J Vis (2020) 23:895–911
https://doi.org/10.1007/s12650-020-00659-6

REGULAR PAPER

Michael Burch · Elisabeth Melby

What more than a hundred project groups reveal about teaching visualization

Received: 27 February 2020 / Revised: 6 May 2020 / Accepted: 13 May 2020 / Published online: 25 June 2020

Abstract The growing number of students can be a challenge for teaching visualization lectures, supervision, evaluation, and grading. Moreover, designing visualization courses by matching the different experiences and skills of the students is a major goal in order to find a common solvable task for all of them. Particularly, the given task is important to follow a common project goal, to collaborate in small project groups, but also to further experience, learn, or extend programming skills. In this article, we survey our experiences from teaching 116 student project groups of 6 bachelor courses on information visualization with varying topics. Moreover, two teaching strategies were tried: 2 courses were held without lectures and assignments but with weekly scrum sessions (further denoted by TS1) and 4 courses were guided by weekly lectures and assignments (further denoted by TS2). A total number of 687 students took part in all of these 6 courses. Managing the ever growing number of students in computer and data science is a big challenge in these days, i.e., the students typically apply a design-based active learning scenario while being supported by weekly lectures, assignments, or scrum sessions. As a major outcome, we identified a regular supervision either by lectures and assignments or by regular scrum sessions as important due to the fact that the students were relatively unexperienced bachelor students with a wide range of programming skills, but nearly no visualization background. In this article, we explain different subsequent stages to successfully handle the upcoming problems and describe how much supervision was involved in the development of the visualization project. The project task description is given in a way that it has a minimal number of requirements but can be extended in many directions while most of the decisions are up to the students like programming languages, visualization approaches, or interaction techniques. Finally, we discuss the benefits and drawbacks of both teaching strategies.

Keywords Information visualization · Interaction · Education · Teaching

1 Introduction

Traditional courses at a university can have varying sizes ranging from only a few students up to several hundreds. Teaching those and finding grades for them in the end is more or less a straightforward process. The teacher might give weekly lectures and a final written exam about the content of the lecture results in the final grade for each individual student.

However, a programming course with up to more than a couple of a hundred of students has other requirements than the standard lectures or visualization projects (Dias et al. 2008; Niklas and Ebert 2012; Müller et al. 2012). To achieve a realistic process, the students should work together to find a common

M. Burch (✉) · E. Melby
Eindhoven University of Technology, Eindhoven, The Netherlands
E-mail: m.burch@tue.nl

solution to the programming problem which is a real-life scenario in software development (Hilburn and Humphrey 2002). This working together in a team (Domik 2009) means weekly collaborations between the students while properly distributing the work among the individuals. Finally, merging the individual solutions into a common end solution can become a tedious and challenging task that has to be managed by the students themselves, but extra surveillance and support is also important from the perspective of the teacher and supervisors.

There are several major steps to be aware of when such projects should lead to successful results with growing numbers of students in the future. Building the groups of students, supervising them, finding the right project, keeping them working at a high engagement level, or doing the evaluation and grading during and after the course are only a few major aspects that have to be considered in such large-size programming and visualization lectures. Moreover, other organizational challenges have to be managed like assigning tutors and supervisors, fixing and reminding them of deadlines, answering emails quickly, participating in discussions, or finding rooms and time slots for the individual groups.

In this article, we describe two different teaching and educational strategies focusing on information visualization topics (Ware 2004), instead of just one teaching strategy as in the original paper (Burch and Melby 2019). Both strategies were successfully applied to 6 bachelor courses with a total number of 687 students working in 116 project groups. More than 95% of the students had a computer and data science background, but even a few studied philosophy, psychology, or industrial design. All 6 courses ran for about 10 weeks in the time period of April 23, 2018, to February 9, 2020. Roughly half a year before each course started, basic information about it was announced in the university-internal Osiris system in which students could decide to enroll or not, based on the provided information, i.e., the only knowledge the students had about the programming project was the general topic, the task to be solved, and the grading strategy. The entire development process was then stepwisely given, starting with a kick-off meeting for TS1 (or a first lecture for TS2) to not overwhelm the students right from the beginning. The students had to do self-study as part of an active learning process (Prince 2004), even if weekly lectures were given, while they could contact tutors, supervisors, or the lecturer all the time. The lecturer follows the open door policy, meaning students can visit and ask whenever they wish, but they could also find regularly an updated information in Canvas, the university-internal learning management system.

The major lessons learned from a teacher perspective were that in general both strategies were successful which was not clear right from the beginning due to the freedom that was provided to the students. The major task of the teacher was to find a good balance between this freedom while at the same time keeping them busy, working, and engaged, targeting a common goal of solving the given project task. However, most of the students felt well-supervised and the tasks were clear most of the time (as reported verbally but also in the course evaluations). Also, the collaboration among the group members worked nearly perfectly in all of the groups. Most of the upcoming questions could be handled either on the lowest supervision level for TS1, i.e., by the group-assigned tutors who are second year students or by the feedback from the lecturer in TS2. However, negative points have been figured out like missing tutors' knowledge about information visualization or programming experiences, facts that could either be solved by the students themselves, or by making usage of the internal supervision hierarchy, i.e., asking the supervisors (responsible for the tutors) or even the lecturer.

As a final result, 633 students passed the courses, 101 with the highest grade of 10. They generated many more visualizations and interaction features than they had to build in the course (see Fig. 1 for some nice examples from one of the courses) which is also a hint that they enjoyed the visualization courses.

2 Related work

Teaching students in visualization courses (Domik and Scateni 2009) is a challenging task, in particular, if those are relatively unexperienced bachelor students with a variety of skills and if the number goes into the hundreds. This difficulty is also caused by the variety of existing visualizations, algorithms, methodologies, and interactions (Domik et al. 2012; Domik and Fischer 2011; Owen et al. 2013), but also by the ways to design, build, and implement a functioning visualization system as a result of a programming project in a collaborative manner (Domik 2009). Finding the right balanced strategy to make the course successful for the students but also for the teacher and supervisors while reaching the learning objectives can become a tedious task and requires a lot of prior planning, meetings, and discussions. The major problem lies in the decisions that have to be made beforehand, i.e., without a lot of knowledge about the students who enrolled

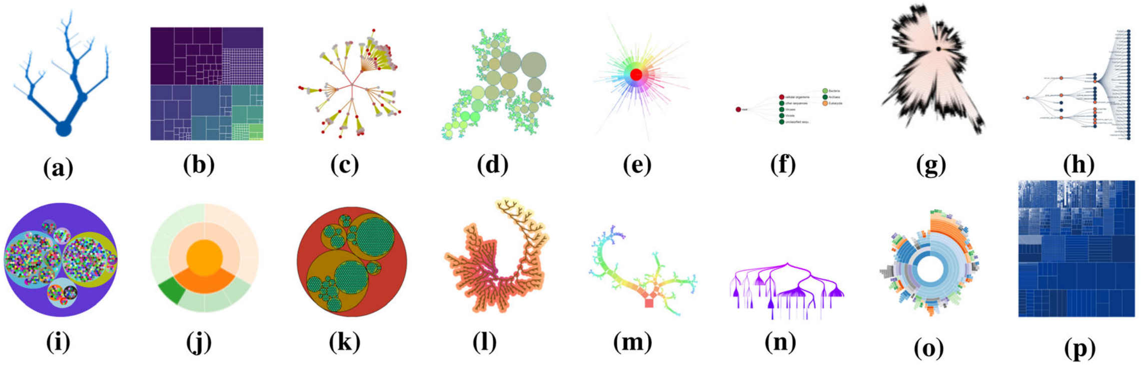

Fig. 1 A small selection of a large repertoire of 125 created hierarchy visualizations, results from one visualization course with teaching strategy TS1 with 272 students focusing on the topic of hierarchy visualization: the students got supervised and guided by weekly scrum sessions and two weekly 30 min meetings with student assistants as tutors. Two visualizations had to be implemented for each group (i.e., 80 in total), but actually, they built many more, showing engagement in the visualization course

in the course. Moreover, if the course only runs over 8–10 weeks of time, it can be hard to let them implement a system from which they do not really know if and how it could meet the aforementioned requirements.

