Teaching Networks to Digital Humanists

Augusto Ciuffoletti

***Abstract*—(Contribution) A technical course in a multidisciplinary university program has to provide high-level skills, with limited lecturing hours and student background. This paper investigates the principles for its design and reports about a study case. (Background) The overall course organization needs to address specific learning targets and teaching techniques, different from those used in traditional courses on the topic. (Research question) A stepwise strategy assists a principled design that allows dynamic, long-term improvements. (Methodology) The evaluation of its applicability requires a years-long record of historical data. The paper studies the evolution of a course over six years using simple monitoring techniques: surveys and rubric-based examinations. (Findings) Monitoring emerges as an essential feature for course evolution: a focused examination provides the best results, while institutional, wide-spectrum surveys appear to be of little help.**

***Index Terms*—Course design; Higher education; Computer Networking; Virtual laboratory; Computational thinking; Constructivism; Rubrics; Course Monitoring**

I. Introduction

Computer-based services are pervasive in our society, and there is a diffused feeling that they should be more respectful of human culture, sensibility, and aesthetics. Teaching institutions are responding to this demand by opening schools that form specialists that will participate in the development of future services. The programs include courses featuring both technical and humanistic disciplines, each giving the student a holistic, not overly detailed, understanding of a topic. Which helps, for instance, to understand the Web influence outside computer science [10]

Digital Humanities is the term for such schools, somewhat extending the original meaning of that term [20]. The idea of such a school inspired projects like *Epoch* [15], which arose a still-growing wave of interest. Its implementation is difficult since several disciplines cohabit in one program so that the time dedicated to each of them is a fraction of that available in a specialized program with a similar study load. Teachers have to select course topics and adopt a steep teaching path while keeping reasonable the effort required from the students.

This paper considers the case of a course in computer networking, a cornerstone in today's information technology. To gain a coherent understanding of this topic, the student has to build some proficiency in five network abstraction layers, from Ethernet to Web Services. A thematic program in information technology dedicates two or three courses to this purpose; on the contrary, a multi-disciplinary school has to reach a comparable result with one. The teacher in charge has to trade-off detail and completeness, but the final achievement from the student perspective has to be a coherent and holistic view of the Internet. The teaching path climbs rapidly across lower networking layers to reach the *Service* one.

Augusto Ciuffoletti is with the Department of Computer Science of the University of Pisa, Pisa, Italy. email: augusto.ciuffoletti@unipi.it

TABLE I: A steep learning path supported by lab activities based on Fuller taxonomy [13]

Producing \ Interpreting	Remember	Understand	Analyse	Evaluate
Create			•	•
Apply		•	•	
none	•	•		

Bloom's taxonomy [7], [3] helps to give a more formal statement of this: the course aims at higher competence levels (analysis, synthesis, evaluation) at expenses of the lower (knowledge, comprehension, application). In a computer-science specific taxonomy [13], the learning path is steep, stepping up to the next stage as soon as the student is ready (see Table I).

Such a course needs to be re-designed from scratch since experiences from traditional curricula are of little help. An elementary methodology mediated from software engineering may provide useful suggestions about how to proceed: a) define the requirements; b) design the solution; c) inject runtime monitoring. Each step corresponds to a fundamental aspect of the course: respectively, the syllabus, the teaching materials, and course assessment [11]. This latter has a role in the progressive refinement of course design, a process that may take years.

This paper investigates the principles that may guide the design of a computer networking course in a *Digital Humanities* school. A case study reporting a six-year-long experience helps to understand their potential and to devise directions for future research.

II. Related works

In a cover story on the *Interactions* journal, Bardzell et al. [5] explain the two-way relationship between computer science and humanities. They analyze the specific case of human-computer interactions, remarking how computer science *reaches out* the humans' ways of life, while humanities *reach in* the walled garden of Computer Science (CS). The reach out of CS is evident, and the interaction with computer-based utilities is more and more frequent. Instead, the humanities *reach in* is point-wise and less pervasive, as if the walls around the CS garden oppose resistance. In such cases, humanities contribute to computer science, as in the case of human language analysis brought by computational linguistics [25], or in the psychological investigations on human behaviors and capabilities that contributed to Artificial

Intelligence (AI) research aiming at their reproduction to make more friendly the interaction [12]. Historical investigations at the core of the Digital History (DH) program [26] are an application ground for Geographic Information Systems (GIS) [31], [16]. Social sciences find a crucial application in the Internet village, to control the creation and spread of opinions and beliefs [10]. In the preservation of the cultural heritage, which is the mission of DHs, computer science programs should be systematically combined with courses related to social sciences to make the students aware of how they will be responsible for the evolution of our society.

Regarding good-practices in designing a course, major educational institutions dedicate a Web page to this topic: they picture course design as a difficult task, with consequences on reputation extending for years and hitting promising students [11]. Such recommendations usually indicate a holistic approach to the course, conceptually similar to the design steps stated in section I. For instance, the *Center for Teaching and Learning* of the Washington University in St. Louis has a Web page entitled *Designing a course* describing a process in four steps: define course goals, determine course content, develop teaching methods and tools, plan assignments and exam, and finally refine course design. The loop-wise figure that concludes the guide is even more adherent to the three-steps methodology explained above. The *Understanding by Design* framework [21], follows a slightly different approach, placing as the second step the learning assessment in a reversed progression.

