U. Schmid, W. Kastner, H. Haberstroh

Technical University of Vienna
Department of Automation
Email: {s, k, hh}@auto.tuwien.ac.at

Process Control Education for Computer Science: From Fiction Back to Facts

Abstract. This paper provides an overview to contents and organization of our (second generation) introductional practical *Process Automation* taught to students of computer science at the Technical University of Vienna since 1993. Justified by two years of experience, we are now convinced that we succeeded in finding a reasonable tradeoff w.r.t. the many issues that must be considered when setting up a practical of this kind.

Keywords. process control education, process automation practical, computer science.

1 Introduction

In a former paper [SHS93], we elaborated on the many difficulties that come up when introducing process control topics to undergraduate computer scientists. Some of the major issues we identified are as follows.

- There are 200–300 students of all branches of computer science that have to attend our course every year.

  How to organize a practical for that many students?

- The course must be run by a very small staff and with limited budget for equipment.

  What kind of practical is suitable in this respect?

- There is only limited time ($\approx$ 2 hours/week lecture and $\approx$ 1 hour/week practical) available for the course.

  What may reasonably be done within that time?

- Only 10% of the participants are students of a technical branch of computer science and therefore interested in process control topics; the remaining 90% will probably never need such knowledge.

  What topics should nevertheless be taught?

- The technical/engineering background (electronics, mechanics, physics, even mathematics) of participating students varies from almost zero to a quite reasonable level.

  What topics can reasonably be taught in spite of that situation?

Planning such a course means answering two questions, subjected to the invariants above:

- *Contents*: What topics should be taught?
- *Organization*: How should those topics be taught?

We have been considering these two questions rather thoroughly during the last few years of setting up and tuning our introductional course *Process Automation*, which is dedicated to an introduction of process control topics to computer science undergraduates. It consists of a more general lecture designed and run by the department's head, Prof. G.-H. Schildt, and a specializing practical designed and run by the authors. Naturally, it was primarily the practical that was changed considerably every year. Apart from annual tuning based on last-year experience, we even had to replace our first generation practical (described in [SS91]) by a second generation one based on totally different contents, equipment and organization in 1993.

This paper presents the current state of our second generation practical[1] (cf. [SKFG95]) as well as the experiences gathered during the last two years. Its remainder is organized as follows: The subsequent Section 2 is devoted to contents and equipment used; organizational matters —both from the point of view of students and staff— are presented in Section 3. The paper is concluded in Section 4 with some experiences and expectations, including future improvements.

2 Contents and Equipment

During the evolution of our course, the contents of the practical have been both continuously narrowed down from multiple subjects to a single one and deepened from a more general to a quite detailled treatment of the topic in question. Starting from classical control theory and petri nets as well as programming process control applications, we now focus entirely upon the latter, cf. [SHS93] for a discussion of the reasons why. More specifically, in view of the background in system software and programming that may reasonably be assumed for computer science undergraduates, it was only natural to choose the process interface part of automation systems as the central issue of the practical.

There are three "milestones" of knowledge aquisition in the course of the practical, corresponding to three different programming exercises. We provided a number of (roughly equivalent) concise examples for each milestone, which are individually assigned to each participant. Each example stresses the major issue directly, i.e., without unnecessary overhead in order to comply to the 1 hour time limit. Note, however, that a certain level of complexity is nevertheless necessary to make the exercises interesting (ideally fun) even for less interested participants. Working on the examples is supported by a carefully written script (see [SKFG95]) that contains all the material required.

2.1 Basics of digital and analog I/O

- *Digital I/O: memory-mapped input/output, change-of-state interrupts.* The exercises are based on switch and LED panels, seven-segment displays and other simple devices directly connected to simple digital process interfaces.

- *Analog I/O: simple analog input/output.* For those exercises, we provided an oscilloscope, an adjustable power supply, and a waveform generator directly connected to analog process interfaces, see Figure 6.

[1] Since practical and lecture are reasonably independent, we may safely omit describing the latter.

2.2 Advanced I/O

To become familiar with the capabilities of modern multifunctional, programmable process interfaces, we provided a multiplex-display and a multiplex-keypad which are directly connected to digital input/output interfaces based on the Zilog Z8536 CIO-chip. The actual exercises deal with programming the "user interface part" of simple devices like a microwave cooker or a digital HIFI-tuner. The following Figure 1 shows our I/O-panel.