However, we got inspired by the approach by Beyer et al. (2016) since our course setups have similar preconditions; however, we have two teaching strategies, either with weekly lectures and assignments (TS2) or without them but with weekly scrum sessions (TS1). Moreover, we must follow a more compressed time schedule [not like in traditional long-durating visualization courses (Müller et al. 2012) or running over one semester (Domik 2012)]. The students typically have only 8–10 weeks to finish all tasks, while the programming task will actually be started a bit later, meaning only up to 8 weeks remain for that. This additional limitation in form of time pressure demands for a well-designed course setup in terms of teaching skills like programming and visualization background (Derry and Fischer 2005). Similar to the approach by Beyer et al. (2016), we follow an active learning model (Motschnig et al. 2016) that forces the students to do collaborative group work, discuss tasks, split them into subtasks, distribute those among the group members, and merge the results into the final product (Evans et al. 2014). Active learning leads to an improvement in students' thinking and writing as summarized by Bonwell and Eison (1991). Following the guidelines by Prince (2004), our visualization courses focus on collaborative, cooperative, and problem-based learning. However, from a programming and visualization perspective we let the students decide which methods to use while Beyer et al. limited the programming environment to JavaScript and D3.

To reach everybody's weekly goals, the students have to do self-studies, like finding the relevant literature, reading, or watching self-found online tutorials, and reflect on the findings in the weekly group meetings, either scheduling those themselves or being guided by scrum sessions and regular tutor meetings. On the negative side, McKeachie (1972) argued that the improvements of discussion over lectures are small. However, to improve the efficiency and effectiveness of the knowledge gaining process, for one teaching strategy we gave extra scrum sessions (Mahnic and Drnovscek 2014). Those also told them how to do regular stand-up meetings to inform everybody about the results, hence focusing on collaboration, cooperation, and problem solving. The scrumming activities support the students in learning how to function as a team since this can be regarded as a real-life scenario and all can benefit from teaching teamwork (Hilburn and Humphrey 2002), not only the project group members but also the tutors. In teaching strategy TS1, we even went one step further than most of the courses held in the past in a way that we actually do not have an accompanying lecture, but the task is only given once in a 45 min kick-off meeting, making it a real problem-based learning task (Woods et al. 2000). In the teaching strategy TS2, we regularly update the project status by weekly assignments and lectures.

After a kick-off meeting in TS1, the students are left alone for the next coming weeks, in the sense that they have to get active and ask for help whenever required. To give support and have a surveillance tool, each group is assigned a tutor who meets the group twice a week. Since the tutors are typically not well trained in education tasks and have less teaching and technical experience, we used a supervision hierarchy of people to reduce the message flow between the students and the lecturer. In most courses, also in the teaching strategy TS2, a flat supervision hierarchy is chosen like in the approach by Beyer et al. (2016) in which 15 students are assigned one teaching assistant who is available once per week. In our course setups, the teaching assistants (tutors) are available in the group meetings twice per week and each has to supervise between 20 and 30 students in 3–4 groups. However, in TS2 there are no tutors involved, the lecturer is the only supervisor.

3 Teaching strategies

Starting such visualization courses was a bit challenging since the lecturer never gave courses with that many bachelor students before. Since he recently moved from another university in a different country, the educational concepts had to be adapted while also the reduced teaching period (from 16 to 8 weeks) had to be taken into account, finally leading to two new teaching concepts combining lots of novel ideas, some of them already adapted in other courses of similar sizes at the university. The teaching concepts included scrum sessions without further guidance by weekly lectures and assignments (TS1) and weekly guidance by lectures and assignments (TS2).

3.1 Course scope and students

The students were bachelor students from different study fields and departments, with computer science and data science as the major ones. This new aspect in education meant that the courses had to be adapted to the varying skills and experiences of the students, with programming skills in different programming languages among the most difficult aspects. Also, the general topics of the courses which were information visualization and interaction limited the choice of possible special topic candidates in order to achieve successfully running courses. Consequently, we decided to focus on simple and easy-to-understand topics like visualizations of hierarchy, network, eye movement, dynamic graph, publication, and genealogical data, one for each of the 6 courses. Those could be easily explained, motivated, and understood by students from various backgrounds.

Moreover, students could be easily motivated to work in such projects since they belong to well-known fields in information visualization with many illustrative examples and enough literature to do self-studies. Also, the underlying data structures and the data sources are quite easy to understand; hence, students could apply formerly learned concepts from the data structures and algorithms course in order to quickly reach the goal of the programming task. The students were also told that making an interactive visualization available in a web browser can be interesting for experts and non-experts in the world, while the tool is easily accessible by just typing in a URL, but also stand-alone tools are accepted. However, web-based tools do not have the burden of installing extra packages and software for the user, in our case the lecturer who has to evaluate the visualization projects. We also explained that a written report might result in a scientific publication in the field of visualization which is a positive side-effect for their CV.

The teaching styles in these courses can be transferred to other courses as long as a programming project is the goal of the course. However, it has to be adapted to the difficulty level of the project and the time period the course is held. The evaluation of the teaching effects is achieved by the written reports and the recorded presentation videos which showed the lecturer that the topic was understood and the visualization project was successful. Moreover, the source code of the implemented tool together with a tutorial showed that the tool was running and applicable to the given dataset scenario.

Finally, since these courses were taught in this style for the first time and we could not judge if they run properly, we also left some options from the teaching perspective. In case the project task was too difficult, we could easily reduce the amount of implementation work by taking out several subtasks. On the other hand, if the task was too easy and trivial, students could do more. Actually, we did some kind of pilot study with a master student and a PhD student (Sect. 5.1) before starting the courses to estimate the workload (see Fig. 2). However, to pass the courses, the students had to solve the basic tasks, but the freedom in this course allowed them to do much more. In general, the students rated the projects overall positively (see Fig. 3 for two courses of category TS1 and TS2) while they also indicated that it is very relevant for their study program (see Fig. 4 for TS1).

