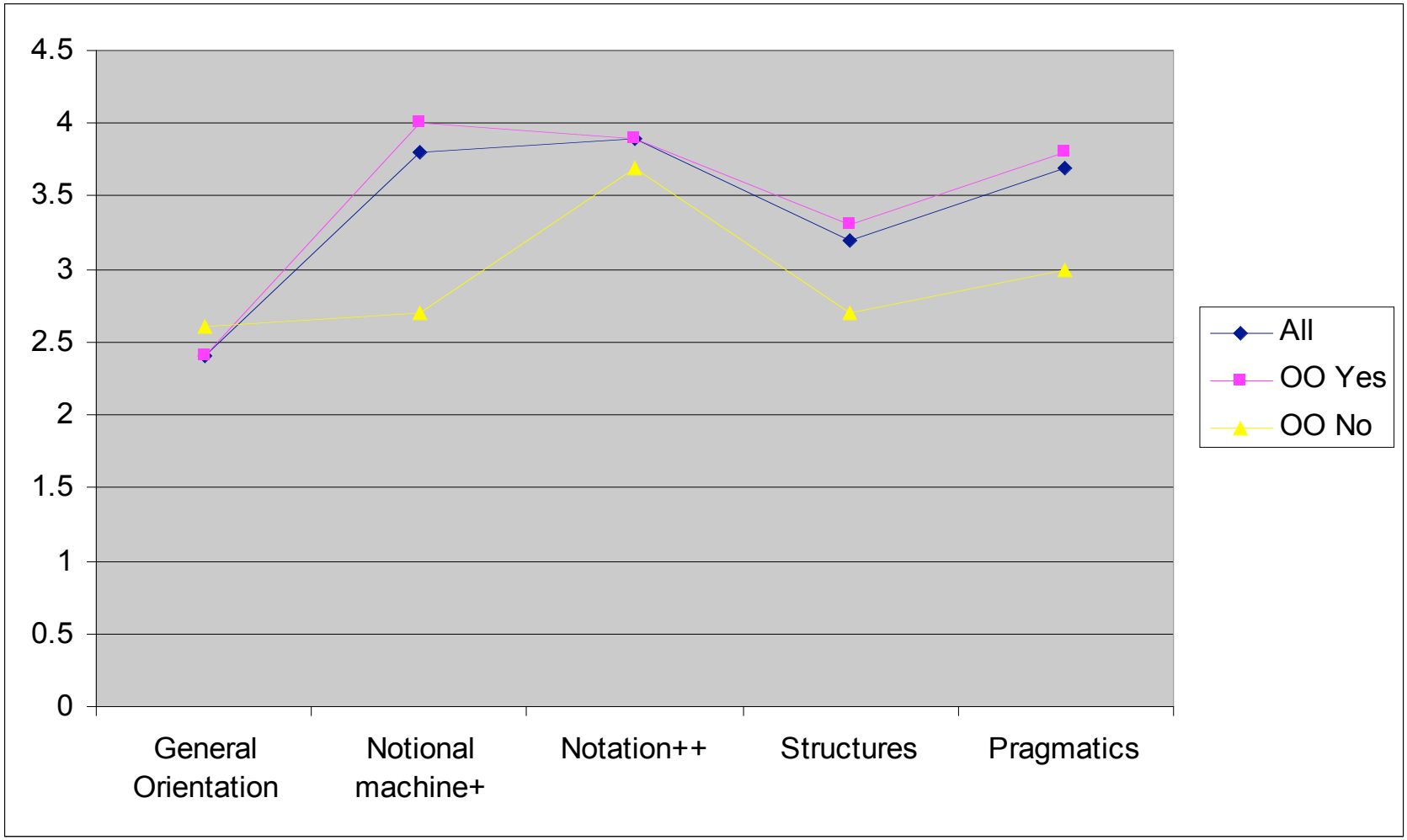
Overall: very similar

Topics assigned to Areas (by the authors)