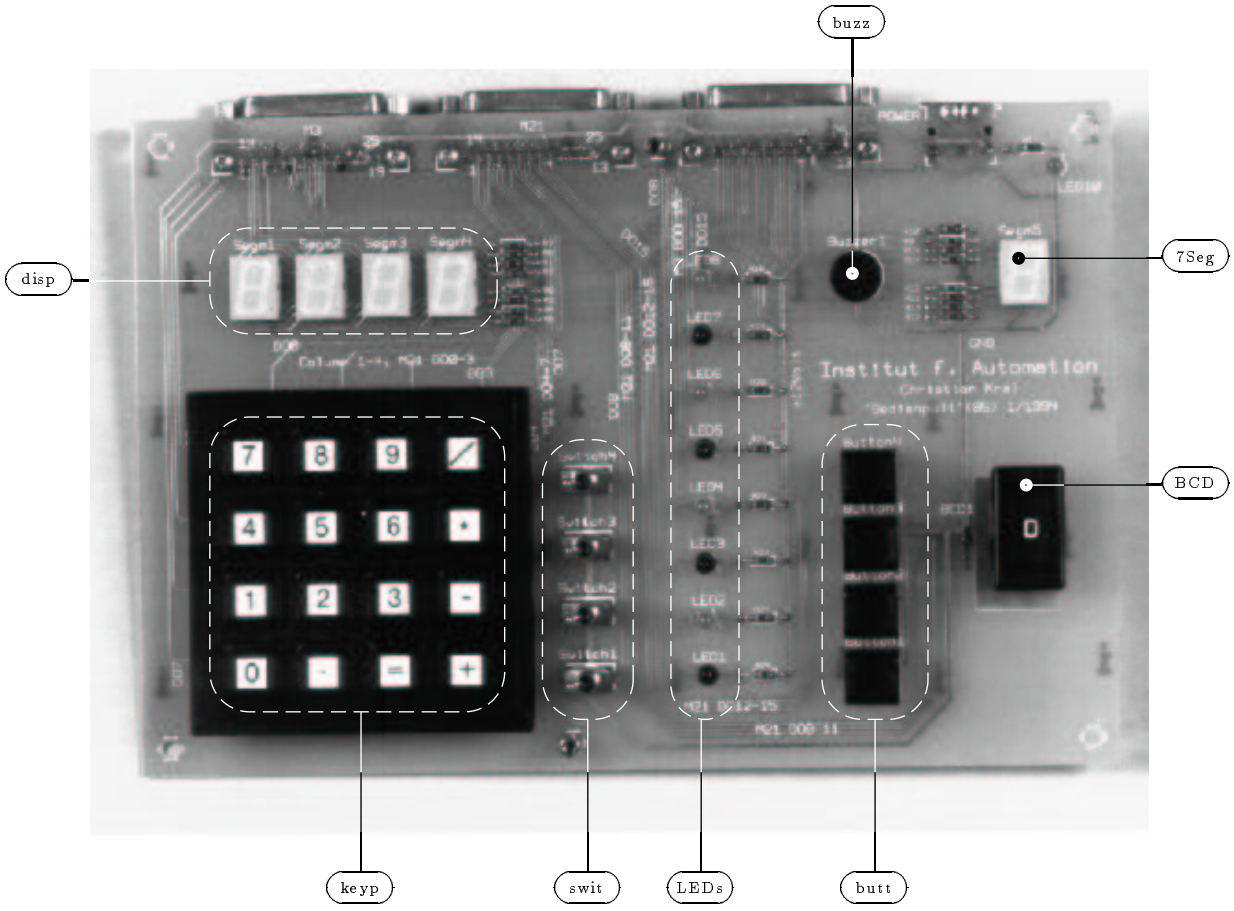

Figure 1: *Photo I/O-panel*

I/O-panel	
position	**description**
keyp	matrix keypad
swit	switches
butt	buttons
BCD	BCD-switch
LEDs	LED's
7Seg	seven segment display
disp	matrix display
buzz	buzzer

2.3 Programming sensor/actor devices

To introduce the capabilities of modern sensor and actor devices, we provided a model of an automation system that incorporates a wide variety of different ones, including a DC- and a stepper-motor as well as an incremental encoder, proximity switches and a differential transformer. More specifically, we designed a model of a testbed for checking the mechanical operation of a 5.25" floppy disk drive. The following figures show our testbed from several different angles.

