HELLENIC REPUBLIC
UNIVERSITY OF MACEDONIA

Using Distributed Pair Programming in a Java Course

Stelios Xinogalos
Department of Applied Informatics,
School of Information Sciences, University of Macedonia,
Egnatia 156, 54636 Thessaloniki, Greece
stelios@uom.edu.gr

Learning Programming

- ✓ **Learning programming has always been difficult for novices**.
- ✓ The difficulties are various and have been heavily studied in the literature.
- ✓ The main difficulties recorded refer to the intrinsic difficulties of programming structures both in terms of their syntax and semantics, the notional machine as defined by du Boulay, the programming environments used that do not support novices in program development and debugging, as well as the problems solved that do not engage students.
- ✓ For several decades, instructors and researchers made efforts to deal with the aforementioned difficulties through specially designed programming environments that utilized various forms of Educational Technology.
- ✓ All these environments aimed at **making programming more accessible and appealing to novices**.

Educational Technology & Programming Environments

- ✓ Programming microworlds
- ✓ Structure editors
  - ✓ Syntax editors
  - ✓ Iconic programming languages – flowchart-based programming environments

DAAD Workshop 2013

- ✓ Software visualization – program animators
- ✓ Improved diagnostic capabilities of compilers
- ✓ Educational games

DAAD Workshop 2014

- ✓ Pair Programming & Distributed Pair Programming
- ✓ Computer Supported Collaborative Learning

DAAD Workshop 2016

Research at UOM

- ✓ Programming microworlds
- ✓ Structure editors
- ✓ Software visualization – program animators
- ✓ Improved diagnostic capabilities of compilers

objectKarel 2002
S. Xinogalos, PhD
M. Satratzemi (Supervisor), V. Dagdilelis & G. Evangelidis (Advisors)

Karel 2006
S. Xinogalos, M. Satratzemi, V. Dagdilelis, G. Evangelidis
Operational Programme "Education and Initial Vocational Training II (EPEAEK)"
Novelty: combining various forms of Educational Technology used in isolation in preceding environments

- ✓ Pair Programming & Distributed Pair Programming
- ✓ Computer Supported Collaborative Learning

SCEPPSys 2015
D. Tsompanoudi, PhD
M. Satratzemi (Supervisor), G. Evangelidis & S. Xinogalos (Advisors)
Novelty: utilizing Collaboration Scripts in a Distributed Pair Programming System

- ✓ Educational games

CMX 2015
C. Malliarakis, PhD Candidate
M. Satratzemi (Supervisor), I. Refanidis & S. Xinogalos (Advisors)
Novelty: customizable (educational content, scenario) MMORPG for programming

Presentation Contents

- ✓ Pair Programming, Distributed Pair Programming & Collaboration Scripts
- ✓ The Educational DPP System of SCEPPSYs
- ✓ Using DPP in an “Object-Oriented Design and Programming” course

  => *student participation and performance*
- ✓ Using DPP in an “Object-Oriented Programming” course

  => *monitoring the course and students*

Presentation Contents

- ✓ **Pair Programming, Distributed Pair Programming & Collaboration Scripts**
- ✓ The Educational DPP System of SCEPPSYs
- ✓ Using DPP in an “Object-Oriented Design and Programming” course

  => *student participation and performance*
- ✓ Using DPP in an “Object-Oriented Programming” course

  => *monitoring the course and students*

PP, DPP & Collaboration Scripts

Pair Programming (PP)	Distributed Pair Programming (DPP)	Collaboration Scripts
• collaboration • continuous knowledge transfer, • negotiation and sharing of programming skills • production of higher quality software in a shorter time • In academic settings: students are more confident, enjoy programming and improve academic performance	• remote collaboration • DPP systems were built for professional, personal or academic purposes • DPP systems are appropriate for distance education and eliminate scheduling problems • research studies indicate that DPP maintains most benefits of PP	• Scaffolding technique used in Computer Supported Collaborative Learning (CSCL) for structuring collaborative interactions • Collaboration scripts were used in SCEPPSys for dealing with unequal student engagement that has been recorded for PP and DPP

PP, DPP & Collaboration Scripts

- ✓ The adoption of PP in the classroom resulted in the development of a considerable number of educational DPP systems.
- ✓ Some of them were built as standalone applications, such as:
  - ✓ **COLLECE**
  - ✓ **COPPER**
- ✓ The majority of educational DPP systems were built as plugins for the Eclipse IDE, such as:
  - ✓ **Sangam**
  - ✓ **RIPPLE**
  - ✓ **Xpairtise**

The evaluation of those systems generally reported positive findings on students' attitude and program quality.

