Teamwork Across Disciplines: High-Performance Computing Meets Engineering

Philipp Neumann[1](✉), Christoph Kowitz[1], Felix Schranner[2], and Dmitrii Azarnykh[2]

[1] Department for Informatics, Technische Universität München (TUM), Boltzmannstr. 3, 85748 Garching, Germany
philipp.neumann@tum.de

[2] Department of Mechanical Engineering, Technische Universität München, Boltzmannstr. 15, 85748 Garching, Germany

Abstract. We present a general methodology to combine interdisciplinary teamwork experience with classical lecture and lab course concepts, enabling supervised team-based learning among students. The concept is exemplarily applied in a course on high-performance computing (HPC) and computational fluid dynamics (CFD). Evaluation and student feedback suggest that competences on both teamwork as well as on lecture material (CFD and HPC) are acquired.

1 Interdisciplinary Education and Teamwork

1.1 Introduction

There is hardly a field which influences other research disciplines as much as computer science. Cutting-edge innovations in robotics support industrial developments, simulation in science and engineering allows to construct and test virtual bridges, airplanes or cars, or pattern recognition and enhanced search algorithms enable big data evaluations in social sciences (and basically any other field), just to name a few examples.

With various levels of parallelism to be exploited within current processors, this arguing particularly holds for parallel and high-performance computing. Compute- and data-intensive software for both industrial or academic applications needs to support this parallelism. This implies a strong need for interdisciplinary education and teamwork between computer science and the related disciplines ("computer science/HPC + X"). While interdisciplinary study programs—such as computational science and engineering, bioinformatics, or information systems—are widely established, an interdisciplinary education at the course level is rarely realized: single lectures, lab courses or seminars are typically provided and organized by a single department and consequently focus on the perspective of the related discipline, cf. Fig. 1(a). Only few examples at German universities exist which invoke two or more departments for interdisciplinary teaching and education at course level, e.g. courses of the project StiL

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 125–134, 2015.
DOI: 10.1007/978-3-319-27308-2_11

[6]. Besides, soft skills such as communication, work management, or components of successful, interdisciplinary teamwork are often not encountered in the study programs at all, except for optional pure soft skill courses. The latter courses consider soft skills independent from the current study program—see Fig. 1(b)—which allows to exchange with other disciplines on this topic on the one hand, but hinders from practising/learning interdisciplinary teamwork in a realistic setting on the other hand.

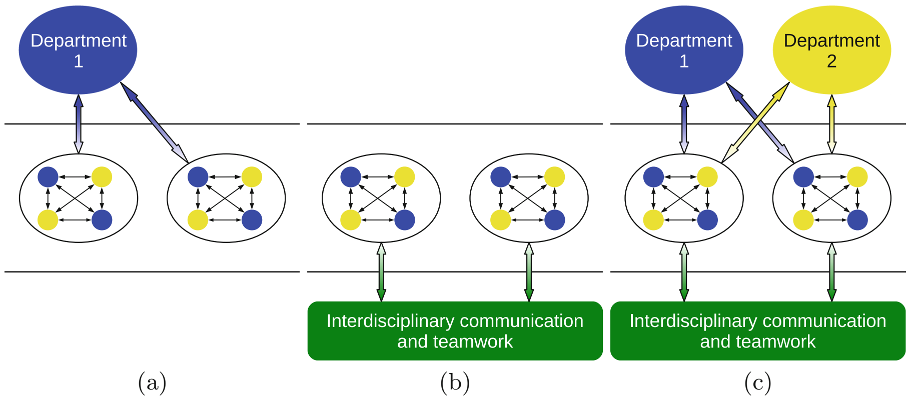
Department 1
Department 1
Department 2
Interdisciplinary communication and teamwork
Interdisciplinary communication and teamwork
(a)
(b)
(c)

Fig. 1. Different course designs. Filled yellow and blue circles represent students interacting in group work and belonging to department 1 (blue) or department two (yellow). (a) Department 1 teaches students from disciplines 1 and 2; focus of the course is put on discipline 1. (b) Interdisciplinary education. All students are taught interdisciplinary communication and teamwork, abstaining from discipline-specifics. (c) Concept *Teamwork Across Disciplines*. Two departments teach mixed student teams, including soft skill aspects (Color figure online).

(Potentially) Interdisciplinary student projects are more common; examples at German universities comprise the one-year student group project at Univ. Paderborn (8–16 students) [4], or the IDP (interdisciplinary project) at TU München (individual or group projects) [3] as part of the master's programs in computer science. Allowing for individual training and practice of interdisciplinary communication and other soft skills, particular guidance and supervision with respect to the latter is rarely provided.