Freie Universität Berlin

Topics	**Du Boulay area**
Sel&Iter, Parameters, Scope, Syntax, *UMLClassDiag*, Ptr&Refs, Library	Notation
SimpDataStr, ProbSolStra, AdvDataStr, Recursion, Generics, *Poly&Inher*, Encapsulation	Structures
ObjComm, MentalModel, *StatNonStat, VarTypes, Obj&Class*	Notional machine
Debugging, IDE, *CRC-cards, DesignSglClass, DesignOfClasses, MethodDesign*, AlgDesign,	Pragmatics
Ethics, AlgEfficiency	General Orientation

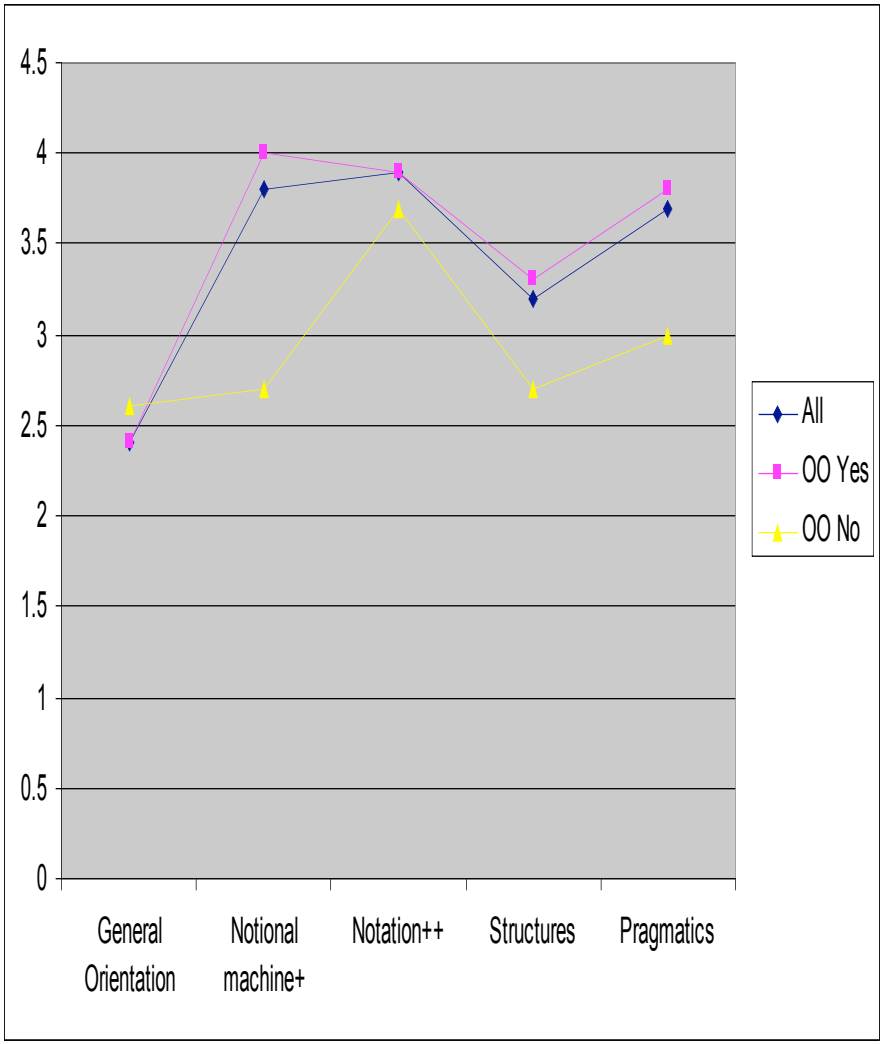
Areas Topics: relevance



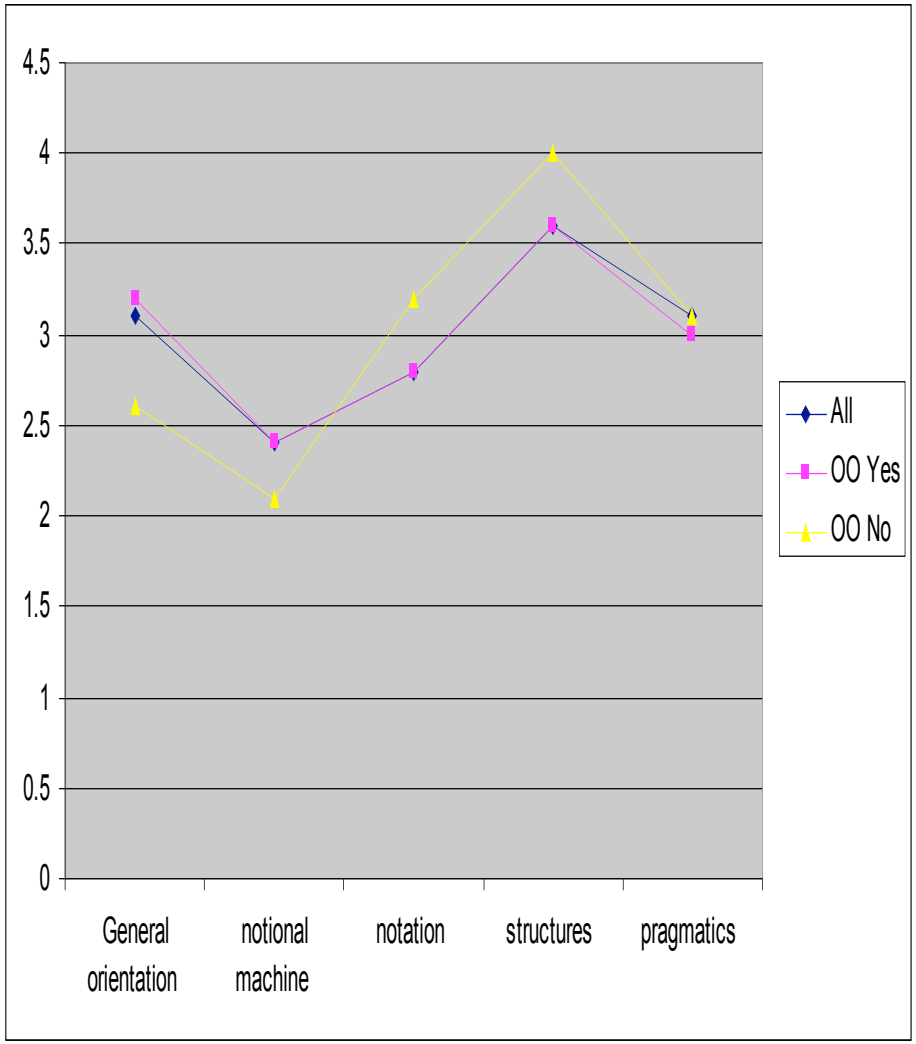
Areas compared

Freie Universität Berlin

topics



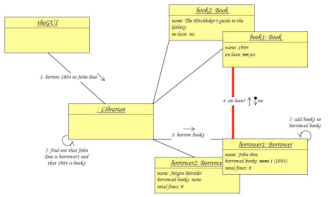
du Boulay



Summary / Conclusions

- 'Classic' topics /iteration syntax) similar relevance, level and difficulty regardless of "teaching-paradigm"
- Including OO in CS1 seems to be an addition of topics
- OO-topics are seen more difficult by those who do not teach them
  - OO Teachers have a tendency to rate 'abstract concepts' as less difficult (table)

- Notional machine
  - Least relevant (areas)
  - More relevant in OO (topics, but…)



- Hidden curriculum (structures vs. notation)
- Focus on coding in OO, too (notation vs. notional machine)

Differences between paradigms

Teaching OO		Teaching PROC
👍	procedural topics	👍
👍	OO topics	👎
👍	Areas	👍
👍	Notional machine	👎
👍	Object Interaction	👍

Thank you!