Concurrent and Parallel Interactive Theoretical Teaching through ICT

A.J. Tomeu\textsuperscript{1} \hspace{1cm} A.G. Salguero\textsuperscript{1} \hspace{1cm} M.I. Capel\textsuperscript{2}

\textsuperscript{1}Department of Computer Sciences
College of Engineering
University of Cádiz (Spain)

\textsuperscript{2}School of Computer Engineering and Telecommunications
University of Granada (Spain)
Email: manuelcapel@ugr.es

1st Annual European Workshop on Parallel and Distributed Computing Education for Undergraduate Students, Vienna (Austria) 24 August, 2015
Summary

1. Introduction
2. Objectives
3. Method
4. Outcome
Summary

1. Introduction
2. Objectives
3. Method
4. Outcome
Parallel Programming learning is a lengthy process that needs *key examples* that contain *non intuitive code*.

- Pedagogics of PP isn’t likely to be modified by now.
- However, we can change the way students face a concurrent problem, and start developing parallel code.

Students must adopt an active role when it comes to learn Programming –that also applies to theoretical teaching.

- Learning–innovation project carried out at University of Cádiz to show that interactive CPP teaching is feasible.
- Early results show an improvement in students CPP education and their appreciation of the new method.
Course on Parallel and Distributed Computing for undergraduate students in the fourth semester (second part of sophomore year) of a CS degree

- Taught in 6 ECTS or one–fifth of a semester effort
- Equivalent to 2 lectures of 1–hour each and 8 hours of individual practical work and self–study per week

The course was followed by an audience of 149 students grouped as it follows,

- Two classes of 75 for theoretical teaching (lectures)
- Eight classes of 25 for practical work (labs)
Means

Full equipped classroom:
- one laptop for each student
- high speed “Virtual Campus”
- WIFI access
- beamer for teacher slides
- whiteboard
- teacher’s tablet
CPP Didactics

- Improvement of students comprehension of CPP: new programming concepts, paradigms and idioms
- Change of ’mood’ regarding Concurrency counter-intuitiveness
- Proactive attitude: theoretical teaching shouldn’t be so dull
- Multipath, individually paced, stop–and–replay, personalized learning process
- Frequent assessment of learning advances on the subject
Temporal constraints:
15 weeks of effective teaching
‘Regular’ course lectures allotment
No extra charge for teachers

Approach taken

1. Careful selection of paradigmatic study cases
2. Design of self-contained ‘didactic units’
3. Implementation of codes on a Moodle–based platform
<table>
<thead>
<tr>
<th>Week</th>
<th>Didactic unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motivation and course intro</td>
</tr>
<tr>
<td>2</td>
<td>Concurrency, creation of processes, race conditions</td>
</tr>
<tr>
<td>3</td>
<td>Mutual exclusion problem solved with active loops</td>
</tr>
<tr>
<td>4</td>
<td>N-processes: Knuth and Peterson algorithms</td>
</tr>
<tr>
<td>5</td>
<td>Distribution fundamentals: Ricart–Agrawala protocol</td>
</tr>
<tr>
<td>6</td>
<td>Predicates and shared resources specification</td>
</tr>
<tr>
<td>7</td>
<td>Concurrent properties formal specification</td>
</tr>
<tr>
<td>8</td>
<td>Review before first test</td>
</tr>
</tbody>
</table>

Tomeu, Salguero, Capel

Theoretical teaching through ICT
### Teaching II

<table>
<thead>
<tr>
<th>Week</th>
<th>Didactic unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Shared resources implement. with C++11 and Java</td>
</tr>
<tr>
<td>10</td>
<td>Task–centric parallelism vs. concurrent threads</td>
</tr>
<tr>
<td>11</td>
<td>Shared resources with locks and conditions</td>
</tr>
<tr>
<td>12</td>
<td>Distributed systems programming</td>
</tr>
<tr>
<td>13</td>
<td>Shared resources implement. with Open MPI library</td>
</tr>
<tr>
<td>14</td>
<td>Review prior to C++11 or Java project deadline</td>
</tr>
<tr>
<td>15</td>
<td>Review before second test</td>
</tr>
<tr>
<td>16</td>
<td>Review prior to Open MPI project deadline</td>
</tr>
</tbody>
</table>

### Grading

<table>
<thead>
<tr>
<th>Written tests</th>
<th>Term projects</th>
<th>Weekly exercises</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 %</td>
<td>40 %</td>
<td>10 %</td>
</tr>
</tbody>
</table>

Tomeu, Salguero, Capel
Theoretical teaching through ICT
What should be innovated?

Traditional teaching style, tends to focus on concurrent capabilities description of one selected language and show how solutions for some given problems can be programmed.

This approach tends to cause an angry refusal, specially on Concurrent Programming topics, among the students.
To empower students with design tools and mental aids to do CPP. A pragmatic approach based on the intuitive understanding of what concurrent languages can do and how they behave.

Re-thinking of the student role during theoretical teaching

- Immediate access to the code shown by the teacher through ‘Virtual Campus’ platform
- Any change on that code can be locally apparent in the student laptop to exercise with any concurrent property
'Virtual Campus', a methodological support for course content–blocks. Through a series of tasks (exercises and research), the students become absolute protagonists of the learning process.
Interactive lecture model

The student is asked to follow the task flow on the diagram, after each new paradigmatic example is presented by the teacher.