On the side of the teachers, several articles and documents witness a dedication to improving course design. Besides an extensive collection of specific syllabi, there are also research articles regarding aspects of course implementation, like the teaching method or the laboratory organization.

The literature counts several examples of syllabi for courses on computer networking technologies, a primary discipline in a computer science curriculum. Recurrent schemas frequently refer to one of two distinct teaching approaches, both related to the layered architecture of the Internet. One proceeds top-down from the application layer, descending the stack, while the other goes in the reverse bottom-up direction.

A textbook by Kurose and Ross [18] gives an example of the top-down approach and promises a smooth learning curve, and is also suitable for courses on similar topics [32]. A well-known textbook by A. Tanenbaum [29] adopts the reverse progression. Technical seminars aiming at network administrators, like those organized by the Cisco Networking Academy, use such an approach. Koo and Kwong [17] carried out a comparative analysis inferring that students inclined to technical aspects prefer a bottom-up approach, while those disposed to programming favor the top-down one.

Schools in humanities seldom offer computer networking courses with a comparable coverage of the topic. One of them is the Rutger's School in Arts and Sciences, with an *Introduction to Computer Networks* — discontinued in 2019 — presenting a well-defined syllabus. In many other cases, like the *Introduction to Digital Humanities* course of George Mason University, computer networking soft-skills are considered sufficient to approach complex topics, like crowd-sourcing and social media. Several such self-contained digital humanities courses are in the list available in the Academic Commons created by the City University of New York [1]. An example of a more structured proposal for a networking course integrated into a multi-disciplinary curriculum is the *Internet Technologies* course included in the draft syllabus of a Digital Humanities course proposed by the EU project *Epoch*.

Switching to the learning tools, the provision of a sandbox for practicing abstract concepts is crucial for students' active learning. Such tools play similar roles in computer network and programming courses, allowing the students to refine a personal model of a concept by way of experimental results, according to a constructionist approach [6], [30]. In a recent doctoral thesis, S. Mvalo investigates the usefulness of simulation tools in the acquisition of Computational Thinking (CT), specifically in the field of computer networking. In this context, a simulator allows the student to arrange network components on a graphical interface and observe their simulated operation. The author points out how similar studies exist for the teaching of programming languages, but not for computer networking. Yet, the two topics share several concepts — like abstraction, decomposition, and generalization — and aim at similar skills, like algorithmic thinking and problem-solving [23]. In his investigation, the author follows a methodology mostly based on qualitative methods applied to students' and teachers' behaviors, analyzing the outcome of a problem-solving task and reflective reports. The conclusions are that the use of a simulator effectively improves the acquisition of CT concepts and skills also in the case of computer networking.

Simulators, like those studied by Mvalo, are one option for carrying out hands-on activities about networking, but the literature provides other alternatives. One consists of a real networking lab with real network devices that the students can use as network administrators [28]. This alternative has cost and maintenance issues [19] and appears to be less effective than other [4].

A third option is in the middle ground between the two above and consists of a virtual network of virtual machines running on a single PC [14], [35] possibly sided by remote networking hardware [8], [34]. A comparative study between network simulators and virtual environments [19] proves that simulators are preferable when students need practicing with link-level devices, like Ethernet switches, as required to train network administrators. However, network simulators are not appropriate for exercises over the transport layer.

Therefore, the approaches to the design of lab tools reflect the *top-down* vs. *bottom-up* debate that opens this section and considers two kinds of course directions: respectively, one for network programmers, another for network administrators. Such a dichotomy is not easy to solve for a course forming the Internet awareness of a digital humanist, and on-field experience on that subject is scarce given the small, although growing, number of instances. This paper suggests a reasoned way to address the issue with the results of a year's long experience.

III. A COURSE TO UNDERSTAND THE INTERNET

This section describes the design of the course following the three-steps methodology defined in Section I.

A. Analyze the requirements

A DH program needs a course about networking that brings the students to understand the Internet and, for the reasons explained in the introduction, to acquire the fundamental skills to develop Web applications. The course is mostly self-contained, and other classes may extend related skills.

During previous courses, the students developed programming abilities and have generic soft-skills in networking, including Web browsing, Web apps utilization, and basic host configuration. Such initial competence is the foundation for the *fast learning* track detailed in the following section.

The Epoch project [15] defines a syllabus for a program in digital humanities with a course named *Internet technologies: HTML, Java, etc.* allocated in the second semester of the first year of the master course. This indication confirms that networking at the application level is considered relevant and better placed near the end of the program. However, the Epoch syllabus overlooks the importance of understanding the lower network layers to build a satisfactory model for the Internet. Such an understanding is needed, for instance, to evaluate the effectiveness of a security measure. Instead, the syllabus should assist the students in gaining a complete model for the Internet.

Once the framework is defined, the taxonomy in [13] helps to describe the course progression (see Table I) without going into syllabus details but concentrating on the target learning path.