Only few novel approaches and concepts which differ from Fig. 1(a) and (b) and aim to improve the current state of education in interdisciplinary teamwork have been proposed over the last years. Leiffer, Graff and Gonzalez point out advantages of project teamwork and provide a list of curriculum measures to support the students' teamwork education [1]. Another example is given by the interdisciplinary, team-based, multi-semester teaching approach to games development which is presented in [2] and allows "students to exercise their technical and artistic skills on a multi-semester project while also developing their teamwork, problem-solving, leadership, and communication skills."

1.2 Challenges

The design of a *course on interdisciplinary education and application* has to take into consideration several criteria; we restrict considerations on invoking two disciplines in the following.

The topic of the course needs to be *really interdisciplinary* (C1). A "toy problem" won't do, since no knowledge transfer would be invoked between the disciplines in this case, and the real challenge of understanding each other would not be revealed. Both disciplines need to be involved at the same or similar amount to stimulate bi-directional knowledge transfer, interdisciplinary learning and discussions (C2). This may also hold for the involvement of two departments in the teaching process, cf. Fig. 1(c). The capability to participate in interdisciplinary discussions requires basic understanding of the topic with respect to both disciplines and soft skill education (C3). This requirement either builds a prerequisite for students entering the course, or implies the necessity of a start-up phase in the course which recapitulates the discipline-specific basics. Finally, to stimulate interdisciplinary, open-minded and creative exchange among students, an activating, flexible course setting is required (C4), adapting to the (individual) students' needs.

1.3 Outline

In the following, we provide details on the course concept *Teamwork Across Disciplines* that we exemplarily deployed in our course *Turbulent Flow Simulation on HPC-Systems*. This course brings together students from mechanical engineering and computer science, working in mixed teams to develop a CFD simulation software for compute clusters.

We describe the course design in Sect. 2, and point out how we address the challenges (C1-C4). Based on the teamwork idea, we particularly point out

– how to form interdisciplinary student teams and support the team building process,
– how to assist students at team-based learning and teaching (*supervised teaching*),
– our approach to examine and evaluate teamwork processes.

We detail feedback and outcomes of the course evaluation in Sect. 3 and draw conclusions from this course realization in Sect. 4.

2 Course Curriculum

2.1 Teamwork Across Disciplines: Concept

To allow for "truly interdisciplinary" education, both disciplines D1 and D2 need to be equally involved. Hence, also both respective departments are involved to equal parts in education and teaching. The arising course is embedded into the curricula of both study programs D1 and D2. Though not a strict requirement, we decided to design the course for 12–32 students at master level to make sure that the students potentially have preliminary knowledge of both subjects, and deeper understanding of one of both subjects.

Next, a “problem” P is designed which consists of subproblems related to D1, D2, or both. A strict coupling of D1 and D2 in the problem description is mandatory. In the scope of the course, student teams of 2–4 students shall solve P. Each student team hence needs to have knowledge of both fields D1 and D2 to successfully work on the problem. Besides, students are supported and taught further details to solve P in accompanying lectures throughout the lecture period.

Some of the lectures are dedicated to soft skill components, such as teamwork, communication, or work distribution, cf. (C3). Besides, *supervised team meetings* are introduced once per week. On the one hand, these meetings build a fixed point of contact of the team members. They can have a team meeting, discuss their work progress and solve problems that, e.g., a single team member could not solve himself. On the other hand, the supervised team meetings also build a fixed point of contact of each team with the other teams and the course supervisors (from departments D1, D2). The teams report on their work status, can ask topic-related questions, or discuss teamwork issues with the supervisors. The latter is a challenge for the course supervisors: formulating a teamwork problem properly often comes close to the actual solution—detecting the problem, however, is very difficult. It is thus also the supervisors who need to grasp a look inside the teams, observe the interaction of the team members, analyse strengths and weaknesses of the team, and give respective advice.

2.2 Realization: Turbulent Flow Simulation on HPC-Systems

Prerequisites, Goals and Course Structure. The concept from Sect. 2.1 was realized in the course *Turbulent Flow Simulation on HPC-Systems*. The course is designed for master students from disciplines/departments of computer science (D1:=CS) and mechanical engineering (D2:=ME). It comprises 2×90 min of lectures/supervised team meetings per week. Both study programs ME and CS contain courses in programming or in a minor such as (computational) fluid dynamics, so that a basic knowledge of both ME/CS can be presumed and dictates a sufficient set of course prerequisites.