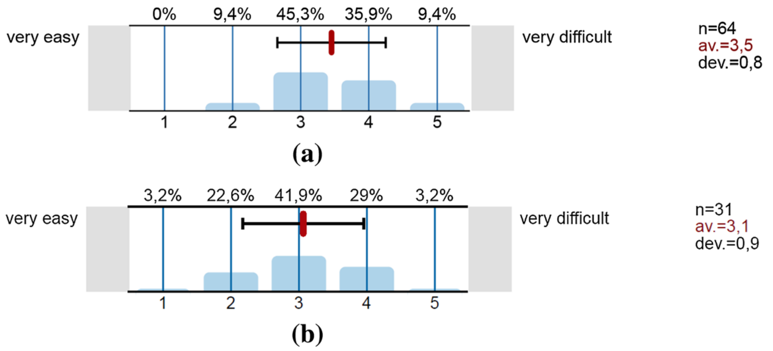

Fig. 2 The level of difficulty in the teaching strategies varies a bit, but the averages are quite similar, i.e., the chosen scenario was neither too easy nor too difficult. **a** The teaching strategy TS1, **b** the teaching strategy TS2

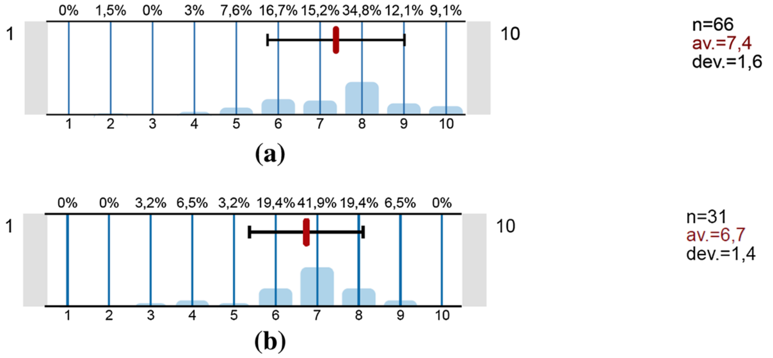

Fig. 3 The rating for the teaching strategies on a scale from 1 to 10 (while 10 is excellent) shows a positive result for both TS1 (**a**) and TS2 (**b**)

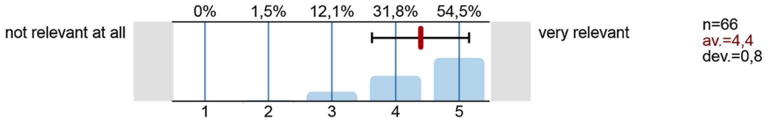

Fig. 4 The relevance of this special project (hierarchy visualization, TS1) for the study program

3.2 Enrolling in the courses

Students could start enrolling in the courses 5 months before the kick-off meeting (TS1) or the first lecture (TS2). They got some information about the courses like a preliminary schedule, ECTS points, the general project description (without many details), and we also told them how they can get their final grade, of which parts it is composed, and how which criteria have to be fulfilled to reach the maximum amount of points in each grading part.

To enroll in the courses, we used the university-internal Osiris information system that supports the students to enroll in their courses. Moreover, it tells them what is required in order to pass a course.

The conversation with the corresponding lecturer is done in Canvas, a learning management system that is used in the academic year in order to guide the teaching process and to handle questions raised by the students. Moreover, Canvas is providing all the details that are needed to successfully take part in the course. These detailed descriptions are given for course materials, project groups, assignments, submission deadlines, grades, frequently asked questions, discussions, and many more. It can be used as an online tool for face-to-face education and to handle upcoming questions as fast as possible, i.e., students and lecturers do not have to wait until lecturing hours to find solutions to problems and challenges.

Finally, 687 students enrolled in the 6 courses, 498 male and 189 female students, typically with a computer science or data science background. Most of those mentioned that they are familiar with programming since they visited an earlier course on programming languages and data structures and algorithms.

3.3 Separation into groups

Building student groups in a programming course is a challenging task. For us, there were three different ways to do that. Students might be grouped randomly, they might group themselves, but they might also be grouped by looking at previously visited programming courses. Each of the groupings has benefits and drawbacks; however, for TS1 we decided to base the grouping on the programming skills and experiences from previously visited programming courses since the project work is based on a programming and implementation task. For TS2, we let the students build the project groups by themselves.

We look into the major aspects that are given by how easy or difficult it is to solve the programming task and what is the maximum learning effect the students can have from such courses. Moreover, collaboration is very important in the courses since this reflects a real-world scenario, for example, when later working in industry in a much larger project, hence not an individual student should do most of the work, but we try to observe the group work as much as possible while letting as much freedom as possible. Based on the students' feedback, we can argue that most of them were satisfied with the group building (see Fig. 5 for TS1).

- *Random groups* People do not know each other, and it can take a quite long time to get familiar with each other. Moreover, planning meetings can be a challenge if students have different courses right from the beginning. Experiencing with everybody's programming skills can lead to the fact that in the first few weeks not much can be done for the actual project. However, the learning rate can be quite steep once they have tackled the effort to get started.
- *Self-built groups* People know each other, and they are quite familiar from the beginning. However, students might take the given programming task not seriously enough and hence, also here a lot of time can be wasted in the first few weeks. A more serious aspect comes from the fact that students worked together in pervious courses and hence, the same ones do the difficult work, hence, the learning rate can be quite low, although they might manage to solve the task finally.
- *Grade-based groups* Putting the strong ones (based on their programming skills) together in project groups as well as the weak ones in other project groups is a fruitful concept. The chance of knowing each other is rather low and the freedom of the visualization project lets the good ones build excellent projects. In most cases, everybody can participate in the programming task of the project. Hence, the programming is pretty easy for them, but the learning rate is increased by themselves since they start searching for even more ideas to implement. For the weaker ones, it is much more difficult to implement the project, but they have to collaborate much more to plan and implement their project work to pass the course. Moreover, the learning rate is pretty high since they have to experiment and train much more than the stronger ones, i.e., those with more programming experience.

In all scenarios, there is a group leader who first collects knowledge about all group members and then finds tasks for everybody that are manageable based on his/her experience levels. In some cases, students must learn new aspects to extend their knowledge or to start a new experience.

3.4 Kick-off meeting and first lecture

The very first step from a lecturing perspective was the kick-off meeting (TS1) or the first lecture (TS2), i.e., all students had to be present to get informed about the project and the task to be solved. Since the number of students was pretty large and there was not a single room available in which all of them fit, the kick-off meetings had been given twice in a row in smaller conference rooms. For TS2 and the first lectures, we did not have this problem because the student numbers were a bit smaller. Each of the kick-off sessions lasted

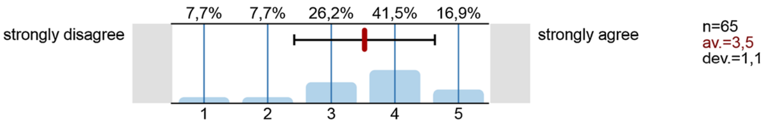

Fig. 5 Satisfaction with the way the groups were established: For TS1, the students are very confident. For TS2, the maximum confidence was nearly 100%

about 45 min and contained most of the information the students needed to start the project. The kick-off meetings also contained an introduction into scrum which lasted another 45 min.

Since not all aspects could be explained in enough detail, we provided a written project guide and a visualization project description in the Canvas system. The students are recommended to read them and to ask questions in case some points are unclear. Moreover, the students were encouraged to ask questions after the kick-off meeting and during or after all of the lectures (not only the first one).