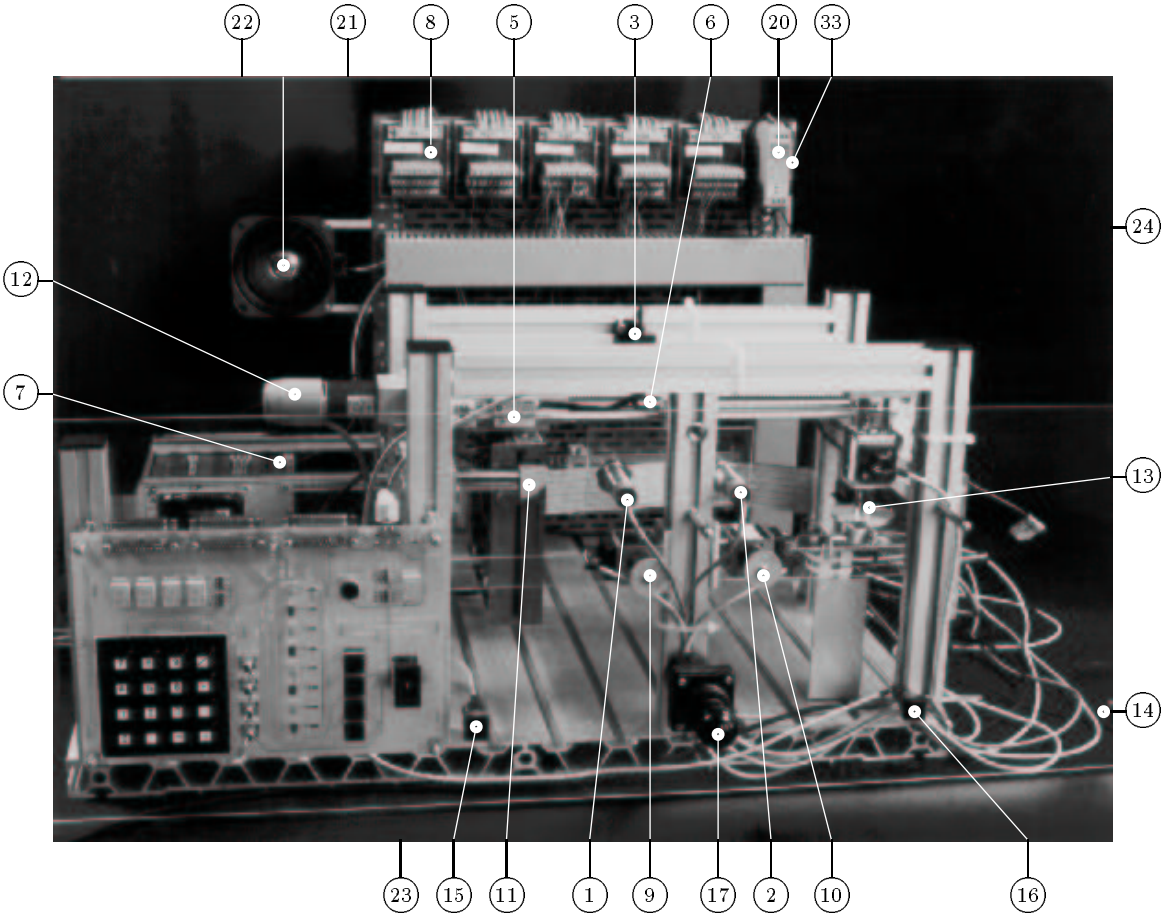

Figure 2: *Photo Floppy-testbed (frontside)*

Floppy-testbed	
position	**description**
1	capacitive proximity switch
2	capacitive proximity switch
3	markreader
4	magnet
5	magnetic proximity switch
6	magnetic proximity switch
7	power supplies
8	screw terminals
9	actuator