Using the CT primary concepts [33], the following is a coarse grain definition of the learning target of the course:

- **abstraction**: how a layered architecture contributes to the implementation and the maintenance of the Internet;
- **decomposition**: at each layer, which are the components of the Internet infrastructure;
- **algorithms**: the role of standard and layer-specific communication protocols
- **data representation**: payload encapsulation, packet formats.

B. Define the course realization

The learning process speeds up when leveraging the student's preexisting competence: Internet soft-skills and the CT of a beginning programmer.

Their experience as Web users sets the starting point of their learning path at the application layer, thus suggesting a top-down learning path. This indication, however, is questionable since the design of higher layers protocols intrinsically depends on lower ones. For instance, the structure of an HTTP session is motivated by TCP features.

Instead of concentrating on the dilemma, consider that the first valuable learning result for the student is understanding the internal layering of the Internet, *per se* a challenging performance in abstraction; the CT attitude matured during a previous programming course facilitates the student. In this phase, the teacher introduces layer-independent concepts and terminology. A top-down description of the Internet, starting from the student's experience, takes a few lecture hours and prepares to build a well-funded model.

So, the course starts concentrating on understanding Internet layering: they start from the application layer with which the student is already acquainted and proceed down to the link one. The teacher uses the everyday experience and metaphors to introduce features all layers have in common and their role in the Internet stack. At the end of such a preliminary step, the student knows that Internet communication uses link-layer frames encapsulated in packets hopping from one router to the other to reach a final destination. Realizing this model is the first horizontal step in Figure I: there is no practical skill involved in what the student has learned up to this point, but several abstract concepts and terms are ready for use.

The next step in the learning path is up, towards the *Apply* row in Table I. Using a traffic analyzer, the student inspects the packet structure, observing the stack of protocol headers that materialize Internet layers. In that way, the protocol's operation is transparent and verifiable. The teacher provides evidence of abstract descriptions, and the student matches understanding with experience.

After this point, the path proceeds from the link-layer up, with lectures associated with fitting lab activities. The student accumulates knowledge through frontal lectures supported by active learning and gradually acquires an analytic model of the complex structure of the Internet. Its layout depends on the layer, and each of them has characteristic functions and components. In Table I, the student performs a right hop towards the *Analysis* column. The syllabus focuses on fundamental protocols (like TCP) together with other of which the students may have already heard: for instance, NAT, DHCP, and BitTorrent. Table II gives an example of such a progression.

The step towards the *Create* row in Table I is gradual and initially guided by detailed instructions. The student practices by generating the traffic to analyze, creating sockets, and adding resources to a Web server. Such activities follow front lectures that explain the concepts, while the lab activities allow the student to improve a personal model. The amount of time dedicated to lecturing depends mostly on the teaching style and may need to fit a rigid schedule, while the length of the lab activity depends on the student's learning style. In principle, the learner should have enough time to obtain a reasoned result. This point is relevant for the practical organization of lab activities described in the next section.

Lab activities foster the development of the evaluation capabilities in the last column of Table I. They need to focus on application layer skills, assuming the student now owns a satisfactory model of the Internet. Working on a simple dynamic Web application allows the student to evaluate different design options, understanding, or even anticipating the results.

The previous discussion stresses the role played by the sandbox to facilitate learning: the students and the teacher share the same contents and modalities when practicing active

TABLE II: Course syllabus

week	topic	comment
1	Overview of the Internet	Introducing abstraction layers
2	IP layer	Routing excluded
3	Link Layer	Ethernet and WiFi
4	UDP and DHCP	A protocol and its application
5,6	BGP, RIP and OSPF	the routing protocols
7	DNS	
8	TCP	
9	NAT and peer-to-peer	the internals of well-known tools
10	HTTP	towards the network of Services
11	Web Frameworks	building Web Services
12	Cloud computing	hosting Web Services

learning. Here sharing means allowing the students to reproduce — either during or after the lecture — the same activities demonstrated by the teacher, thus promoting a personal understanding.

To this end, the sandbox, or laboratory, exhibits the following features [9]:

a) to be usable outside lecture hours,
b) to provide the students with a realistic experience,
c) to be financially sustainable,
d) to require limited maintenance.

The option of a dedicated networking laboratory does not fit requirements c) and d) being both expensive and difficult to maintain, and also for security reasons related to point a). A simple NATted LAN is sufficient for the planned activities since router configuration and the management of link-layer devices are not in the syllabus, in contrast with Nabhen et al. [24]. For this reason, the use of conventional hardware is preferable [22]. Letting the students install the sandbox on a personal computer is optimal for requirement a). However, to allow the required experience sharing, the sandbox design needs to take into account that, despite student's devices are very heterogeneous, the lab practice needs to be comparable.

Condition b) prevents using the *localhost* loop-back interface to attain this result, being a source of artifacts that are confusing for the student. Network simulators are not of interest since they are more oriented to practicing link-layer techniques [23] so that the final option is a cross-platform solution based on virtualization.

The VirtualBox tool fits the requirements: this hypervisor is cross-platform, open-source, well-documented, and exhibits an acceptable performance also on modest hardware. The technical staff (or the teacher) builds Virtual Machines (VMs) images that the student imports in VirtualBox, while each student installs the software and VMs on the owned PC. Long term sustainability depends on periodic VM upgrades.