From the 687 students enrolled in the courses, 603 showed up in the first meetings. We split them into groups beforehand and announced that in Canvas in order to inform them about which kick-off meeting they should attend. After the presentation, students were explicitly asked about questions and if they feel too shy to ask they could either visit the lecturer's office, send emails, or start discussions in Canvas. Only a few questions were asked immediately, with clarification questions about the grading and the provided dataset being the major ones. We also announced that there was no lecture for TS1, i.e., students had to do self-study or ask for help. However, for teaching visualization without lectures the students got project-relevant state-of-the-art reports and extra literature to guide them into the right direction. Positively, in the first week after the kick-off meeting only three questions have been asked, indicating that the setup and given tasks seemed to be understandable.

3.5 Scrum

To improve the group collaboration and the distribution of the weekly tasks among the group members, we explicitly made use of scrum (Mahnic and Drnovscek 2014) for TS1. This agile framework for managing software development was considered useful for our courses in TS1 (as also mentioned by the students in the course evaluation, see Sect. 6.4).

Consequently, we also had to teach the students about scrum to successfully apply it over the coming weeks. For this, we invited an expert from a software company with a long experience in scrumming. Extra lectures on scrumming were given over the semester after several development phases for all students, for the supervisors and tutors, as well as for special scrum masters that had to be elected by each of the student groups.

Scrum is in particular a good concept for our project work since it is designed for three to nine group members planning to cut down their tasks into subtasks that have to be finished after a certain time period. These 'sprints' typically do not last longer than 2 weeks followed by asking for project progress, re-planning, and finally, stand-up meetings lasting for only several minutes (Mahnic and Drnovscek 2014). The regular stand-up meetings were useful to inform everybody about the results, hence focusing on collaboration, cooperation, and problem solving. The scrumming activities support the students in learning how to function as a team since this can be regarded as a real-life scenario and all can benefit from teaching teamwork.

The scrum master choices were made by each student project group individually, since the group itself and the elected person know this best in order to efficiently and effectively complete the programming tasks. However, the scrum master training sessions and the actual practice in the project group helped to identify if the scrum master choice was correct or not. If problems occurred, the scrum master could be exchanged by another one.

4 Supervision hierarchy and grading

To better manage the large number of students, we followed a hierarchy-based organization of the project for TS1 (see Fig. 6). The hierarchy consists of the students on the lowest level, the group with each group leader supervised by a tutor on the second lowest level, supervisors above them, and finally, the lecturer (project coordinator) on top serving as the root node of the hierarchy. The setting in the TS1 courses was that we had one project coordinator with 5 supervisors, each of them is responsible for 2–3 tutors, while each of the 11 tutors had to supervise 3–4 student groups.

The general idea of the hierarchical structure is that the flow of information can be better controlled and managed. This means if problems occur on the lowest level, those can be solved in the next higher level, i.e., by the tutor. If this is not possible, the next higher level of people is asked, finally, asking the project coordinator. This structure is also useful to provide new information or announcements from top to bottom, i.e., vice versa, flowing into the lowest level of the hierarchy.

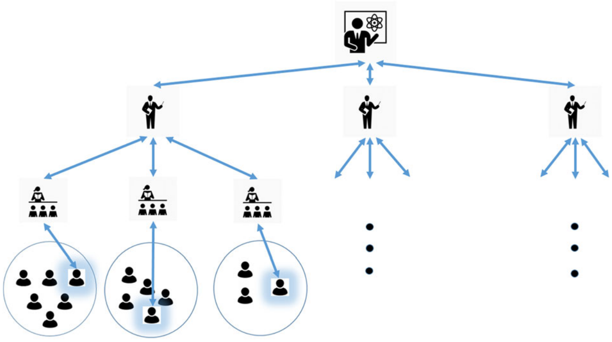

Fig. 6 For TS1, the supervision hierarchy consists of the lecturer as the topmost element, the supervisors, the tutors, and the project group with an elected group leader/scrum master

4.1 Message flow

We told the students that questions via email can only be asked by the scrum master/group leader and only on the lowest hierarchy level first for TS1. For TS2, the lecturer should be contacted. This means if problems occurred they were solved on the lowest level between group leader and tutor, avoiding conversation conflicts if all group members were enabled to ask or for TS2 by the lecturer. However, in the weekly group-tutor meetings everybody could ask questions. If tutors were not able to answer, they could ask the corresponding supervisor and finally, if still no solution is found, the lecturer is asked for help. In rare situations, students could also break free from this hierarchy and directly ask the lecturer for help, like in the TS2 scenario.

The hierarchical organization of this supervision process for TS1 was quite successful since the lecturer was asked only seven questions from students directly over the entire project phase. All of the other several hundred questions could be managed on the lowest supervision level. For TS2, however, hundreds of questions have been asked during the lecturing weeks, either in the lecture, after it, by emails, or by students visiting the lecturer in the office. Consequently, for the lecturer TS2 was more supervision-intensive.

Frequently asked questions were announced in Canvas, but only if the same topic was asked several times, then we decided to make everybody aware of it. This tremendously reduced the number of emails to be sent to each group and vice versa.

4.2 Benefits of supervision

In earlier versions of this visualization project-based course, student groups were left alone with the programming tasks and although they mostly solved the tasks successfully, they also reported difficulties and challenges. This meant, many more questions were raised, increasing the workload for the lecturer. However, these courses only consisted of up to 50 students and 10 project groups, but the courses described here have a larger scale and hence, a better supervision process has to be tried out.

The supervision hierarchy has several benefits, with distributing the teaching and grading workload as the major points. For example, tutors had to visit each student group twice per week for 30 min. In these meetings, students could mention their progress and which challenges and problems they are facing. In this process, the tutor could give feedback since the experience level of the tutors was a bit higher than those of the students. However, tutors were no experts in visualization, nor do they have a lot of programming experience, meaning also the tutors can ask for advice, for example, the supervisors.

The tutors were the most important part in this hierarchy since they build the interface between teachers and students. They got direct input in the development process and where the problems arise. Students felt much safer in TS1 than in TS2 because they had a contact person (see Fig. 7), at least twice per week. For the stronger groups, the tutor did not have to say much, for the weaker ones the tutor was of great help. Another benefit of tutoring was the way we handled the effect of keeping each student busy. To proof that, students had to write minutes for the meetings they made, that had to be handed in to the tutors each week. The tutor corrected them and only reported the supervisors if problems occurred or students were not willing to participate or collaborate in the team work.

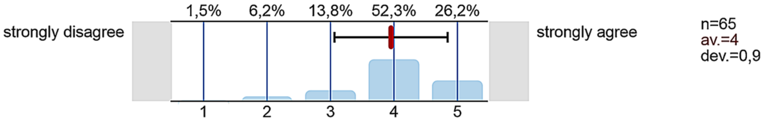

Fig. 7 Satisfaction about the tutor support in TS1

Following all these principles made grading much easier, fairer, and most important much more time-efficient for TS1. However, for TS2 the weekly assignments had to be graded which resulted in much more grading work.

4.3 Final grading

We already announced the grading procedure and how to get the individual grades in Osiris, i.e., when students enrolled in the course. For TS1, those were in detail 10% for the interim report, 45% for the final report, 10% for the video presentation, and 35% for the final project, i.e., the source code and the running interactive visualization tool. For TS2, we split it into four times 25% for the four components: assignments, written reports (interim and final), video presentation, and final project.