As result, students can check “hands on” any concurrent property on the code.

Finally, reinforcement exercises are assigned as homework to each student.
Centered around an abstract concurrent shared resource.

A concurrent system is then made up of
- Set of active components (processes)
- A concurrent ADT (shared resource)
- Communication and synchronization, only through the resource.
What has been innovated II

Figure: Suggested development process
Example: Peterson’s algorithm

Initial: advanced(0..N-1), turn(0..N-2) ← 0

Critical Section = stage(N-1)

Safety: □ # Process(i) = 1

Reachability: ● □ Process(●) at stage(N − 1)

Liveness: □ ● Process(i) at stage(N − 1)

Requested system initial design

- Students do not design a system from scratch
- Initial design has been validated by instructors
- Students are provided with tests to check their implementation

Students do not design a system from scratch
Initial design has been validated by instructors
Students are provided with tests to check their implementation
Peterson’s algorithm II

Predicative specification model

- Students have to develop a formal specification (pre, post, invariant) of the initial shared resource design
- To choose the correct idioms (notify(), locks, conditions...) for correct synchronization of resource’s operations
- Finally, to check liveness and safety properties

Queue of stages specification for the algorithm

StageQueue
Operations
  readStageTurn(processId)
  writeStageTurn(processId)
Semantics Domain:
  Type: StageQueue(0..N-2) == seq N, processId:0..N-1
  Invariant: #at (N-1) == 1
Peterson’s algorithm III

Queue of stages specification for the algorithm (contd.)

CPre: $\forall k:0..i-1,i+1,N-1$: advanced(i)>advanced(k) \ or turn(j) != i
void writeStageTurn(i){}
Post: advanced(i)++;

CPre: {True}
int readStageTurn(i)
Post: [advanced(i)] \in 0..N-2

Validation

- Validation of StageQueue in isolation
- Validation of the complete System
  - In this validation scenario, stronger invariants can be proved
Mechanical code generation

- Students are told to deliver a piece of code implementing the concurrent shared resource behaviour.
- A set of design patterns can be used to transform resource’s specification into C++11 or Java code.
- Concurrent properties (safety, fairness, ...) must have been assured through correct programming of synchronization.
- Three synchronization idioms are provided to students: notify–notifyAll, locks and conditions, and MPI operations.
Low-level system model for automatic trace-generation

```c
int[N-1] advanced; //process-i must go through
int[N-2] turn; //last process that reached the stage
while (true){
    //Remain
    (1) for(j=0; j<N-2; j++){ 
    (2)    advanced[i]= j;
    (3)    turn[j]= i;
    (5)    for(k=0; k<N-1; k++){
    (6)        if (k!=i){ //otherwise continue
    (7)            while(advanced[k]>=j and turn[j]==i)
    (8)                ; //busy wait;  }}
    (9)    advanced[i]= N-1; //meta-instruction
    (10)   //Critical Section: (N-1) stage
    (11)   advanced[i]= -1 //post--protocol
}
```
Automatic generation of tests

Traces–based animations

- One tester is made up of a huge set of traces that explore all the system’s states up to a given depth.
- A typical tester executes between 500 and 1,000 different traces of the system to be checked.
- By exploring traces it becomes possible to detect any misbehaviours of the system.
- Students can use testers to find out what is wrong with their implementations.
Assessing the advantages of the method

**Pre-assessment**

<table>
<thead>
<tr>
<th>Method usefulness</th>
<th>Date of survey</th>
<th>Cohort size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (totally disagree) – 5 (completely agree)</td>
<td>first semester 2014</td>
<td>78</td>
</tr>
</tbody>
</table>

**Aspects of interest to be assessed**

- Understanding improvement of CPP concepts presented
- The number of reinforcement–exercises assigned
- The time required for the resolution of exercises
- Compliance level with the new model of theoretical teaching

Tomeu, Salguero, Capel

Theoretical teaching through ICT
Method assessment from students

Figure: Understanding Improvement
Student perception of assignments

**Figure**: Number of Exercises Adjustment
Student perception of assignments II

Figure: Resolution Time of Exercises
Figure: Agreement Level with the Method
Assessing the advantages of the method II

### Post-assessment

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Students succeeding</td>
<td>35%</td>
<td>55.6%&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

<sup>a</sup>First examination taken by the students

### What students said..

“Like your theoretical lessons, the way they are now”

### And instructors ...

Do not complain about the extra effort needed to put the method up provided the better results and involvement of students

---

Tomeu, Salguero, Capel

Theoretical teaching through ICT
## Conclusions

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| - Excellent results obtained during screening and post-assessment phases  
- Student understanding of key concepts improved w.r.t. traditional teaching  
- Favourable change in students personal work habits, higher involvement and improved self-confidence  
- Practical skills to deal with assignments in shorter time have been boosted  
- Effective ICT deployment has proved to help teaching a difficult subject  
- The success rate of students taking the course has remarkably improved | - A drawback is that formal specification of resources and checking of correctness is, as of now, complex and not completely automated, which can be an added difficulty for the average undergrad student w.r.t. the traditional teaching of Concurrency |
References I

Area Moreira, M. Enseñar y aprender con TIC: más allá de las viejas pedagogías. Aprender a educar con tecnología, n.2, diciembre 2012, pgs. 4-7.