As a drawback, the evaluation of XPairtise revealed less active interactions between pair programmers. Instead of equal contributions in the program code, the pairs did rarely switch roles and one team member dominated each DPP session.

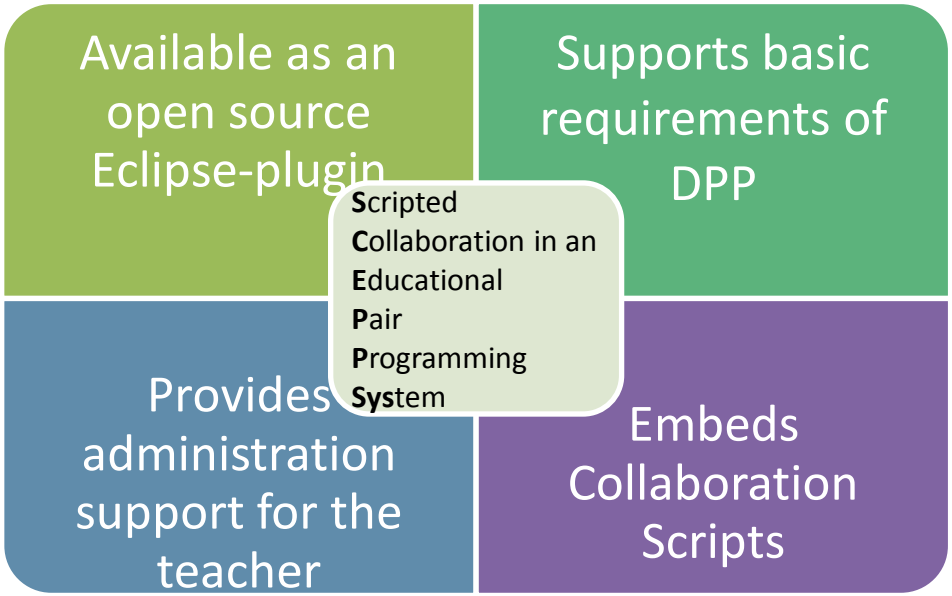
Presentation Contents

- ✓ Pair Programming, Distributed Pair Programming & Collaboration Scripts
- ✓ **The Educational DPP System of SCEPPSYs**
- ✓ Using DPP in an "Object-Oriented Design and Programming" course

  => *student participation and performance*
- ✓ Using DPP in an "Object-Oriented Programming" course