At the beginning of the course, the students download the C++ code frame NS-EOF for incompressible, laminar, three-dimensional Navier-Stokes-based flow simulations [8]. NS-EOF makes use of most prominent object-oriented software design patterns: data structures, algorithmic and parallelization components are encapsulated and can be exchanged or extended easily. A key component of most incompressible Navier-Stokes solvers is provided by a Poisson solver for the pressure unknowns. In NS-EOF, either a simple SOR-based solver can be employed for this problem, or a more sophisticated implementation based on PETSc [7]. The latter is particularly used in this course.

In the scope of the course, NS-EOF shall be extended by distributed memory parallelism using the message passing interface (MPI) library and by incorporation of at least one turbulence model; the respective components of NS-EOF are thus missing in the code frame. The simulation shall be executed on the mid-sized MAC-cluster [5] administrated by the Leibniz Supercomputing Centre/TUM. The cluster consists of AMD bulldozer, Intel SandyBridge and other partitions and thus builds a perfect "playground" for the students to experiment with different pieces of hardware. Parallelization/turbulence modeling is new to the students from ME/CS. Joint work for successul completion of this problem (P:=extension of code frame by parallelization and turbulence modeling) is hence required.

The course is structured in three phases: *single phase*, *group phase*, and *project phase*. The complete structure for a semester period of twelve weeks is detailed in Table 1 and laid out in the following.

Single Phase. The *single phase* represents the introductory part of the course. Spanning over 2 weeks, the students are introduced to the basics of the (Reynolds averaged) Navier-Stokes equations. Besides, the methodology to numerically solve the Navier-Stokes equations is repeated and illustrated at the given code frame. The latter topic is presented along with a precise introduction to NS-EOF and principles of object-oriented programming. The introductions are designed in form of four plenary lectures; the code introduction is given in form of a mix of lecture and interactive session.

During the single phase, each student is obliged to extend NS-EOF by visualization routines to output VTK-compatible files; these can be read by the free visualization software ParaView. Individually working on this extension, the single phase guarantees that all students become familiar with the concepts of the object-oriented code frame and with the principles of laminar flow simulations (cf. (C3)), visualizing and interpreting results of three benchmark flow problems: lid-driven cavity, free channel flow, and backward-facing step flow. The *lid-driven cavity* problem works on a cubic domain and is a good scenario for qualitative understanding of vortex formation in fluid flow. For *free channel flow*, the analytical solution is known. A quantitative convergence evaluation of the finite-difference solver NS-EOF and the respective pressure Poisson solver is thus possible. Channel flow over a *backward-facing step* yields the formation of vortices on the one hand, but still relies on simple channel-like geometries on the other hand. This allows to study, e.g. turbulence, using relatively simple, yet effective turbulent boundary layer models.

An oral exam is scheduled at the end of the single phase to control the learning outcome.

Group Phase. At the beginning of the *group phase*, the problem P—MPI parallelization and turbulence modeling in NS-EOF—is revealed to the students. A concise task description is provided, including all important steps and subtasks such as domain decomposition, rank-to-coordinate mapping of processes,

Table 1. Structure of course *Turbulent Flow Simulation on HPC-Systems*. Single phase: weeks 1–2. Group phase: weeks 3–8. Project phase: weeks 9–12.

Week	Contents	Learning Outcomes	Format
1A	Reynolds Averaged Navier-Stokes (RANS)	– Comprehension of RANS – Ability to mathematically apply RANS filtering	Lecture
1B	Numerically solving Navier-Stokes	– Knowledge of solving procedure and different algorithmic steps – Ability to derive finite difference expressions for Navier-Stokes and mixed derivatives	Lecture
2A	Introduction to C++-programming	– Ability to implement classes and functions – Comprehension of C++ syntax – Comprehension of principles of object-oriented programming	Lecture
2B	Introduction to NS-EOF	– Comprehension of software design of NS-EOF – Ability to add functionality to NS-EOF	Lecture, Interactive code review
3A	Soft skills: Teamwork	– Knowledge of team structures, roles and processes	Lecture
3B	Team building (for group phase)	– Reflection of personal competences on computational fluid dynamics, turbulence and HPC	Interactive session
4	Parallelization of NS-EOF	– Ability to apply domain decomposition algorithm in NS-EOF – Knowledge of MPI library and usage	Lecture
5A	Supervised team meeting	– Ability to successfully collaborate within team structure – Ability to split work among team members and define/distribute tasks	Group work,interactive session
5B	Turbulence modeling	– Comprehension of principles of turbulence modeling (energy cascade; algebraic, one-, and two-equation models)	Lecture
6A	Supervised team meeting	– see 5A	Group work,interactive session
6B	Soft skills: Communication	– Knowledge of four-sides communication model [10] – Comprehension of potential issues in communication – Knowledge of measures to improve communication (effective listening, feedback)	Lecture, group work