C. Introduce the monitoring tools

Monitoring is useful to fine-tune, year after year, the implementation of the course, and requires the availability of a series of homogeneous records to spot the aspects in need of refinement and to understand how to improve them. The way to collect them needs to be invariant and produce coherent data to avoid distortions in the representation of course evolution.

The results of students' final examinations are a handy source of data about course effectiveness. To this end, the exam must provide insight into the learning process of the student — especially when the result is not much satisfactory — since the teacher wants to understand which course aspect raised learning problems. Thus, the examination process needs to discriminate among course features in an objective way.

Surveys provide another insight into the student's experience, but they do not feel committed to providing a qualified performance, as in the case of final examinations. Besides, the teacher needs to take into account the degree of involvement of the student in test results, which may produce a sort of *conflict of interest* that may make useless the survey. Another caveat is the degree of participation: a limited sample prevents statistical validity, but the teacher can nonetheless infer an unresolved lack of interest.

Laboratory activity evaluation and continuous assessment are other sources of information that may become extremely expensive in terms of teaching resources. When the class includes tens of students, continuous assessment requires a team of qualified assistants, which may not be available. Instead, the informal evaluation of lab activities by samples or on-demand requires the teacher a minor responsibility and workload.

IV. On field experience

This section supports the previous analysis by reporting about a course (*Telematica*, now renamed as *Protocolli e Servizi di Rete*) designed according to the principles discussed above. The classes take place during the second semester of the third (and last) year of the *Digital Humanities* program of the University of Pisa. The teacher in charge of the course has a background in teaching courses related to networking for the school of Computer Science at the same institute.

The report follows the design steps announced in section I: syllabus, tools, and monitoring tools. A discussion summarizing the benefits derived from adopting the proposed principles concludes the section.

The syllabus underwent a significant revision after the first year. The initial organization included many subjects with coarse detail. At the end of the year, the survey listed all topics asking the student to designate which of them was easier to understand. The syllabus of the following year took into account such suggestions by privileging the indicated subjects but describing them in depth. Later on, a minor tweak introduced new service layer concepts — namely, the labs on cloud platforms and the Flask Web framework — at expenses of Real-time Transport Protocol (RTP) and Session Initiation Protocol (SIP): its current form is in Table II.

The sandbox used for lab activities uses VirtualBox and consists of a network of two VMs. An exhaustive description is in a software repository (https://github.com/AugustoCiuffoletti/labreti.git). There, the interested reader finds technical details together with the scripts to create the virtual network and lab materials.

Three monitoring tools helped to manage the course, including syllabus refinement. The rest of this section evaluates the effectiveness of each of them.

The first one is an official survey promoted by the University asking each student that enrolls for the final examination of

TABLE III: Results from the official survey since 14/15

Question/Year	14/5	15/6	16/7	17/8	18/9
n. of responses	28	33	46	45	53
B-02 (interest)	3.2	3.3	3.2	3.1	3.2
B-06 (activities)	3.2	3.6	3.6	3.2	3.3
BS-02 (overall)	3.4	3.4	3.3	3.2	3.0
reference	3.3	3.2	3.2	3.2	3.2

a course to give a score to sixteen of its features. Although not compulsory, participation is strongly encouraged since the year 2014/15.

The survey content is the same from year to year: so that the results respond to the homogeneity requirement discussed above. However, they are the same for all University courses and therefore necessarily generic, and they concern the quality of teaching — like teacher's timeliness and availability — with a limited interest in the learning experience. Table III shows the three of them that may be relevant for course organization: the "interest raised by the content of the course" (row B-06), the "usefulness of additional activities" (row B-08), and a generic quality indicator (row BS-02). The reference row is the average of all courses and features. The scores are in the range $[0, 4]$.

According to these data, the course is overall slightly above the reference for all considered features. However, the teacher has no clues about possible improvements: such data are useful for school management, not for the teacher.

The second measurement derives from a course-specific survey managed by the teacher addressing features considered critical. Participation is not compulsory, so the number of data points is sometimes insufficient for a valid sample. Useful results are available only for the years 2013/14, 2014/15, and 2018/19, here studying only the parts related to the learning experience (Table IV) and the lab utilization (Table V).

The former asks to indicate the mood after participating in a lecture using six informal terms: sleepy, curious, tired, satisfied, puzzled, indignant. To avoid a grading impression, they appear in that random order on the form. The intent is to capture several dimensions of student participation in the lecture:

- *tired* means that the quantity of content conveyed is relevant,
- *puzzled/sleepy* suggest problems in content coordination at different degrees,
- *satisfied/indignant* reflect if the lecture meets student expectation,
- *curious* indicates a rising interest.

The survey is delivered by email and filled offline: the intent is to filter out extemporary feelings related to a specific lecture or incidental. Table IV) summarizes the results. The prevalent sentiment is curiosity, which nicely meets course intent. Tiredness significantly grows from the first year to the second, reflecting the program variation explained above. A limited number of students feel puzzled and even less are sleepy: the connection between the topics is clear, and the students mostly follow the teacher. They are satisfied with their learning, and they do not feel deceived in their expectations.