  => *monitoring the course and students*

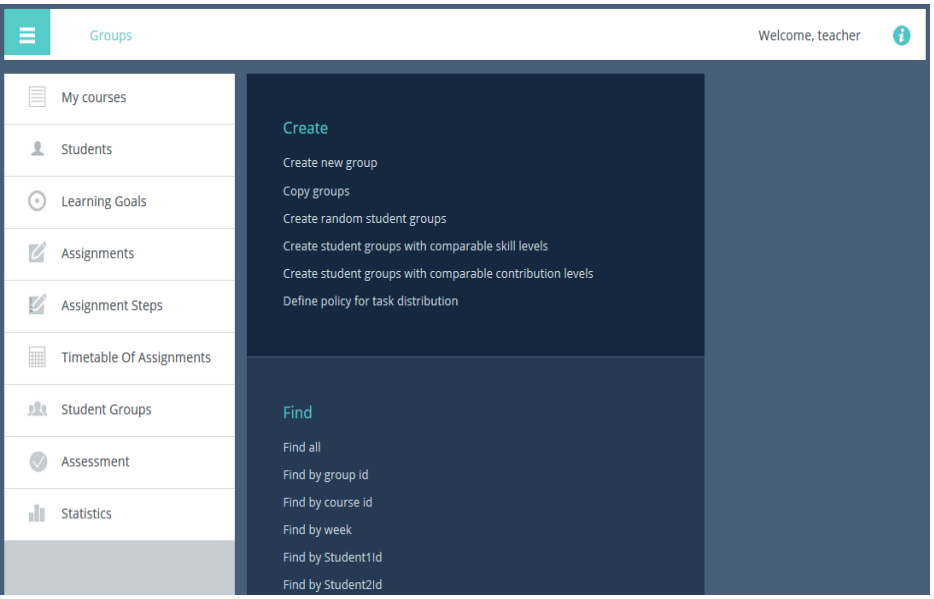
SCEPPSYs – an educational DPP system



https://sites.google.com/a/uom.edu.gr/despinats/

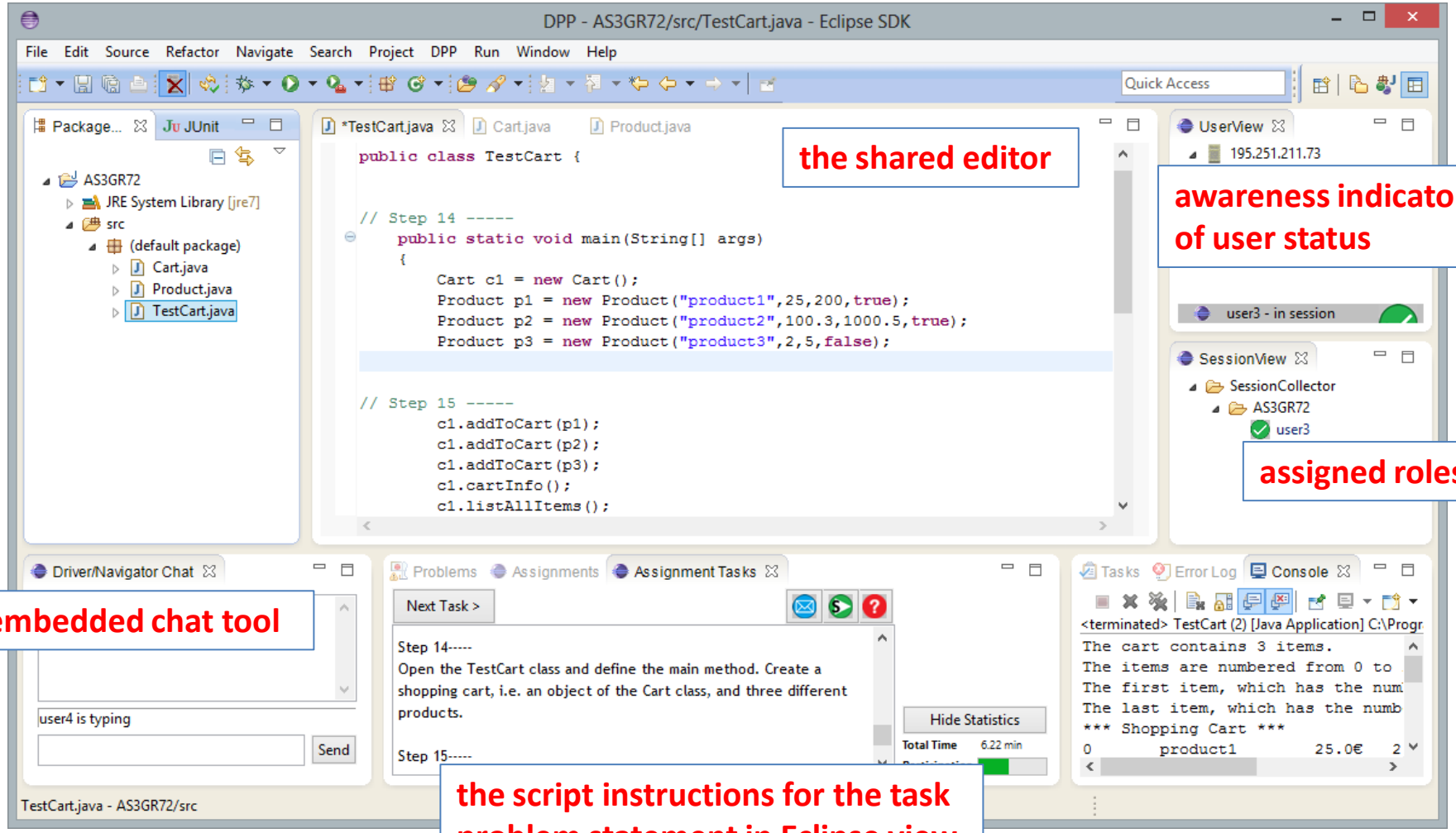
- ✓ a *server* for dispatching messages between the clients
- ✓ a *database* for storing user's accounts, information about courses, groups and assignments, shared projects and statistics
- ✓ a *web-based authoring tool* used by instructors for scripting DPP
- ✓ an *Eclipse plugin* installed by students