(*Continued*)

Table 1. (*Continued*)

Week	Contents	Learning Outcomes	Format
7A	Supervised team meeting	– see 5A – Ability to successfully communicate in team discussions	Group work,interactive session
7B	Parallelization: Scaling and Efficiency	– Knowledge of Amdahl's law – Ability to analyze and improve scaling of MPI-parallelized software	Lecture
8A	Supervised team meeting	– see 7A	Group work,interactive session
8B	Turbulence: Error analysis	– Knowledge of boundary layer categorization – Comprehension of turbulent viscosity effects in channel flows – Ability to analyze and validate turbulence simulations	Lecture
9A	Scenario: Backward-facing step	– Knowledge of flow setup – Comprehension of physical effects in (turbulent) backward-facing step flow	Lecture
9B	Presentation of potential project topics	– Knowledge of potential code extensions and model improvements	Lecture
10	Decision on/Choice of project topics		Interactive session
11	Wind tunnel tour	– Knowledge of wind tunnel types and measurement techniques in experimental fluid dynamics	Interactive session
12	Project presentations	– Application of and practising presentation skills	Interactive session

or the evaluation of parallel performance. The problem involves both disciplines: computer scientists are required for the MPI parallelization and software design, mechanical engineers know about turbulence models and respective physical limitations. Moreover, the incorporation of turbulence models also implies modifications in the parallelization process: the global time step reduction required by NS-EOF needs to be adapted and turbulent flow quantities such as the turbulent viscosity need to be communicated between neighbouring processes. The task thus satisfies (C1). Besides, students from both disciplines need to be highly involved in the development of this code extension (C2). After introducing the building blocks of teamwork in a lecture, teams are formed based on individual reflection of personal skills and exchange with the other course participants. Every student can hence search for fellows with orthogonal skills and so become member of an optimal, interdisciplinary team.

Further details on the subtasks are discussed and taught in the following weeks 3–8. In this period, the student teams implement the extensions in NS-EOF, organizing and managing the workload autonomously. Each week, one plenary lecture on parallelization, turbulent flows, or soft skills is held. The other weekly time slot is reserved for the supervised team meetings to allow for exchange with and among the student teams, cf. Sect. 2.1.

The group phase is closed with an individual oral examination and code review to control the contribution of each team member to the team's success. Besides, the knowledge transfer within the team is controlled, e.g. via posing ME/CS questions to CS/ME students.

Project Phase. In the *project phase* (weeks 9–12), students work on an individual project in the scope of turbulent HPC simulations. Potential project topics are presented at the beginning of this phase, amongst others:

– Extension of the MPI-parallelized code NS-EOF by shared memory parallelization: for ME students, this is helpful since shared memory parallelization is typically preferred over message passing for reasons of simplicity and application on Desktop systems. For CS students, this is also a natural extension due to today's heterogeneous and many/multicore hardware devices such as Intel's Xeon Phi accelerator.
– Extension of NS-EOF by checkpointing mechanisms to allow for a restart mechanism and resilience,
– Improvement of parallel I/O routines using, e.g., MPI-IO methods,
– Incorporation of an advanced turbulence model (e.g., the Spalart-Allmaras model [9]),
– Validation of NS-EOF by comparing the simulation results to experimental results.

Students are free to also come up with own project ideas and can hence focus on their individual interests and skills, cf. (C4). Both team- or individual work on the projects is possible. Original teams from the group phase may be dissolved, and new team structures may be formed to allow optimal team constellations in terms of the preferred project topic, previous teamwork experiences etc. To further link the topic of numerical simulation on parallel systems to experimental fluid dynamics, a visit at the TUM wind tunnels and experimental fluid dynamics labs is organized.

The project phase ends with oral examinations for the project groups[1] and a plenary session in which each group presents their project results to the other students.

3 Evaluation

Based on the design described in Sect. 2, the course was conducted in the winter term 2014/15. Fourteen students, split into four groups during the group phase, participated in the course. We evaluated the course concept and design by three means: oral feedback by the students (1), centralized, department-wide mid-term

[1] As pointed out, a group may also be an individual student.

course evaluation (2), and a course-internal evaluation round at the end of the course (3).