TABLE IV: Feeling at the end of a lecture from the unofficial survey

Category/Year	13/14	14/15	18/19
n. of responses	37	24	32
Tired	13.7%	27.6%	31.5%
Curious	49.1%	34.5%	37.0%
Sleepy	2.0%	6.9%	3.7%
Satisfied	21.6%	17.2%	14.8%
Puzzled	13.7%	10.3%	13.0%
Indignant	0.0%	3.5%	0.0%

TABLE V: Virtual laboratory utilization

Category/Year	13/14	14/15	18/19
n. of responses	37	24	32
Never	21.6%	20.8%	3.1%
Sometimes	40.5%	50%	50%
Often	35.1%	25%	34.4%
Systematic	2.7%	4.2%	12.5%

Compared with the results of the institutional survey, the teacher has much more information about the learning experience, enough to plan a syllabus or teaching style improvement, as occurred at the end of the first year.

Regarding the utilization of the virtual lab, Table V shows a favorable trend of the lab utilization: the part of students that do not use the laboratory decreases from 20% to 3%. The 50% of moderate users are possibly those that practice only for preparing the exercise for the final examination, as detailed below. Systematic users rise from 4% to 12%, which is still low. In summary, the trend is encouraging, but success is still ahead. However, the interest in lab activities is beyond any doubt. In the last survey, to a direct question (not shown in table), 90% of the students tagged practical exercises as primary and preferably carried out during lectures, not as homework.

The third data source is the outcome of final examinations. Section III-C explains why this process should discriminate against the accomplishment of specific educational targets and provide comparable results from different examination sessions. The definition of precise guidelines for the trial helps to meet such requirements.

In the case under study, the final examination is oral and involves three steps. Each of them assesses a distinct kind of ability: conceptual, operational, and practical. In CT terms, the conceptual one is related to abstraction and decomposition, while the operational one corresponds to algorithms and data representation. The practical one matches a generic problem-solving capability. The resulting rubric [27], [2] is simple and easy to follow since each step centers on one assessment criteria. The students know about this process from the course Web page dedicated to exam preparation.

The first step (Q1) consists of describing a course topic — e.g., exterior routing — indicated by the teacher. Terminology and coherence are the evaluation parameters, and a negative performance determines a reject. The second step (Q2) requires the description of a fine-grain aspect — e.g., TCP sliding window — and the failure does not preclude passing the test. The third one (Q3) is the live demonstration of a lab activity selected by the student; the evaluator scores comprehension and problem-solving capacity. The student has

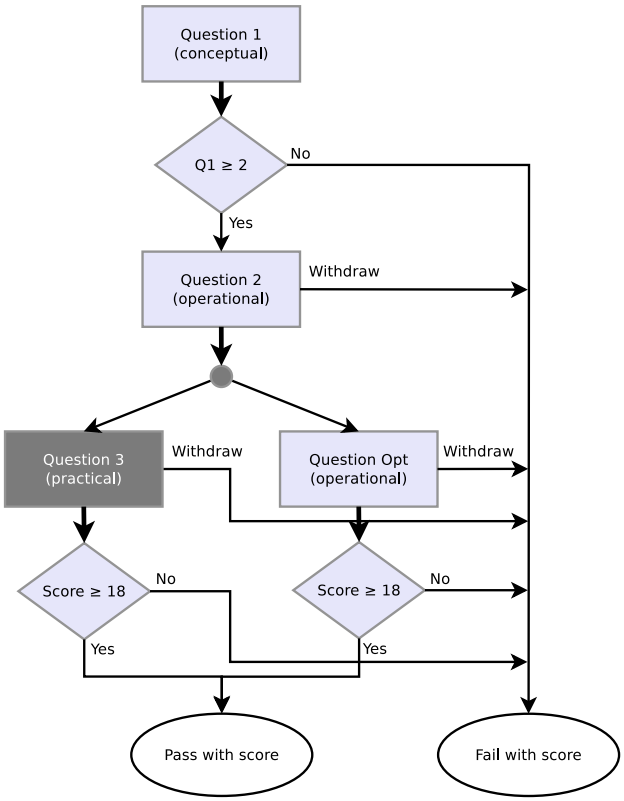
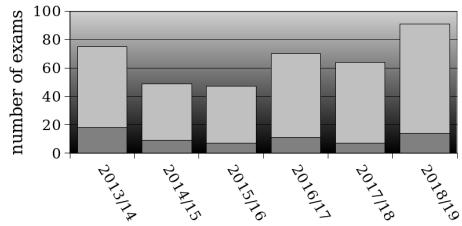
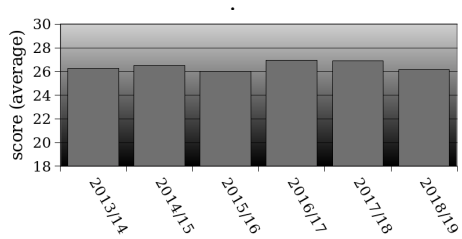


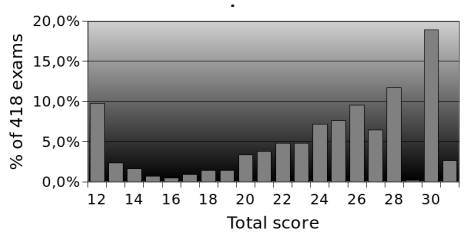
Fig. 1: Flow chart of an examination. Dark grey elements indicate that the step is partially controlled by the student. Fat arrows indicate the recording of the score by the software assistant. The final score is always recorded by the assistant .



