Concurrent and Parallel Interactive Theoretical Teaching through ICT

A.J. Tomeu[1] A.G. Salguero[1] M.I. Capel[2]

[1]Department of Computer Sciences
College of Engineering
University of Cádiz (Spain)

[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

Summary

Summary

Summary

Motivation

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

Context

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

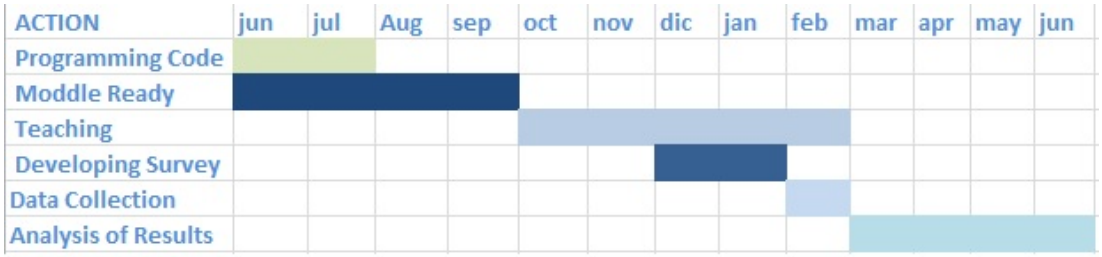
Full equipped classroom:

-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

Agenda



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

Teaching

Week	Didactic unit
1	Motivation and course intro
2	Concurrency, creation of processes, race conditions
3	Mutual exclusion problem solved with active loops
4	N-processes: Knuth and Peterson algorithms
5	Distribution fundamentals: Ricart–Agrawala protocol
6	Predicates and shared resources specification
7	Concurrent properties formal specification
8	Review before first test

Teaching II

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

Grading

Written tests	Term projects	Weekly exercises
50 %	40 %	10 %

What should be innovated?

CONTROL DE *Threads*: Ejemplo de Cesión de Prioridad Voluntaria (yield)

```
public class replaniYield
  extends Thread
{
   private boolean hY;//indicara si el hilo cede prioridad o no…
   private int     v;

   public replaniYield(boolean hacerYield, int vueltas)
   {hY = hacerYield; v = vueltas;}

   public void run()
   {
    for(int i=0; i<v; i++)
     if(i==20&&hY==true){this.yield();}//indica cesion de prioridad…
     else System.out.println("Hilo "+this.getName()+" en iteracion "+i);
   }
    public static void main(String[] args)
    {
       replaniYield h0 = new replaniYield(false, 50);
       replaniYield h1 = new replaniYield(false, 50);
       replaniYield h2 = new replaniYield(true , 50); //cedera prioridad y
       h0.setName("1-NoYield");                       //sera o no considerarda
       h1.setName("2-NoYield"); h2.setName("3-SIYield");
       h0.start(); h1.start(); h2.start();
    }
}
```

© Antonio Tomeu Creación y Control de Thread en Java 27

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

Didactic paradigm


TEMA 3: CREACIÓN Y CONTROL DE THREADS EN JAVA

Textos del Tema

Conceptos sobre Threads en Java
Conceptos sobre Threads en Java (Texto Complementario)
Códigos del Tema

Diapositivas del Tema

Diapositivas del Tema 3 (ACTUALIZADAS A 2 DE NOVIEMBRE)
Diapositivas del Tema 3 (versión para imprimir, ACTUALIZADAS A 2 DE NOVIEMBRE)

Planificación Semanal de las Actividades del Alumno

Actividades de la Semana IV: 22-10 a 26-10
Actividades de la Semana V: 29-10 a 2-11

Lecturas Adicionales de Interés del Tema

API de la clase Thread

Entrega de Prácticas del Tema

Practica 3
Tiempos
Subida de Productos de la Práctica Número 4
Subida de Productos de la Práctica Número 5