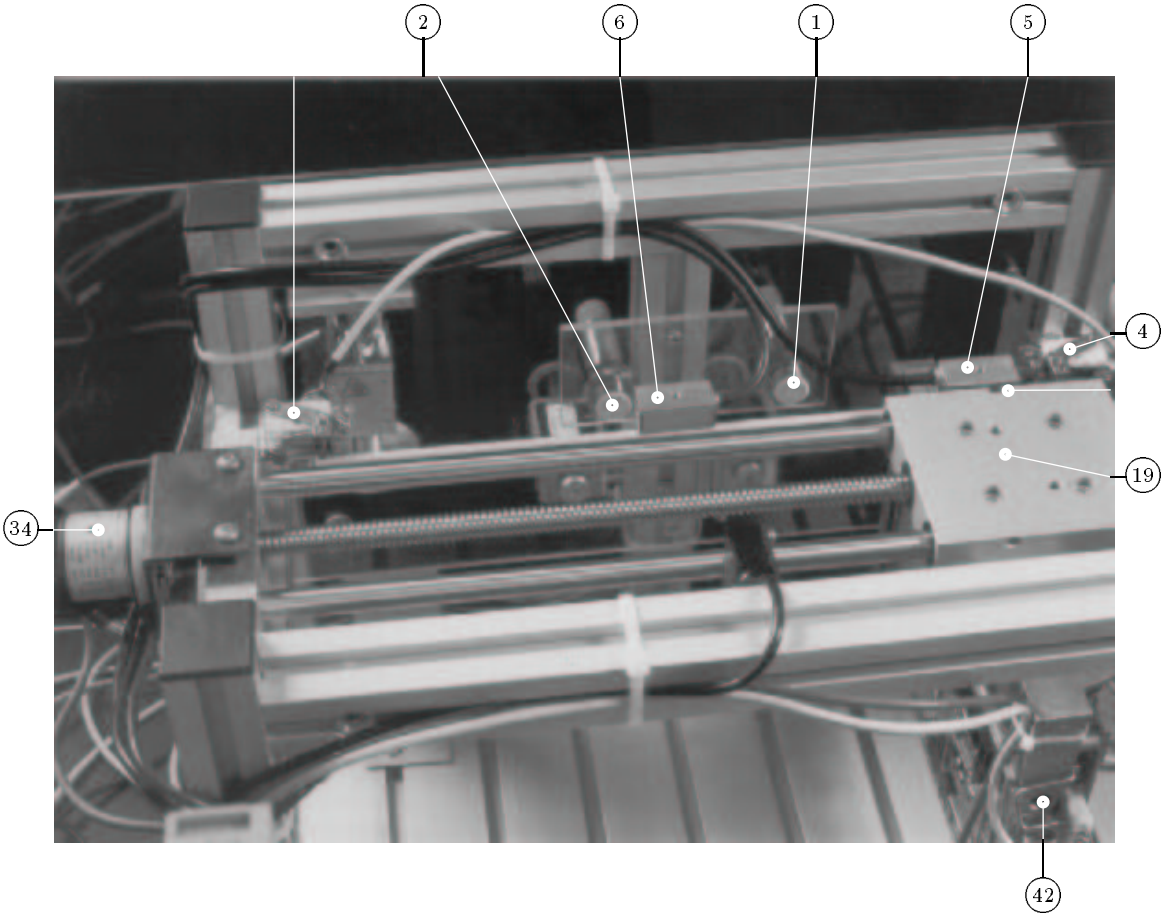

Figure 3: *Photo Floppy-testbed (backside)*

Floppy-testbed cont.	
position	**description**
(10)	actuator
(11)	closing equipment
(12)	DC-motor
(13)	actuator
(14)	light barrier
(15)	manual control switch
(16)	manual control switch
(17)	joystick
(19)	sledge
(20)	transducer
(21)	frame
(22)	switchbox
(23)	limit switch
(24)	limit switch
(30)	induktive proximity switch
(32)	optical fibre

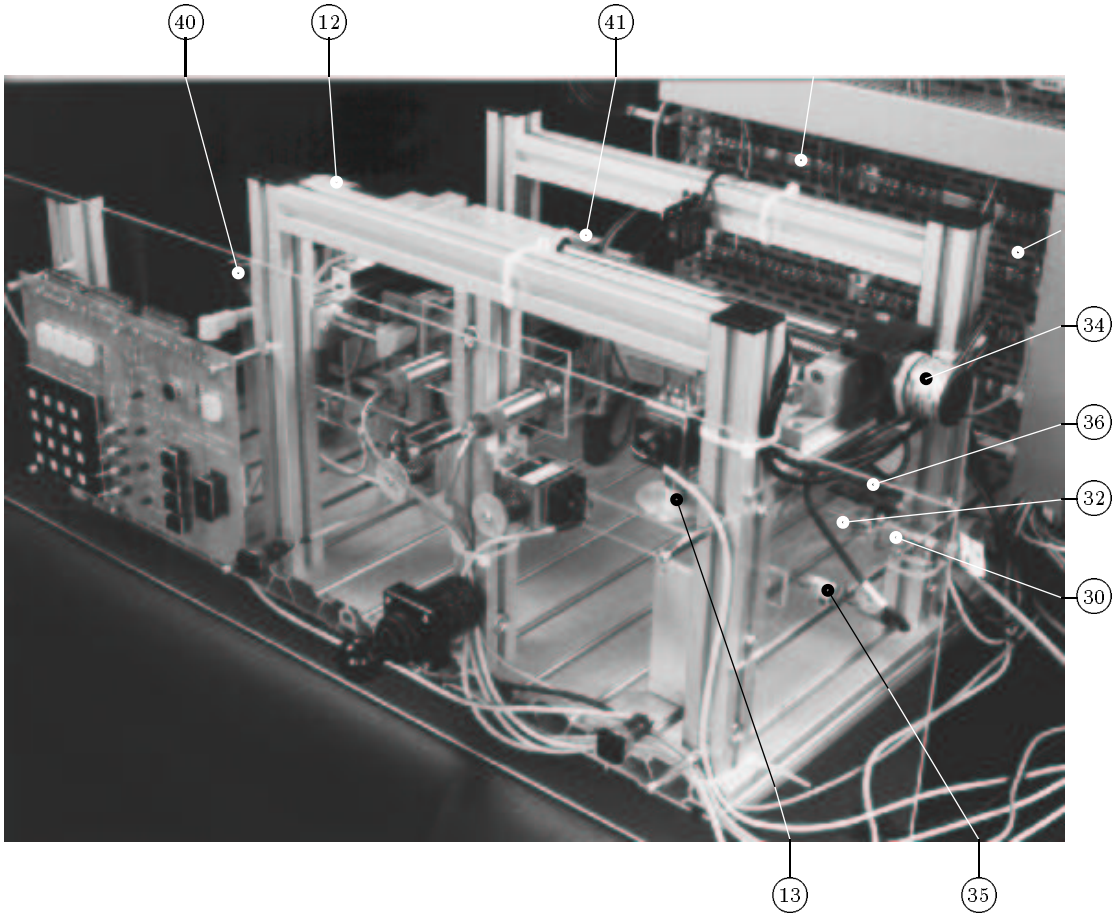

Figure 4: *Photo Floppy-testbed (right)*

Floppy-testbed cont.	
position	**description**
(33)	photosensor with fibre optic
(34)	incremental encoder
(35)	temperature sensor
(36)	markreader
(40)	distance sensor (triangulation-based)
(41)	barcode
(42)	differential transformer

Programming the computer system controlling the testbed to perform some meaningful operation is of course a complex issue. To make the exercises concise, we provided our participants with most parts of the application software, leaving open only one specific task, like

- measurement of speed or temperature,
- stepper motor positioning,
- velocity and/or position control,
- barcode reading.