The grading of the written reports was split into two parts for both TS1 and TS2. The students had to hand in an interim report half-way of not more than 4 pages. This report was read by the lecturer only and annotated with comments. If it was not understandable or the content was very weak, the grade was reduced accordingly. The final report was twice as long as the interim report and the grade was based on content, reproducability, technical descriptions, story, figures, and clarity of presentation. This report was read and checked by the lecturer, while the grade was not only on the criteria above, but also on the fact if the students revised the interim report based on the lecturer's comments. This process is similar to the reviewing process in academic publications.

The video grading was also straightforward. Each supervisor got the list of videos to watch for TS1. After 1 week, each video had to be graded based on the five criteria like story, content, agility, understandability, and presentation form. For TS2, the lecturer did that by himself.

Also, the final projects could easily be graded since we had the video presentations that showed most of the working features. We could also get an impression about scalability issues or interactivity by such a live demo. All supervisors graded the projects for TS1, for TS2 only the lecturer did that. Finally, all grades were taken into account and last but not least we looked into the peer grading to eventually lower individual grades based on the fact that students did not take part too much in the group work. However, before lowering the grade we tried to confirm this by inviting the student to a final oral exam. However, if we got the impression that a student did not participate at all, we could let him/her fail.

4.4 Staff working hours

The time spent to supervise and grade the students was acceptable which was also caused by our hierarchical organization for TS1. However, for TS2 the working hours have been spent by just one person, the lecturer.

For TS1, the supervisors worked about 40 h each. The tutors worked about 60 h each including the scrum training sessions. The lecturer also worked about 60 h including grading and meeting student groups. Moreover, the scrum expert worked for 10-and-a-half days including developing material and giving presentations.

In total, all involved people worked about 1000 h to get the course running for TS1, i.e., for lecturing, supervising, reading, correcting, video watching, grading, and so on. For TS2, the lecturer worked about 200 h in 10 weeks during the course was running.

5 Student project work

There were two different tasks in the courses, the general programming task and the presentation task that was split into two parts, i.e., the written reports (interim and final) and a video presentation.

The programming task was first given in Osiris, but only the general description leaving out many details, to not overwhelm the students with too much information right from the beginning. Later on, the

Canvas webpage was updated in order to inform the students and allow them for questions and feedback. A very detailed project guide in form of a 10-page pdf file was provided that the students had to read.

The very first personal meeting with the students, the lecturer, the supervisors, and the tutors was useful to give an overview about the project and further information. Here, also the goal of the project was given and which tasks the students had to solve to pass the course from the perspective of the implementation.

Since the courses were on visualization, the students had different backgrounds like computer and data science, also varying programming skills, and no prior knowledge in visualization, it was decided to give them an easy-to-understand topic based on typical application areas like hierarchy, network, eye movement, dynamic graph, publication, or genealogical data.

5.1 Programming task

When first discussing the structure of the courses, we decided to base them on interactive visualizations that might run in a web browser. Our first intention was to let the students build four different visualizations showing the provided dataset that are interactively linked.

However, before announcing the project details we conducted pilot studies recruiting a second year bachelor student and a PhD student who is an expert in information visualization. The PhD student could easily implement the project tasks in one day up to a week, but the bachelor student reported it took several weeks to implement while several difficulties were detected. The major goal of the pilot studies was to show us if the task is doable at all, meaning there is at least one way for the students to succeed. In the worst case, we could get rid of the freedom aspect in the courses again and give the students detailed instructions to reimplement everything based on the results of the pilot studies. However, this would have reduced the learning effect in the course.

In some cases and based on the pilot study results, we decided to limit the number of required visualizations to only two or three, but they still had to be interactively linked, shown next to each other as in multiple coordinated views. In our pilot studies, the project was developed by only one person, however, in the course up to 7 students can collaboratively build the tool by splitting each task into subtasks, hence the main task might be solved much faster.

We recommended Python, JavaScript, D3, and Bokeh as standard ways to get started and provided some online tutorials. However, due to the freedom in the course students could decide about their own technologies and programming languages.

We provided example visualizations (see Fig. 8 for a node-link tree diagram) and gave them literature to get started. Moreover, we made them familiar with famous webpages surveying visualizations from a certain topic, for example the TreeVis.net webpage (Schulz 2011) to find inspirations for their hierarchy visualizations. Since these are information visualization courses, students could be creative when building their own visual solutions.

As an extra feature in the implemented tool, it should be possible to upload other datasets that can be shared with other people in the world, for example, to build visualizations for publications or presentations.

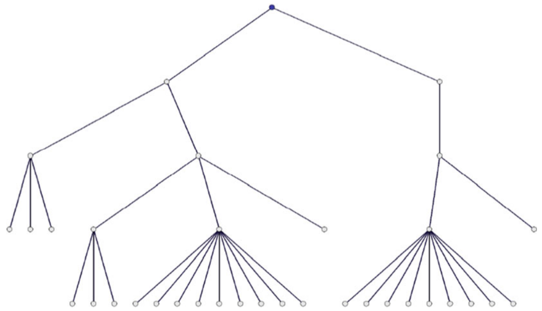

Fig. 8 An example node-link hierarchy visualization to illustrate the students the general concept in one of the visualization courses for TS1. This example was also used in the kick-off meeting to easily demonstrate how a simple hierarchy visualization might look like

5.2 Presentation task

During and after the visualization projects, the students had to write down their results and also present them in a corresponding video presentation of 5–6 min.

The written report was built by using the VGTC LaTeX templates in order to have similar preconditions and grading conditions for all of the reports. The interim report was limited to 3-and-a-half up to 4 pages while only 1 page of figures (total area covered) was allowed. For the final report, 5-and-a-half up to 6 pages of text and 2 pages of figures had to be generated by revising the interim report and by adding extra features, e.g., interaction techniques, a data model, a performance testing, or a user evaluation.

Although not explicitly explained, the students did not complain about working with LaTeX, but looking at the reports they had some trouble with standard problems like hyphenation patterns, references, wrong capitalization, missing figure captions, and the like. However, most of these issues could be clarified during the step from the interim report to the final report when attending the face-to-face discussion phase with the lecturer.

The goal of the written reports is to learn to write academically as early as possible, a task needed all the time while studying. This is in particular important in other courses but also when writing down the bachelor and master thesis or even later after they have left the university for industry or academia.

A second way of presentation was the video that they had to record. We did not tell the students how to produce a video, but everybody owns a mobile phone and can easily produce a video; however, they were allowed to take more advanced equipment for the video recording. The only restrictions were the time limitation of 5–6 min, and all students must be visible for a short time in the beginning as an introduction. Moreover, the grading focuses on agility, content, story, presentation form, and understandability while the video should be designed in a way to advertise the tool to customers.

The change from 20 to 30 min oral presentations to video presentations tremendously reduced the amount of hours watching project presentations. Moreover, supervisors could decide when to watch and grade the videos, i.e., the grading must not be done immediately but teaching assistants could distribute the work over several days, meaning less fatigue effects occur leading to unfair decisions in the end.