Consultas del Tema

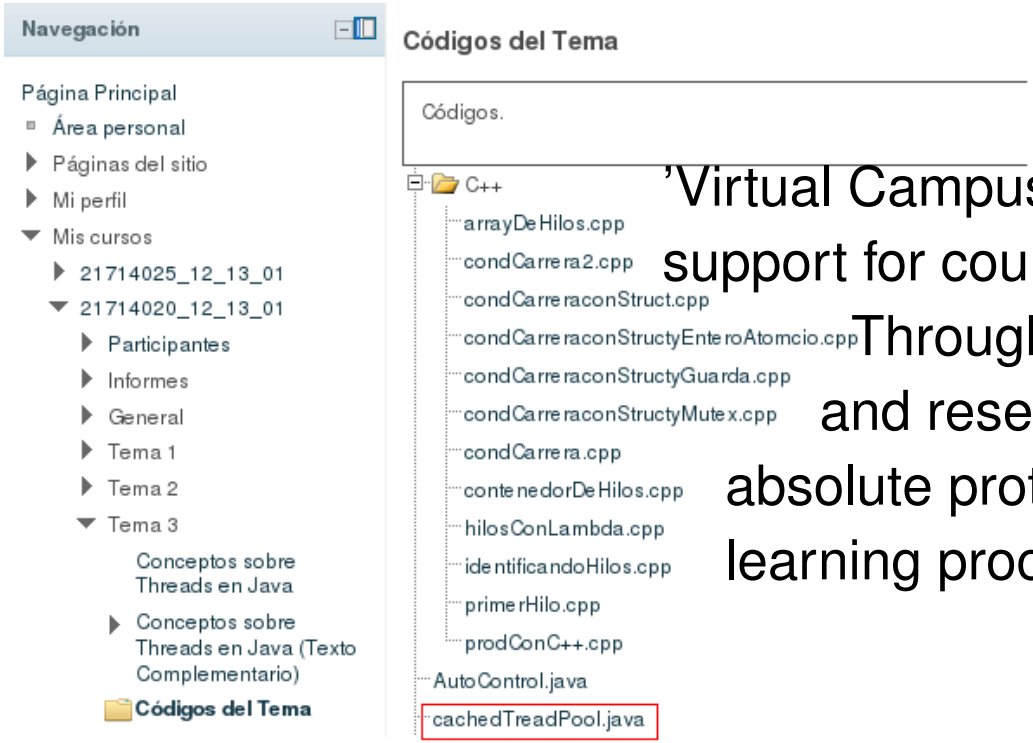
Recuperación de la Clase del Pasado día 12-11 (ALUMNOS GRUPO B)

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

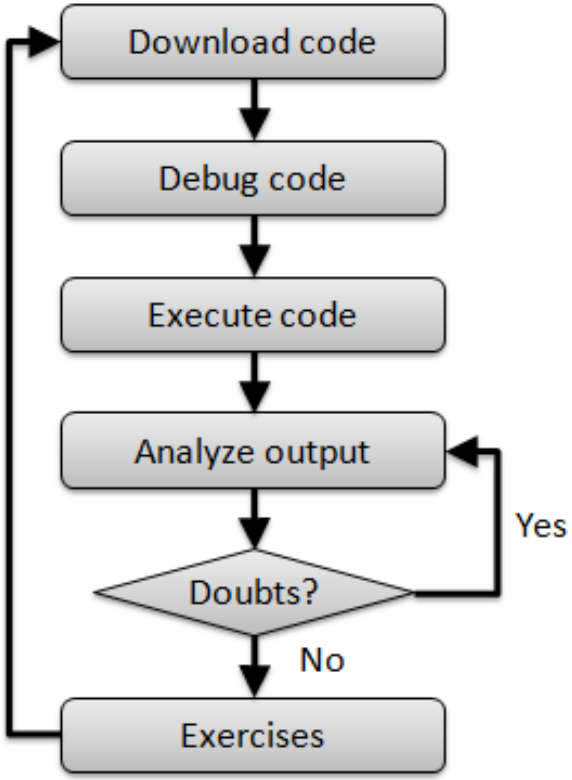
- Immediate access to the code shown by the teacher through ‘Virtual Campus’ platform
- Any change on that code can be locally apparent in the student laptop to exercise with any concurrent property