(a) Successful (light gray) and unsuccessful (dark gray) exams per year (font: software assistant records)



(b) Average score of successful exams per year (font: institutional records)



(c) Distribution of exam scores, including unsuccessful ones (font: software assistant records)

Fig. 2: Statistics about the final examination

the option, incurring a penalty of 4 points, to replace the lab demonstration with another question (QX) similar to the second one. The procedure is summarized in Figure 1. A spreadsheet-based automated assistant helps the evaluator to adhere to the procedure, suggesting random questions from a pool, recording partial scores, and suggesting the final one.

The minimum score to pass the examination is 18/30, and the evaluator assigns a partial score in the interval $[0-6]$ to student's performance in each step. They are combined in a global evaluation using the equation:

$$\text{score} = 12 + \begin{cases} Q1+Q2+Q3 & \text{regular exam} \\ Q1+Q2+Qx-4 & \text{w/o lab demo} \end{cases} \quad (1)$$

Figure 2a shows the success rate for final examinations, showing in dark-grey the number of failures. During six years, the success rate (per year) is between 76% in 2013/14 and 89% in 2017/18. The average score (in Figure 2b) is approximately stable and slightly above 26/30.

The score distribution in Figure 2c exhibits a saddle shape, approximately centered around the score needed to pass the exam. This aspect means that the examination modality sharply differentiates candidates' preparation. Besides, the number of results falling in the fuzzy region just below the threshold is limited: only 2.2% falls in the interval $[15-17]$, where a lesser evaluation error may mean a relevant penalty for the candidate. The monotonically increasing shape towards higher scores indicates that students' preparation meets the expectations, and the stable trend confirms that students' motivation is persistent. The depletion at a score of 29 depends on the teacher's decision to systematically tilt a score of 29 to 30, corresponding to *full marks*, which makes a relevant difference in the student's curriculum. The score of 31 stands for a *cum laude*, appointed outside the defined schema in case of distinctive merit. Overall, the final examination appears to be reasonably challenging, with a failure rate of 25%. Frequently, the students passing the test achieve a grade better than 26 (36%), while the others have several possibilities during the year to repeat the exam.

Further insight into the learning process descends from the analysis of the student performance for each question, as shown in Figure 3. Every group of four adjacent columns represents a level of performance, from 0 (unsatisfactory) to 6 (very good). Each column in a group corresponds to a question: the first column, black, is for Q1, while the last one, light gray, is for Qx.

The column reporting the grades to the conceptual question (Q1) exhibits the same saddle shape seen for the distribution of the overall evaluation (in Figure 2c), and nearly 50% of the results are concentrated in the range $[5, 6]$. The first observation suggests that this question is sharply selective and

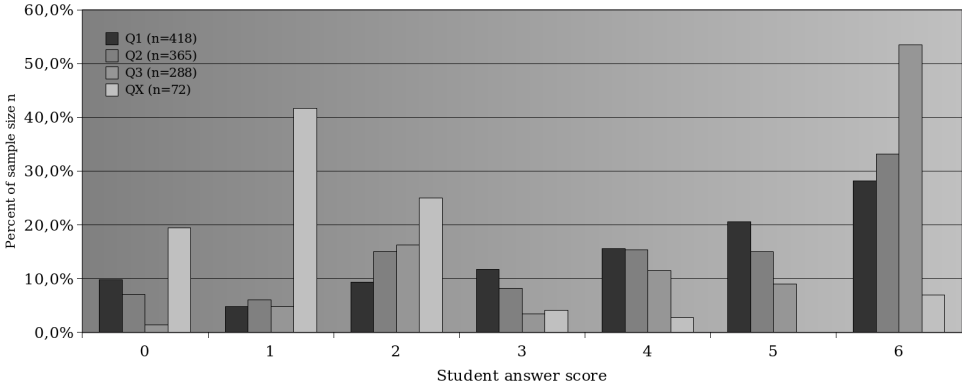


Fig. 3: Score per question during final examination (font: software assistant records)

effectively discriminates against the achievement of a sufficient understanding; when this is the case, the performance is frequently clear and accurate, and Q1 obtains a high score.

The second column, relative to question Q2, is also polarized but does not exhibit a saddle. Nearly 50% of the exams receive the two top scores, while the two lower ones represent less than the 15% of the total. The student has access to this step only if Q1 is acceptable, indicating a fair conceptual understanding. Therefore the operational details are relatively easy to learn and memorize, thus justifying the unbalanced score distribution.

The third column reports students' performance in the laboratory activity, Q3. The trend is similar to the one of Q2, showing 50% of scores on the top value, which means that the demo is usually well prepared. Likewise, the student with a sufficient conceptual and operational understanding generally performs a consistent demonstration on a selected topic.

