COURSE DESCRIPTION CARD - SYLLABUS

Course name
Distributed Processing Tools

Course

Field of study
Computing

Year/Semester
1/1

Area of study (specialization)
Distributed Systems

Profile of study
general academic

Level of study
Second-cycle studies

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other (e.g. online)

Tutorials

Projects/seminars

Number of credit points

Lecturers

Responsible for the course/lecturer:
Dariusz Wawrzyniak, Ph.D.

Responsible for the course/lecturer:

Prerequisites

Learning outcomes of the first cycle studies defined in the resolution of the PUT Academic Senate (verified in the admission process to the second-cycle studies, available at the website of the faculty www.cat.put.poznan.pl or www.fc.put.poznan.pl), especially programming skills in commonly used programming languages (e.g. C/C++, Java), knowledge of operating systems and concurrent processing, basic experience in network programming and distributed computing.

Course objective

1. Provide students with fundamental knowledge of tools for building distributed systems, related services and mechanisms, as well as techniques and paradigms of distributed programming.
2. Develop the ability to choose an appropriate approach to solving distributed processing problems and constructing distributed systems.

Course-related learning outcomes

Knowledge

1. Has a structured and theoretically founded general knowledge related to the paradigms of distributed system programming.

2. Has advanced detailed knowledge regarding environments and tools applied in building distributed application.
3. Knows development trends in environments and tools for distributed system construction.
4. Has advanced and detailed knowledge of the processes occurring in the life cycle of distributed systems.
5. Knows advanced methods, techniques and tools used to solve complex engineering tasks in a selected area of distributed system construction.

Skills

1. Is able to plan and carry out experiments, including computer measurements and simulations, interpret the obtained results and draw conclusions and formulate and verify hypotheses related to complex engineering problems of distributed processing tools.
2. Can use analytical, simulation and experimental methods to formulate and solve engineering problems of distributed processing tools.
3. Is able to assess the suitability and the possibility of using new achievements (methods and tools) and new IT products related to distributed processing tools.
4. Can carry out a critical analysis of existing technical solutions related to distributed processing tools and propose their improvements (streamlines).
5. Is able to assess the usefulness of distributed processing methods and tools for solving an engineering task, consisting in the construction or evaluation of an IT system or its components, including the limitations of these methods and tools.
6. Is able - in accordance with a given specification, taking into account non-technical aspects - to design a complex device, IT system or process and implement this project - at least in part - using appropriate distributed processing methods, techniques and tools, including adapting to this purpose existing tools or developing new ones.

Social competences

1. Understands that in the field of IT the knowledge and skills related to distributed processing tools quickly become obsolete.
2. Understands the importance of using the latest knowledge in the field of distributed processing in solving research and practical problems.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Lecture: written exam with about four open-ended questions or problems with about 100 points to score and 50 points to pass.
Classes: practical assignments focused on the application of distributed processing tools.
Extra points for an outstanding activity or elaboration of additional aspects of a problem/tool.

Programme content

The lecture covers the following topics:

- basic terms and concepts, including the distinction between network system, distributed system, and middleware;

- the classification of interprocess communication mechanisms;

- remote procedure call, including the issue of RPC protocol and failure semantics, the question of idempotent procedure and stateless RPC services;

- object-oriented approach to distributed system construction, covering object classification in the context of distributed processing, interface definition and its particular function as a parameter type, inheritance;

- concurrency in executing remote services, resource sharing and synchronisation, the control of threads;

- message-oriented middleware: the concept, queuing and publish-subscribe paradigms, broker-based and non-broker examples;

- tuple spaces: the concept of Linda, associative addressing, the specification of JavaSpeces with examples;

- Ada-95: principles of programming (data types, programming constructs, packages, object-orientedness), concurrency (tasks, rendezvous, protected objects), distributed processing according to the specification in Annex E (units categories, system description language).

Laboratory classes are split into 15 units conducted at workstation running under Linux. Students are assigned tasks to be performed individually or occasionally in small groups. The topics include:

- OS level concurrency: POSIX threads and their synchronisation, the concept of a monitor;

- a project of distributed monitor based on ZeroMQ;

- SunRPC: verification of failure semantics based on a remote counter, complex network service (with concurrent handling and callback);

- JavaRMI or Internet Communications Engine as an example of the object-oriented approach, the comparison of the remote counter in procedural and object-oriented approach, object-based remote buffer;

- MOM systems (remote buffer generalisation), the comparison of queuing and publish-subscribe communication mode;

- JavaSpaces: an example of associative addressing in interprocess communication;

- principles of Ada programming based on an implementation of a simple algorithm (e.g. bubble sort, selection sort);

- concurrency in Ada: task creation, synchronisation through rendezvous, protected objects, implementation of a semaphore and bounded buffer as an example;

- distributed programming based on Annex E with examples involving units categorised as RCI, RT, SP.

Teaching methods

1. Lectures: presentation of slides (multimedia showcase), discussion of problems, solving tasks on blackboard.

2. Classes: solving tasks, practical exercises, discussion, conducted in a computer laboratory under the control of Unix-like operating system.

Bibliography

Basic

1. G. Coulouris, J. Dollimore, T. Kindberg, Systemy rozproszone. Podstawy i projektowanie, WNT, W-wa, 1999.

2. A.S. Tanenbaum, M. van Steen, Systemy rozproszone. Zasady i paradygmaty, WNT, W-wa, 2006.

Additional

1. M. Gabassi, B. Dupouy, Przetwarzanie rozproszone w systemie UNIX, Lupus, W-wa, 1995.

2. E. Freeman, S. Hupfer, K. Arnold, JavaSpaces Principles, Patterns, and Practice, Addison-Wesley, 1999.

3. Z. Huzar, Z. Fryźlewicz, I. Dubielewicz, i in., Ada 95, Helion, Gliwice, 2001.

4. J. Barnes, Ada 2012, Cambridge University Press, 2014.

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,0
Classes requiring direct contact with the teacher	60	2,5
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation) [1]	65	2,5

[1] delete or add other activities as appropriate