What has been innovated



'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

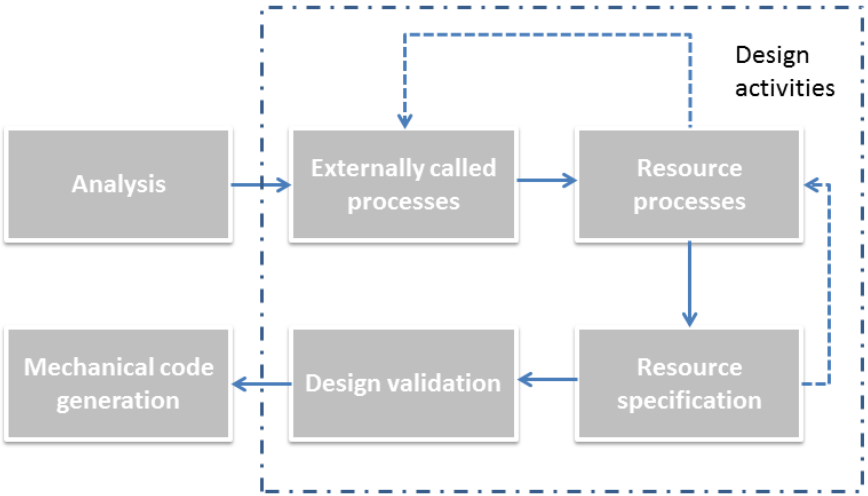


The student is asked to follow the task flow on the diagram, after each new paradigmatic example is presented by the teacher.

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

Finally, reinforcement exercises are assigned as homework to each student.

Term project model



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

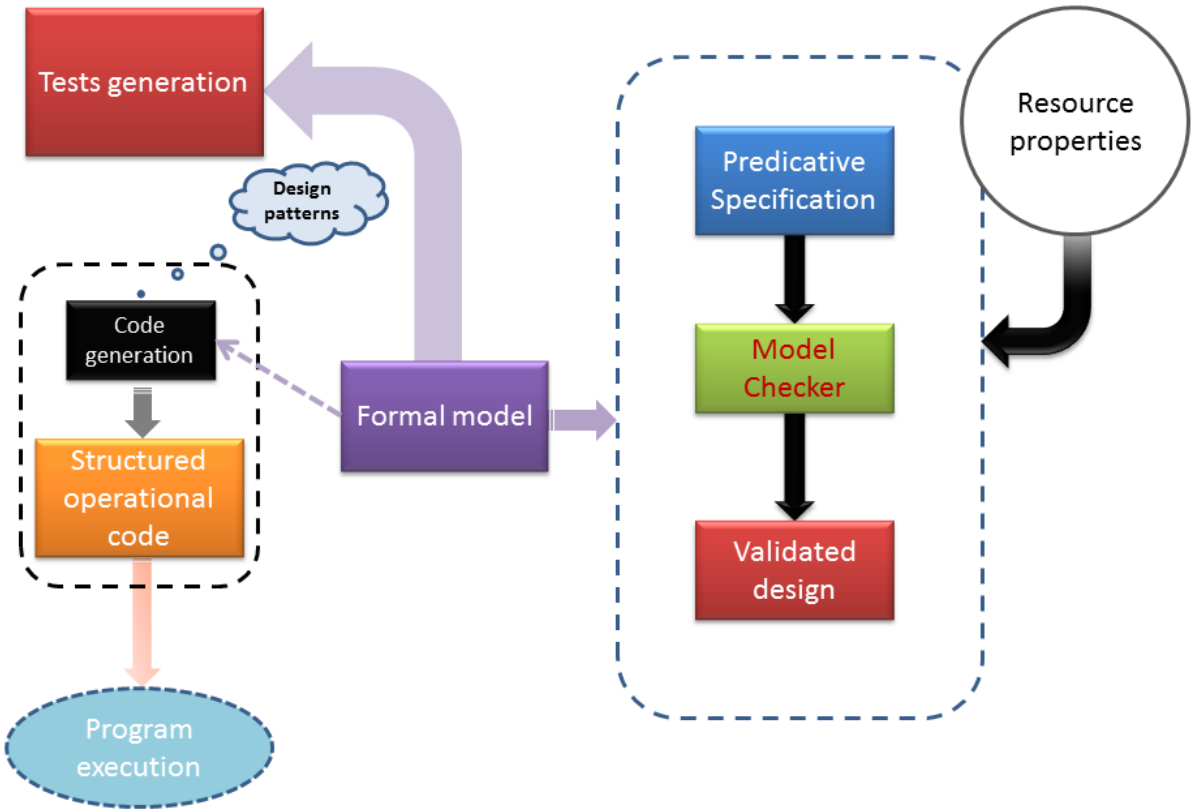
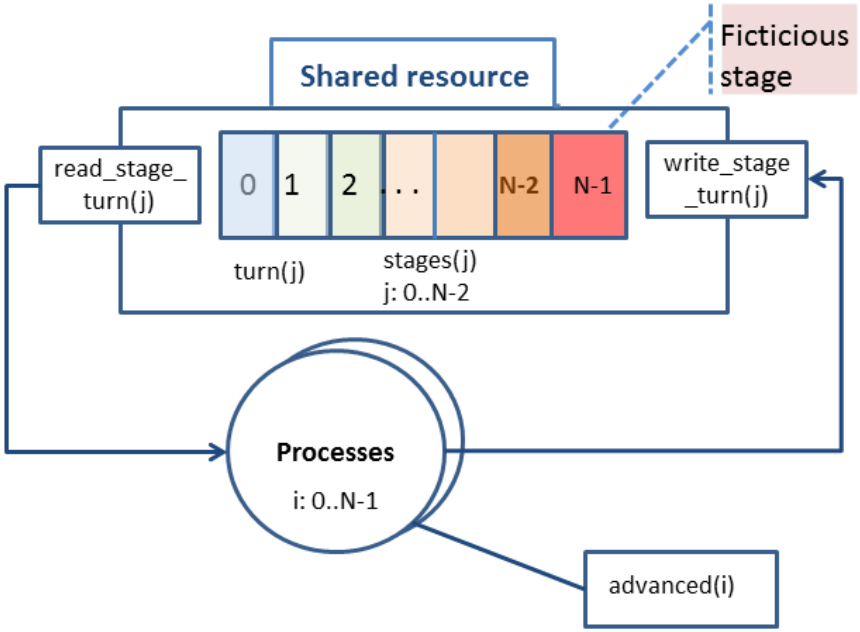