This procedure allows focussing on details within the framework of a complete solution that is fun to work on. Moreover, students learn to read and to adhere to the specification of the provided part of the solution, a necessary prerequisite for future team work.

2.4 Equipment

All of the abovementioned process models are kept as simple and robust as possible. They are all connected to an M68030-based VME microprocessor *target system* via the low-cost, multi-vendor *M-Modules* I/O-System, which provides dozens of different process interface modules at low cost, see [MMSa] and [MMSb] for details. Programming the more advanced exercises is supported by the target's real-time multitasking operating system (ISI/SCG's pSOS^{+m} and add-on's).

The development environment employed for the practical consists of 3 identical *development cells*, each comprising four PC-based *workstations* and one of the (abovementioned) target system (including all process models) connected via Ethernet. The workstations provide a *C* cross-development environment (Microtec MCC68K^{+}, XRAY68K^{+}) supporting our target systems. All development cells are also connected to an additional DEC Alpha *(file) server*. The following Figure 5 shows the schematics of a single development cell; Figure 6 is an appropriate photo.

Programs may be developed and compiled at any workstation simultaneously; the actual degree of simultaneity depends on the number of software licences we can afford. Debugging is also commanded and controlled from the workstations, but requires (of course) the target system and is therefore mutual exclusive (that is, per development cell). Debugging access to the target system is in fact controlled by a suitable reservation system. Note that any workstation provides the same capabilities since all user data are located at the file server and all development cells are identical.