5.3 Weekly tasks and other obligations

Apart from the programming and presentation tasks, students had some weekly tasks to solve for TS1 and TS2. Since this course is mainly built on group collaboration, it is most important for students to show up in the regular meetings. To reach this goal for TS1, they had to write down minutes describing exactly what they did and when they participated. For TS2, they had to solve the weekly assignments which were split into personal and group assignments.

The minutes had to be handed in to the tutors to keep the correction burden low. These were taken into account for individual grade modifiers, for example, when team members report on problems with students, for example, not showing up regularly or not working properly. This information also flows into the peer grading.

6 Results

In this section, we reflect on the course results in terms of the visualization techniques with interactions, working hours, final grades, student feedback, problems, and lessons learned in a positive as well as negative way.

6.1 Final meeting

In the mandatory final meeting, that took place 10 weeks after the kick-off meeting or first lecture, about 91% of the students showed up. The lecturer first started with the project description from the kick-off meeting or first lecture. Most of the students raised their hands when the lecturer asked if they had learned how to interpret such topic-specific visualizations which means they learned already a lot from a visualization perspective.

However, the goal of the final meeting was to show the best video presentations and to award the four best ones based on criteria like scalability, aesthetics, presentation, and final project.

6.2 Developed visualizations and interactions

Due to the freedom the students had, we got solutions based on several programming languages. Most of the results were based on Python (82%) and JavaScript (11%) making use of Bokeh and D3, but we also got other solution, for example, in C++.

In general, the students developed many more solutions than they had to implement in all of the courses. For the hierarchy visualization topic, for example, the students had to build 2 hierarchy visualizations in each group, resulting in 80 hierarchy visualizations in total. After 8 weeks of development time they came up with 125 hierarchy visualizations from nearly any category. Nineteen groups developed 2 visualizations, 8 groups 3 visualizations, 6 groups 4 visualizations, 5 groups 5 visualizations, 1 group 6 visualizations, and another group even 8 visualizations. We depicted a list of hierarchy visualizations developed by the students in Fig. 1.

The variety of the number of visualizations showed us that they enjoyed the work. Some students enjoyed the course and even invented new ones, of which we published already two in conference papers (Burch et al. 2019a; van de Wetring et al. 2020) and one more is currently under review. Twenty-two of the visualizations were based on node-link diagrams (Eades 1992), radial and non-radial ones (Burch et al. 2011), 21 on the Sunburst metaphor (Stasko and Zhang 2000), 17 were based on treemaps (Shneiderman 1992), 16 on bubblemaps (Hlawatsch et al. 2014), 9 on Pythagoras trees (Beck et al. 2014; Bosman 2004), 6 on icicle plots (Kruskal and Landwehr 1983), 5 on bubble hierarchies (Grivet et al. 2004), 4 on indented plots (Burch et al. 2010), while also rare concepts like polygonal Sunburst, a galaxy plot, a level overview, or a hexagrid were implemented just to mention a few. Some students even tried 3D hierarchy visualizations (Carrière and Kazman 1995; Kleiberg et al. 2001).

From an interaction perspective (Yi et al. 2007), most of the students integrated typical ones like select, explore, reconfigure, encode, abstract/elaborate, filter, and connect, although we did not explicitly mention they should implement all of them. Moreover, also scalability issues were handled in some projects, i.e., students experimented with larger hierarchy examples and measured the rendering times as well as the responsiveness of the interaction techniques.

6.3 Working hours and final grades

Due to the variety of experiences and skills, students also worked differently long for the project, see Fig. 9. Most of the working hours are close to 140 h which corresponds to the 5 ECTS criterion (1 ECTS considered 28 h), see Fig. 10.

The final grades of the visualization course were quite positive. In our university, we have a grading scheme from 10 (outstanding) to 0 (very poor), i.e., 10 is the best grade and at least a 6 is required to pass the course. 10 and 9 are considered similar to A+ in the US grading scheme. From the 687 students, only 54 failed. A total of 101 students got the highest grade of 10, 180 got a 9, 146 an 8, 113 a 7, and 93 a 6.

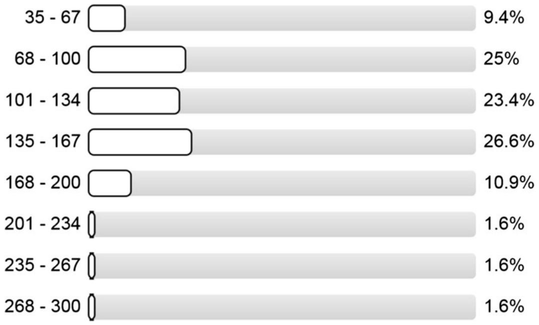

Fig. 9 The working hours of the students vary a lot, an effect that was mainly caused by the different experience levels and programming skills. Note that 1 ECTS corresponds to 28 working hours

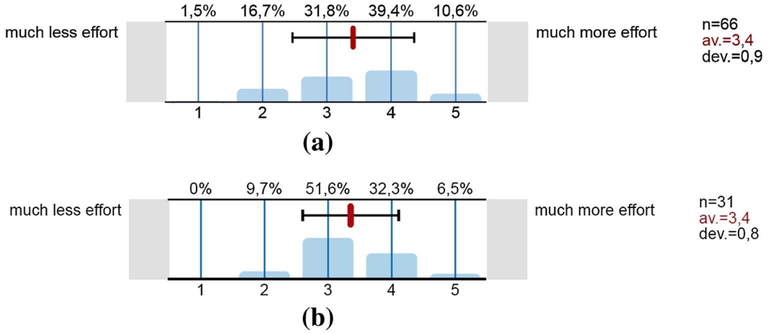

Fig. 10 The effort applied to complete the courses for both teaching strategies corresponds with the number of credit points: **a** TS1, **b** TS2

6.4 Student feedback

We were very thankful for the students to give feedback for our visualization courses. The course evaluations showed some positive remarks, but also negative ones which are more important for us to improve the courses for the future.

Some negative aspects are the difficulty to get started, in particular, if students have not much background in programming, visualization, or web-based development. The students wanted to see more concrete examples to get started in the courses. However, having no programming experience at all and enrolling in programming courses is a tough decision. Another problem was the fact that the chosen programming language in a project group has to be known by each team member. Finding a good consensus is a challenging task right from the beginning. Finally, some group members just decided which programming language to use as the major one by ignoring the experience of the remaining group members. Hence, for some group members the programming tasks become much easier than for others.

However, for the next edition of the course we will limit the number of students to those having attended programming, visualization, and data structures and algorithms courses successfully.

We got more positive feedback than negative one in our visualization course. The students liked the teaching and learning of practical skills and that they could apply formerly learned concepts and programming experiences to a real-world problem. Moreover, they enjoyed the teamwork and the group forming on programming skills while scrum was a fruitful concept in TS1 to get the group work distributed and done. However, they also liked the weekly assignments in TS2 but found the weekly lectures sometimes not needed; hence, we had a quite low number of attendees. Some students mentioned that the hierarchical supervision structure helped them a lot to find answers to their questions on every difficulty level while they felt their problems got solved quickly. The given freedom in the course was appreciated by many students that also left room for creativity which is in particular useful for visualization courses. In general, students described that they learned a lot in the course and how they tried to tackle the open problem step by step, testing new technologies and programming languages.

In general, the students claimed that they gained a lot of new knowledge, see Fig. 11 for an example course in TS1.