Figure: Suggested development process

Example: Peterson's algorithm



Initial: advanced(0..N-1), turn(0..N-2) $\leftarrow$ 0

Critical Section = stage(N-1)

Safety: $\Box \# Process(i) = 1$

Reachability: $\diamond \Box Process(\bullet)$ *at stage*$(N-1)$

Liveness: $\Box \diamond Process(i)$ *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

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 trough correct programming of synchronization
- Three synchronization idioms are provided to students: notify–notifyAll, locks and conditions, and MPI operations

Tests

Low–level system model for automatic trace-generation

```
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

Method usefulness	Date of survey	Cohort size
1 (totally disagree) – 5 (completely agree)	first semester 2014	78

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

Method assessment from students

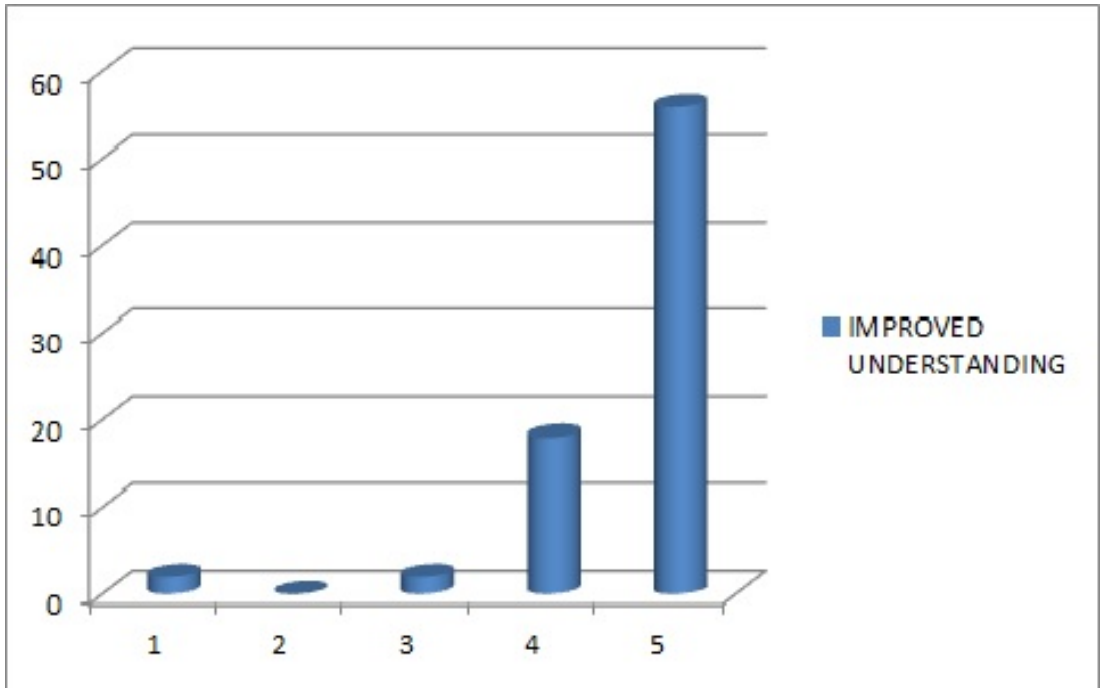


Figure: Understanding Improvement

Student perception of assignments

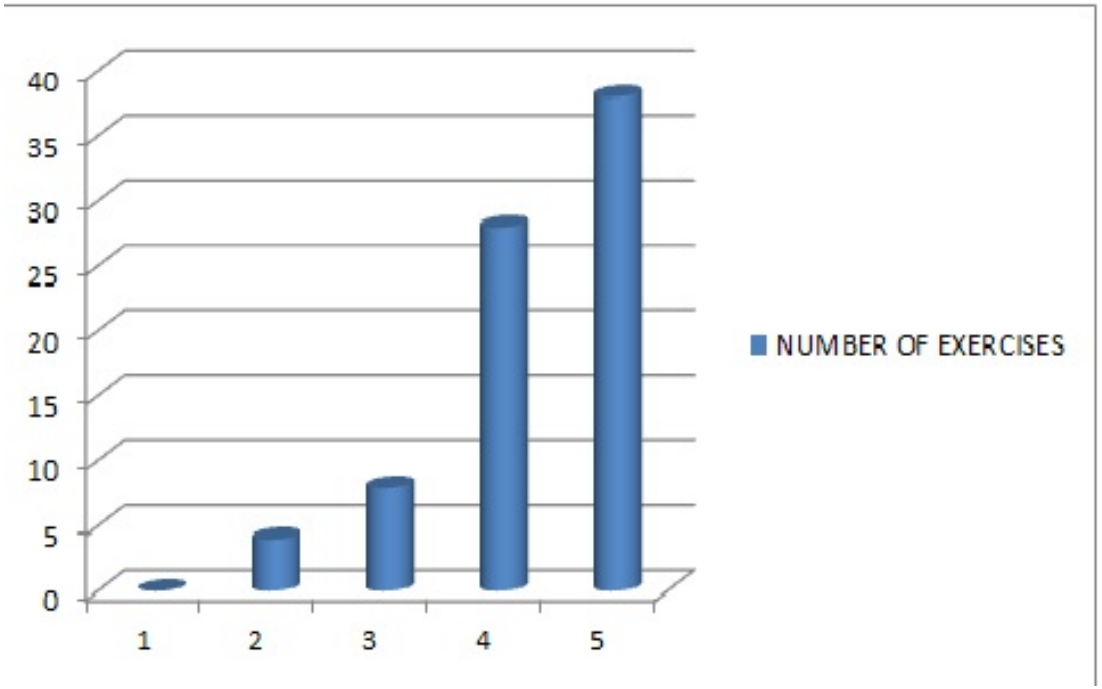


Figure: Number of Exercises Adjustment

Student perception of assignments II

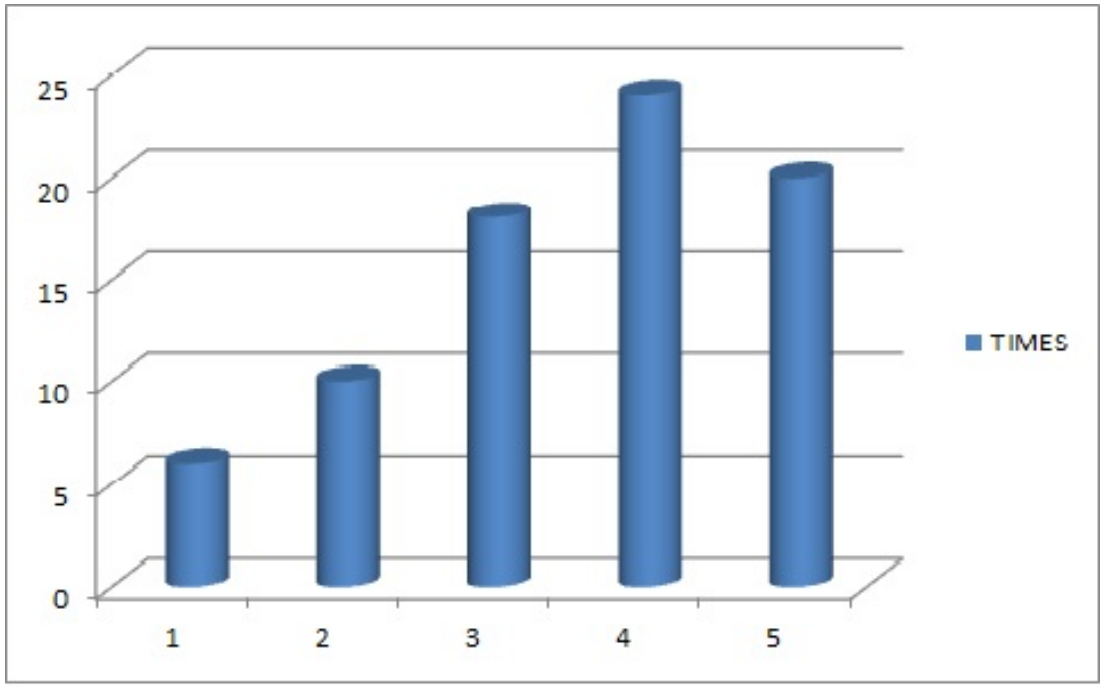


Figure: Resolution Time of Exercises

Compendium

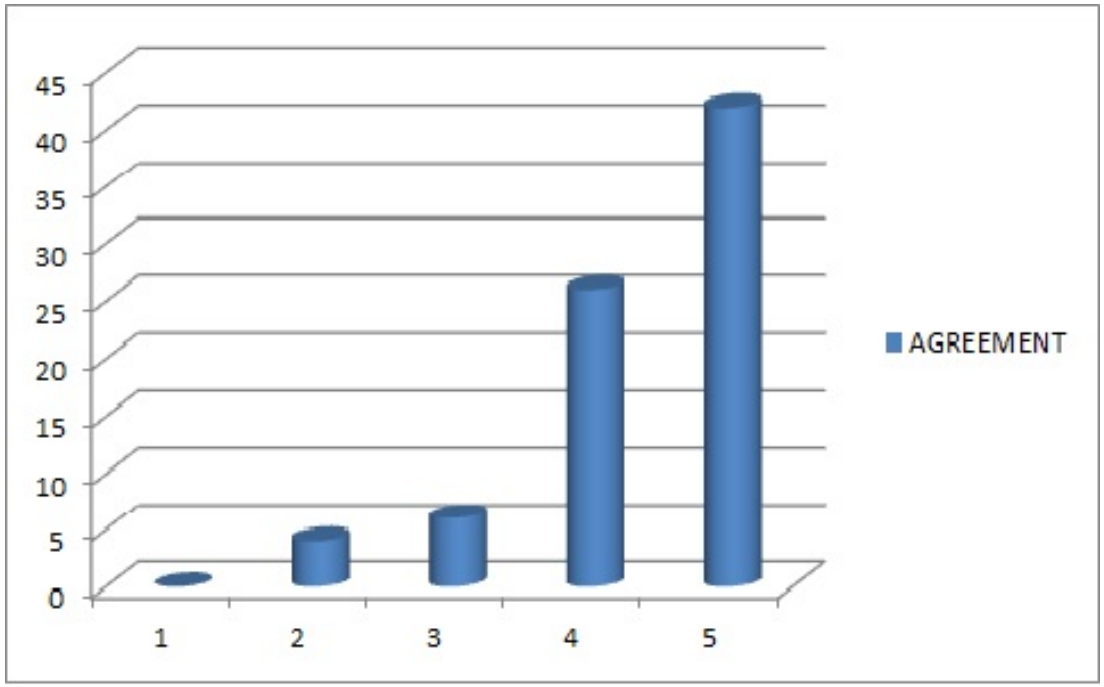


Figure: Agreement Level with the Method

Assessing the advantages of the method II

Post-assessment

	No Method (2012/2013)	Method (2013/2014)
Students succeeding	35%	55.6%[a]

[a] 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

Conclusions

Pros

- 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

Cons

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.

Ben-Ari, M. A Suite of Tools for Teaching Concurrency. Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2004, Leeds, UK, June 28-30, 2004.

Bloom, B.S., et al. Taxonomy of Educational Objetives: Handbook I, Cognitive Domain. New York: David McKay, 1956.

Carro, M., Herranz, A. & Mariño, J. A Model-Driven Approach to Teaching Concurrency. ACM Transactions on Computing Education, vol. 13, issue 1, 2013.

Fethi A. Inan and Deborah L. Lowther and Steven M. Ross and Dan Strahl. Pattern of classroom activities during students use of computers: Relations between instructional strategies and computer applications. Teaching and Teacher Education, vol. 26, no. 3, 540-546, 2010.