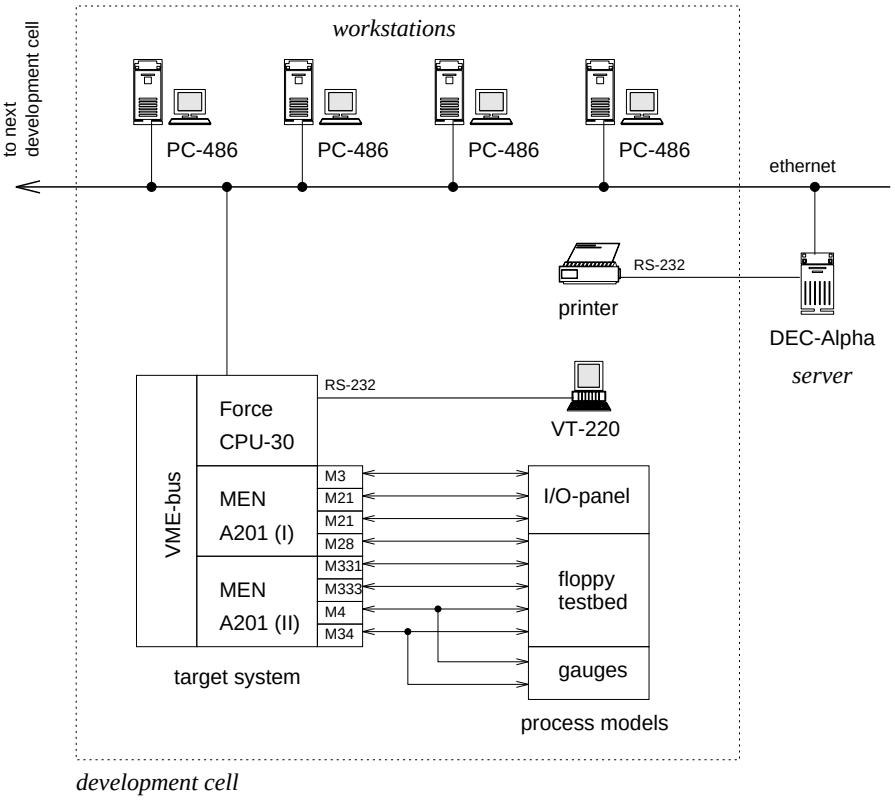

Figure 5: *Schematics development cell*

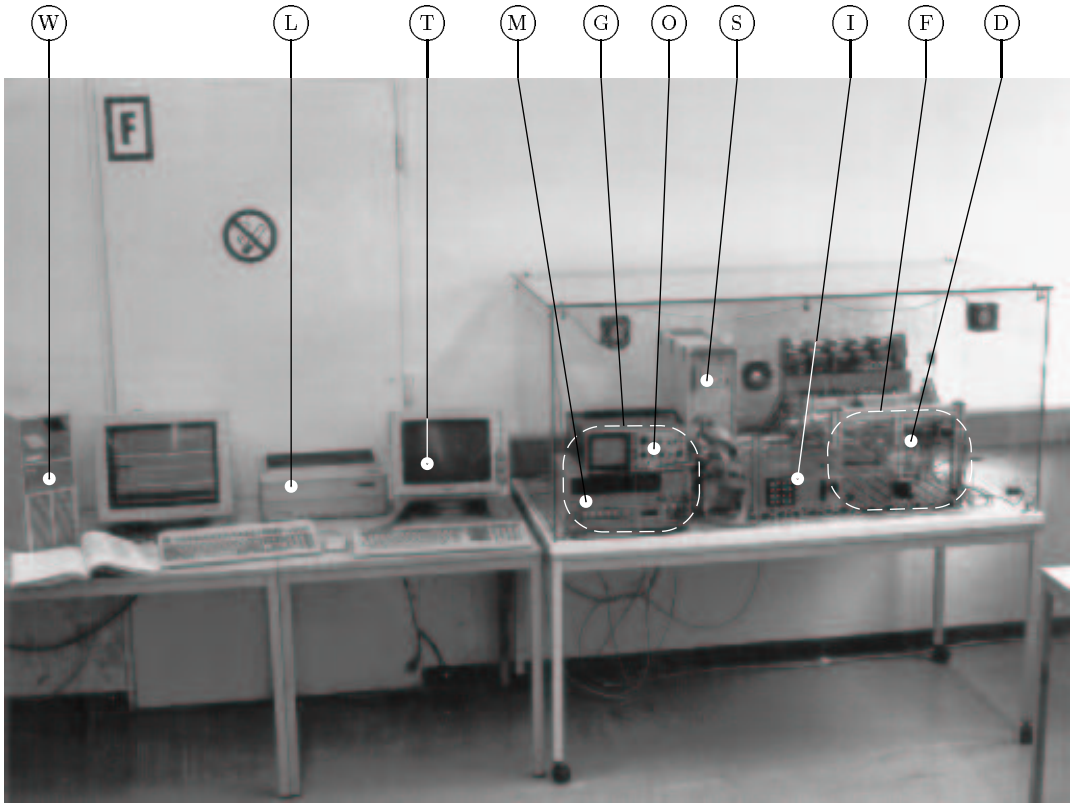

Figure 6: *Photo development cell*

development cell	
position	**description**
(W)	workstation
(L)	line printer
(T)	VT-220 terminal
(S)	target system
(G)	measuring instruments
(O)	oscilloscope
(M)	multifunctional instrument including adjustable power supply, waveform generator, frequency counter and digital voltmeter
(I)	I/O-panel
(F)	floppy testbed
(D)	floppy drive

3 Organization

Despite of the large number of participants, we decided to organize our practical largely as a classical one, cf. [SHS93]. Each student has to work on indiviually assigned exercises and to present the solutions at three pre-scheduled dates to their supervisor. All presentations are accompanied by a certain examination of the participants related/background knowledge.

This allows a relatively objective determination of the individual capabilities, based on several ratings.

In order to run this type of practical for 200–300 students with acceptable effort, it must of course be organized carefully and strictly. In fact, to support organization and examination, we have developed several tools that greatly reduce the burden imposed on the staff. In the following sections we will describe organizational matters, both from the students' and the staffs' view.

3.1 The Student's View

Starting with the end of the preceeding term, students may register for the practical on a dedicated workstation (outside of our department) and may purchase the supporting teaching material, i.e., the comprehensive script that also includes a guideline through the practical and the collection of introductional exercises. The registration program records the student's data, selects an assistant professor supervising the participant throughout the whole course, and assigns a first exercise (out of the collection of exercises) to be solved.

During the first week of the practical, students may familiarize themselves with the hardware and software environment. Trained tutors and assistant professors explain how to use the equipment, and a demonstration program helps in getting acquainted with compiler, linker and debugger. To work on their exercises, students may use our workstations at any time between 8:00 a.m. and 10:00 p.m. every day for a maximum of 6 hours a week; this restriction is enforced by our workstation reservation system.

According to Section 2, there are three exercises to be solved during the practical, with approximately one month of time available for each of them. To solve questions and problems arising when working on the exercises, assistant professors and tutors hold regular consultation hours every week. Students may also direct their questions (anonymous) to a FAQ[2] system via email and get the responses —within one day— via a WWW-homepage set up for the pracical (http://www.cslab.tuwien.ac.at/pa/pa-home.html). The solution to each of the exercises must be documented in a comprehensive lab report, forming the basis of the subsequent presentation.

The examination associated with the presentation of the first exercise is a written test, whereas the following ones are oral ones. Scheduling the supervisors' time for presentations/examinations is done by means of a reservation tool, aiding the participants in choosing a suitable time slot. During the presentation, each student has to demonstrate that his/her solution works and to discuss the underlying ideas by means of the lab report. The students background knowledge and familiarity with the presented solution is examined by asking certain questions related to the exercise. Oral examinations are supported by a data base tool developed specifically for that purpose, see 3.2. Finally, the examination ends with the assignment of the next exercise.