In summary, the second and third evaluation steps have a limited impact on examination results: they are useful to motivate non-superficial study and practice, and the students meet the required target with a moderate effort.

The scores distribution for the question that optionally replaces the laboratory one (Qx, last column in Figure 3) confirms the value of practical activities: the students that decide to pay the four points penalty to avoid preparing the laboratory activity end up exhibiting a poor operational understanding.

A. Discussion

There is a two-way dependency between the act of designing that course and the guidelines outlined in this paper, which is relevant from an epistemological point of view. Most of the ideas explained in the previous sections were part of the teacher's approach during the initial steps of course design, and they helped in this process: for instance, the learning path definition and the relevance of lab activities. In a few other cases, the guidelines descend from experience: for example, in the case of the role of uniformity in the monitoring activity. The need for *comparable* monitoring results from different course editions emerged — and then included in the guidelines — when the series was long already.

The case study investigates three such monitoring techniques. One is an institutional survey providing raw results, which are useful for governance and do not provide an analytic insight into the course progress. Two other techniques are on the teacher's initiative: a survey and a focused final examination. The former provides relevant results, but its accuracy depends on students' collaboration, which is uncertain. Instead, students are motivated to give their best during the latter. In summary, the teacher that keeps the examination modality invariant in time and with a defined and focused rubric extracts from the results a wealth of reliable insights. Incidentally, the students are happy with fixed and fair rules.

The study reveals that lab utilization is an issue of the course: in principle, this result confirms monitoring efficacy in support of the paper's thesis and does not need further discussion. However, the story is worth a short note. In 2019/20, the teacher decided to give more time to lab activity during the lectures. One week after the beginning of the course, frontal lecturing was suspended as a measure against the spreading of the Coronavirus disease 2019 (COVID-19), nullifying the teacher's plan. Most of the activity moved on the Web, and the teacher asked the students to upload screenshots of at least two lab practices to have access to the final examination. Students' participation went over the expectation: 35 students uploaded the screenshots of over 50% of the proposed lab activities — 7 over 13 instead of two —, which at the moment is a more precise and favorable estimate compared with that given in Table V.

V. Conclusions

Several aspects contribute to the design of a course, including its educational target, students' expectations, time resources, and logistics. This paper deals with principles and guidelines that help the teacher in charge of planning a course.

The paper explores how such guidelines apply to a course in computer networking for a digital humanities school. The task is quite challenging, given that the topic is among the fundamental ones and the approach necessarily non-traditional. The report starts by defining an educational target adherent to the peculiar demand and planning the course implementation. The analysis of a six-year-long experience gives an insight into course dynamics over a long period. A relevant outcome

is the recognition of the role played by course monitoring and a study of how to improve its effectiveness; to this end, surveys are useful in and course-specific ones, designed by the teacher, are preferable since they focus on course-critical details. A suitable design of the final examination modalities allows inspecting the student's learning experience, providing clues for its improvement. References to exhaustive course materials enable the reproduction of that experience or its adaptation to different frameworks.

REFERENCES

[1] DH programs and syllabi. https://wiki.commons.gc.cuny.edu/DH_Programs_and_Syllabi/. last accessed in August 2020.

[2] Teaching concerns. https://usm.maine.edu/sites/default/files/assessment/Article-RubricsGrading.pdf, 2006. last accessed: July 2020.

[3] Lorin W. Anderson, Benjamin Samuel Bloom, et al. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman,, 2001.

[4] M. Anisetti, V. Bellandi, A. Colombo, M. Cremonini, E. Damiani, F. Frati, J. T. Hounsou, and D. Rebeccani. Learning computer networking on open paravirtual laboratories. *IEEE Transactions on Education*, 50(4):302–311, 2007.

[5] Jeffrey Bardzell and Shaowen Bardzell. Humanistic HCI. *Interactions*, 23(2):20–29, February 2016.

[6] Mordechai Ben-Ari. Constructivism in computer science education. *SIGCSE Bull.*, 30(1):257–261, March 1998.

[7] B.S. Bloom, D.R. Krathwohl, and B.B. Masia. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Number v. 1 in Taxonomy of Educational Objectives: The Classification of Educational Goals. D. McKay, 1956.

[8] Ka Ching Chan and M. Martin. An integrated virtual and physical network infrastructure for a networking laboratory. In *Computer Science Education (ICCSE), 2012 7th International Conference on*, pages 1433–1436, July 2012.

[9] A. Ciuffoletti. Teaching networks in the cloud. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 340–345, March 2016.

[10] Randy Connolly. Why computing belongs within the social sciences. *Communications of the ACM*, 63(8):54–59, 2020.

[11] Richard M. Felder and Rebecca Brent. The ABC's of engineering education: ABET, Bloom's taxonomy, cooperative learning, and so on. In *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*, volume 1, 2004.

[12] Asbjørn Følstad and Petter Bae Brandtzæg. Chatbots and the new world of HCI. *Interactions*, 24(4):38–42, June 2017.