SCEPPSYs – setting up a course



- ✓ **Administration panel:** allows instructors to organize programming assignments, to monitor students' progress and to extract collaboration related analytics.
- ✓ DPP is practiced in the form of **collaboration scripts** that are adapted to the requirements of DPP and can be defined through the administration panel of SCEPPSys.
- ✓ A collaboration script includes the definition of participants, groups, programming tasks and turn-taking policies that specify the distribution of the driver/navigator roles among the programmers.

SCEPPSYs – carrying out a typical DPP session



SCEPPSYs – carrying out a typical DPP session

- ✓ Group members meet online and request a PP session.
- ✓ A **shared project** is automatically generated inside the workspace of both students and the programming tasks are displayed in a separate area.
- ✓ Students solve the tasks by adopting the **roles of the driver and navigator** and switch roles according to the task distribution policy.
- ✓ During the session a **text-based chat** provides a means of communication and coordination between the team members.
- ✓ To motivate students, metrics like **driving time** and **individual participation rates** are displayed and students may retrieve useful **hints** for each step during the problem solving process.
- ✓ Students may submit the assignment on session close or continue the DPP session at another time.

Presentation Contents

- ✓ Pair Programming, Distributed Pair Programming & Collaboration Scripts
- ✓ The Educational DPP System of SCEPPSYs
- ✓ **Using DPP in an "Object-Oriented Design and Programming" course**

  => *student participation and performance*
- ✓ Using DPP in an "Object-Oriented Programming" course

  => *monitoring the course and students*

Experimental Design

- ✓ **Course**: "Object-oriented design and programming" (**2013-14**)
- ✓ **Department**: Technology Management direction, Department of Applied Informatics, University of Macedonia, Greece
- ✓ **Semester**: 4
- ✓ **Duration**: 13 weeks (1 hour lecture + 2 hours lab)
- ✓ **Programming language**: Java
- ✓ **Participants**: 74 students

⇒*Students were asked to solve eight projects in pairs instead of individually as homework.*

⇒ *SCEPPSYs was used for applying DPP.*

⇒ *The projects counted for 20% of the final grade.*

Main Results

- ✓ The incorporation of DPP in the course significantly **improved course pass rates**:

	2011	2012	2013	**2014**
Pass Rate	43%	51%	37%	**69%**

- ✓ Students **enjoy to work in teams** and **acknowledge the value of DPP**:
  - ✓ 83% of the students stated they would collaborate again in future programming assignments
  - ✓ As an overall experience, DPP was rated with an average score of 3.81 (SD = 0.74) on a scale of 1 (very poor) to 5 (very good)

Main Results

✓ Students confirmed the main benefits of DPP:

With DPP…	Mean
Students share knowledge and problem solving skills	3.83
Errors in program code can be found sooner	3.79
Learning programming is facilitated	3.74
Learning to program is more enjoyable	3.79
Students can solve more problems on their own	3.58
Students are more confident in their assignment solutions	3.61
Students become more responsible in completing the assignments	4.21

Student feedback on DPP and Collaboration (Likert scale: 1 (strongly disagree) – 5 (strongly agree))

Main Results

- ✓ Combining collaboration scripts and DPP yields comparable student efforts:
  - ✓ Although previous PP and DPP studies report asymmetries in student participation levels, the evaluation of SCEPPSys could not confirm these findings.
  - ✓ The use of collaboration scripts to distribute user roles during DPP sessions, proved a successful approach to address the most common problem of group work, and had a positive impact on students' contribution.