6.5 Awards

In the kick-off meeting and first lecture, we announced that the students can get an award for the most aesthetical and most scalable approach as well as for the best project and the best presentation. This was an additional engagement factor in the visualization courses and was well accepted by the students mentioned in the final course evaluation. The 5 supervisors voted for the award winners for TS1 based on predefined

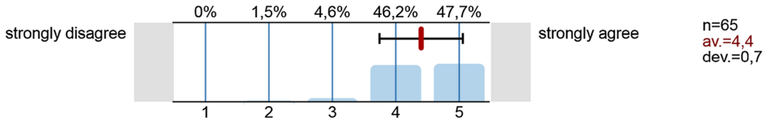

Fig. 11 Knowledge and skills gained from a course in TS1

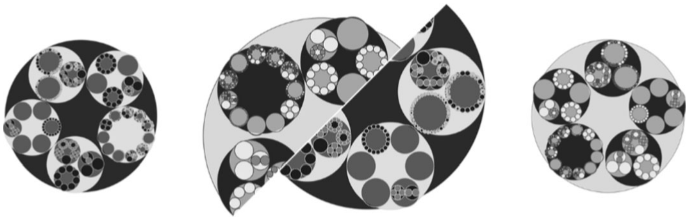

Fig. 12 This figure won the aesthetics award for the hierarchy visualization project

criteria which resulted in clear winners in each award category. For TS2, the lecturer decided about the winners. The award winners were announced in the final meeting after which the corresponding group presentation video was shown. Figure 12 shows the winning figure of the aesthetics award for the hierarchy visualization project.

6.6 Publications

Many visualization conferences provide a way to let students submit their own work in form of a conference submission. We decided to submit the best ideas and concepts to visualization conferences whenever the scope of the conference and the call for papers made this possible. However, before submitting the paper, we improve the writing quality since students do not have that much background in writing scientific papers, but we were pretty much impressed, how much could be taught, in order to keep the workload low when revising the written report into a conference submission.

Following this strategy, we got several papers from the visualization courses published at peer-reviewed well-known visualization conferences and workshops while another paper is currently under review submitted to another conference. Both TS1 and TS2 led to publications, i.e., for TS1 we published (Burch et al. 2019a; Burch and Melby 2019; van de Wetring et al. 2020) and for TS2 we published (Burch et al. 2019b, c; Kumar et al. 2019). Publications are another kind of project evaluation that show that even bachelor students can be successful in paper writing, in case they are supervised well enough in visualization courses.

6.7 Problems

Apart from the many positive points, we also figured out that several problems and challenges occurred that could not be fixed that easily. But however, the negative points were a learning experience for the teacher that must be taken into account for the next edition of this course, even if the number of students will be much higher.

One student group had problems with the programming language, i.e., all of them knew how to program, but in different languages. The tutor could help out and motivate them to implement in Python since this language will be used most often in other courses at the university. Also related to the programming language was the fact that one group decided to switch to another language after a few weeks which meant they lost some time for the implementation; however, this group managed it to pass the course.

Since this visualization course also included web technologies, some student groups had problems with getting the tool runnable on a server. However, most of the computer science students managed this problem on their own by using their own server. They never mentioned they run into problems with that. It turned out that students not from computer science with less programming background asked for providing a server. Finally, only one project group could not get the tool running as an online version due to the fact that one library could not be installed on the university server. Also, this group passed the course since choosing a strange library due to the freedom in the course was not their fault, i.e., in this case we also accepted the local version of the tool, but it still had to run in a web browser.

Although we decided to provide as much freedom as possible in the course, we had one group in which the students complained about the missing rules, restrictions, and guidelines. It took them nearly 4 weeks to adapt to this teaching style, but then they managed the challenges and passed the course. The reason for this problems was that these students were typically given strict rules in courses at their home university, they never had this experience.

Problems that can always arise in each course come from the fact that students get ill from time to time. In our courses, we had to merge groups together, but this happened very early in the project, hence not much had to be changed in the groups, they had to adapt to this new situation. Moreover, one group was managed by only one student, although the other students were physically present in the meetings. However, they contributed, but the group leader had a stubborn attitude. The tutor and even the supervisor tried to mitigate this situation, but it was pretty difficult to solve. For example, they let all students speak during the meetings and point out their ideas and decisions.

7 Lessons learned and recommendations

Teaching visualization courses with various students from varying backgrounds and programming skills can be a great fun, not only because the topic of visualization is a good choice since it can be understood easily and students can be creative. Also, the large number of students gives a teacher the opportunity to experiment with teaching strategies (TS1 and TS2) and to figure out which aspects are liked and which ones disliked.

But to be successful, several rules have to be taken into account. Although we did these courses in these styles for the first time, we did not get that many problems as we expected before. The major reason for that was the freedom for the students, meaning if one way was not working, they tried another one. This is not possible in courses in which everybody has to use the same programming language, the same visualizations, and in general, exactly the same rules and restrictions. On the other hand, this freedom also brought some new challenges, that we tackled with a supervision hierarchy and scrum training sessions for the tutors, supervisors, and the students for TS1 and weekly assignments and lectures for TS2. We could tackle the problems when they arose, meaning we more or less took the students by their hands and led them to the final solution, while we still let them the freedom to do their own decisions and to be creative.

From an engagement perspective, the freedom in the course was the right choice. Also, the active learning worked, even without having an accompanying weekly lecture. Telling the students the task and the project goal, they could split the work into project phases and in these phases/sprints they could distribute the work among the group members. Scrumming helped a lot to reach this goal in TS1. The students collaborated and cooperated while they solved the weekly problems by themselves, only guided by the tutors.

Grouping the strong students and the weak ones from a programming experience perspective was the right decision for TS1, in TS2 we had some problems with group building but that also worked. This was also mentioned in the course evaluation. This grouping procedure in TS1 did not lead to the typical problem of the fact that weak students hide behind the strong ones in a project group. This also meant that the different skill levels in the whole course can be handled easier since students feel more accepted in groups with similar experience levels. We actually split the course into small subcourses of different skill levels while in each group the levels are similar.

The supervision hierarchy was a great success in TS1. Questions and problems could be solved on the lowest supervision level, by the student assistants/tutors. A benefit of the tutor supervision is that each tutor knows the group members quite well after a few weeks, which is not possible if only one teacher has to do this job. Moreover, in cases the tutors were not able to answer the questions, the supervisors on the next hierarchy level could help, if not the lecturer managed the problem and announced it as frequently asked question or problem in Canvas that everybody could read about it. Another benefit of the supervision hierarchy is that the final grading became fair, it was very unlikely that students did not participate in the course while still getting a high grade. Everybody had to work regularly and had to hand in minutes and reports. To make the grading even fairer, we asked for peer grading and checked the students who got bad grades by their team members.

8 Conclusion and future work

We developed an education concept for large visualization courses with many students. We tested two teaching strategies, but we cannot say that one was in favor of the other. Both teaching strategies led to the common goal. However, from the number of hours spent, TS2 was more efficient, but the evaluation results were a bit worse than for TS1, but still on a similar level. The courses were considered successful since the learning objectives have been reached and also the students/teacher liked the course and gave a good teaching evaluation. In particular, the number of created visualizations indicated that the students liked the courses. Only a small percentage of students failed, while the learning effect was immense, for both information visualization, interaction, and even web development, but also for the improvement of the programming skills. Moreover, and most important is the fact that students practiced how to collaborate in a group of people typically not knowing each other before. As a positive side-effect, we even published several of the student projects at well-known visualization and eye tracking conferences. For future work, we will have the same course again in a few months, but this time at a different scale, with 284 students.