[13] Ursula Fuller, Colin G. Johnson, Tuukka Ahoniemi, Diana Cukierman, Isidoro Hernán-Losada, Jana Jackova, Essi Lahtinen, Tracy L. Lewis, Donna McGee Thompson, Charles Riedesel, and Errol Thompson. Developing a computer science-specific learning taxonomy. *SIGCSE Bull.*, 39(4):152–170, December 2007.

[14] F. Galan, D. Fernandez, J. Ruiz, O. Walid, and T. de Miguel. Use of virtualization tools in computer network laboratories. In *Information Technology Based Proceedings of the FIfth International Conference onHigher Education and Training, 2004. ITHET 2004.*, pages 209–214, 2004.

[15] Sorin Hermon. Towards a curriculum in digital approaches to cultural heritage introduction. In *A Proposed Curriculum for Digital Heritage Studies Policies, Practices and Developments in Europe - Volume 3*. Information Society Technologies, 2008. Last accessed March 31 2020.

[16] Levin John. DH GIS Projects. http://anterotesis.com/wordpress/mapping-resources/dh-gis-projects/. Accessed on 2020/03/23.

[17] S. G. M. Koo and Sze Wan Kwong. Teaching computer communication networks: Top-down or bottom-up? In *Proceedings Frontiers in Education 35th Annual Conference*, pages S2H–S2H, Oct 2005.

[18] James F. Kurose and Keith W. Ross. *Computer networking: a top-down approach*. Addison Wesley.

[19] Liangxu Sun, Jiansheng Wu, Yujun Zhang, and Hang Yin. Comparison between physical devices and simulator software for Cisco network technology teaching. In *2013 8th International Conference on Computer Science Education*, pages 1357–1360, 2013.

[20] Peter Lunenfeld, Anne Burdick, Johanna Drucker, Todd Presner, and Jeffrey Schnapp. Digital_humanities. *Cambridge, MA: MIT Press. Retrieved January*, 12:2014, 2012.

[21] Jay McTighe and Grant Wiggins. Understanding by design framework. White paper 312, Association for Supervision and Curriculum Development, Alexandria, VA, 2012.

[22] Robert Montante. Using Scapy in teaching network header formats: Programming network headers for non-programmers (abstract only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 1106–1106, New York, NY, USA, 2018. ACM.

[23] Steve Mvalo. *Developing computer networks students' computational thinking: the case for the use of simulation software*. PhD thesis, University of Reading, 2019.

[24] Ricardo Nabhen and Carlos Maziero. Some experiences in using virtual machines for teaching computer networks. In Deepak Kumar and Joe Turner, editors, *Education for the 21st Century — Impact of ICT and Digital Resources*, pages 93–104, Boston, MA, 2006. Springer US.

[25] Johanna Nichols and Tandy Warnow. Tutorial on computational linguistic phylogeny. *Language and Linguistics Compass*, 2(5):760–820, 2008.

[26] Douglas Seefeldt and William G. Thomas III. What is digital history? a look at some exemplar projects. https://digitalcommons.unl.edu/historyfacpub/98/, 2009.

[27] Dannelle D. Stevens and Antonia J. Levi. *Introduction to rubrics: An assessment tool to save grading time, convey effective feedback, and promote student learning*. Stylus Publishing, LLC, 2013.

[28] Zeljko Stojanov, D. Dobrilovic, and Tamara Zorić. Exploring students' experiences in using a physical laboratory for computer networks and data security. *Computer Applications in Engineering Education*, 25, 02 2017.

[29] Andrew S. Tanenbaum and David J. Wetherall. Computer networks. *Prentice Hall*, 2011.

[30] Michael Thuné and Anna Eckerdal. Analysis of students' learning of computer programming in a computer laboratory context. *European Journal of Engineering Education*, 44(5):769–786, 2019.

[31] Richard White. What is spatial history? https://web.stanford.edu/group/spatialhistory/cgi-bin/site/pub.php?id=29, 2010. Last accessed 2020/03/23.

[32] Barry Wilkinson and Clayton Ferner. Towards a top-down approach to teaching an undergraduate grid computing course. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, page 126–130, New York, NY, USA, 2008. Association for Computing Machinery.

[33] Jeannette M. Wing. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3717–3725, 2008.

[34] Nursel Yalcin, Yalcin Altun, and Utku Kose. Educational material development model for teaching computer network and system management. *Computer Applications in Engineering Education*, 23(4):621–629, 2015.

[35] Li Yang. Teaching system and network administration using virtual pc. *J. Comput. Sci. Coll.*, 23(2):137–142, December 2007.

Augusto Ciuffoletti Augusto Ciuffoletti was born in Genova (Italy) in 1956. He graduated in Computer Science in 1980 at the University of Pisa, with a thesis on the layered representation of distributed systems. Since 1987 he is with the University of Pisa as a Researcher. From 2001 to 2010, he had a collaboration with the Italian Institute of Nuclear Physics for funded projects on Grid and Cloud monitoring. Later he joined the Open Grid Forum (OGF) as a member of the Open Cloud Computing Interface (OCCI) Working Group. He is currently investigating networks of restricted devices. Meanwhile, the University of Pisa assigned him several teaching roles: since 2000, he has been teaching computer networking for the Course in Computer Science, and in 2013 he happily landed on the Course in Digital Humanities, with the experience summarized in this paper.