In (2), students graded various course aspects using an integer system from 1 (positive) to 5 (negative). The course schedule (incl. deliverables) as well as communication of learning outcomes and goals were found to be highly satisfactory (graded 1 (∼50 %), or 2 (∼50 %)). With the clear focus on lab course-like interdisciplinary teamwork, autonomous working phases (graded 1 (∼80 %) and 2 (∼20 %)) and group size/formation (graded 1 (∼85 %) and 2 (∼15 %)) were also evaluated particularly positive. The course-internal evaluation (3) further supports (2): 13 out of 13 students consider the teamwork to be most supportive for their learning process in the course, as well as the support by the course supervisors. For 11 out of 13 students, "learning by doing" in the scope of this course is highly essential for the learning process. Successful learning based on autonomous and supervised working phases is hence enabled by our course concept.

The level and extent of the autonomous work program was considered challenging in (2): on a scale from 1 (much work/high difficulty) to 5 (small amount of work/low difficulty), the course level/work extent were ranked 1.29/1.67 on average. A potential reduction of the workload is currently revised.

In (2), it was also the acquisition of competences that was considered. The students consider themselves capable to work on typical tasks similar to the course contents (avg. grade 1.57 on the same scale 1–5), that is tasks from the fields HPC and CFD. The capability to transfer (scientific) methods to other concrete tasks was rated 2.0. With the evaluation (2) taking place in the middle of the group phase due to the centralized time schedule for department-wide course evaluations, this statement is, however, only of limited value. Oral and individual discussion (1) with the students after the course suggest an even higher level of satisfaction with this respect. We conclude that the supervised team-based learning combined with the lecture blocks allows the transfer of knowledge on turbulence and HPC aspects and respective acquisition of competences.

4 Conclusion

We presented a general methodology to combine interdisciplinary teamwork experience with classical lecture and lab course concepts, enabling supervised team-based learning among students. With both disciplines equally involved with respect to both course contents and student supervision, a "really interdisciplinary" setting could be established. The concept was exemplarily applied in a course on high-performance computing and computational fluid dynamics. Three-fold evaluation/student feedback suggests that the team-based learning experience supports the respective acquisition of competences. The consideration and evaluation of the course concept in other interdisciplinary settings would be of big interest to share experiences on the team-based learning process and to draw respective analogues for best practices.

Acknowledgements. The *Hochschulreferat für Studium und Lehre* of the Tech-ni-sche Uni-ver-si-tät Mün-chen is acknowledged for the financial support to realize the new course concept in the scope of the Ernst-Otto-Fischer teaching price. We thank the Munich Centre of Advanced Computing for providing the MAC-cluster and related computational resources for the student groups of our course. We particularly thank Linda Roppelt and Rudolph Aichner of the TUM-ProLehre team for their continuous advice and support on the course concept and evaluation. P. Neumann acknowledges the financial support by the priority program *1648 Software for Exascale Computing*, funded by the German Research Foundation (DFG).

References

1. Leiffer, P.R., Graff, R.W., Gonzalez, R.V.: Five curriculum Tools to enhance interdisciplinary teamwork. In: Proceedings of the ASEE Annual Conference & Exposition, pp. 6459–6469 (2005)
2. Kuhl, S.A., Pastel, R., George, R., Meyers, C.M., Freitag, M.L., Lund, J.M., Stefaniak, M.P.: Teaching interdisciplinary teamwork through hands-on game development. In: Proceedings of the ASEE Annual Conference & Exposition (2014)
3. Department of Computer Science, Technische Universität München (2015). http://www.in.tum.de/fuer-studierende/master-studiengaenge/informatik/interdisziplinaeres-projekt/fpo-2007-und-fpso-2012.html
4. Department of Computer Science, University Paderborn (2015). http://www.cs.uni-paderborn.de/en/students/study-components/project-group.html
5. Munich Centre of Advanced Computing (2015). http://www.mac.tum.de
6. StiL, University Leipzig (2015). http://www.stil.uni-leipzig.de, in particular http://www.stil.uni-leipzig.de/wp-content/uploads/2015/06/Poster-Gesellschaftliche-Strukturen-im-Wandel.pdf
7. Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., Curfman McInnes, L., Rupp, K., Smith, B., Zhang, H.: PETSc Web page (2014). http://www.mcs.anl.gov/petsc
8. Griebel, M., Dornseifer, T., Neunhoeffer, T.: Numerical Simulation in Fluid Dynamics: A Practical Introduction. SIAM, Philadelphia (1997)
9. Spalart, P., Allmaras, S.: A one-equation turbulence model for aerodynamic flows. In: AIAA Paper 92-0439 (1992)
10. Schulz von Thun, F.: Miteinander reden 1: Störungen und Klärungen. Rowohlt Verlag GmbH (2010)