Presentation Contents

- ✓ Pair Programming, Distributed Pair Programming & Collaboration Scripts
- ✓ The Educational DPP System of SCEPPSYs
- ✓ Using DPP in an "Object-Oriented Design and Programming" course

  => student participation and performance

- ✓ **Using DPP in an "Object-Oriented Programming" course**

  => *monitoring the course and students*

Case Study

- ✓ **Course**: "Object-oriented programming" (**2015-16**)
- ✓ **Department**: Department of Applied Informatics, University of Macedonia, Greece
- ✓ **Semester**: 3
- ✓ **Programming language**: Java
- ✓ **Duration**: 13 weeks (3 hours lab)
- ✓ **Participants:** 94 students (47 pairs) submitted at least one out of the six projects assigned

$\Rightarrow$ *optional programming assignments*
(bonus in case of passing the final exams)

Statistics reported by SCEPPSYs

Status (project submitted, not submitted, not found)	Contribution of first student (in %)
Task Distribution policy (roles rotating, balanced knowledge, free)	Contribution of second student (in %)
Total time spent to solve a project (min)	Number of steps solved according to role distribution policy
Driving time spent to solve a project (min)	Driving time of first student
Driving / Total time ratio	Driving time of second student
Number of sync runs	Non driving time of first student
Number of role switches	Non driving time of second student
Number of retrieved hints	Messages sent by first student
Contribution of first student (number of characters)	Messages sent by second student
Contribution of second student (number of characters)	

Statistics reported by SCEPPSYs

The statistics calculated and reported by SCEPPSys in combination with the projects' grades can be used as *metrics* for:

- ✓ *monitoring the fulfillment of the courses' goals in general*
- ✓ *detecting difficulties with specific OOP concepts/constructs*
- ✓ *detecting students' progress in programming*
- ✓ *detecting undesirable behaviors (e.g. plagiarism)*
- ✓ *detecting problems in collaboration between the members of a pair*

Indications provided by statistics

STATISTIC	INDICATIONS
Total and driving time	• level of difficulty of an assignment • difficulties of students with the underlying OOP concepts • help realize students' workload
Driving/total time ratio	detect odd or even extreme behaviors , such as "copying a solution" or "working offline"
Number of retrieved hints	• the more difficult an assignment is • the less confident students are for their solution a bigger number of hints is retrieved
Messages sent during problem solving	• degree of cooperation and communication between the members of a pair • difficulty of an assignment
Number of Synchronized Program Executions	• monitor students' problem solving strategies (e.g. incremental development and testing) • in combination with other statistics can indicate potential difficulties in achieving the goals of an assignment

Statistics for the projects

PROJECT	LEARNING UNIT	NUMBER OF CLASSES (STEPS)	LOC (Lines Of Code)	NUMBER OF PROJECTS	GRADE (in scale 0..10)	TOTAL TIME (MIN)	DRIVING TIME (MIN)	DRIVING / TOTAL TIME RATIO	NUMBER OF SYNC RUNS	NUMBER OF RETRIEVED HINTS	MESSAGES SENT BY EACH GROUP
#1	Class definition, main	2 (13)	90	45	9.25	190	36	25%	8	9 (69%)	103
#2	Class associations - relationship	3 (16)	120	46	**8.72**	231	55	28%	13	**13 (81%)**	106
#3	Object collections – ArrayList	3 (23)	160	39	9.15	**262**	63	30%	**25**	15 (65%)	**153**
#4	Inheritance & polymorphism	4 (16)	114	35	9.21	127	40	35%	9	9 (56%)	86
#5	GUI, event handling (+inheritance)	6 (24)	135	**28**	9.36	**243**	50	25%	**18**	16 (67%)	**128**
#6	Binary files (+inheritance, ArrayList, Comparator)	5 (5)	210	**25**	**8.76**	174	34	**21%**	15	-	100

Case Study – results (1/4)

- ✓ **Grades**: students' mean *grades* in all the projects were very good (at least 8.72).

- ✓ **Number of projects**: students tend to give up the effort as the course reaches the end (last two assignments):
  - ✓ cognitively demanding course
  - ✓ the cognitive overload of students leads a certain percentage of them in dropping out of their responsibilities at some point
  - ✓ assignments were not obligatory
  - ✓ 41% of the students had not passed the first semester "Procedural Programming" course based on C.

  Understanding the reasons for dropping out needs a much deeper analysis of the available data in order to draw solid conclusions.

Case Study – results (2/4)

- ✓ **Duration**: implementing programs is a time consuming activity - students working in pairs spent approximately two to four hours for writing the code for an assignment.
  - ✓ nearly one fourth to one third of this time was spent on actually writing code

- ✓ **Number of sync runs**: students do run the programs, but it cannot be told in certainty that they do it during problem solving for testing their program, or because they have to make continuous corrections due to raised exceptions and/or logical errors that lead to wrong output.
  - ✓ at least 8 sync runs were recorded in average for each project, which is an indication of incremental development and testing.