References

Beck F, Burch M, Munz T, Di Silvestro L, Weiskopf D (2014) Generalized pythagoras trees for visualizing hierarchies. In: Proceedings of the 5th international IVAPP conference, pp 17–28

Beyer J, Strobelt H, Oppermann M, Deslauriers L, Pfister H (2016) Teaching visualization for large and diverse classes on campus and online. In: Proceedings of IEEE VIS workshop on pedagogy data visualization

Bonwell CC, Eison JA (1991) Active learning: creating excitement in the classroom. In: ASHEERIC Higher Education Report No. 1, George Washington University, Washington, DC

Bosman AE (2004) Pythagoras tree, reprinted in Bruno Ernst: Bomen van Pythagoras—Variaties van Jos de Mey Aramith Uitgevers Amsterdam, 1985

Burch M, Melby E (2019) Teaching and evaluating collaborative group work in large visualization courses. In: Proceedings of the 12th international symposium on visual information communication and interaction, VINCI. ACM, pp 17:1–17:8

Burch M, Raschke M, Weiskopf D (2010) Indented pixel tree plots. In: Proceedings of international symposium on visual computing, pp 338–349

Burch M, Konevtsova N, Heinrich J, Höferlin M, Weiskopf D (2011) Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. IEEE Trans Vis Comput Graph 17(12):2440–2448

Burch M, Aerts W, Bon D, McCarren S, Rothuizen L, Smet O, Wöltgens D (2019a) Combining interactive hierarchy visualizations in a web-based application. In: Proceedings of the 14th international joint conference on computer vision, imaging and computer graphics theory and applications, VISIGRAPP, pp 191–198

Burch M, Kumar A, Mueller K, Kervezee T, Nuijten W, Oostenbach R, Peeters L, Smit G (2019b) Finding the outliers in scanpath data. In: Krejtz K, Sharif B (eds) Proceedings of the 11th ACM symposium on eye tracking research and applications, ETRA. ACM, pp 83:1–83:5

Burch M, Kumar A, Timmermans N (2019c) An interactive web-based visual analytics tool for detecting strategic eye movement patterns. In: Krejtz K, Sharif B (eds) Proceedings of the 11th ACM symposium on eye tracking research and applications, ETRA. ACM, pp 93:1–93:5

Carrière J, Kazman R (1995) Research report: interacting with huge hierarchies—beyond cone trees. In: Proceedings of information visualization, pp 74–81

Derry SJ, Fischer G (2005) Toward a model and theory for transdisciplinary graduate education. In: Proceedings of American Educational Research Association's annual meeting and symposium

Dias P, Madeira J, Santos BS (2008) Education: Teaching 3D modelling and visualization using VTK. Comput Graph 32(3):363–370

Domik G (2009) Who is on my team: building strong teams in interdisciplinary visualization courses. In: Proceedings of international conference on computer graphics and interactive techniques, SIGGRAPH ASIA

Domik G (2012) Fostering collaboration and self-motivated learning: best practices in a one-semester visualization course. IEEE Comput Graph Appl 32(1):87–91

Domik G, Fischer G (2011) Transdisciplinary collaboration and lifelong learning: fostering and supporting new learning opportunities. In: Rainbow of computer science—dedicated to Hermann Maurer on the occasion of His 70th birthday, pp 129–143

Domik G, Scateni R (2009) Education programme at Eurographics 2009. Comput Graph Forum 28(6):1723–1724

Domik G, Ebert DS, Kohlhammer J, Rushmeier HE, Santos BS, Weiskopf D (2012) Visualization curriculum panel—or the changes we have made to our visualization courses over the last 10 years. In: Eurographics 2012—education papers, pp 1–2

Eades P (1992) Drawing free trees. Bull Inst Comb Appl 5:10–36

Evans LPG, Adams NM, Anagnostopoulos C (2014) When does active learning work? CoRR arXiv:1408.1319 (2014)

Grivet S, Auber D, Domenger J-P, Melançon G (2004) Bubble tree drawing algorithm. In: Proceedings of international conference on computer vision and graphics, pp 633–641

Hilburn TB, Humphrey WS (2002) Teaching teamwork. IEEE Softw 19(5):72–77

Hlawatsch M, Burch M, Weiskopf D (2014) Bubble hierarchies. In: Proceedings of the workshop on computational aesthetics, pp 77–80

Kleiberg E, van de Wetering H, van Wijk JJ (2001) Botanical visualization of huge hierarchies. In: Proceedings of information visualization, pp 87–94

Kruskal J, Landwehr J (1983) Icicle plots: better displays for hierarchical clustering. Am Stat 37(2):162–168

Kumar A, Timmermans N, Burch M, Mueller K (2019) Clustered eye movement similarity matrices. In: Krejtz K, Sharif B (eds) Proceedings of the 11th ACM symposium on eye tracking research and applications, ETRA. ACM, pp 82:1–82:9

Mahnic V, Drnovscek S (2014) Agile project management with scrum. Microsoft Press, Redmond

McKeachie W (1972) Research on college teaching. Educ Perspect 11(2):3–10

Motschnig R, Sedlmair M, Schröder S, Möller T (2016) A team-approach to putting learner-centered principles to practice in a large course on human–computer interaction. In: Proceedings of 2016 IEEE frontiers in education conference, FIE, pp 1–9

Müller C, Reina G, Burch M, Weiskopf D (2012) Large-scale visualization projects for teaching software engineering. IEEE Comput Graph Appl 32(4):14–19

Niklas E, Ebert DS (2012) Leveraging multidisciplinarity in a visual analytics graduate course. IEEE Comput Graph Appl 32(3):84–87

Owen S, Domik G, Ebert DS, Kohlhammer J, Rushmeier HE, Santos BS, Weiskopf D (2013) How visualization courses have changed over the past 10 years. IEEE Comput Graph Appl 33(4):14–19

Prince M (2004) Does active learning work? A review of the research. J Eng Educ 93(3):223–231

Schulz H-J (2011) Treevis.net: a tree visualization reference. IEEE Comput Graph Appl 31(6):11–15

Shneiderman B (1992) Tree visualization with tree-maps: 2-D space-filling approach. ACM Trans Graph 11(1):92–99

Stasko JT, Zhang E (2000) Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In: Proceedings of the IEEE symposium on information visualization, pp 57–65

van de Wetring H, Klaasen N, Burch M (2020) Space-reclaiming icicle plots. In: Proceedings of the pacific visualization symposium. **(Accepted, but still not published)**

Ware C (2004) Information visualization: perception for design (interactive technologies), 2nd edn. Morgan Kaufmann, Burlington

Woods D, Felder R, Rugarcia A, Stice J (2000) The future of engineering education. III. Developing critical skills. Chem Eng Educ 34(2):108–117

Yi JS, ah Kang Y, Stasko JT, Jacko JA (2007) Toward a deeper understanding of the role of interaction in information visualization. IEEE Trans Vis Comput Graph 13(6):1224–1231