Apart from the exercises, which constitute the core of our practical, there are also some accompanying measures. For example, to help undergraduates in getting aquainted with real business life, we introduced regular presentations of a suitable company during our course. People from process automation industries are invited to present their company and, in particular, the demands they make on their staff. Such additional service seem to be well appreciated by most of our students.

[2]Frequently Asked Questions.

3.2 The Staff's View

The staff's work for the practical already starts in the preceeding semester. A few of the most important routine tasks that need to be done are:

- invitation for company presentation,
- reservation of lecture rooms for test and presentation,
- updating examination questions in the data base,
- installation and test of (new) tools for the practical,
- installation of the programming environment,
- testing hardware and software components,
- recruiting tutors.

Tutors are usually selected out of those students who have completed the practical with best ratings in former years. Note that an exhaustive training of tutors immediately prior to the course is nevertheless inevitable. We normally employ ten tutors holding two hours consultation per week.

Based on the students' registration information, the preparations for the course are eventually completed by

- updating and copying teaching material,
- setting up accounts/home directories and the examination database

With the beginning of the semester, the staff is fully in charge of the practical. Consulatation hours are held in the lab, the FAQ system has to be maintained, and problems with the hardware and/or software system that may arise must be solved quickly.

The most exhaustive tasks from the staff's point of view arise from holding oral presentations/examinations, which are usually performed within one specified week for each exercise. Since there are three assistant professors (and also three target systems) available for examinations, we run only two examinations concurrently so that a third assistant professor (or target system) can step in when one of the two drops out. Participants may register for a specific (usually 20 minutes) time slot of their supervisor's time within the specified week by means of a reservation tool.

To reduce the load imposed by the many participants, we actually

- perform a written test for the first examination, which is usually "survived" only by a certain percentage of the initial participants,

- aid subsequent oral examinations by our data base tool developed specifically for that purpose.

The abovementioned data base tool contains both an exhaustive collection of questions (and solutions) related to the exercises and the students' records for each examination. The following Figure 7 shows a snapshot of the user interface part of the data base tool when collecting/updating examination questions. This feature is normally used only in the preparation phase, prior to running the course.

To assist supervisors during an actual examination, the data base tool provides easy access to a student's previous record (name, assigned exercise, marks of previous examinations, etc.).

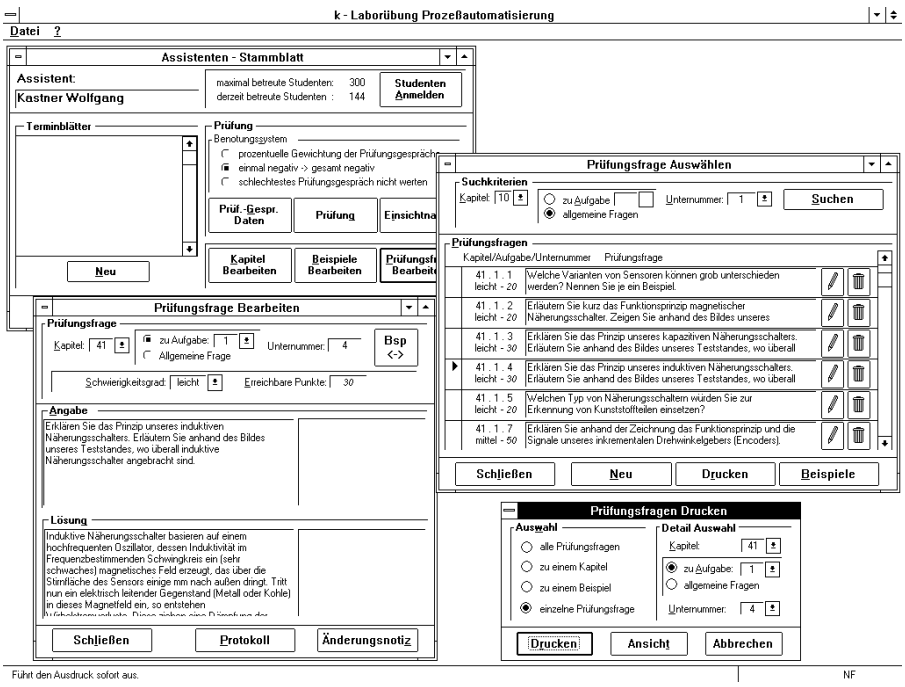

Figure 7: *Snapshot registration of examination questions*

Questions may be selected and chosen quickly, and predefined rating criteria help greatly in objectivly assessing the answers given. Figure 8 gives an impression of the user interface part for the examination phase.

When an examination is finished, the resulting mark is computed by the system and a protocol documenting questions and given answers is printed. In conjunction with the lab report collected from the student, this provides us with printed documents satisfying all legality aspects. After the final examination, the system automatically computes final marks and also related statistics.