Case Study – results (3/4)

- ✓ **Messages sent by each group**: 86 to 153 messages were sent by the members of each pair in average for each project, although Skype and Facebook was also used.
  - ✓ this is definitely a strong indication of collaborative work and exchange of perceptions and knowledge.

- ✓ **Coverage of syllabus (assignments)**: SCEPPSys helps the instructor monitor the coverage of the intended OOP concepts/constructs by reporting the frequency that each learning goal has been addressed in the context of the assignments.
  - ✓ in the on-line questionnaire the majority of students agreed or completely agreed that the quality of the assignments was good (86.3%) and that the assignments covered in a high degree the content of the course (84.5%).

Case Study – results (4/4)

- ✓ **Difficulty of the assignments**: taking into account the statistics recorded (number of projects, total time, number of hints retrieved, ..) in combination with the mean grade, conclusions were drawn regarding the difficulty of the assignments.
- ✓ These were partly confirmed by students' replies in the questionnaire regarding the difficulty of the assignments:

PROJECT	LEARNING UNIT	NOT SUBMITTED	EASY	OF LOW DIFFICULTY	OF MEDIUM DIFFICULTY	DIFFICULT	OF HIGH DIFFICULTY
#1	Class definition, main	6.90%	53.40%	29.30%	6.90%	3.40%	0%
#2	Class associations – relationship	1.70%	36.20%	51.70%	8.60%	1.70%	0%
#3	Object collections – ArrayList	1.70%	17.20%	39.70%	37.90%	3.40%	0%
#4	Inheritance & polymorphism	5.20%	8.60%	20.70%	**50%**	**13.80%**	**1.70%**
#5	GUI, event handling (+inheritance)	15.50%	0%	10.30%	15.50%	**43.10%**	**15.50%**
#6	Binary files (+inheritance, ArrayList, Comparator)	17.20%	1.70%	5.20%	12.10%	**32.00%**	**31%**

Conclusions (1/2)

- ✓ Using **DPP** in a Java course:
  - ✓ can improve students' performance
  - ✓ students enjoy team work
  - ✓ students acknowledge the benefits of DPP

- ✓ Applying DPP with **Collaboration Scripts** solves the problem of asymmetries in student participation levels.

- ✓ The statistics reported by **SCEPPSys** can be utilized for monitoring:
  - ✓ an OOP course based on Java
  - ✓ difficulties with specific concepts
  - ✓ the quality of collaboration between the members of a pair
  - ✓ students' progress in programming

Conclusions (2/2)

- ✓ An important extension of SCEPPSys would be the ability to provide more advanced reports that would automate the process of detecting:
  - ✓ *potential difficulties with specific OOP concepts/constructs*: by calculating the average grade achieved by students for each learning goal.
  - ✓ *students' progress in programming*: by calculating and reporting important changes in the grades of his/her projects, as well as the contribution of a student in the projects.
  - ✓ *undesirable behaviors*: by comparing the total and driving time for a project, and also with a minimum required time defined by the instructor.

References

Tsompanoudi, D., & Satratzemi, M. (2011). Enhancing Adaptivity and Intelligent Tutoring in Distributed Pair Programming Systems to Support Novice Programmers, *Proceedings of the 3rd International Conference on Computer Supported Education, CSEDU 2011*, pp. 339-344.

Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2015). Distributed Pair Programming using Collaboration Scripts: An Educational System and initial Results, *Informatics in Education*, Vol. 14, No 2, 291-314.

Xinogalos, S., Malliarakis, C., Tsompanoudi, D. and Satratzemi, M. (2015). Microworlds, Games and Collaboration: three effective approaches to support novices in learning programming. In *Proceedings of the 7th Balkan Conference on Informatics Conference* (BCI '15).ACM, New York, NY, USA, Article 39, 8 pages.

Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2016). Evaluating the Effects of Scripted Distributed Pair Programming on Student Performance and Participation, *IEEE Transactions on Education*, Volume 59, Number 1, 24-31, DOI: 10.1109/TE.2015.2419192.

Xinogalos, S., Satratzemi, Tsompanoudi, D. & Chatzigeorgiou, A. (2016). Monitoring an OOP Course Through Assignments in a Distributed Pair Programming System, *5th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, Budapest, Hungary, 29.-31.08.2016.

Thank you!