4 Conclusions

Our second generation practical has been run by the authors two times (1994, 1995), with the following results:

- Programming of process interfaces for real process models seems to be interesting even to a non-technical computer scientist; there are some students playing around day and night with our target systems.
- The floppy-testbed proved to be not as robust as we expected it to be. We had (and still have) to modify a few things to avoid maintenance overhead and even damage.
- Personal supervision and oral examination provides both good assessment of actual understanding and feedback. However, to reduce the resulting load for the staff, it is inevitable to organize the first examination as a written test.
- Collecting lab reports documenting the solution, experiences, and complaints from all participants provides optimal feedback for annual tuning of the practical. Various sta-

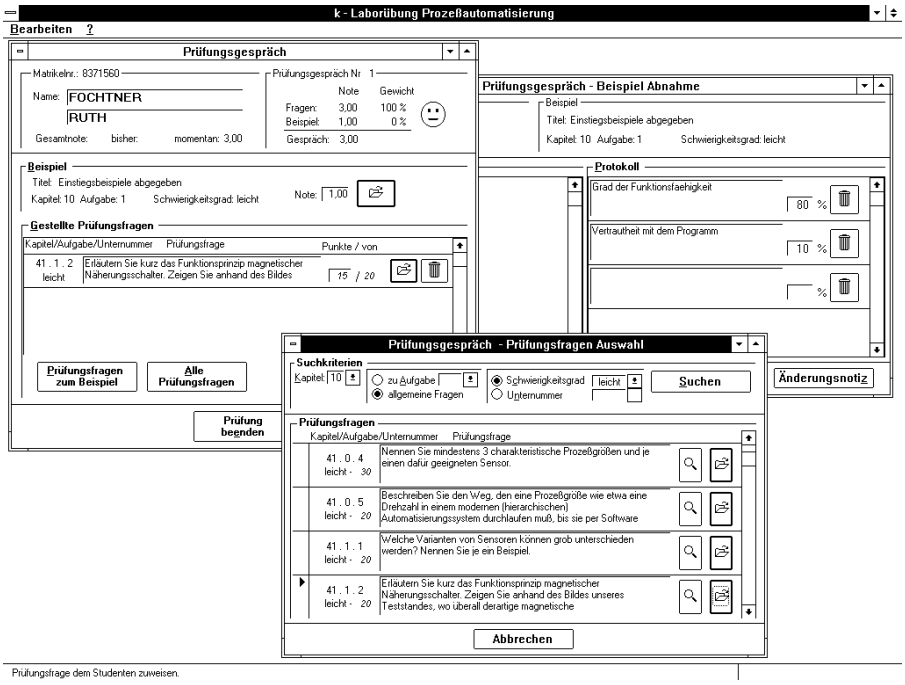

Figure 8: *Snapshot examination*

tistics computed by our data base tool help also in identifying issues our students have problems with.

- Exhaustive training of tutors is absolutely mandatory.

There are also some issues we expect our practical to fulfill, without being able to confirm whether this is actually true by now.

- The practical should be flexibile and scalable. It is affordable to discard inappropriate exercises and even models, and to add new subject matters and related examples.

Finally, there are obviously numerous possibilities to improve the current practical. Among them are:

- Computer supported tutoring should be introduced to make individual scheduling of work time more flexible.
- In view of the limited time it is of vital importance to exploit existing knowledge. Therefore, we cannot afford the overhead associated with introducing a new programming language like *PEARL*. However, relying on the *C* language, which is the "heart" of an (earlier) course in *System Programming*, is what industry calls for, but is not really suitable for real-time programming.

Acknowledgments

Many people contributed to contents and organization the practical described in this paper. The efforts of Christian Kral, Martin Laubach, Johann Klasek, Fred Fajtak, Bernhard Gödel, Bettina Weiss, Thomas Chiba, Fritz-Peter Mörth, Mario Weilguni, and Günther Gridling are explicitely acknowledged.

References

[MMSa] MUMM e.V.: *Basic M-Modules Specification*, Nürnberg, Germany.

[MMSb] MUMM e.V.: *M-Module Directory*, Nürnberg, Germany, August 1993.

[SS91] Schmid, U.; Stöckler, St.: *Konzept der Laborübung Prozeßautomatisierung*, Techn. Univ. Wien, Inst. f. Automation, Report 183-1/21, June 1991. (in german)

[SHS93] Schmid, U.; Haberstroh, H.; Stöckler, St.: *Process Control Education for Computer Science: Facts and Fiction*, Proc. SEFI/TEMPUS JEP 2011 - IMPACT Workshop on Computer Science Topics for Control Engineering Education, Vienna, September 1993, p. 13–22.

[SKFG95] Schmid, U.; Kastner, W.; Fajtak, F.; Gödel, B.: *Übungsskriptum Laborübung Prozeßautomatisierung*, Techn. Univ. Wien, Inst. f. Automation, 1995. (in german)